

2nd Conference on Information-Theoretic Cryptography

ITC 2021, July 23–26, 2021, Virtual Conference

Edited by

Stefano Tessaro



Editor

Stefano Tessaro

University of Washington, Seattle, WA, USA
tessaro@cs.washington.edu

ACM Classification 2012

Security and privacy → Cryptography; Mathematics of computing → Information theory; Theory of computation → Computational complexity and cryptography

ISBN 978-3-95977-197-9

Published online and open access by

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at <https://www.dagstuhl.de/dagpub/978-3-95977-197-9>.

Publication date

July, 2021

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <https://portal.dnb.de>.

License

This work is licensed under a Creative Commons Attribution 4.0 International license (CC-BY 4.0):
<https://creativecommons.org/licenses/by/4.0/legalcode>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.ITC.2021.0

ISBN 978-3-95977-197-9

ISSN 1868-8969

<https://www.dagstuhl.de/lipics>

LIPICs – Leibniz International Proceedings in Informatics

LIPICs is a series of high-quality conference proceedings across all fields in informatics. LIPICs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

Editorial Board

- Luca Aceto (*Chair*, Reykjavik University, IS and Gran Sasso Science Institute, IT)
- Christel Baier (TU Dresden, DE)
- Mikolaj Bojanczyk (University of Warsaw, PL)
- Roberto Di Cosmo (Inria and Université de Paris, FR)
- Faith Ellen (University of Toronto, CA)
- Javier Esparza (TU München, DE)
- Daniel Král' (Masaryk University - Brno, CZ)
- Meena Mahajan (Institute of Mathematical Sciences, Chennai, IN)
- Anca Muscholl (University of Bordeaux, FR)
- Chih-Hao Luke Ong (University of Oxford, GB)
- Phillip Rogaway (University of California, Davis, US)
- Eva Rotenberg (Technical University of Denmark, Lyngby, DK)
- Raimund Seidel (Universität des Saarlandes, Saarbrücken, DE and Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, DE)

ISSN 1868-8969

<https://www.dagstuhl.de/lipics>

■ Contents

Preface	
<i>Stefano Tessaro</i>	0:vii

Regular Papers

Group Structure in Correlations and Its Applications in Cryptography	
<i>Guru-Vamsi Policharla, Manoj Prabhakaran, Rajeev Raghunath, and Parjanya Vyas</i>	1:1–1:23
More Communication Lower Bounds for Information-Theoretic MPC	
<i>Ivan Bjerre Damgård, Boyang Li, and Nikolaj Ignatieff Schwartzbach</i>	2:1–2:18
On Prover-Efficient Public-Coin Emulation of Interactive Proofs	
<i>Gal Arnon and Guy N. Rothblum</i>	3:1–3:15
On the Randomness Complexity of Interactive Proofs and Statistical Zero-Knowledge Proofs	
<i>Benny Applebaum and Eyal Golombek</i>	4:1–4:23
Line-Point Zero Knowledge and Its Applications	
<i>Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky</i>	5:1–5:24
ZK-PCPs from Leakage-Resilient Secret Sharing	
<i>Carmit Hazay, Muthuramakrishnan Venkatasubramanian, and Mor Weiss</i>	6:1–6:21
Secure Merge with $O(n \log \log n)$ Secure Operations	
<i>Brett Hemenway Falk and Rafail Ostrovsky</i>	7:1–7:29
Perfectly Oblivious (Parallel) RAM Revisited, and Improved Constructions	
<i>T-H. Hubert Chan, Elaine Shi, Wei-Kai Lin, and Kartik Nayak</i>	8:1–8:23
On the Complexity of Anonymous Communication Through Public Networks	
<i>Megumi Ando, Anna Lysyanskaya, and Eli Upfal</i>	9:1–9:25
Broadcast Secret-Sharing, Bounds and Applications	
<i>Ivan Bjerre Damgård, Kasper Green Larsen, and Sophia Yakoubov</i>	10:1–10:20
Locally Reconstructable Non-Malleable Secret Sharing	
<i>Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, Sruthi Sekar, and Jenit Tomy</i>	11:1–11:19
Linear Threshold Secret-Sharing with Binary Reconstruction	
<i>Marshall Ball, Alper Çakan, and Tal Malkin</i>	12:1–12:22
Doubly-Affine Extractors, and Their Applications	
<i>Yevgeniy Dodis and Kevin Yeo</i>	13:1–13:23
Online Linear Extractors for Independent Sources	
<i>Yevgeniy Dodis, Siyao Guo, Noah Stephens-Davidowitz, and Zhiye Xie</i>	14:1–14:14
Code Offset in the Exponent	
<i>Luke Demarest, Benjamin Fuller, and Alexander Russell</i>	15:1–15:23



P_4 -free Partition and Cover Numbers & Applications <i>Alexander R. Block, Simina Brânzei, Hemanta K. Maji, Himanshi Mehta, Tamalika Mukherjee, and Hai H. Nguyen</i>	16:1–16:25
Replacing Probability Distributions in Security Games via Hellinger Distance <i>Kenji Yasunaga</i>	17:1–17:15
Differentially Private Approximations of a Convex Hull in Low Dimensions <i>Yue Gao and Or Sheffet</i>	18:1–18:16
Differentially Oblivious Database Joins: Overcoming the Worst-Case Curse of Fully Oblivious Algorithms <i>Shumo Chu, Danyang Zhuo, Elaine Shi, and T-H. Hubert Chan</i>	19:1–19:24
Communication Complexity of Private Simultaneous Quantum Messages Protocols <i>Akinori Kawachi and Harumichi Nishimura</i>	20:1–20:19
Quantum-Access Security of the Winternitz One-Time Signature Scheme <i>Christian Majenz, Channele Matadah Manfouo, and Maris Ozols</i>	21:1–21:22
On the Security of Proofs of Sequential Work in a Post-Quantum World <i>Jeremiah Blocki, Seunghoon Lee, and Samson Zhou</i>	22:1–22:27
Fooling an Unbounded Adversary with a Short Key, Repeatedly: The Honey Encryption Perspective <i>Xinze Li, Qiang Tang, and Zhenfeng Zhang</i>	23:1–23:21
T_5 : Hashing Five Inputs with Three Compression Calls <i>Yevgeniy Dodis, Dmitry Khovratovich, Nicky Mouha, and Mridul Nandi</i>	24:1–24:23
Post-Compromise Security in Self-Encryption <i>Gwangbae Choi, F. Betül Durak, and Serge Vaudenay</i>	25:1–25:23
Generic-Group Identity-Based Encryption: A Tight Impossibility Result <i>Gili Schul-Ganz and Gil Segev</i>	26:1–26:23

■ Preface

In its second edition, the conference on Information-Theoretic Cryptography (ITC) was once again affected by the COVID-19 pandemic. After an initial optimistic attempt to organize a hybrid conference in Bertinoro, Italy, with Daniele Venturi as the general chair, the organizers had to finally revert, once again, to a virtual event.

The importance of ITC in the landscape of cryptography conferences is very evident, as information-theoretic cryptography continues to flourish, in old and new forms. It is rare to find cryptographic problems whose study does not give rise to interesting information-theoretic questions, and this year's program is a clear testament to this. It covers a diverse and exciting range of topics, from foundations all the way to real-world applications.

We have received a total of forty-six submissions, and the nineteen members of our program committee, helped by a number of external reviewers, accepted twenty-six of them. As in the previous edition, we have set a high bar for selection, without targeting a particular number of accepted papers. The unusually high acceptance rate is solely an indication of the high quality of these submissions. And despite our best efforts, I am quite certain we have still managed to reject submissions that were well deserving of acceptance. The conference also includes six spotlight talks, and possibly other events, which are still being arranged at the time of finalizing this volume.

None of this would be possible without all of those who have contributed to making ITC a success. First of all, I would like to thank the authors of all papers (accepted or not) for submitting their works. I am also indebted to all PC members for their tireless reviewing efforts and their insightful discussions, and to all external reviewers for dedicating their time to this effort. Once again, the members of the ITC Steering Committee, led by Benny Applebaum, have been giving extremely valuable advice while organizing a conference in such uncertain times. I also want to particularly thank Daniele Venturi, this year's general chair, for his work on the logistics of the conference, and for his attempts to bring ITC physically to Italy. And of course, I want to thank all invited speakers, presenting authors, and participants for committing their time to making this second edition a success, despite all circumstances.



■ Steering Committee

- Benny Applebaum (Chair, Tel-Aviv University)
- Ivan Damgård (Aarhus University)
- Yevgeniy Dodis (New York University)
- Yuval Ishai (Technion)
- Ueli Maurer (ETH Zurich)
- Kobbi Nissim (Georgetown)
- Krzysztof Pietrzak (IST Austria)
- Manoj Prabhakaran (IIT Bombay)
- Adam Smith (Boston University)
- Yael Tauman Kalai (Microsoft Research New England)
- Stefano Tessaro (University of Washington)
- Vinod Vaikuntanathan (MIT)
- Hoeteck Wee (NTT Research)
- Daniel Wichs (Northeastern University and NTT Research)
- Mary Wootters (Stanford)
- Chaoping Xing (Nanyang Technological University)
- Moti Yung (Google)



■ Organization

General Chair

- Daniele Venturi (Sapienza University of Rome)

Program Chair

- Stefano Tessaro (University of Washington)

Program Committee

- Anat Paskin-Cherniavsky (Ariel University)
- Arpita Patra (Indian Institute of Science)
- Christian Majenz (QuSoft and CWI)
- Divesh Aggarwal (National University of Singapore)
- Eyal Kushilevitz (Technion)
- Fang Song (Portland State University)
- Gilad Asharov (Bar-Ilan University)
- Ignacio Cascudo (IMDEA Software Institute)
- Kai-Min Chung (Academia Sinica)
- Krzysztof Pietrzak (IST Austria)
- Mark Bun (Boston University)
- Marshall Ball (Columbia University and University of Washington)
- Martin Hirt (ETH Zurich)
- Mary Wootters (Stanford University)
- Mohammad Mahmoody (University of Virginia)
- Sidharth Jaggi (Chinese University of Hong Kong and University of Bristol)
- Siyao Guo (NYU Shanghai)
- Uri Stemmer (Ben-Gurion University)
- Vipul Goyal (CMU and NTT Research)

External Reviewers

Navid Alamati, Daniel Apon, Fabio Banfi, Mihir Bellare, Varsha Bhat, Eldon Chung, Alexandru Cojocaru, Wei Dai, Dean Doron, Antonio Faonio, Serge Fehr, Kai Gallert, Konstantin Gegier, Emanuele Giunta, Aarushi Goel, Jesse Goodman, Alex B. Grilo, Koki Hamada, Aditya Hegde, Yao-Ching Hsieh, Mi-Ying Huang, Shih-Han Hung, Joseph Jaeger, Eliran Kachlon, Ilan Komargodski, Ashutosh Kumar, Rajendra Kumar, David Lanzemberger, Zeyong Li, Jyun-Jie Liao, Wei-Kai Lin, Yao-Ting Lin, Chen-Da Liu Zhang, Pasin Manurangsi, Maciej Obremski, Periklis A. Papakonstantinou, Protik Paul, Alice Pellet-Mary, Christopher Portmann, Youming Qiao, Guilherme Rito, Fu Shiuan, Yifan Song, Jana Sotáková, Ziteng Sun, Ajith Suresh, Prashant Vasudevan, Daniel Wichs, Zhiye Xie, Takashi Yamakawa, Maki Yoshida, Yanbao Zhang, Sebastian Zur

2nd Conference on Information-Theoretic Cryptography (ITC 2021).
Editor: Stefano Tessaro



Leibniz International Proceedings in Informatics
LIPICIS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Group Structure in Correlations and Its Applications in Cryptography

Guru-Vamsi Policharla

Indian Institute of Technology, Bombay, Mumbai, India

Manoj Prabhakaran

Indian Institute of Technology, Bombay, Mumbai, India

Rajeev Raghunath

Indian Institute of Technology, Bombay, Mumbai, India

Parjanya Vyas

Indian Institute of Technology, Bombay, Mumbai, India

Abstract

Correlated random variables are a key tool in cryptographic applications like secure multi-party computation. We investigate the power of a class of correlations that we term *group correlations*: A group correlation is a uniform distribution over pairs $(x, y) \in G^2$ such that $x + y \in S$, where G is a (possibly non-abelian) group and S is a subset of G . We also introduce bi-affine correlations, and show how they relate to group correlations. We present several structural results, new protocols and applications of these correlations. The new applications include a completeness result for black box group computation, perfectly secure protocols for evaluating a broad class of black box “mixed-groups” circuits with bi-affine homomorphisms, and new information-theoretic results. Finally, we uncover a striking structure underlying OLE: In particular, we show that OLE over \mathbb{F}_{2^n} , is isomorphic to a group correlation over \mathbb{Z}_4^n .

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases Group correlations, bi-affine correlations, secure computation

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.1

Related Version *Full Version*: <https://eprint.iacr.org/2021/624>

Funding *Manoj Prabhakaran*: Manoj Prabhakaran was supported by the Joint Indo-Israel Project DST/INT/ISR/P-16/2017 and a Ramanujan Fellowship by the Dept. of Science and Technology, India.

Acknowledgements We thank Yuval Ishai and various anonymous reviewers for helpful comments and pointers.

1 Introduction

A central concept in secure multiparty computation (MPC) is that of correlated random variables. If Alice and Bob are given correlated random variables, they can later use them to securely compute any function, with information-theoretic security [22, 20]. This model has been a key ingredient in many theoretical and practical results in MPC. While the class of 2-party correlations that information-theoretically secure computation *can be* based on (i.e., “complete” correlations) is well-understood [23, 24], not all complete correlations *are* used in practical protocols. Instead, several “standard” correlations which have additional structure, like Oblivious Transfer (OT), Oblivious Linear function Evaluation (OLE) and Beaver’s Multiplication Triplets (BMT) [2] are used in practice. The main motivation in this work is to systematically study the additional structure that protocols can exploit, and develop a deeper and broader foundation for such correlations.



© Guru-Vamsi Policharla, Manoj Prabhakaran, Rajeev Raghunath, and Parjanya Vyas;
licensed under Creative Commons License CC-BY 4.0

2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 1; pp. 1:1–1:23



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Apart from uncovering the beautiful mathematical structures from which these correlations derive their power, another motivation for our work is to expand the applicability of correlated random variables to secure computation involving black-box algebraic structures which can be less structured than finite fields or rings. Consider the following seemingly disparate problems of information-theoretically secure 2-party computation:

- **Blackbox Group Computation:** If the function is given as a circuit over a blackbox (non-abelian) group, how can two parties securely compute it with *perfect security*? The complete correlation proposed in [10] (namely, oblivious transfer of group elements), yielded only statistical security.
- **Generating and Processing Correlations over a Blackbox Ring:** If correlated random variables over a blackbox ring (e.g., OLE) are acquired by a pair of parties from a trusted server, can they be efficiently *rerandomized* (e.g., for “forward security” against future corruption of the server)? Efficiency relates to both the use of correlations as well as communication and number of rounds.
How efficiently can such correlations be generated, using a less structured primitive like string OT?
- **Circuits Using Alternate Algebraic Structures:** Traditionally, MPC literature has considered algebraic circuits to be over fields or rings, and these protocols breakdown if the algebraic structure underlying the circuit has less structure. Can alternate protocols be devised for computation over (say) distributive near-rings or non-associative algebras, or when multiple such algebraic structures are used in the same circuit?

We introduce *bi-affine correlations* as an abstraction of a broad class of cryptographically interesting correlations, and address all of the above problems in terms of them. Perhaps more importantly, we undertake a study of the fundamental properties of bi-affine correlations and the underlying mathematical structure of bi-affine homomorphisms, without being confined to immediate applications. This leads us to the definition of *Group Correlations* and *Subgroups Correlations* as a generalization of bi-affine correlations, that brings out additional hidden structure of bi-affine correlations.

Interestingly, while “additive correlations” (the abelian version of group correlations) and “bilinear correlations” (a special case of bi-affine correlations) have been explicitly considered before in various applications, most notably in the rich line of work on function/homomorphic secret-sharing (F/HSS) and pseudorandom correlation generators (PCG) [8, 9, 5, 7, 6],¹ it was not realized that the former is a generalization of the latter, underlining the need for studying them abstractly.

1.1 Our Contributions

We develop a theory of *group correlations* and *subgroups correlations*, with a focus on the subclass of bi-affine correlations. A group correlation, specified by a group G and a subset $S \subseteq G$, is simply an additive secret-sharing of a random element in S , or equivalently, a uniform distribution over $\{(x, y) \mid x, y \in G, x + y \in S\}$. A subgroups correlation is a restriction of such a group correlation correlation to the universe $G_1 \times G_2$ where G_1 and G_2 are subgroups of G , with a regularity condition on S (so that the resulting correlation has

¹ In these works, bi-linear correlations were often termed *simple bi-linear correlations*. For consistency with the terminology in the current work, we avoid this term. What was termed (general) bi-linear correlations there would correspond to correlations of the form $\text{BA}_{\sigma(2)}$ in this work.

uniform marginal distributions). Within this simple framework, a rich variety of structures arise based on how the groups and the set S are defined. Our contributions include the following:

- **A Theory of Group Correlations:** This includes several new definitions of structures and properties, as well as connections between them. (Section 3).
- **Information-Theoretic Results:** We give new results on information theoretic quantities (specifically, *residual information*) that can be used to analyze the optimality of secure protocols. (Section 4).
- **New Protocol Building-Blocks:** We present a suite of protocols for various functionalities involving bi-affine correlations, with applications to 2-Party secure computation. (Section 5).
- **Applications:** The above building-blocks can be put together to yield various information-theoretically secure computation protocols. In particular, we show:
 - There exists a *complete correlation* for 2-party *perfectly* passive-secure evaluation of a black-box (non-abelian) group circuit – called the Zero Alternating Sum (ZAS) correlation. ZAS is a bi-affine correlation, and hence this could be seen as a special case of the following results. In contrast, previously the complete correlation proposed in [10] (namely, OT with group elements), yielded only statistical security. When the circuit has logarithmic depth, or is in the form of polynomial-sized formula, we obtain a 2-round UC secure protocol.
 - A GMW-style 2-Party protocol for evaluating a black-box “mixed-group circuit” with homomorphism and bi-affine homomorphism gates, which requires 2 rounds of interaction per layer.
 - 2-Party protocols for rerandomizing and testing bi-affine correlations obtained from a semi-trusted source (who will not collude with either party until after the protocol is over) (Section 5.1, Section 5.4). We also discuss how this can be viewed as a solution to sampling correlations in the single-server version of the commodity based model [4].
 - A 2-Party protocol for securely sampling bi-affine correlations using string OTs, generalizing a protocol of Gilboa [19]. Using our information-theoretic results, we establish its optimality for a class of bi-affine correlations (including the ones considered in [19]). (Section 5.3).
- **A Surprising Structure.** Finally, we uncover a striking structure underlying OLE. In particular, we show that OLE over \mathbb{F}_{2^n} , is isomorphic to a group correlation over \mathbb{Z}_4^n . Given that OLE has been widely studied and used, it is remarkable that such a structure has remained hidden so far.

Details of the protocols and applications can be found in the full version.

Discussion

Here we elaborate on some of the above contributions.

Hidden Structures. We point out two instances of hidden structure in well-studied objects that are revealed by our abstractions. OLE and BMT are two correlations that have been extensively studied both in terms of their applications, and in terms of protocols generating them. However, while abstracting them as bi-linear correlations (see Footnote 1), they are treated somewhat differently. For instance, in [5], PCGs for bi-linear correlations are given, which directly applies to OLE; and then a PCG for BMT is provided by reducing BMT to OLE. However, a consequence of our results is that BMT is already a (simple) bi-linear correlation, but with a bi-linear operator different from that of OLE: while OLE uses a

map $\sigma(a, b) = ab$, BMT uses $\sigma((a, b), (c, d)) = ad + bc$ (all variables belonging to a ring). This results in a more efficient protocol since reducing one BMT to two OLE correlations is wasteful (a reduction in the opposite direction is not possible).

The second instance of a hidden structure is that of OLE which has a complicated structure due to the interaction of field multiplication with the addition structure of the field. As such, one may not expect OLE (over large fields) to be a group correlation. But we show that every symmetric bi-affine correlation (of which OLE is an example) is in fact a group correlation. Even more surprisingly, for the special case of OLE over the field \mathbb{F}_{2^n} , the underlying group turns out to be \mathbb{Z}_4^n . Thus OLE over \mathbb{F}_{2^n} can be seen as sampling an element uniformly from a (non-obvious) set $S \subseteq \mathbb{Z}_4^n$, and then simply additively secret-sharing it coordinate-wise. While we do not offer any immediate applications of this particular structure, as a fundamental property of an extremely useful cryptographic primitive, it is an interesting result.

ZAS: A Bi-Affine Correlation in a Group. An interesting application we present is that of a complete correlation for 2-party secure computation over a black-box group, with *perfect security*. In contrast to the prior approach which relied on OT with group elements, and only obtained statistically secure protocols [10], we rely on a deceptively simple correlation, called the Zero Alternating Sum (ZAS) correlation. In a ZAS correlation over a (non-abelian) group G , Alice and Bob get random pairs $(a, c) \in G^2$ and $(b, d) \in G^2$ such that $a + b + c + d = 0$.

Note that defining ZAS does not require anything more than the group operation. This demonstrates the generality of bi-affine homomorphisms, compared to bi-linear maps. While bi-linear maps are used to capture the multiplication operation in a *ring*, bi-affine homomorphisms can equally well capture the alternating sum structure in a group. Concretely, the function $\sigma : G^2 \rightarrow G^{\text{op}}$, defined as $\sigma(x, y) = -(x + y)$ where G^{op} is the *opposite group* of G (whose group operation is the same as that of G , but applied to the operands in the opposite order), is a bi-affine homomorphism w.r.t. the subgroups $T = G \times \{0\}$ and $U = \{0\} \times G$ of the group G^2 .

Optimality of Gilboa’s Reduction. As a corollary of our information-theoretic results pertinent to bi-affine correlations, we show that Gilboa’s reduction from OLE over \mathbb{F}_{2^n} to string OT [19] is optimal in the number of string OTs used (n string OTs per OLE instance), and cannot be improved upon even with amortization. In fact, this extends to OLE over \mathbb{F}_{p^n} if Gilboa’s protocol is modified to use 1-out-of- p string OTs.

Mixed-Groups Circuit with Bi-Affine Homomorphism Gates. Conventionally, MPC literature has considered boolean or arithmetic circuits over a given ring or field. A variant of this considers the underlying algebraic structure to be given as a black box to the protocol (e.g., [11, 21] for rings and [17, 16, 10] for groups). Motivated by practical applications, MPC protocols for computation that uses multiple representations has received attention (e.g., the ABY framework [15] and subsequent works). More recently, circuits with bi-linear gates over multiple black box groups has been considered in [9].

Our applications use a similar circuit paradigm as [9], and use two types of gates (1) group operations (2) gates for group homomorphisms and bi-affine homomorphisms. Bi-Affine Homomorphisms are quite general, and can correspond to multiplication in distributive near-rings or non-associative rings, or even (negated) addition in a non-abelian group. As such, this is a powerful computational model that subsumes arithmetic circuits over a

ring. Nevertheless, the bi-affine homomorphism structure lets us build perfectly secure 2-party protocols for all such circuits, using bi-affine correlations for the corresponding bi-affine homomorphisms (if necessary, along with “Zero Alternating Sum” correlations for the non-abelian groups).

1.2 Related Work

Correlations have received much attention in cryptography, especially since Beaver’s proposal of using them as cryptographic commodities [3] and the emergence of the pre-processing model as a common approach to theoretically and practically efficient MPC. They have been put to great use for MPC, both in the passive and active corruption settings, in theory and practice (see. e.g., the SPDZ family of protocols [14] and subsequent work). All these works develop and use several building blocks like self-reduction and self-testing for their correlations.

The recent line of works on Pseudorandom Correlation Generators and Function Secret Sharing [9, 5, 7, 8, 6], which consider bi-linear and additive correlations are most closely related to our work. Briefly, they answer two important questions. (1) how to perform secure computation over bi-linear gates (2) how to efficiently generate these correlations. In contrast to our work, these results were focussed on exploiting computational hardness, and restricted themselves to bi-linear correlations and abelian groups.

Secure Multi Party Computation over non-abelian Black-Box groups has been well studied in the honest-majority setting [17, 16, 10]. In the two-party setting Cohen et al. [10] gave a passive statistically secure protocol for evaluating circuits over black-box groups in the OT hybrid model and used the IPS compiler [20] to achieve security against active corruption. In this work, we use a stronger primitive – namely Zero Alternating Sum correlations – but are able to obtain a simple perfectly secure protocol against active adversaries without the use of expensive compilers for log-depth circuits.

Protocols for rerandomization and testing of correlations have appeared previously in the literature but their focus has remained on specific correlations such as BMT, squaring tuples etc., [13]. The commodity based model first introduced by Beaver in [4] has been revisited recently in [12, 27] to sample OLE and BMT correlations.

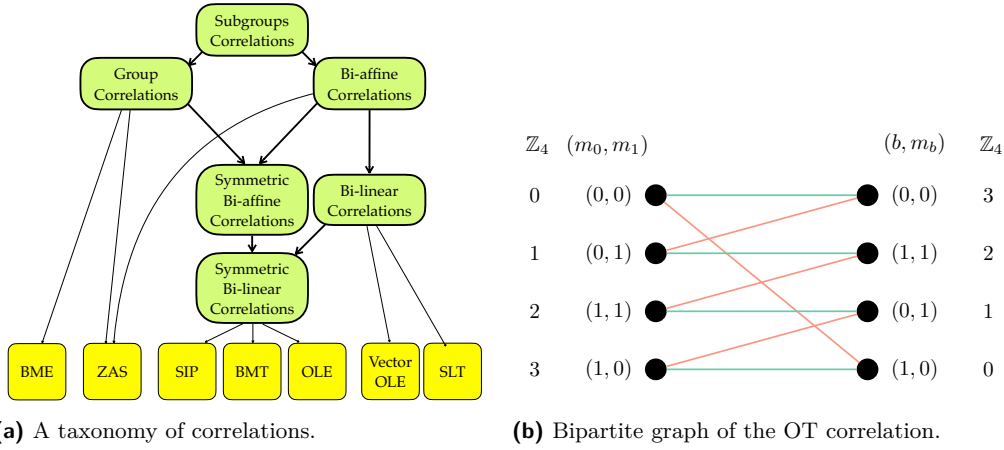
1.3 Technical Overview

In this section we present the highlights of our results, informally. Several additional technical details and generalizations are deferred to the subsequent sections and the full version.

1.3.1 Definitions

We consider several classes of *flat* correlations – i.e., distributions that are uniform over their support. Below we use support and distribution interchangeably.

Group Correlations and Subgroups Correlations. A group correlation defined w.r.t a group G and a subset $S \subseteq G$ is the uniform distribution over all pairs $(g_1, g_2) \in G^2$ such that $g_1 + g_2 \in S$. A subgroups correlation *embedded in* this group correlation is obtained by requiring $g_1 \in G_1$ and $g_2 \in G_2$, where G_1, G_2 are subgroups of G with the property that the marginal distributions of g_1 and g_2 are both uniform. This subgroups correlation is said to be *compact* if $|G| < |G_1||G_2|$.



(a) A taxonomy of correlations.

(b) Bipartite graph of the OT correlation.

Bi-Affine Homomorphisms. A linear function (or a group homomorphism) $\phi : G \rightarrow H$ satisfies $\phi(a + b) = \phi(a) + \phi(b)$ (where the addition and subtraction are in the appropriate groups). An “affine” function ψ is such that ϕ defined by $\phi(x) := \psi(x) - \psi(0)$ is linear; i.e., $\psi(a + b) = \psi(a) - \psi(0) + \psi(b)$. A bi-affine function could be defined as a function of two inputs, which is affine in each of them; i.e., for groups T, U, H , a function $e : T \times U \rightarrow H$ such that

$$e(t, u + u') = e(t, u) - e(t, 0) + e(t, u') \quad \text{and} \quad e(t + t', u) = e(t, u) - e(0, u) + e(t', u). \quad (1)$$

Note that if we required $e(t, 0) = e(0, u) = 0$, then the conditions above would collapse to e being *bi-linear*. Examples of functions that satisfy (1) but are not bi-linear include $e : G \times G \rightarrow G$ defined as $e(a, b) = a + b$ or as $e(a, b) = -a - b$.

For notational simplicity in our results, we define a *bi-affine homomorphism* as a *unary* function $\sigma : Q \rightarrow H$, (Q, H being groups) with respect to subgroups $T, U \leq Q$ so that $e : T \times U \rightarrow H$ defined as $e(t, u) := \sigma(t + u)$ satisfies (1). An equivalent definition, in terms of group homomorphisms, is given in Definition 7.

Bi-Affine Correlation. Given a bi-affine homomorphism σ as above, the support of the corresponding bi-affine correlation correlation $\text{BA}_\sigma \subseteq (T \times H) \times (U \times H)$ is defined as

$$\text{BA}_\sigma = \{((t, a), (u, b)) \mid \sigma(t + u) = a + b\}.$$

Examples. As shown in Figure 1a, the most commonly used correlations indeed fall under the class of bi-affine correlations.

- Oblivious Linear Evaluation (OLE): Defined over a ring A as $((t, a), (u, b))$ such that $a + b = tu$, OLE is isomorphic to a bi-affine correlation with $\sigma(t, u) = tu$, where $\sigma : A^2 \rightarrow A$ is a bi-affine homomorphism with respect to $T = A \times \{0\}$ and $U = \{0\} \times A$.
- Beaver’s Multiplication Triples (BMT): Defined over a ring A as $((t_1, u_1, a), (t_2, u_2, b))$ such that $a + b = (t_1 + t_2)(u_1 + u_2)$, BMT is isomorphic to a bi-affine correlation with $\sigma((t_1, u_1), (t_2, u_2)) = t_1 u_2 + t_2 u_1$, where $\sigma : A^4 \rightarrow A$ is a bi-affine homomorphism with respect to $T = A^2 \times \{0\}^2$ and $U = \{0\}^2 \times A^2$.
- Zero Alternating Sum (ZAS): Defined over a (possibly non-abelian) group D as $((a, c), (b, d))$ such that $a + b + c + d = 0$, ZAS is isomorphic to a bi-affine correlation BA_σ , where $\sigma : D^2 \rightarrow D^{\text{op}}$ defined as $\sigma(c, d) = -(c + d)$ is a bi-affine homomorphism with respect to $T = D \times \{0\}$ and $U = \{0\} \times D$.

Powers of a Bi-Affine Homomorphism. Given a bi-affine homomorphism $\sigma : Q \rightarrow H$ w.r.t. subgroups T, U , we can define new bi-affine homomorphisms as “powers” of σ . There are a few different useful notions of such powers that emerge in the sequel, which we call σ^n , $\sigma^{(n)}$ and $\sigma^{(n)}$.

- $\sigma^n : Q^n \rightarrow H^n$ is simply the coordinate-wise application of σ .
- $\sigma^{(n)} : Q^n \rightarrow H^n$ corresponds to a “vector” variant of σ , generalizing how string-OT or vector-OLE are vector variants of OT and OLE respectively; it is in fact the same as σ^n , but considered as a bi-affine homomorphism w.r.t. T^n and $U^{(n)} = \{(u, \dots, u) | u \in U\} \subseteq U^n$. $\text{BA}_{\sigma^{(n)}}$.
- $\sigma^{(n)} : Q^n \rightarrow H$ is an *inner-product* version of σ , generalizing how BMT is isomorphic to $\text{BA}_{\sigma^{(2)}}$, where σ is the multiplication in a ring (so that BA_{σ} corresponds to OLE over that ring).

There exists a non-interactive, UC-secure protocol to securely sample one instance of $\text{BA}_{\sigma^{(\ell, m)}}$ from $\ell + m$ instances of BA_{σ} . A special case of this protocol is the reduction of a BMT correlation to two OLE correlations. See full version for details.

1.3.2 Connections

We uncover some surprising connections between the different classes of correlations mentioned above (Theorem 9).

1. Every symmetric bi-affine correlation is a group correlation. In particular, OLE over a ring A is isomorphic to a group correlation w.r.t the group \mathbb{K}_A over $A \times A$ whose group operation is defined as $(a, b) \odot (c, d) = (a + c, b + d - ac)$, and subset $S = \{(a, 0) | a \in A\}$.
2. Every bi-affine correlation is a *compact* subgroups correlation. Note that an asymmetric bi-affine correlation, like a vector OLE, cannot be a group correlation. But this result shows that it is a subgroups correlation compactly embedded in a group correlation. In particular, n -dimensional vector OLE over a ring A is embedded in the group correlation over the group $A^n \times A \times A^n$ with subset $S = \{(t, u, tu) | t \in A^n, u \in A\}$. Interestingly, when instantiated for OLE ($n = 1$), it shows that OLE is embedded in the BMT correlation.
3. If σ is a semi-abelian bi-affine homomorphism, then BA_{σ} is embedded in $\text{BA}_{\sigma^{(2)}}$. This serves as an alternate way of viewing the embedding of OLE in BMT, since OLE is BA_{σ} and BMT is $\text{BA}_{\sigma^{(2)}}$ where σ is the 1multiplication operation in the (possibly non-commutative) ring.

As mentioned, OLE over a ring is a group correlation, over the group \mathbb{K} . We explore this group and discover more unexpected structure:

- When A has an element η such that $\eta + \eta = 1$, \mathbb{K}_{σ} is isomorphic to the group $A \times A$ (considering only the addition operation in the ring).
- When A is \mathbb{F}_{2^n} then \mathbb{K}_{σ} is isomorphic to \mathbb{Z}_4^n . (See Section 1.3.5).

1.3.3 Information-Theoretic Results

Wyner residual information (RI_W) (5) is an information theoretic measure which describes how “correlated” two random variables are. This measure is a monotone and cannot be increased through communication. Concretely, Prabhakaran et. al. [25] showed that if m independent instances of one type of correlation (C) can be reduced to n independent instances of another type of correlation (C'), then $m \cdot RI_W(C) \leq n \cdot RI_W(C')$ (Proposition 10).

In this work, we compute the Wyner Residual Information for a subset of bi-affine correlations which satisfy the *non-defective* property (Definition 7). A consequence of our results is that, for any field F , $RI_W(\text{OLE}_F^n) = \log |F|$. In fact, the above result extends to

domains rather than fields. (A domain is a ring with the “zero-product property,” i.e., if $ab = 0$ then $a = 0$ or $b = 0$.) These results play a crucial role in later sections where we prove optimality of reductions from bi-affine correlations to oblivious transfer. Furthermore, we show that the bi-partite graph of a group correlation is a single connected component iff the set $\{s - s' \mid s, s' \in S\}$ is a generating set for the group G by appealing to the Gács-Körner common information (Lemma 11).

1.3.4 Constructions

We present several constructions (Section 5), which relate to various conditional sampling functionalities that *complete* a bi-affine correlation. Let \mathcal{F}_σ be an ideal sampling functionality that samples an instance of the correlation and gives each party its side of the correlation. Similarly, let $\tilde{\mathcal{F}}_\sigma$ be a biasable sampling functionality (where the adversary is allowed to pick its side of a valid correlation). Now, we define three completion functionalities – depending on how many variables are fixed – for bi-affine correlations.

<p>Conditional Sampling Functionalities $\mathcal{F}_{\sigma U}$, $\mathcal{F}_{\sigma TU}$ and $\mathcal{F}_{\sigma TAU}$ (where $\sigma : Q \rightarrow H$ is a bi-affine homomorphism w.r.t. $T, U \leq Q$)</p> <p>Inputs: t, a from Alice, and $u \in U$ from Bob, where</p> <p style="text-align: center;">$t = a = \perp$ for $\mathcal{F}_{\sigma U}$ $t \in T, a = \perp$ for $\mathcal{F}_{\sigma TU}$ $t \in T, a \in H$ for $\mathcal{F}_{\sigma TAU}$.</p> <p>Outputs: (\tilde{t}, \tilde{a}) to Alice and (\tilde{u}, \tilde{b}) to Bob, where $((\tilde{t}, \tilde{a}), (\tilde{u}, \tilde{b})) \leftarrow \text{BA}_\sigma$ conditioned on $\tilde{u} = u$, $\tilde{t} = t$ if $t \neq \perp$, and $\tilde{a} = a$ if $a \neq \perp$.</p>

We then present various protocols that implement the above functionalities (Section 5):

- UC secure protocols for $\mathcal{F}_{\sigma|U}$, $\mathcal{F}_{\sigma|TU}$ and $\mathcal{F}_{\sigma|TAU}$ in the \mathcal{F}_σ -hybrid model (Figure 2). The protocols remain secure even if \mathcal{F}_σ is replaced by an “adversarially controlled” version $\tilde{\mathcal{F}}_\sigma$ (which still only provides instances in the support of the correlation BA_σ).
 - These protocols, denoted as $\text{Comp}_{\sigma|U}$, $\text{Comp}_{\sigma|TU}$ and $\text{Comp}_{\sigma|TAU}$, can be used for multiple purposes. In particular, it allows for *rerandomizing* a sample, and also as a tool for non-destructively checking the validity of a sample (in the protocols TRSamp_σ and altTRSamp_σ below). Our protocols are optimal in multiple ways: there is only one message (or in the case of $\text{Comp}_{\sigma|TAU}$, two messages) and a single instance of the correlation is “consumed” per instance produced. For the basic forms of these tasks (without the extension to $\tilde{\mathcal{F}}_\sigma$), similar constructions have been previously developed, but only for specific correlations like OLE, BMT etc., [13].
- We also develop a new set of protocols for realizing the above functionalities using a “tamperable” version $\hat{\mathcal{F}}_\sigma$ (which, when the two parties are honest, allows the adversary to specify arbitrary pairs, possibly outside the support of BA_σ), instead of $\tilde{\mathcal{F}}_\sigma$. We present two such protocols, trading-off generality with efficiency.
 - The first protocol, TRSamp_σ (Figure 5) works for all bi-affine homomorphisms σ , but consumes $\omega(\log \lambda)$ (purported) samples of BA_σ to produce a single (good) instance. This protocol relies on an *error-preservation property* of the protocol $\text{Comp}_{\sigma|TAU}$, whereby it can be checked if two purported samples have identical “error,” by consuming only one of them. This allows checking that a set of samples all have the same error, while leaving one of them unconsumed. This still admits the possibility that *all of the samples* have the same non-zero error. To detect this (except with negligible probability), a cut-and-choose step is employed.

- The second protocol, altTRSamp_σ (Figure 6) achieves a rate of $1/2$, but relies on additional algebraic structure in the groups underlying σ . The main component of this protocol is an *error rerandomization* step (Figure 7), which we instantiate for a variety of bi-affine homomorphisms $\sigma : Q \rightarrow H$, where:
 - * σ corresponds to multiplication in a vector space over a large field (or more generally, a module of appropriate complexity),
 - * H is abelian and its order has no small prime factors,
 - * H is non-abelian and $|\{r + x - r \mid r \in H\}|$ is large for all $x \neq 0$.
- We give a semi-honest secure protocol (Figure 4) for efficiently sampling a bi-affine correlation BA_σ from string-OTs. This protocol relies on additional structure in the groups underlying the σ , and requires (slight) non-blackbox access to them. The additional structure is used to represent every group element as a small sum of elements from a “basis.” The protocol is a generalization of a protocol by Gilboa [19] for sampling OLE over a ring using string OTs, to bi-affine correlations over a wide range of groups. We also show, using our results on residual information from above, that when the basis allows a tight representation of the group elements, then, with some additional constraints on σ , the protocol is *optimal in the number of string-OTs used* (Lemma 15).

1.3.5 A Surprising Structure for OLE

It is easy to see that OT (i.e., OLE over \mathbb{F}_2) can be written as a group correlation over \mathbb{Z}_4 , by “drawing” the correlation as a bipartite graph and observing that it forms a cycle (see Figure 1b). A surprising result we obtain is that OLE over \mathbb{F}_{2^n} is in fact a group correlation over \mathbb{Z}_4^n . We illustrate this for $n = 2$ in the full version.

We give a detailed description and proof in the full version, but provide a high level overview here. To show that $\text{OLE}_{\mathbb{F}_{2^n}}$ is a group correlation we give an isomorphism ϕ from $\mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$ to \mathbb{Z}_4^n along with a subset $S \subset \mathbb{Z}_4^n$ and show that field elements $(t, a), (u, b)$ form an OLE correlation ($a + b = tu$) iff the sum of elements $g_1 = \phi(t, a), g_2 = \phi(u, b)$ lies within S . The isomorphism itself is highly non-trivial as it needs to handle the interaction of multiplicative and additive operations of the field in a purely additive sense. The isomorphism and subset are given by

$$\phi(x, y) = [x] - 2 \cdot \left[\sqrt{\sum_{i:x_i=1} \xi^{(i)}(x)_i} \right] + 2 \cdot [\sqrt{y}]$$

$$S = \left\{ [x] - 2 \cdot \left[\sqrt{\sum_{i:x_i=1} \xi^{(i)}(x)_i} \right] \mid x \in \mathbb{F}_{2^n} \right\}$$

where $[x]$ denotes the embedding from \mathbb{F}_{2^n} to \mathbb{Z}_4^n , obtained by interpreting $x \in \{0, 1\}^n$ as $x \in \{0, 1, 2, 3\}^n$, $\{\xi^{(i)}\}_{i \in [0, n-1]}$ is an arbitrary basis of \mathbb{F}_{2^n} with $\xi^{(0)} = 1$, and $(x)_i$ is the field element obtained by zeroing out all coordinates greater than or equal to i .

1.3.6 Applications

Using our constructions from Section 5 we show how to perform secure 2-Party computation of “mixed-groups” circuits in the semi-honest setting. The mixed-groups circuit model uses wires which carry group elements and group/bi-affine homomorphism gates in addition to gates implementing standard group operations.

- The first setting is semi-honest 2-Party computation in the $\mathcal{F}_\sigma, \mathcal{F}_{ZAS}$ hybrid model, where σ is the bi-affine homomorphism corresponding to the bi-affine homomorphism gate being evaluated. Throughout the evaluation we maintain the invariant that all wires are secret shared between the two parties. At each bi-affine gate, two bi-affine correlations and one ZAS correlation (in the group of the output wire) is consumed and at most two rounds of communication are needed to evaluate each level of the circuit. We achieve perfect security in this setting.
 - As a corollary, we show that the ZAS correlation is complete for passively secure 2-Party secure computation over black-box groups. This is immediate as all group operations can be implemented using ZAS correlations only.
 - For the special case of formulas (or log-depth circuits) we present a two round perfectly secure protocol where the communication is proportional to the number of terms in the formula. Note that a formula can be written as an alternating sum of Alice and Bob's private inputs $f(x_1, \dots, x_n, y_1, \dots, y_n) = \sum_{i=1}^n (x_i + y_i)$. Alice pads each term of the formula with randomness and sends terms which contain her input in the clear. Alice and Bob invoke \mathcal{F}_{ZAS} to compute terms containing Bob's inputs. Bob then sums up the terms sent by Alice and his output from \mathcal{F}_{ZAS} invocations to compute $f(x_1, \dots, x_n, y_1, \dots, y_n) = \sum_{i=1}^n (x_i + y_i)$.
 - We also show how the same task can be achieved in a different manner using the Function Secret Sharing based approach of Boyle et al. [9].
- The second setting we consider is the commodity based model introduced by Beaver [4]. Here a semi-trusted server which provides Alice and Bob with (possibly incorrect) correlations and is guaranteed to not collude with either party. Incorrect correlations are identified by using either TR Samp_σ or altTR Samp_σ , after which the computation can be done in a manner identical to the previous setting.

Full descriptions of these protocols can be found in the full version.

2 Preliminaries

All the sets (and in particular, groups, rings and fields) we consider in this work are finite. For groups, we typically use additive notation. When several groups are used together, we often assign different symbols like \odot , \oplus and $+$ for their operators. The unary negation symbol $(-x)$ is used across all groups to indicate the inverse; also, the binary subtraction symbol $(x - y)$ is used to denote $x + (-y)$, when the group operation is $+$. We use upright capital letters to denote random variables, as X, Y etc. Through out the paper, 2-party secure computation, unless otherwise qualified, refers to information-theoretic security against passive corruption.

We recall that given a subgroup T of a group $(G, +)$, its right and left cosets containing an element $g \in G$ are defined as $T + g = \{t + g \mid t \in T\}$, $g + T = \{g + t \mid t \in T\}$. We define “shifted groups” over these cosets, by redefining the group operation.

► **Definition 1 (Shifted Group Operation).** *Given a group $(G, +)$, and $g \in G$, the operation $+_g$ is defined as $x +_g y = x - g + y$.*

It can be seen that $+_g$ is associative, as $(x +_g y) +_g z = x +_g (y +_g z) = x - g + y - g + z$. For any subgroup $T \subseteq G$, it can be verified that $(T + g, +_g)$ and $(g + T, +_g)$ are both groups with identity element g and the inverse of x given by $g - x + g$. They are both subgroups of $(G, +_g)$.

► **Definition 2** (Flat Correlation). A flat correlation over sets X, Y is defined to be the uniform distribution over a set $C \subseteq X \times Y$. It is said to be regular if there are integers d_X, d_Y such that $\forall x \in X, |C \cap (\{x\} \times Y)| = d_X$ and $\forall y \in Y, |C \cap (X \times \{y\})| = d_Y$.

Above, C is called the *support* of the correlation, and is also used to denote the correlation itself. Given a flat correlation over X, Y with support C , its graph \mathbb{G}_C is defined as the bipartite graph with vertices $X \dot{\cup} Y$ (disjoint union) and the set of edges C .

► **Definition 3** (Isomorphic Correlations). Flat correlations $C \subseteq X \times Y$ and $C' \subseteq X' \times Y'$ are said to be isomorphic to each other if there exist bijections $\alpha : X \rightarrow X'$ and $\beta : Y \rightarrow Y'$ such that $C' = \{(\alpha(x), \beta(y)) \mid (x, y) \in C\}$.

► **Definition 4** (Sampling Functionalities $\mathcal{F}_C, \tilde{\mathcal{F}}_C, \hat{\mathcal{F}}_C$). For a flat correlation C , we define three functionalities as follows.

- **Sampling Functionality** \mathcal{F}_C : Uniformly samples a pair $(x, y) \leftarrow C$, and gives x to Alice and y to Bob.
- **Biasable Sampling Functionality** $\tilde{\mathcal{F}}_C$: If Alice is corrupt, then it takes $x \in X$ from Alice, and outputs $y \leftarrow \{y' \mid (x, y') \in C\}$ to Bob; similarly, if Bob is corrupt, it takes y from Bob and outputs $x \leftarrow \{x' \mid (x', y) \in C\}$ to Alice. But if both parties are honest then it lets the adversary specify a valid sample, i.e., $(x, y) \in C$, instead of sampling one itself.
- **Tamperable Sampling Functionality** $\hat{\mathcal{F}}_C$: It behaves like $\tilde{\mathcal{F}}_C$, but if both Alice and Bob are honest, then it lets the adversary specify an arbitrary pair (x, y) (rather than only a valid pair).

3 Definitions and Connections

3.1 Group Correlations and Subgroups Correlations

► **Definition 5** (Group Correlation). A flat correlation $C \subseteq X \times Y$ is said to be a group correlation if there exists a group G and a subset $S \subseteq G$ such that C is isomorphic to the flat correlation $C' \subseteq G \times G$ given by $C' = \{(x, y) \mid x + y \in S\}$. In this case, we say that C is a group correlation of the form $GC^{G,S}$. A group correlation of the form $GC^{G,S}$ is said to be abelian if the group G is abelian.

Regularity. Let G_1, G_2 be subgroups of G , and $S \subseteq G$. S is said to be regular with respect to (G_1, G_2) if, for all $g_2, g'_2 \in G_2$, we have $|S \cap (G_1 + g_2)| = |S \cap (G_1 + g'_2)|$, and for all $g_1, g'_1 \in G_1$, we have $|S \cap (g_1 + G_2)| = |S \cap (g'_1 + G_2)|$. We call $\deg_L = |S \cap (g_1 + G_2)|$ and $\deg_R = |S \cap (G_1 + g_2)|$ the left and right degree of the subgroups correlation respectively.

We say that a group correlation $GC^{G,S}$ is regular w.r.t. a pair of subgroups (G_1, G_2) of G if S is regular w.r.t. (G_1, G_2) .

► **Definition 6** (Subgroups Correlation). A flat correlation $C \subseteq X \times Y$ is said to be a subgroups correlation if there exists a group correlation C' that is regular w.r.t. a pair of subgroups (G_1, G_2) , and C is isomorphic to the correlation $C'' \subseteq G_1 \times G_2$ defined as $C'' = C' \cap (G_1 \times G_2)$. In this case, we say C is of the form $GC_{G_1, G_2}^{G,S}$, and is embedded in C' . Further, if $|G| < |X||Y|$, we say that C is a compact subgroups correlation.

If C is a regular flat correlation, then it can be seen to be a (non-compact) subgroups correlation of the form $GC_{G_1, G_2}^{G,S}$ where, identifying X and Y with arbitrary groups of the same sizes (say $\mathbb{Z}_{|X|}$ and $\mathbb{Z}_{|Y|}$), we let $G = X \times Y$, $G_1 = X \times \{0_Y\}$, $G_2 = \{0_X\} \times Y$, and $S = C$. Conversely, a subgroups correlation is a regular flat correlation. Hence, without restricting to being compact, subgroups correlations and regular flat correlations are the same. A compact subgroups correlation entails more structure than just being regular.

3.2 Bi-Affine Correlations

We start by defining a generalization of the notion of a homomorphism, called *bi-affine homomorphism*. Note that the definition below refers to homomorphisms between “shifted” groups, using the shifted group operation (Definition 1).

► **Definition 7** (Bi-Affine Homomorphism). *For groups $(Q, +)$ and (H, \oplus) , and subgroups $T, U \leq Q$, a function $\sigma : Q \rightarrow H$ is said to be a bi-affine homomorphism w.r.t. (T, U) , if the following are group homomorphisms*

$$\begin{aligned} \sigma|_{T+u} : (T + u, +_u) &\rightarrow (H, \oplus_{\sigma(u)}) && \forall u \in U \\ \sigma|_{t+U} : (t + U, +_t) &\rightarrow (H, \oplus_{\sigma(t)}) && \forall t \in T. \end{aligned}$$

Further, σ is said to be semi-abelian if H is an abelian group; it is said to be abelian if both Q and H are abelian. It is said to be symmetric if it is semi-abelian and $Q = D \times D, T = D \times \{0\}, U = \{0\} \times D$ for some group D . If either $\sigma|_{T+u}$ is surjective for every $u \in U$, or $\sigma|_{t+U}$ is surjective for every $t \in T$, σ is called a surjective bi-affine homomorphism. If there is no pair $(t, u) \in (T \setminus \{0\}) \times (U \setminus \{0\})$ such that $\sigma(t + u) = \sigma(t) - \sigma(0) + \sigma(u)$, σ is said to be non-defective².

These homomorphism conditions over the shifted groups can be equivalently written as, $\forall t, t' \in T, u, u' \in U$,

$$\begin{aligned} \sigma(t + t' + u) &= \sigma(t + u) \oplus -\sigma(u) \oplus \sigma(t' + u) \\ \sigma(t + u + u') &= \sigma(t + u) \oplus -\sigma(t) \oplus \sigma(t + u'). \end{aligned}$$

(where we used $(t + u) +_u(t' + u) = t + t' + u$ and $(t + u) +_t(t + u') = t + u + u'$).

► **Definition 8** (Bi-Affine Correlation). *Given groups $(Q, +)$ and (H, \oplus) , and a bi-affine homomorphism $\sigma : Q \rightarrow H$ w.r.t. (T, U) , the correlation $\text{BA}_\sigma \subseteq (T \times H) \times (U \times H)$ is defined as*

$$\text{BA}_\sigma = \{((t, a), (u, b)) \mid \sigma(t + u) = a \oplus b\}$$

A flat correlation C is said to be a bi-affine correlation if there exists σ as above such that it is isomorphic to BA_σ . Further, C is said to be semi-abelian, abelian or symmetric if σ has the corresponding property.

Bi-linear correlations. It is instructive to compare bi-affine homomorphisms with bi-linear maps. For groups $(T, +)$, $(U, +)$ and (H, \oplus) , where the last one is abelian, a function $e : T \times U \rightarrow H$ is said to be a bi-linear map if e left and right distributes over the group operations: i.e., for all $t_1, t_2 \in T$ and $u_1, u_2 \in U$, $e(t_1 + t_2, u_1) = e(t_1, u_1) \oplus e(t_2, u_1)$, and $e(t_1, u_1 + u_2) = e(t_1, u_1) \oplus e(t_1, u_2)$.

It is easy to see that a bi-linear map is a special case of a bi-affine homomorphism: Let $Q = T \times U$, $T' = T \times \{0_U\}$ and $U' = \{0_T\} \times U$. Then, $\sigma : Q \rightarrow H$ is a bi-linear map iff it is a semi-abelian bi-affine homomorphism w.r.t. (T', U') , with the additional property that $\sigma(x) = 0$ for all $x \in T' \cup U'$. If a bi-affine homomorphism σ is a bi-linear map, then we say that a correlation of the form BA_σ is a *bi-linear correlation*. For bi-linear σ , non-defective

² This condition corresponds to $K_{2,2}$ freeness of the bi-affine correlation. Proof can be found in the full version.

reduces to not having non-zero $t \in T, u \in U$ such that $\sigma(t + u) = 0$. An example of such a bi-affine correlation is given by OLE (or vector OLE) over a *domain*. A domain is a ring with the “zero-product property,” i.e., if $ab = 0$ then $a = 0$ or $b = 0$ (with fields being a special case of domains).

3.3 Powers of Bi-Affine Homomorphisms

Given a bi-affine homomorphism σ , one can define related bi-affine homomorphisms as various “powers”. In this section, we describe some standard transformations to do this, and in Section 3.5 give some important examples of correlations in the literature that illustrate these transformations. Let $\sigma : Q \rightarrow H$ be a bi-affine homomorphism w.r.t subgroups T, U .

- We define $\sigma^n : Q^n \rightarrow H^n$ as simply the coordinate-wise application of σ . That is, $\sigma^n(q_1, \dots, q_n) = (\sigma(q_1), \dots, \sigma(q_n))$. If σ is a bi-affine homomorphism w.r.t. subgroups $T, U \leq Q$, then σ^n is readily seen to be a bi-affine homomorphism w.r.t. subgroups $T^n, U^n \leq Q^n$.
- It is interesting to view σ^n as a bi-affine homomorphism w.r.t. other subgroups within T^n, U^n . In particular, we define $\sigma^{(n)}$ to be the same as σ^n but considered as a bi-affine homomorphism w.r.t. $T^n, U^{(n)}$, where $U^{(n)} = \{(u, \dots, u) \mid u \in U\} \subseteq U^n$.
- When H is abelian, we also define an aggregating version $\sigma^{(\ell, m)} : Q^{\ell+m} \rightarrow H$, as $\sigma^{(\ell, m)}(q_1, \dots, q_\ell, q'_1, \dots, q'_m) = \sum_{i=1}^{\ell} \sigma(q_i) \oplus \sum_{i=1}^m \sigma(-q'_i)$ where the summations refer to the operation \oplus in the group H . $\sigma^{(\ell, m)}$ can be seen to be a bi-affine homomorphism w.r.t. $(T^\ell \times U^m, U^\ell \times T^m)$. We shall simply write $\sigma^{(n)}$ for the symmetric bi-affine homomorphism $\sigma^{(\lceil n/2 \rceil, \lfloor n/2 \rfloor)}$.

These powers of a bi-affine homomorphism are in fact bi-affine homomorphisms. We prove this in the full version. We can now define BA_{σ^n} , $\text{BA}_{\sigma^{(n)}}$ and $\text{BA}_{\sigma^{(\ell, m)}}$ as the bi-affine correlations corresponding to σ^n , $\sigma^{(n)}$ and $\sigma^{(\ell, m)}$ respectively.

3.4 Group Structure of Bi-Affine Correlations

In this section we show connections between (sub)group correlations and bi-affine correlations, which can be summarized as follows:

- ▶ **Theorem 9.** *For any bi-affine homomorphism σ ,*
1. BA_σ is a compact subgroups correlation;
 2. if σ is symmetric, then BA_σ is a group correlation;
 3. if σ is semi-abelian, then BA_σ is embedded in $\text{BA}_{\sigma^{(2)}}$, and more generally, $\text{BA}_{\sigma^{(\ell, m)}}$ is embedded in $\text{BA}_{\sigma^{(2m')}}$ for all $m' \geq \max(\ell, m)$.

We present the key ingredients of the above connections here. Details omitted from here can be found in the full version.

Groups \mathbb{J} and \mathbb{K} . To capture the structure of bi-affine correlations as (sub)group correlations, we define two groups.

- If $\sigma : Q \rightarrow H$ is a bi-affine homomorphism w.r.t. (T, U) , the group \mathbb{J}_σ is defined as the direct product $T \times U \times H$. Then it is easy to see that BA_σ is a subgroups correlation of the form $\text{GC}_{G_1, G_2}^{G, S}$ where $G = \mathbb{J}_\sigma$ and $S = \{(t, u, \sigma(t + u)) \mid t \in T, u \in U\}$, with $G_1 = T \times \{0\} \times H, G_2 = \{0\} \times U \times H$. This is a compact subgroups correlation because $|G_1||G_2| = |T||U||H|^2 > |T||U||H| = |G|$.
- If $\sigma : D \times D \rightarrow H$ is a symmetric bi-affine homomorphism, then \mathbb{K}_σ is defined as $(D \times H, \odot)$, where \odot is given by $(d, h) \odot (d', h') = (d + d', h \oplus h' \oplus \sigma(d, 0) \oplus \sigma(0, d') \oplus -\sigma(d, d'))$. It can now be shown that BA_σ is a group correlation of the form $\text{GC}_{\mathbb{K}_\sigma, S}^{\mathbb{K}_\sigma, S}$, where $S = \{(d + d', \sigma(d, 0) \oplus \sigma(0, d')) \mid d, d' \in D\}$.

In particular, if $\sigma : A \times A \rightarrow A$ for a ring A , with $\sigma(a, b) = ab$ (multiplication in the ring), then the operation \odot is defined as $(t, a) \odot (u, b) = (t + u, a + b - tu)$. This group, which we denote as \mathbb{K}_A , encodes both the addition and multiplication operations in the ring (as $(0, a) \odot (0, a') = (0, a + a')$, and $(a, 0) \odot (a', 0) = (a + a', -aa')$).

3.5 Some Noteworthy Examples

Here we consider several cryptographically interesting examples and show that they are (sub)group correlations and also explore connections between them. More examples along with a tabular summary can be found in the full version.

Oblivious Linear function Evaluation and Beaver’s Multiplication Triples. OLE and BMT over an arbitrary ring A are defined as follows:

$$\begin{aligned} \text{OLE}_A &:= \{(p, a), (q, b) \mid a + b = pq\}, \\ \text{BMT}_A &:= \{(a_1, b_1, c_1), (a_2, b_2, c_2) \mid c_1 + c_2 = (a_1 + a_2)(b_1 + b_2)\}. \end{aligned}$$

Consider the symmetric bi-affine homomorphism $\sigma : A \times A \rightarrow A$ defined with respect to subgroups $T = A \times \{0\}$ and $U = \{0\} \times A$ as $\sigma(p, q) = pq$. It can be seen that the bi-linear correlation BA_σ is isomorphic to OLE_A . OLE_A is also a group correlation (Theorem 9).

It is straightforward to see that BMT is a group correlation with $G = A \times A \times A$ and $S = \{(a, b, ab) \mid a, b \in A\}$. Furthermore, BMT is isomorphic to the bi-linear correlation

$$\text{BA}_{\sigma^{(2)}} := \{((\tilde{a}_1, 0), (0, \tilde{b}_2), \tilde{c}_1), ((0, \tilde{b}_1), (\tilde{a}_2, 0), \tilde{c}_2) \mid \tilde{a}_1 \tilde{b}_1 + \tilde{a}_2 \tilde{b}_2 = \tilde{c}_1 + \tilde{c}_2\},$$

This can be seen by defining isomorphisms

$$\alpha(a_1, b_1, c_1) = ((a_1, 0), (0, b_1), c_1 - a_1 b_1) \text{ and } \beta(a_2, b_2, c_2) = ((0, b_2), (a_2, 0), c_2 - a_2 b_2).$$

It can now be checked that

$$((a_1, b_1, c_1), (a_2, b_2, c_2)) \in \text{BMT}_A \Leftrightarrow (\alpha(a_1, b_1, c_1), \beta(a_2, b_2, c_2)) \in \text{BA}_{\sigma^{(2)}}.$$

Zero-Alternating Sum Correlation. We introduce an important correlation, called Zero Alternating Sum (ZAS) correlation over any (possibly non-abelian) group $(D, +)$. ZAS is a flat correlation $\text{ZAS}_D \subseteq D^2 \times D^2$, defined as

$$\text{ZAS}_D := \{(a, c), (b, d) \mid a + b + c + d = 0\}.$$

We remark that if D is an abelian group, then ZAS_D is a trivial correlation.³

zas_D as a Bi-Affine Correlation. Somewhat surprisingly, ZAS turns out to be a bi-affine correlation. We define the corresponding bi-affine homomorphism $\sigma : D \times D \rightarrow H$, where $H = D^{\text{op}}$, the *opposite group* of D (i.e., H has the same elements as D and has a group operation \oplus defined by $a \oplus b = b + a$). We let $\sigma(x, y) = -(x + y)$. Then, clearly, ZAS is isomorphic to the flat correlation $\{(c, a), (d, b) \mid \sigma(c, d) = a + b\}$. It is straightforward to verify that σ is indeed a bi-affine homomorphism. For completeness, we present a proof in the full version. Later, we refer to the bi-affine homomorphism σ defined above as σ_D^{ZAS} .

³ A secure protocol for sampling from ZAS_D , when D is abelian, is as follows: Alice samples $x \leftarrow D$ to Bob; Alice then picks a random $a \leftarrow D$ and outputs $(a, x - a)$; Bob samples $b \leftarrow D$ and outputs $(b, -x - b)$.

zas_D as a Group Correlation. When D is not abelian, σ defined above is not semi-abelian, and hence ZAS_D is *not* symmetric. As such, Theorem 9 *does not* apply to ZAS_D . Nevertheless, we show below that ZAS_D over any group D is a group correlation of the form $GC^{G,S}$, where the group G is $D \times D$, with coordinate-wise addition, and $S = \{(g, -g) \mid g \in D\}$.

$$\begin{aligned} ((a, c), (b, d)) \in ZAS_D &\Leftrightarrow a + b + c + d = 0 \Leftrightarrow a + b = -(c + d) \\ &\Leftrightarrow (a + b, c + d) \in S \Leftrightarrow (a, c) + (b, d) \in S. \end{aligned}$$

4 Information Theoretic Results

Common-Information. For a pair of correlated random variables (X, Y) , two important information-theoretic measures of correlation are the well-known quantity of *mutual information* $I(X; Y)$ [26] and the lesser known notions of *common information*. Specifically, there are two measures of common information due to Gács and Körner [18] and due to Wyner [28], which can be defined as below:

$$CI_{\text{GK}}(X; Y) = I(X; Y) - RI_{\text{GK}}(X; Y) \quad (2)$$

$$CI_{\text{W}}(X; Y) = I(X; Y) + RI_{\text{W}}(X; Y) \quad (3)$$

$$RI_{\text{GK}}(X; Y) = \inf_{\text{Q}} I(X; Y|Q), \text{ such that } H(Q|X) = H(Q|Y) = 0 \quad (4)$$

$$RI_{\text{W}}(X; Y) = \inf_{\text{Q}} I(Y; Q|X) + I(X; Q|Y), \text{ such that } I(X; Y|Q) = 0 \quad (5)$$

where the infimum is over all random variables Q that are jointly distributed with (X, Y) . Here RI_{GK} and RI_{W} are (respectively) Gács-Körner and Wyner *residual information*.

We shall write $RI_{\text{W}}(C)$ etc. as a short hand for $RI_{\text{W}}(X; Y)$, where the random variables (X, Y) are uniformly distributed over C . We will use the following proposition that is a special case of a “monotonicity” result in [25].

► **Proposition 10** ([25]). *If m independent instances of \mathcal{F}_C can be securely computed using n independent instances of $\mathcal{F}_{C'}$, then $m \cdot RI_{\text{W}}(C) \leq n \cdot RI_{\text{W}}(C')$.*

Also, C is a trivial correlation – i.e., there exists an information theoretically secure 2-party protocol to sample from C – iff $RI_{\text{W}}(C) = 0$ (or equivalently, $RI_{\text{GK}}(C) = 0$).

► **Lemma 11.** *Suppose C is a group correlation of the form $GC^{G,S}$. Then:*

1. C is trivial iff S is a (left or right) coset of a subgroup of G .
2. $CI_{\text{GK}}(C) = 0$ iff the set $\{s - s' \mid s, s' \in S\}$ is a generating set for the group G .
3. If for all $s_1, s_2, s_3, s_4 \in S$, $s_1 - s_2 + s_3 - s_4 = 0 \Rightarrow \{s_1, s_3\} = \{s_2, s_4\}$, then $RI_{\text{W}}(C) = \log |S|$ viz. C is $K_{2,2}$ free.

Now, we state our main technical result in this section. Recall that in a non-defective bi-affine homomorphism, there is no pair $(t, u) \in (T \setminus \{0\}) \times (U \setminus \{0\})$ such that $\sigma(t + u) = \sigma(t) - \sigma(0) + \sigma(u)$.

► **Lemma 12.** *If σ is a non-defective bi-affine homomorphism w.r.t. (T, U) , then $RI_{\text{W}}(\text{BA}_{\sigma}) = \log \min(|T|, |U|)$.*

An example of a non-defective bi-affine homomorphism is multiplication in a *domain*. As a result, we have $RI_{\text{W}}(\text{OLE}_A^n) = \log |A|$ if A is a domain.

5 Protocols for Bi-Affine Correlations

In this section, we present several protocols pertinent to bi-affine correlations. These protocols realize several basic functionalities related to “completing” a correlation (i.e., sampling from a correlation conditioned on certain variables being fixed), given access to a random instance of the same correlation which could be obtained from a semi-trusted source modeled by the biasable sampling functionality. The same protocols can also be used to “rerandomize” for forward security. Missing details of the constructions and their proofs can be found in the full version.

In the following, let $\sigma : Q \rightarrow H$ be a bi-affine homomorphism from a group $(Q, +)$ to group (H, \oplus) w.r.t subgroups $T, U \leq Q$.

5.1 Completing a Bi-Affine Correlation

We first define the conditional sampling functionality that *completes* a bi-affine correlation, by sampling an instance of the correlation conditioned on its inputs.

<p>Conditional Sampling Functionalities $\mathcal{F}_{\sigma U}$, $\mathcal{F}_{\sigma TU}$ and $\mathcal{F}_{\sigma TAU}$ (where $\sigma : Q \rightarrow H$ and $T, U \leq Q$)</p> <p>Inputs: $t \in T, a \in H$ from Alice, and $u \in U$ from Bob, where</p> <p style="text-align: center;">$t = a = \perp$ for $\mathcal{F}_{\sigma U}$ $t \in T, a = \perp$ for $\mathcal{F}_{\sigma TU}$ $t \in T, a \in H$ for $\mathcal{F}_{\sigma TAU}$.</p> <p>Outputs: (\tilde{t}, \tilde{a}) to Alice and (\tilde{u}, \tilde{b}) to Bob, where $((\tilde{t}, \tilde{a}), (\tilde{u}, \tilde{b})) \leftarrow \text{BA}_\sigma$ conditioned on $\tilde{u} = u$, $\tilde{t} = t$ if $t \neq \perp$, and $\tilde{a} = a$ if $a \neq \perp$.</p>
--

Functionalities $\mathcal{F}_{\sigma|T}$ and $\mathcal{F}_{\sigma|TUB}$ are defined symmetric to $\mathcal{F}_{\sigma|U}$ and $\mathcal{F}_{\sigma|TAU}$, respectively. All functionalities allow the adversary to selectively abort output delivery to honest parties (after seeing its own output, if any).

Figure 2 contains UC secure protocols for the functionalities $\mathcal{F}_{\sigma|U}$, $\mathcal{F}_{\sigma|TU}$ and $\mathcal{F}_{\sigma|TAU}$ in the $\tilde{\mathcal{F}}_\sigma$ hybrid model (Definition 4) with only one invocation of $\tilde{\mathcal{F}}_\sigma$. The first two protocols require one round of communication while $\text{Comp}_{\sigma|TAU}$ needs two rounds of communication.

► **Lemma 13.** *$\text{Comp}_{\sigma|U}$, $\text{Comp}_{\sigma|TU}$ and $\text{Comp}_{\sigma|TAU}$ (Figure 2) UC-securely realize functionalities $\mathcal{F}_{\sigma|U}$, $\mathcal{F}_{\sigma|TU}$ and $\mathcal{F}_{\sigma|TAU}$ respectively in the $\tilde{\mathcal{F}}_\sigma$ hybrid.*

We prove this lemma in the full version. Here, we point out that if both parties are honest, then Alice and Bob output (t, a) and (u, b) such that:

$$\begin{aligned}
 a \oplus b &= [\sigma(t + \Delta_u) \oplus -\sigma(t)] \oplus [\tilde{a} \oplus \tilde{b}] \oplus [-\sigma(\tilde{u}) \oplus \sigma(\Delta_t + \tilde{u})] \\
 &= [\sigma(t + \Delta_u) \oplus -\sigma(t)] \oplus [\sigma(\tilde{t} + \tilde{u})] \oplus [-\sigma(\tilde{u}) \oplus \sigma(\Delta_t + \tilde{u})] \\
 &= [\sigma(t + \Delta_u) \oplus -\sigma(t)] \oplus [\sigma((\tilde{t} + \tilde{u}) +_{\tilde{u}}(\Delta_t + \tilde{u}))] \\
 &= \sigma((t + \Delta_u) +_t(t + \tilde{u})) \\
 &= \sigma(t + u)
 \end{aligned}$$

where, we use the properties of σ (Definition 7) and the fact that $\tilde{a} \oplus \tilde{b} = \sigma(\tilde{t} + \tilde{u})$. Also note that to prove $\text{Comp}_{\sigma|TAU}$ realizes $\mathcal{F}_{\sigma|TAU}$, it is sufficient to show that Π_σ is a secure realization of $\mathcal{F}_{\sigma|TAU}$ (and then appeal to the UC theorem to implement $\mathcal{F}_{\sigma|TU}$ with protocol $\text{Comp}_{\sigma|TU}$ in the $\tilde{\mathcal{F}}_\sigma$ hybrid model). Correctness of Π_σ , when the parties are honest, follows from the fact that $a \oplus b = a \oplus \Delta_a \oplus \tilde{b} = \tilde{a} \oplus \tilde{b} = \sigma(t + u)$. UC security follows from the observation that in Π_σ , the inputs to $\mathcal{F}_{\sigma|TU}$ and the message that Alice sends to Bob can be arbitrary and would still correspond to valid input choices of the parties (or aborting).

Protocols $\text{Comp}_{\sigma U}$ and $\text{Comp}_{\sigma TU}$ in the $\tilde{\mathcal{F}}_\sigma$ hybrid model
<ul style="list-style-type: none"> ■ Inputs: Bob receives $u \in U$. In $\text{Comp}_{\sigma TU}$, Alice receives $t \in T$, as well. ■ Invocation of $\tilde{\mathcal{F}}_\sigma$: Alice gets (\tilde{t}, \tilde{a}) and Bob gets (\tilde{u}, \tilde{b}) from $\tilde{\mathcal{F}}_\sigma$, s.t. $\tilde{a} \oplus \tilde{b} = \sigma(\tilde{t} + \tilde{u})$. ■ In $\text{Comp}_{\sigma U}$, Alice sets $t = \tilde{t}$. ■ Alice \leftrightarrow Bob: <ul style="list-style-type: none"> ■ Alice sends Δ_t to Bob, where $\Delta_t := -\tilde{t} + t$. (In $\text{Comp}_{\sigma U}$, $\Delta_t = 0_T$ and this message can be omitted.) ■ Bob sends Δ_u to Alice, where $\Delta_u := u - \tilde{u}$. ■ Output: Alice outputs (a, t) where $a := \sigma(t + \Delta_u) \oplus -\sigma(t) \oplus \tilde{a}$, and Bob outputs (u, b) where $b := \tilde{b} \oplus -\sigma(\tilde{u}) \oplus \sigma(\Delta_t + \tilde{u})$. (In $\text{Comp}_{\sigma U}$, $b = \tilde{b}$.)
Protocol Π_σ in the $\mathcal{F}_{\sigma TU}$ hybrid model
<ul style="list-style-type: none"> ■ Inputs: Alice receives $(t, a) \in T \times H$, and Bob receives $u \in U$. ■ Invocation of $\mathcal{F}_{\sigma TU}$: Alice inputs t, Bob inputs u to $\mathcal{F}_{\sigma TU}$, and receive outputs (t, \tilde{a}) and (u, \tilde{b}) respectively s.t. $\tilde{a} \oplus \tilde{b} = \sigma(t + u)$. ■ Alice \rightarrow Bob: Alice sends Δ_a to Bob, where $\Delta_a := -a \oplus \tilde{a}$. ■ Output: Alice outputs (t, a) and Bob outputs (u, b), where $b := \Delta_a \oplus \tilde{b}$.
Protocol $\text{Comp}_{\sigma TAU}$ in the $\tilde{\mathcal{F}}_\sigma$ hybrid model
<p>$\text{Comp}_{\sigma TAU}$ is obtained by composing Π_σ with $\text{Comp}_{\sigma TU}$ (as an implementation of $\mathcal{F}_{\sigma TU}$).</p>

■ **Figure 2** UC-secure protocols for $\mathcal{F}_{\sigma|T}$, $\mathcal{F}_{\sigma|TU}$ and $\mathcal{F}_{\sigma|TAU}$ in the $\tilde{\mathcal{F}}_\sigma$ hybrid model. All protocols use a single invocation to the functionality $\tilde{\mathcal{F}}_\sigma$. The first two protocols have a single round of message exchange, while the latter requires two rounds.

5.2 Inner-Product Bi-Affine Correlations from Bi-Affine Correlations

If Alice and Bob hold $\ell + m$ instances of any semi-abelian bi-affine correlation BA_σ (in appropriate directions), they can non-interactively extract an instance of $\text{BA}_{\sigma^{(\ell+m)}}$.

Protocol to sample $\text{BA}_{\sigma^{(\ell,m)}}$ in the \mathcal{F}_σ hybrid model
<ul style="list-style-type: none"> ■ Invocation of \mathcal{F}_σ: <ul style="list-style-type: none"> ■ \mathcal{F}_σ is invoked ℓ times, at the end of which Alice holds $(r_1, \dots, r_\ell, x_1, \dots, x_\ell)$ and Bob holds $(s_1, \dots, s_\ell, y_1, \dots, y_\ell)$ such that $\sigma(r_i + s_i) = x_i \oplus y_i$ where $r_i \in T, s_i \in U$ and $x_i, y_i \in H$ for all $i \in [\ell]$. ■ \mathcal{F}_σ is invoked m times in the opposite direction, at the end of which Alice receives $(s'_1, \dots, s'_m, y'_1, \dots, y'_m)$ and Bob receives $(r'_1, \dots, r'_m, x'_1, \dots, x'_m)$, such that $\sigma(r'_i + s'_i) = x'_i \oplus y'_i$ where $r'_i \in T, s'_i \in U$ and $x'_i, y'_i \in H$ for all $i \in [m]$. ■ Outputs: Alice outputs $t_i = r_i, u'_j = -s'_j, h_1 = \sum_{k=1}^{\ell} x_k \oplus \sum_{k=1}^m y'_k$ and Bob outputs $t'_j = -r'_j, u_i = s_i, h_2 = \sum_{k=1}^{\ell} y_k \oplus \sum_{k=1}^m x'_k$ for all $i \in [\ell], j \in [m]$.

■ **Figure 3** A protocol for sampling $\text{BA}_{\sigma^{(\ell,m)}}$ in the $\mathcal{F}_\sigma, \mathcal{F}_{ZAS|TU}$ hybrid model.

The correctness of the protocol in Figure 3 can be seen as follows. Recall that the support of $\sigma^{(\ell, m)}$ is defined as $((t_1, \dots, t_\ell, u'_1, \dots, u'_m, h_1), (u_1, \dots, u_\ell, t'_1, \dots, t'_m, h_2))$ satisfying

$$\sigma^{(\ell, m)}(t_1 + u_1, \dots, t_\ell + u_\ell, u'_1 + t'_1, \dots, u'_m + t'_m) = h_1 \oplus h_2 \quad (6)$$

The L.H.S of (6) can be expanded to verify correctness.

$$\begin{aligned} \sum_{i=1}^{\ell} \sigma(t_i + u_i) + \sum_{i=1}^m \sigma(-t'_i - u'_i) &= \sum_{i=1}^{\ell} \sigma(r_i + s_i) + \sum_{i=1}^m \sigma(r'_i + s'_i) \\ &= \sum_{i=1}^{\ell} (x_i + y_i) + \sum_{i=1}^m (x'_i + y'_i) \\ &= h_1 \oplus h_2. \end{aligned}$$

5.3 Bi-Affine Correlations from String OT

We sample bi-affine correlations by first constructing a protocol for $\mathcal{F}_{\sigma|_{\text{TAU}}}$ in the string OT hybrid model. This implies a semi-honest secure protocol for \mathcal{F}_{σ} when Alice and Bob sample their inputs uniformly at random. As the first step in a protocol for $\mathcal{F}_{\sigma|_{\text{TAU}}}$, Alice and Bob agree upon a *generator matrix* M_U of dimensions $k \times d$ such that every element $u \in U$ can be expressed as $u = \sum_{i=1}^k M_U(i, c_i)$ where $M_U(i, j)$ denotes the element in the i -th row and j -th column and the vector c is the decomposition of element u w.r.t the generator matrix M_U . Given such an generator matrix, our protocol needs k instances of $\binom{d}{1}$ -OT $^\ell$ string OTs.⁴ Figure 4 describes the protocol for $\mathcal{F}_{\sigma|_{\text{TAU}}}$ in the string OT hybrid model.

Protocol $\text{Comp}_{\sigma|_{\text{TAU}}}$ in the $\binom{d}{1}$ -ot $^\ell$ Hybrid model

Parameters: Groups $(T, +)$, $(U, +)$, (H, \oplus) and a generator matrix of U , $M_U \in U^{k \times d}$.

- **Inputs:** Alice has input $t \in T, a \in H$ and Bob has input $u \in U$.
 - Alice samples $\{r_i\}_{i \in [2, k]} \leftarrow H$ and sets $r_1 = a$.
 - For each $i \in [k - 1]$, Alice and Bob invoke $\binom{d}{1}$ -OT $^\ell$. Alice's input is the tuple $\{-r_i \oplus \sigma(t + M_U(i, j)) \oplus -\sigma(t) \oplus r_{i+1}\}_{j \in [d]}$ and Bob's input is a choice integer $c_i \in [d]$ such that $u = \sum_{j=1}^k M_U(j, c_j)$ where $M_U(i, j)$. Bob receives $m_i = -r_i \oplus \sigma(t + M_U(i, c_i)) \oplus -\sigma(t) \oplus r_{i+1}$.
 - For $i = k$, Alice's input is the tuple $\{-r_k \oplus \sigma(M_U(i, j))\}_{j \in [d]}$ and Bob's input is the choice integer $c_k \in [d]$. Bob receives $m_k = -r_k \oplus \sigma(t + M_U(k, c_k))$.
- Bob combines the messages he received to compute $b = \sum_{i=1}^k m_i$

■ **Figure 4** A semi-honest secure protocol realising $\mathcal{F}_{\sigma|_{\text{TAU}}}$ in the $\binom{m}{1}$ -OT $^\ell$ -Hybrid model.

► **Lemma 14.** $\text{Comp}_{\sigma|_{\text{TAU}}}$ (Figure 4) is a semi-honest secure protocol realising $\mathcal{F}_{\sigma|_{\text{TAU}}}$.

Note that $|U| \leq d^k$, since every element in U can be represented as the summation of k elements, each chosen from a d -dimensional row of M_U . The following lemma considers the case when this representation is tight.

⁴ Effectively, we require oblivious transfer over group elements and hence the length of strings must be long enough to send the description of an element.

► **Lemma 15.** *If σ is non-defective and $|U| = d^k \leq |T|$, then $\text{Comp}_{\sigma|_{\text{TAU}}}$ is optimal in the number of instances of $\binom{d}{1}\text{-OT}^\ell$ used (for any length ℓ) for semi-honest securely realizing one instance of BA_σ .*

► **Lemma 16.** *If C is a regular correlation then $RI_W(C) \leq \log \min(\deg_L(C), \deg_R(C))$. Further, if C is $K_{2,2}$ -free, then $RI_W(C) = \log \min(\deg_L(C), \deg_R(C))$.*

Lemma 15 follows from the fact that $RI_W(\binom{d}{1}\text{-OT}^\ell) \leq \log(d)$.⁵ Also, by Lemma 12, $RI_W(\text{BA}_\sigma) \leq \log |U| = k \log d$. Then, by Proposition 10, at least k instances of $\binom{d}{1}\text{-OT}^\ell$ are needed to securely sample one instance of BA_σ , proving the lemma.

Comparison with Gilboa’s protocol. In [19], Gilboa gave a protocol to generate OLE correlations over a ring A . Their protocol requires the ring to have a bit-decomposition which is equivalent to demanding the existence of a generator matrix M_A of dimension $\log |A| \times 2$. When $A = \mathbb{F}_{(2^n)}$, Gilboa’s protocol uses n instances of $\binom{2}{1}\text{-OT}^\ell$. By appealing to Lemma 16, Proposition 10 and the fact that $RI_W(\binom{2}{1}\text{-OT}^\ell) = 1$, it can be argued that this is the minimum number of OTs that must be invoked (in either direction and per correlation if amortised) to obtain an information-theoretically secure 2-Party protocol that samples $\text{OLE}_{\mathbb{F}_{2^n}}$ correlations.

5.4 Biasable Correlations from Tamperable Correlations

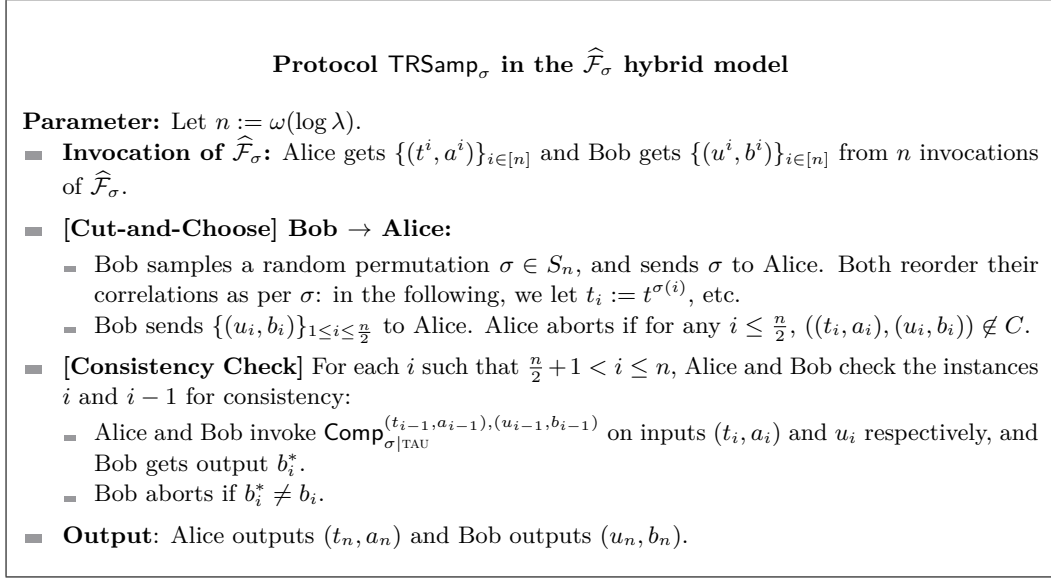
The protocol TR Samp_σ in Figure 5 gives a secure protocol for $\tilde{\mathcal{F}}_\sigma$ in the $\hat{\mathcal{F}}_\sigma$ hybrid model. With no assumptions on the structure of the correlation, Alice and Bob can consume $\log(\lambda)$ correlations and output one correlation which they are guaranteed is correct with overwhelming probability. The main insight in our tamper resistant protocols is to use the following error preservation property of $\text{Comp}_{\sigma|_{\text{TAU}}}$ to check correlations against each other in a “tournament” style and thereby amplify the probability of catching incorrect correlations.

Error-Preservation Property. When $\text{Comp}_{\sigma|_U}$, $\text{Comp}_{\sigma|_{\text{TU}}}$ and $\text{Comp}_{\sigma|_{\text{TAU}}}$ are instantiated in the $\hat{\mathcal{F}}_\sigma$ -hybrid, errors in the correlation output by parties is related to the error in the correlation which parties receive from $\hat{\mathcal{F}}_\sigma$. Recall that when both Alice and Bob are honest, $\hat{\mathcal{F}}_\sigma$ allows the adversary to feed an arbitrary pair $((\hat{t}, \hat{a}), (\hat{u}, \hat{b}))$ to the parties. Suppose, $\hat{a} \oplus \hat{b} = \sigma(\hat{t} + \hat{u}) \oplus \hat{e}$. In this case, the outputs (t, a) and (b, u) are such that $a \oplus b = \sigma(t + u) \oplus e$, where $e = x \oplus \hat{e} \oplus -x$ (for $x = -\sigma(t + \hat{u}) \oplus \sigma(\hat{t} + \hat{u})$). In particular, $e = 0_H$ iff $\hat{e} = 0_H$; further, when H is abelian, $e = \hat{e}$.

► **Lemma 17.** *TR Samp_σ (Figure 5) securely realizes the functionality $\tilde{\mathcal{F}}_\sigma$ against passive corruption, with statistical security.*

A More Efficient Version. While applicable to all bi-affine correlations, TR Samp_σ has a rate of $o(1/\log \lambda)$ in the security parameter λ . Here we present a template which can be used to obtain (a much better) constant rate (in our instantiations, $1/2$, without amortization) in many common examples of bi-affine correlations over large groups. This template is in the form of a passive-secure protocol for $\tilde{\mathcal{F}}_\sigma$ in the $(\hat{\mathcal{F}}_\sigma, \mathcal{E}_\sigma)$ -hybrid, where \mathcal{E}_σ is an “error

⁵ An upperbound on $\binom{d}{1}\text{-OT}^\ell$ can be computed by setting $Q = Y$ in (5), where $X = (m_1, \dots, m_d)$ and $Y = (b, m_b)$. Then $I(Y; Y|X) = H(Y|X) = \log(d)$ since the only remaining entropy in Y given X is the d different choices of b .



■ **Figure 5** A passive secure protocol for $\widetilde{\mathcal{F}}_{\sigma}$ in the $\widehat{\mathcal{F}}_{\sigma}$ hybrid model.

randomization” functionality. Then, \mathcal{E}_{σ} itself is securely realized in the $\widehat{\mathcal{F}}_{\sigma}$ -hybrid, depending on the specifics of the map σ . We implement this latter step only for large groups which satisfy one of three different structural properties.

► **Lemma 18.** *altTRSamp $_{\sigma}$ (Figure 6) passively-securely realizes the functionality $\widetilde{\mathcal{F}}_{\sigma}$, in the $\widehat{\mathcal{F}}_{\sigma}, \mathcal{E}_{\sigma}$ hybrid model*

Error Randomization Functionality. The error randomization functionality \mathcal{E}_{σ} outputs two instances of the correlation $((t_1, a_1), (u_1, b_1))$ and $((t_2, a_2), (u_2, b_2))$ such that either the latter is a valid correlation in BA_{σ} , or the former has a “high min-entropy error”. Relying on this altTRSamp $_{\sigma}$ checks one correlation against the other and catches erroneous correlations with overwhelming probability. In our instantiations of \mathcal{E}_{σ} , the latter is obtained through an invocation of $\widehat{\mathcal{F}}_{\sigma}$ and the former is a “randomised” version of the latter such that the new error (if non-zero) has large min-entropy. For details of the error randomization functionality see Figure 7. Depending on the structure of the bi-affine homomorphism $\sigma : Q \rightarrow H$, the instantiations need different algebraic properties from the group H :

- **Modules:** A group H is said to be a right-module of a ring R if there is a bi-linear map $\sigma : H \times R \rightarrow H$ (i.e., $\sigma((h+h'), r) = \sigma(h, r) + \sigma(h', r)$ and $\sigma(h, (r+r')) = \sigma(h, r) + \sigma(h, r')$) with the additional properties that $\sigma(\sigma(h, r), r') = \sigma(h, (rr'))$ (where the multiplication rr' is from the ring) and $\sigma(h, 1) = h$, where 1 stands for the multiplicative identity in R . Let $\text{units}(R)$ denote the set of ring elements $r \in R$ that have a multiplicative inverse in the ring. We define $\text{minimg}_R(H)$ to be the minimum size of the image of $\text{units}(R)$ under the map $r \mapsto x \cdot r$, over all non-zero elements x in the module H . i.e.,

$$\text{minimg}_R(H) = \min_{x \in H \setminus \{0_H\}} |\{x \cdot r \mid r \in \text{units}(R)\}|.$$

We require that $\text{minimg}_R(H)$ is super-polynomial in the security parameter. An example is the case when R is a large enough field and H is a vector-space over R , then $\text{minimg}_R(H) = |R| - 1$.

- **Semi-Abelian Bi-affine correlations:** For a group H , we define $\text{minord}(H)$ as the order of the smallest non-trivial subgroup of H . Consequently, for all $0 < k < \text{minord}(H)$, for all $h \in H \setminus \{0_H\}$, we have $\underbrace{h + \dots + h}_{k \text{ times}} \neq 0$. $\text{minord}(H)$ equals the smallest prime factor of the order of H . For security we require that $\text{minord}(H)$ is super-polynomial in the security parameter. An example is a large prime order group H , where $\text{minord}(H) = |H|$.
- **Surjective Bi-affine Correlations:** For a (non-abelian) group D , we define $\text{minorbit}(D)$ to be the size of the smallest conjugacy class of D , excluding $\{0\}$. That is,

$$\text{minorbit}(D) := \min_{x \in D \setminus \{0\}} |\{r + x - r \mid r \in D\}|.$$

This instantiation requires the $\text{minorbit}(D)$ must be super-polynomial in security parameter. As an example consider the group $\text{SL}(2, 2^n)^6$ – i.e., 2×2 matrices over \mathbb{F}_{2^n} , with determinant 1, where $\text{minorbit}(\text{SL}(2, 2^n)) \geq 2^n$ [1].

Descriptions of instantiations for the above algebraic objects can be found in the full version.

Protocol altTRSamp $_{\sigma}$ in the $\widehat{\mathcal{F}}_{\sigma}$, \mathcal{E}_{σ} hybrid model

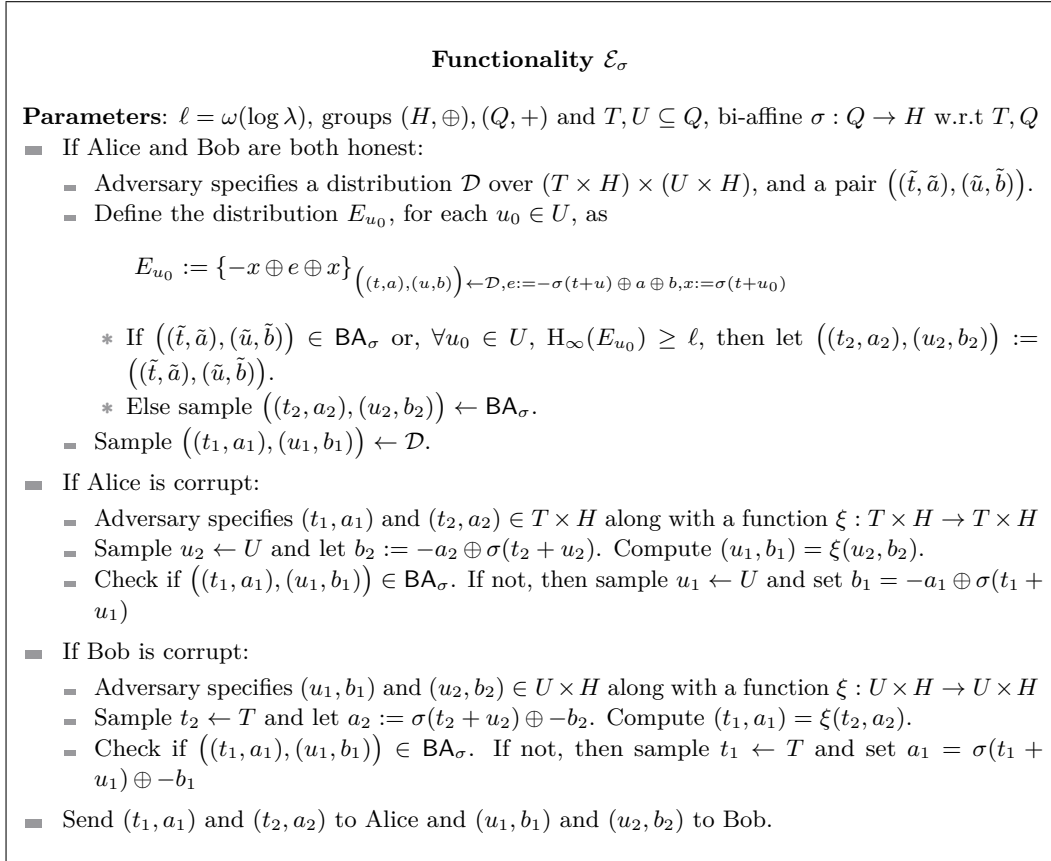
- **Invocation of $\widehat{\mathcal{F}}_{\sigma}$:** Alice gets (t_0, a_0) and Bob gets (u_0, b_0) from $\widehat{\mathcal{F}}_{\sigma}$.
- **Error-Rerandomization:** Alice and Bob invoke \mathcal{E}_{σ} and receive (t_1, a_1) , (t_2, a_2) and (u_1, b_1) , (u_2, b_2) respectively.
- **Verification:**
 - Alice and Bob invoke $\text{Comp}_{\sigma|\text{TAU}}^{(t_0, a_0), (u_0, b_0)}$ on inputs (t_1, a_1) and u_1 respectively, and Bob gets output b^* .
 - Bob aborts if $b^* \neq b_1$.
- **Output:** Alice outputs (t_2, a_2) and Bob outputs (u_2, b_2) .

■ **Figure 6** A passive-secure protocol for $\widetilde{\mathcal{F}}_{\sigma}$ in the $\widehat{\mathcal{F}}_{\sigma}$, \mathcal{E}_{σ} hybrid model.

References

- 1 Edith Adan-Bante and John M Harris. On conjugacy classes of $\text{gl}(n, q)$ and $\text{sl}(n, q)$. *arXiv preprint arXiv:0904.2152*, 2009.
- 2 Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Annual International Cryptology Conference*, pages 420–432. Springer, 1991.
- 3 Donald Beaver. Foundations of secure interactive computing. In *Annual International Cryptology Conference*, pages 377–391. Springer, 1991.
- 4 Donald Beaver. Commodity-based cryptography. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 446–455, 1997.
- 5 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO*, volume 11694, pages 489–518. Springer, 2019.
- 6 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density lpn. *Cryptology ePrint Archive, Report 2020/1417*, 2020.

⁶ Every element in the group also has a succinct representation using $O(n)$ bits.



■ **Figure 7** The error randomization functionality for bi-affine homomorphism σ .

- 7 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-lpn. In *CRYPTO*, pages 387–416. Springer, 2020.
- 8 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: optimizations and applications. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2105–2122, 2017.
- 9 Elette Boyle, Niv Gilboa, and Yuval Ishai. Secure computation with preprocessing via function secret sharing. In *Theory of Cryptography Conference*, pages 341–371. Springer, 2019.
- 10 Gil Cohen, Ivan Bjerre Damgård, Yuval Ishai, Jonas Kölker, Peter Bro Miltersen, Ran Raz, and Ron D Rothblum. Efficient multiparty protocols via log-depth threshold formulae. In *Annual Cryptology Conference*, pages 185–202. Springer, 2013.
- 11 Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient multi-party computation over rings. In *EUROCRYPT*, pages 596–613, 2003. URL: <http://link.springer.de/link/service/series/0558/bibs/2656/26560596.htm>.
- 12 Ivan Damgård, Helene Haagh, Michael Nielsen, and Claudio Orlandi. Commodity-based 2pc for arithmetic circuits. In *IMA International Conference on Cryptography and Coding*, pages 154–177. Springer, 2019.
- 13 Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P Smart. Practical covertly secure mpc for dishonest majority—or: breaking the spdz limits. In *European Symposium on Research in Computer Security*, pages 1–18. Springer, 2013.

- 14 Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pages 643–662. Springer, 2012.
- 15 Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In *NDSS*. The Internet Society, 2015.
- 16 Yvo Desmedt, Josef Pieprzyk, and Ron Steinfeld. Active security in multiparty computation over black-box groups. In *International Conference on Security and Cryptography for Networks*, pages 503–521. Springer, 2012.
- 17 Yvo Desmedt, Josef Pieprzyk, Ron Steinfeld, Xiaoming Sun, Christophe Tartary, Huaxiong Wang, and Andrew Chi-Chih Yao. Graph coloring applied to secure computation in non-abelian groups. *J. Cryptology*, 25(4):557–600, 2012.
- 18 P. Gács and J. Körner. Common information is far less than mutual information. *Problems of Control and Information Theory*, 2(2):149–162, 1973.
- 19 Niv Gilboa. Two party rsa key generation. In *CRYPTO*, pages 116–129, 1999. URL: <http://link.springer.de/link/service/series/0558/bibs/1666/16660116.htm>.
- 20 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008. doi:10.1007/978-3-540-85174-5_32.
- 21 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC*, pages 294–314, 2009. doi:10.1007/978-3-642-00457-5_16.
- 22 Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.
- 23 Joe Kilian. More general completeness theorems for secure two-party computation. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 316–324, 2000.
- 24 Hemanta Maji, Manoj Prabhakaran, and Mike Rosulek. A unified characterization of completeness and triviality for secure function evaluation. In *INDOCRYPT*, pages 40–59, 2012.
- 25 Vinod M Prabhakaran and Manoj M Prabhakaran. Assisted common information with an application to secure two-party sampling. *IEEE Transactions on Information Theory*, 60(6):3413–3434, 2014.
- 26 Claude Shannon. A mathematical theory of communications. *Bell System Technical Journal*, 27:379–423, July 1948.
- 27 Nigel P Smart and Titouan Tanguy. Taas: Commodity mpc via triples-as-a-service. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 105–116, 2019.
- 28 A. D. Wyner. The common information of two dependent random variables. *IEEE Transactions on Information Theory*, 21(2):163–179, 1975.

More Communication Lower Bounds for Information-Theoretic MPC

Ivan Bjerre Damgård ✉

Department of Computer Science, Aarhus University, Denmark

Boyang Li ✉

Institute of Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

(This work was done while visiting Aarhus University).

Nikolaj Ignatieff Schwartzbach ✉

Department of Computer Science, Aarhus University, Denmark

Abstract

We prove two classes of lower bounds on the communication complexity of information-theoretically secure multiparty computation. The first lower bound applies to perfect passive secure multiparty computation in the standard model with $n = 2t + 1$ parties of which t are corrupted. We show a lower bound that applies to secure evaluation of any function, assuming that each party can choose to learn or not learn the output. Specifically, we show that there is a function H^* such that for any protocol that evaluates $y_i = b_i \cdot f(x_1, \dots, x_n)$ with perfect passive security (where b_i is a private boolean input), the total communication must be at least $\frac{1}{2} \sum_{i=1}^n H_f^*(x_i)$ bits of information.

The second lower bound applies to the perfect maliciously secure setting with $n = 3t + 1$ parties. We show that for any n and all large enough S , there exists a reactive functionality F_S taking an S -bit string as input (and with short output) such that any protocol implementing F_S with perfect malicious security must communicate $\Omega(nS)$ bits. Since the functionalities we study can be implemented with linear size circuits, the result can equivalently be stated as follows: for any n and all large enough $g \in \mathbb{N}$ there exists a reactive functionality F_C doing computation specified by a Boolean circuit C with g gates, where any perfectly secure protocol implementing F_C must communicate $\Omega(n g)$ bits. The results easily extends to constructing similar functionalities defined over any fixed finite field. Using known techniques, we also show an upper bound that matches the lower bound up to a constant factor (existing upper bounds are a factor $\lg n$ off for Boolean circuits).

Both results also extend to the case where the threshold t is suboptimal. Namely if $n = kt + s$ the bound is weakened by a factor $O(s)$, which corresponds to known optimizations via packed secret-sharing.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases Multiparty Computation, Lower bounds

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.2

Funding *Ivan Bjerre Damgård*: The European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO).

Nikolaj Ignatieff Schwartzbach: The European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO).

1 Introduction

In secure multiparty computation (MPC) a set of n parties compute an agreed function on inputs held privately by the parties. The goal is that the intended result is the only new information released and is correct, even if t of the parties are corrupted by an adversary.

In this paper we focus on unconditional security where even an unbounded adversary learns nothing he should not, and we ask what is the minimal amount of communication one needs to compute a function securely. To be clear, we will only consider functions where



© Ivan Bjerre Damgård, Boyang Li, and Nikolaj Ignatieff Schwartzbach;
licensed under Creative Commons License CC-BY 4.0

2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 2; pp. 2:1–2:18



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

the size of the output is much shorter than the input, so we avoid trivial cases where the communication is large, simply because the parties need to receive a large output. Note that one can always compute the function without security by just sending the inputs to one party and let them compute the function, so the question to consider is: compared to the size of the inputs, what overhead in communication (if any) is required for a secure protocol? Note that a different and probably much harder question is if, in general, the communication must be larger than the circuit size of the function.

These questions only seem interesting for unconditional security: for computational security we can use homomorphic encryption to compute any function securely with only a small overhead over the input size.

There is a lot of prior work on lower bounding the communication required in interactive protocols, and we survey some of this below. However, the most relevant existing work for us is [6] which considers exactly the questions we ask here for the case of honest majority, $n = 2t + 1$, and passive (semi-honest) security. They show that a factor n overhead over the input size is required for a variant of the inner product function, where parties may privately choose to learn or not to learn the output. The result extends to the case of suboptimal threshold where $n = 2t + s$, and then the overhead becomes n/s .

Note that this result leaves open two important questions:

Firstly, a natural next step after the results from [6] is to ask which functions in general require large communication. However, applying the result from [6] to functions other than the inner product is nontrivial because they leverage a particular property of the inner product, namely that it can be used to implement a PIR, which is of course not the case in general. In this work, we therefore ask:

Can we show lower bounds for perfect passive secure evaluation of functions other than the inner product?

Second, it is well known that perfect malicious security can be achieved if and only if $t < n/3$ and the result from [6] has nothing to say about this case: to apply it, one would need to set s to be $\Theta(n)$ and then their lower bound becomes trivial. Hence, the final open question we consider is:

Can we show lower bounds for perfect malicious security in the case where $n = 3t + 1$?

We answer both questions in the affirmative.

1.1 Our results

1.1.1 Bounds for passive security

In this paper, we prove lower bounds for the model with n parties of which t are statically corrupted. The network is synchronous, and we assume that the adversary can learn the length of any message sent (in accordance with the standard ideal functionality modeling secure channels which always leaks the message length). We consider information-theoretically secure protocols with static corruption in the maximal threshold model.

On the technical side, what we show are actually lower bounds on the entropy of the messages sent on the network when the inputs have certain distributions. This then implies similar bounds in general on the average number of bits to send: an adversary who corrupts no one still learns the lengths of messages, and must not be able to distinguish between different distributions of inputs. Hence message lengths cannot change significantly when we change the inputs, otherwise the protocol is insecure.

For our passive lower bounds, we require that protocols securely implement the standard functionality for secure function evaluation, where we add the option that each player P_i can privately choose to learn or not to learn the output, by selecting an additional input bit b_i . What we show is that there exists a mapping H^* which takes any function f , such that in any n -party protocol securely evaluating the output $y_i = b_i \cdot f(x_1, \dots, x_n)$ for player P_i , the total average communication must be at least $\frac{1}{2} \sum_{i=1}^n H_f^*(x_i)$ where x_i is the input of player P_i .

Very roughly speaking, the function $H_f^*(x_i)$ measures how much uncertainty remains in the function output given that we know x_i . Specifically, it is defined as the maximum uncertainty that remains on any subset of inputs of size t among the remaining $2t$ inputs. The lower bounds that we establish are tight in some cases: for the inner product we get a bound of $\Omega(n)$ times the input length, so we recover the lower bound of [6]. Since the inner product can be computed by a circuit of linear size, this bound is tight up to constant factors. For the XOR function we get a trivial lower bound which only states that each party must communicate their input. As the XOR function is linear, it is of course not surprising that the bound is trivial in this case. Indeed, for two parties the bound is tight since a passively secure protocol is for one of the two parties to simply reveal their input to the other party. A final interesting example is a function called “ranking”, that provides each party with the index of their input in the sorted list of all inputs. For this example, we get again a non-trivial bound of $\Omega(n)$ times the input size. This bound may not be tight, assuming there is no linear-sized circuit for sorting integers.

On the technical side, our bound is established by considering a fixed party P_i and choosing a bipartitioning of the remaining $2t$ parties into two groups of size t . We show that the entropy such a group provides to the function output is a lower bound on the communication of party P_i so we choose the maximum value among all such partitions. This corresponds to the definition of the function H_f^* mentioned above. Since the adversary is not allowed to distinguish between different distributions of messages we can essentially add all the lower bounds for the communication of all parties to obtain our lower bound.

The lower bound extends to the case where the threshold is submaximal, i.e. $n = 2t + s$. The bound can be established by considering a partitioning of the parties into sets of size s (it is allowed that a party belongs to no set). For each such set, we take the supremum over all ways of bipartitioning the remaining $2t$ parties into two sets of size t to get a communication bound. Since the adversary is not allowed to distinguish between different distributions of messages, again we can add the communication for each such set of size s . This means we get a communication lower bound for each such partition of the parties into sets of size s so we take the maximum among all such partitions. For functions which are “symmetric”, any partition of the parties into sets of size s gives the same lower bound which means the final supremum can be omitted. In this case, the lower bound is weakened by a factor $O(s)$ such that the total communication can be shown to be $\Omega(\sum_{i=1}^n H_f^*(X_i))/s$. For functions which are not symmetric it is not possible to make a general statement about what happens in the submaximal threshold case, though highly asymmetric functions likely have weaker lower bounds since only a few parties contribute a large amount of entropy to the function output.

1.1.2 Bounds for active security

For our lower bounds for active security, we make use of the Universal Composability (UC) model for secure protocols. Recall that, in the UC model, we specify an “ideal functionality” in order to state what a protocol is supposed to do. The functionality accepts input from the parties and computes outputs in a specific way that an adversary by definition cannot modify. A protocol securely implements the functionality if running the protocol is, in a certain well-defined sense, “equivalent” to interacting with the functionality.

In our case, we consider a functionality F_f computing a function f with a specific structure. Namely, the functionality first receives input from all parties and sends an acknowledgement to everyone. Then it receives a second batch of inputs, computes the desired function and sends the result to all parties. This structure of F_f implies that in any protocol implementing F_f , the first set of inputs must be chosen and committed before the second set of inputs are chosen, and this is important for the proof of our lower bound. However, even if this particular structure is a limitation, the model still covers some natural applications. For instance, the concrete function we study models a case where a long string (a database) is determined in the first phase, and the function to be evaluated then returns a bit in a certain position chosen later (an entry in the database).

We assume UC security mainly for simplicity of exposition, we can actually make do with significantly weaker assumptions, this is detailed in Section 3.5. What we show is that for all n and any sufficiently large S , there exists a function f_S with input size S such that any protocol that evaluates F_{f_S} securely must communicate $\Omega(nS)$ bits.

Even more is true: we are able to construct functions f_S as we just claimed such that they can be evaluated by circuits of size $O(S)$. This means we also get the following result: for any n and all large enough $g \in \mathbb{N}$ there exists a Boolean circuit C with g gates specifying the computation to be done by functionality F_C , such that any protocol that evaluates F_C securely must communicate $\Omega(ng)$ bits.

We emphasize that our result leaves open the question of overhead over the circuit size when the circuit is much bigger than the inputs. However, there is still something we can say about this general question. Note that the general MPC protocols we know are not, strictly speaking, protocols. Rather, they are protocol compilers that take a circuit C as input, and produce a protocol for computing C securely. Our results do imply that any such compiler must produce a protocol with large communication overhead over the circuit size when applied to circuits in the family we build. Now, if this overhead would no longer be present when applying the compiler to other circuits, it would mean that it was able to exploit in some non-trivial way the structure of the circuit it is given. Doing this would require protocol compilers of a completely different nature than the ones we know, which do “the same thing” to any circuit they are given.

This bound also extends to the case where the threshold t is suboptimal. Namely, if $n = 3t + s$, then the lower bound is $O(ng/s)$ and this shows that the improvement in communication that we know we can get using so-called packed secret sharing, is the best we can achieve. The bound does not, however, extend to statistical security. We show in Section 3.6 that there exists a statistically secure protocol breaking the bound already in the 4-party case.

We also show an upper bound that matches the lower bound up to a constant factor for all values of $t < n/3$. This is motivated by the fact that the existing upper bound from [14] is a factor $\lg n$ off for Boolean circuits. We do this by exploiting recent results by Cascudo et al. [4] on so-called reverse multiplication friendly embeddings. Other than establishing the exact communication complexity for this particular class of functions, it also shows that our result is the best possible general lower bound we can have.

To show our results, we start from a lower bound for the communication complexity of a specific function for the case of four parties including one maliciously corrupt player. We then “lift” this result to the multiparty case. This high-level strategy is similar the one used in [6], however our proof for the four party case as well as the concrete lifting technique are very different from what was done in [6]. In fact it is easy to see that new techniques are necessary to achieve our result. Namely, in our case where $t < n/3$, [6] only gives a trivial

result, as mentioned above. Nevertheless [6] is known to be optimal for passive security, even in the case of suboptimal threshold. This means that there is no way to use their proof for our question, one must somehow exploit the fact that the considered protocols are assumed to be maliciously secure.

1.2 Related work

Prior work on lower bounding communication in interactive protocols includes [15, 12, 5, 11, 15, 16, 2, 13] (and see [10] for an overview of these results). The previous work most relevant to us is [10]. They consider a special model with three parties where only two have input and only the third party gets output, and consider perfect secure protocols. This paper was the first to show an explicit example of a function where the communication for a (passive and perfectly) secure protocol must be larger than the input.

Later, in [8], a lower bound was shown on the *number of messages* that must be sent to compute a certain class of functions with statistical security. When the corruption threshold t is $\Theta(n)$, their bound is $\Omega(n^2)$. This of course implies that $\Omega(n^2)$ bits must be sent. However, we are interested in how the communication complexity relates to the input and circuit size of the function, so once the input size becomes larger than n^2 the bound from [8] is not interesting in our context.

In [9], lower bounds on communication were shown that grow with the circuit size. However, these bounds only hold for a particular class of protocols known as gate-by-gate protocols, and we are interested in lower bounds with no restrictions on the protocol.

2 Lower bounds for arbitrary functions

In this section we prove a lower bound on the communication complexity for perfect passive secure multiparty computation. The lower bound applies to any function in which the parties can choose to learn or not to learn the output. For some functions, the lower bound can be shown to be tight.

Let X be a random variable with pdf $p : \mathcal{X} \rightarrow [0, 1]$. We define the (Shannon) entropy of X as:

$$H(X) = - \sum_{x \in \mathcal{X}} p(x) \lg p(x)$$

where \lg is base 2. The entropy measures the uncertainty of X : to communicate the outcome of X , an average of $H(X)$ bits have to be communicated. We define the conditional entropy $H(Y | X)$ as the amount of information in Y left, given that we know X . If $H(Y; X)$ is the joint entropy we define:

$$H(Y | X) = H(Y; X) - H(X)$$

A related measure is the *mutual information* $I(Y; X)$ that measures how much information the two random variables X, Y have in common. It is defined as:

$$I(X; Y) = H(X) - H(X | Y)$$

We will use this measure for our lower bound, we need the following identities:

► **Lemma 1.** $I(X; Y) = I(X; Z) + [H(X | Z) - H(X | Y)]$.

Proof. Follows from the definition of mutual information:

$$I(X; Y) = H(X) - H(X | Y) = I(X; Z) + H(X | Z) - H(X | Y) \quad \blacktriangleleft$$

► **Lemma 2.** Let X, Y, Z be random variables such that $I(Y; Z) \geq H(Y)$. Then $H(X | Z) \leq H(X | Y)$.

Proof. By using the chain rule for entropy twice we get:

$$\begin{aligned} H(X, Y, Z) &= H(Y) + H(X | Y) + H(Z | Y, X) \\ &= H(Z) + H(X | Z) + H(Y | Z, X) \end{aligned}$$

Since $I(Y; Z) \geq H(Y)$ we have $H(Y | Z, X) = 0$ and so we find that:

$$H(X | Y) = H(X | Z) + [H(Z) - H(Z | Y, X)] - H(Y)$$

Noting that $H(Z | Y, X) \leq H(Z | Y)$ we get:

$$H(Z) - H(Z | X, Y) \geq H(Z) - H(Z | Y) = I(Y; Z) \geq H(Y)$$

In particular, we have $[H(Z) - H(Z | Y, X)] - H(Y) \geq 0$ which concludes the proof. ◀

► **Lemma 3.** Let X, Y, Z be random variables such that $I(X; (Y, Z)) \geq \ell$, and $I(X; Y) = 0$. Then $H(Z) \geq \ell$.

Proof. We use the chain rule for mutual information to obtain:

$$\ell \leq I(Y, Z; X) = I(X; Y) + I(Z; X | Y)$$

By assumption we have $I(X; Y) = 0$. The bound $I(Z; X | Y) \leq H(Z)$ is not hard to see and concludes the proof. ◀

We now define the *functional entropy* of a random variable which is used to establish our communication lower bound. Informally, the functional entropy measures how much uncertainty an input to a function provides to its output.

2.1 Functional entropy

We start by considering the binary case: let $f : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Y} \times \mathcal{Y}$ be a binary function, and let X_1, X_2 be random variables over \mathcal{X} . We define the f -expansion of X_1 to be the following exponential-sized string (the case for X_2 is similar):

$$\mathcal{E}_f(X_1) = \left\| \left\| f(X_1, x_2) \right\|_{x_2 \in \mathcal{X}} \right\|$$

where $\|$ denotes string concatenation. The order of the concatenation matters in the sense that it must be a fixed order, but the specific order is not important. We now define the *functional entropy* of X_i as:

$$H_f(X_i) = H(\mathcal{E}_f(X_i))$$

Loosely speaking, this quantity measures how much uncertainty remains in the function output, given that we remove all randomness from variables other than X_i . Since the value of $\mathcal{E}_f(X_i)$ can be computed from X_i , the functional entropy must be upper bounded by the regular Shannon entropy, i.e. $H_f(X_i) \leq H(X_i)$.

We now extend the notion to an n -ary function $f : \mathcal{X}^n \rightarrow \mathcal{Y}^n$ for $n = 2t + 1$ and some t . Let $T \subset \{1, 2, \dots, n\}$ be a set of indices, and define $-T$ as its complement. Note that we can write any f as f' where

$$f(X_1, X_2, \dots, X_n) = f'(X_T, X_{-T})$$

We define the functional entropy of a set of random variables T as:

$$H_T = H_{f'}(X_T)$$

Finally, we define the maximum functional entropy of a variable X_i as:

$$H_f^*(X_i) = \max_{T, |T|=t, i \notin T} H_T$$

Loosely speaking, $H_f^*(X_i)$ measures how much uncertainty we can have in the function output, if we fix all but t inputs, where these t inputs do not include X_i . We will use this quantity to establish our lower bound.

2.2 Lower bound, arbitrary functions, maximal threshold

In this section we establish the communication lower bound for perfect security and maximal threshold. Let $n = 2t + 1$ be an integer, and consider a set of parties P_1, P_2, \dots, P_n computing a function where the i^{th} party learns $y_i = b_i \cdot f_i(\mathbf{X})$, where $b_i \in \{0, 1\}$ is a private boolean input, and $f : \mathcal{X}^n \rightarrow \mathcal{Y}^n$ is a vector function. Consider a partition of the n parties into groups of size $t, t, 1$ where X_1 is the concatenated inputs of parties in the first group, X_2 the second group, and X_3 is the input of the single party (not including the b_i inputs). Let $C_{i,j}$ be the (ordered) concatenation of all messages sent between groups i and j .

In the following two lemmas we consider a situation where only the single players in group 3 learns the output, while all other players have their output selection bit set to 0. Now, since no group has more than t players, the adversary can corrupt all players in each single group, and hence neither the first, nor the second group must learn anything new from the protocol.

► **Lemma 4.** $I(X_1; (C_{1,2}; C_{1,3})) \geq H_f(X_1)$.

Proof. By privacy against group 1, the variables $C_{1,2}, C_{1,3}$ are independent of X_2 and X_3 . This means group 2 and P_3 can resample randomness and use $(C_{1,2}, C_{1,3})$ as an oracle to compute $f(x_1, x_2, x_3)_{2,3}$ for any values of x_2, x_3 . This implies that the mutual information between the communication and the expansion is at least the entropy of the expansion:

$$I(\mathcal{E}_f(X_1); (C_{1,2}; C_{1,3})) \geq H(\mathcal{E}_f(X_1)) = H_f(X_1) \quad (1)$$

On the other hand, as $\mathcal{E}_f(X_1)$ is determined by X_1 we have that,

$$I(X_1; \mathcal{E}_f(X_1)) = H_f(X_1) \quad (2)$$

We now compute:

$$\begin{aligned} & I(X_1; (C_{1,2}, C_{1,3})) \\ &= I(X_1; \mathcal{E}_f(X_1)) + (H(X_1 | \mathcal{E}_f(X_1)) - H(X_1 | (C_{1,2}; C_{1,3}))) \quad \text{by Lemma 1} \\ &= H_f(X_1) + [H(X_1 | \mathcal{E}_f(X_1)) - H(X_1 | (C_{1,2}; C_{1,3}))] \quad \text{by Equation (2)} \end{aligned}$$

By Equation (1) we can apply Lemma 2 to conclude the value in the brackets is nonnegative which concludes the proof. ◀

► **Lemma 5.** $H(C_{1,3}) \geq H_f(X_1)$.

Proof. Immediate consequence of Lemmas 3 and 4 because of privacy against group 2 which implies $I(X_1; C_{1,2}) = 0$. ◀

► **Theorem 6.** *In any MPC protocol of maximal threshold $n = 2t + 1$ that evaluates $y_i = b_i \cdot f_i(x_1, x_2, \dots, x_n)$ with perfect passive security, the total communication is at least*

$$\frac{1}{2} \sum_{i=1}^n H_f^*(X_i)$$

bits of information.

Proof. Consider any party P_i . Then for any partition of the remaining $2t$ parties into two groups of size t , Lemma 5 gives a lower bound on $H(C_{1,3})$ for a certain setting of the inputs. We then choose the maximum such lower bound which is precisely $H_f^*(X_i)$. By perfect passive security, the adversary is not allowed to distinguish between different distributions of messages so we can add the lower bound obtained for each choice of P_i . Finally, we divide by two because each bit is counted exactly twice: once at the sender and once at the receiver. ◀

2.3 Lower bound, arbitrary functions, submaximal threshold

In this section we consider the case when the number of corruptions is submaximal, i.e. $n = 2t + s$ for some $s > 1$. We extend our definition of H_f^* to apply to groups of variables. Let S be some group of parties of size s we then define:

$$H_f^*(X_S) = \max_{T, |T|=t, S \cap T = \emptyset} H_T$$

We consider a fixed partition of the parties into groups of size t, t, s : we call the concatenated inputs of the parties in each group for X_1, X_2, X_3 , and let $C_{i,j}$ denote the correspondence between groups i, j .

Let \mathcal{S} be a partition of the parties into sets of size s , and let \mathcal{S} be the set of all such partitions. Note that a party is allowed to belong to no set in \mathcal{S} , say if $2t$ is not divisible by s .

► **Theorem 7.** *In any MPC protocol of submaximal threshold $2t + s$ that evaluates $y_i = b_i \cdot f_i(x_1, x_2, \dots, x_n)$ with perfect passive security, the total communication is at least*

$$\max_{\mathcal{S} \in \mathcal{S}} \left[\frac{1}{2} \sum_{S \in \mathcal{S}} H_f^*(X_S) \right]$$

bits of information.

Proof. Consider some fixed partition \mathcal{S} of the parties into sets of size s . We can let any element $S \in \mathcal{S}$ define a partition of the parties into sets of size t, t, s . The third group can be regarded as a single party with the concatenated inputs as their input. In doing so, we obtain the result of Lemma 5 for any such partition. This means the communication for the third group must be at least $H_f^*(X_S)$. Since the adversary is not allowed to distinguish between different distributions of messages, we can add the communication for all $S \in \mathcal{S}$ to get a lower bound on the communication. Finally, any such \mathcal{S} yields a lower bound, so we choose the partition $\mathcal{S} \in \mathcal{S}$ that maximizes the lower bound. ◀

The statement of the theorem allows for the function to be “asymmetric” in the sense that some ways of partitioning the parties gives stronger bounds. If any choice of \mathcal{S} gives the same lower bound, we say the function f is symmetric. For symmetric functions, the above lower bound can be simplified. We assume that s divides n for simplicity.

► **Corollary 8.** *Let f be a symmetric function. Then in any MPC protocol of submaximal threshold $2t + s$ that evaluates $y_i = b_i \cdot f_i(x_1, x_2, \dots, x_n)$ with perfect passive security, the total communication is at least*

$$\frac{n}{2s} H_f^*(X_1)$$

bits of information.

2.4 Examples

We briefly provide some examples of different choices of f . In the following, we let I denote an upper bound on the bit length of each participants input, i.e. $X_i \in \{0, 1\}^I$ for every i .

2.4.1 Inner product

We consider a variant of the inner product function where $2t$ parties provide inputs, while the last party only provides a value for b . Now consider any single party. We can then divide the remaining $2t$ parties along the “aisle” of the inner product function. Closer study reveals that almost all information in X_i matters, meaning we get:

$$H_f^*(X_i) = tI$$

Summing this up reestablishes the lower bound of [6]. We note that this lower bound is tight up to constant factors.

2.4.2 XOR

Consider the bitwise XOR function, that takes n inputs $x_1, \dots, x_n \in \{0, 1\}^I$ and outputs $y = \bigoplus_{i=1}^n x_i$. We note that two inputs X_T, X'_T provide the same expansion iff $\bigoplus_{i \in T} X_i = \bigoplus_{i \in T} X'_i$. This means we get:

$$H_f^*(X_i) = I$$

Summing this up gives a lower bound of $\Omega(nI)$ which only states that each party must communicate their input. However, for two parties this is tight since a passively secure two-party protocol for XOR is for one of the parties to simply reveal their input.

2.4.3 Ranking

Consider a function where each party P_i inputs an integer x_i and learns the index of their input in the sorted list of all inputs. Note that two inputs X_T, X'_T have the same expansion if and only if they are permutations of each other. Strictly speaking, the “if” part is not necessary to be proven since we are calculating a lower bound (not the upper bound), but to illustrate, from perspective of outside T , the output mostly does not change when values within T are permuted among themselves. (This could change some tie-breaking if the mechanics depend on index, but it is true if we allow ties). For the “only if” part, for any two X_T and X'_T that are different in the multi-set of values they contain, there must be an

integer that is larger than a values in the multi-set of $\{X_T\}$ and b for $\{X'_T\}$ and $a \neq b$: thus there exists an entry in the expansion where someone outside T holds the integer, and it would rank differently the two cases, thus the expansion is different. For a list of n items, the information content of a permutation on n elements is bounded by $\lg n! \leq \lg n^n \leq n \lg n$. This means we get a communication lower bound of $\Omega(ntI - nt \lg t)$ bits, which for large inputs is $\Omega(ntI)$.

Regarding the corresponding upper bound, we can use a construction by Parberry ([17]) of a sorting network with $O(n(\lg n)^2)$ gates. We can now use any passively secure MPC protocol with linear complexity per gate to compute ranking in time $O(ntI(\lg n)^2)$. This has a discrepancy of a factor $O(\lg n)^2$ and gives a communication lower bound of $\Omega(n/(\lg n)^2)$ per gate. This bound is not tight unless there is a passively secure MPC protocol for sorting with sublinear communication complexity per gate; or if there is a circuit for sorting with linear size. The latter is not true unless sorting can be done in linear time. As a result, it is unlikely that our bound is tight for the ranking function.

3 Lower bounds for malicious security

In this section we prove that there is an n -party functionality that can be described by a circuit with g gates such that each party needs to communicate at least $\Omega(g)$ bits. We show this using a series of lemmas that bound the entropy on the communication. We first show the special case for four parties, and then “lift” this to the general case with n parties.

Let P_1, \dots, P_n be parties connected by pairwise secure channels. We denote by I the *input size* (in bits) of each party, and O the *output size*. For simplicity we assume all parties receive the same output, and denote by $f : \{0, 1\}^{nI} \rightarrow \{0, 1\}^O$ the function to compute.

We assume an active adversary that is allowed to statically corrupt up to t parties where $3t < n$. To define security we use the universal composability (UC) model by Canetti. A quick reminder (for details, see [3]): The model includes the environment Z , a machine that models everything that is external to the protocol, include adversarial attacks. π stands for the protocol, i.e., a set of machines modelling the parties that executes it. The symbol \diamond stands for “compose”, so $Z \diamond \pi$ denotes the “real process” where Z interacts with (attacks) the protocol. The model also includes an ideal functionality F that specifies what the protocol is supposed to do. To compare F to π , we need a simulator S that in a nutshell converts the interface offered by F to the interface Z sees when attacking the protocol. Thus $Z \diamond S \diamond F$ denotes the “ideal process” where Z interacts with S and F and hence only attacks allowed by F are possible. We then say that π securely implement F , if there exists a simulator S so that no environment Z can tell if it is doing the real or the ideal process. A bit more formally:

► **Definition 9** (UC Security). *A protocol π is said to securely realize a functionality F with perfect malicious security if there exists a simulator S such that for any environment Z , we have that $Z \diamond \pi$ is perfectly indistinguishable from $Z \diamond S \diamond F$.*

We will consider protocols that implement a reactive ideal functionality F_f for computing f securely. The functionality first receives input from each party, and sends an acknowledgement to all parties once the inputs have been received. Finally, it accepts an additional input from all parties, it then computes the function and sends the output to all parties. As we shall see, it is important towards proving our lower bound that we consider this reactive case, rather than the simpler version where the functionality gets all the inputs in one go.

Note that any protocol implementing F_f will naturally consist of two phases: one that implements the part where the first inputs are sent, which we call the input phase, and the rest, which we call the computation phase. This implies that the first batch of inputs are committed in the first phase, before any information on the second batch of inputs or the output is revealed.

Note that the structure imposed by our choice of F_f still allows us to model quite natural tasks. The concrete function we consider below is one where a long bit string (a “database”) is committed in the first phase, and then the function computed will securely extract a particular entry in the database.

3.1 Lower bound, malicious security, four parties

We start by considering a special case of active MPC with four parties P_1, \dots, P_4 . In the input phase, the functionality receives an input bit string X_i from each P_i . We assume that $X_1 \in \{0, 1\}^I$ (we do not need to assume anything about the lengths of the other inputs). Let L be the length of the concatenation $\mathbf{X} = X_1 || X_2 || X_3 || X_4$. In the second phase, the functionality receives an integer u_i from P_i , where $u_i \in \mathbb{Z}_L$. It outputs $(u, \mathbf{X}[u])$, where $u = \sum_i u_i \bmod L$.

We call this function $f_{I,4}$. It has the important property that if the input X_1 of P_1 is changed, there is always a setting of the other inputs for which the change of X_1 will cause the output to change, namely if u points to a position in X_1 that was changed. One consequence of this is the following lemma:

► **Lemma 10.** *Assume protocol π computes n -party function f with perfect security, and it is the case that for any $x'_1 \neq x_1$, there are values x_2, \dots, x_n of the other inputs such that $f(x_1, \dots, x_n) \neq f(x'_1, x_2, \dots, x_n)$. Assume further that P_1 has input x_1 , is corrupt but plays honestly. Then the simulator for π must always send x_1 as input to the functionality for f on behalf of P_1 .*

Proof. If all players are honest and have inputs x_1, \dots, x_n , then by perfect security the output must be $f(x_1, \dots, x_n)$. If instead P_1 is corrupt but plays honestly, the protocol does exactly the same as if all players are honest so the output is still $f(x_1, \dots, x_n)$. Hence, when simulating this case, the simulator must send x_1 to the functionality, for any other value x'_1 it may send, the output in the simulation will be incorrect for some choice of x_2, \dots, x_n , by assumption in the lemma. ◀

Before continuing, we define some terminology: suppose we are given a player P that takes part in a protocol π , and let t be a transcript, that is, the ordered set of all messages sent and received during an execution of the protocol. Now, *sampling random coins consistent with t* means to sample uniformly a random tape r that could have been used to create t if P had done the protocol honestly. In other words, r has the property that if P starts π with random tape r and receives in each round the messages specified in t , he would send the messages specified in t in each round. Of course, such a sampling is not always efficient, but remember that we consider perfectly secure protocols that must be robust against unbounded adversaries.

► **Theorem 11.** *In any reactive protocol that implements $F_{f_{I,4}}$ with perfect malicious security, P_4 must use average communication $\Omega(I)$.*

Proof. Consider a protocol π that computes the function with perfect security. We will consider the messages sent in π as random variables as follows: fix the inputs of P_2, P_3 and P_4 to arbitrary values x_2, x_3, x_4 , and let the input of P_1 be chosen uniformly. Assume π is

executed such that all parties follow the protocol. Now, we let T_i for $i = 1, 2, 3, 4$ be the random variable that represents concatenation of all messages sent to and from P_i in the execution of the input phase.

Since the communication pattern must not depend on the inputs, it suffices to show that $H(T_4) \geq H(X_1)$. We first show this follows from the following two equations:

$$H(X_1 | T_2) = H(X_1) \tag{3}$$

$$H(X_1 | T_2, T_4) = 0 \tag{4}$$

To see this, we apply the chain rule for Shannon entropy:

$$H(T_4) \geq H(T_4 | T_2) + H(X_1 | T_2, T_4) = H(X_1, T_4 | T_2) \geq H(X_1 | T_2) = H(X_1)$$

We now show each claim separately:

1. Perfect malicious security implies there is a simulator for a corrupt P_2 that plays honestly. The messages created by the simulation are distributed exactly as in a real execution. However, while simulating the input phase, the simulator does not have access to the output, and hence has no information on X_1 . It follows that $H(X_1 | T_2) = H(X_1)$.
2. Suppose for the sake of contradiction that X_1 is not determined by T_2, T_4 . This means there must exist (at least) two different executions of the input phase where P_1 has different inputs, but the messages seen by P_2, P_4 are the same. More formally, there exist sets of values of (T_1, T_2, T_3, T_4) , say (t_1, t_2, t_3, t_4) and (t'_1, t_2, t'_3, t_4) both with non-zero probability where the first case can occur with $X_1 = x_1$ and the second with $X_1 = x'_1$, where $x_1 \neq x'_1$. We define a value e such that $x_1[e] \neq x'_1[e]$. Now consider the following two attacks on the input phase, represented by environments Z, Z' :
 - a. Z chooses inputs x_1, x_2, x_3, x_4 for the respective parties, corrupts P_3 , but lets her play honestly in the input phase. If at the end of the input phase P_3 obtains transcript $T_3 = t_3$, she will pretend that she saw $T_3 = t'_3$ instead. She samples random coins r'_3 consistent with t'_3 and completes the protocol honestly, assuming that her view of the input phase was (x_3, r'_3, t'_3) . In the last phase, Z sets the inputs u_i in some fixed way such that $e = \sum_i u_i \bmod L$.
 - b. Z' chooses inputs x'_1, x_2, x_3, x_4 for the respective parties, corrupts P_1 , but lets her play honestly in the input phase. If at the end of the input phase P_1 obtains transcript $T_1 = t'_1$, she will pretend that she had x_1 as input and saw $T_1 = t_1$ instead. She samples random coins r_1 consistent with t_1 and completes the protocol honestly assuming her view of the input phase was (x_1, r_1, t_1) . In the last phase, Z sets the inputs u_i in some fixed way such that $e = \sum_i u_i \bmod L$.

We can now observe that when the real protocol executes in the first attack, with non-zero probability, it is the case that P_1 has input x_1 and transcripts t_1, t_2, t_3 and t_4 were produced in the input phase. Likewise in the second attack it may happen that P_1 received input x'_1 and transcripts t'_1, t_2, t'_3 and t_4 were produced in the input phase.

Assuming these events, we see that the protocol execution after the input phase will be the same in the two scenarios: in both cases the players will do the last part of the protocol honestly starting from views $(x_1, r_1, t_1), (x_2, r_2, t_2), (x_3, r'_3, t'_3), (x_4, r_4, t_4)$, where all random coins are uniform, given the corresponding transcripts. Since these views are identically distributed in the two cases and the inputs chosen in the last phase are the same, the same output distribution D is generated in both cases.

Now consider the simulation of the two attacks. Note that the ideal functionality always computes the output from x_1, x_2, x_3, x_4 in the first case, and from x'_1, x_2, x_3, x_4 in the

second, by Lemma 10. This means that the output is $x_1[e]$ in the first case and $x'_1[e]$ in the second. Assume without loss of generality that $x_1[e] = 0$ and $x'_1[e] = 1$.

On the other hand, we have just seen that the real protocol may sometimes generate output distribution D under both the first and the second attack. Clearly, the probability that D outputs 0 is non-zero, or the probability of output 1 is non-zero. Assume the second case, without loss of generality.

Now, Z can break perfect security: if it sees output 0, it guesses that it has been talking to the simulation, and otherwise it guesses that it is in the real case. Clearly Z will always guess simulation in the ideal (simulated) case but will guess real with non-zero probability in the real case, contradicting perfect indistinguishability. ◀

► **Remark 12.** We can now explain why it is not clear that our proof technique would work if we had used the standard functionality for secure function evaluation where all inputs are given in one go: In order to show that the input phase can produce the same state for the protocol from both input x_1 and x'_1 for P_1 , we need to restrict to a particular subset of the transcripts that might occur. But if that same phase also includes provision of the inputs u_i and perhaps the computation of u , the possible values of u might be similarly restricted, so it is not clear that the environment can still choose the index e so that it will “catch” the difference between x_1 and x'_1 .

3.2 Lower bound, malicious security, n parties, maximal threshold

We now show that the bound generalizes to multiple parties. Let $n = 3t + 1$ and denote the parties by $P_{1,1}, \dots, P_{1,t}, P_{2,1}, \dots, P_{2,t}, P_{3,1}, \dots, P_{3,t}, P_4$. Define by $IP_{I,n}$ the following functionality: each party first provides an I -bit input. When all inputs have been received they are concatenated to form \mathbf{X} , then each party provides number $u_i \in Z_L$ where L is the length of \mathbf{X} . We set $u = \sum_i u_i \bmod L$ and (u, \mathbf{X}_u) is returned.

► **Lemma 13.** $IP_{I,n}$ can be computed by a circuit C with $O(nI)$ gates.

Proof. Let $S = nI$ be input size and assume for simplicity that $S = 2^k$ is an exact power of two. We assume the circuit takes $O(\lg S)$ additional bits which we will denote by \mathbf{r} , it corresponds to the index u above. Strictly speaking, we should take the u_i as input and compute their sum modulo L , but the size of the circuit for doing this is insignificant, it is clearly $o(nI)$ for all large enough I . We now proceed using induction in k :

- Base-case $k = 0$: the circuit simply outputs its input bit. This is clearly uniform in the input.
- Induction $k > 0$: we may split the input into two 2^{k-1} sized halves \mathbf{X}_0 , and \mathbf{X}_1 . By induction there are circuits C_0, C_1 each with $O(2^{k-1})$ gates computing $\mathbf{X}_0, \mathbf{X}_1$, let y_0, y_1 be the output gates. It suffices to combine C_0, C_1 using a constant number of gates. We now construct the circuit $y = (y_0 \wedge \overline{\mathbf{r}_k}) \vee (y_1 \wedge \mathbf{r}_k)$: this takes at most four gates which is clearly constant. In addition both C_0, C_1 choose their elements uniformly at random: if \mathbf{r}_k is indeed a random bit then y is also uniform.

The result now follows since $t = \Theta(n)$. ◀

► **Lemma 14.** Any reactive protocol that realizes $IP_{I,n}$ with perfect malicious security must have total average communication $\Omega(ntI)$.

Proof. Consider any party P . We may group the remaining $3t$ parties arbitrarily into 3 groups, each consisting of t parties to produce a functionality equivalent to $F_{f_{t,4}}$ where P plays the role of P_4 . Corrupting any party in the 4-party case corrupts at most t parties in

$\text{IP}_{I,n}$, and the inputs of a t party group are formed by combining the inputs of the individual players in the group (using concatenation or addition modulo L). By this translation, the player P_1 in the 4-player setting has an input of length tI in the first phase, and hence, by Theorem 11, P must communicate at least $\Omega(tI)$ bits. We can apply this argument to each of the $3t + 1$ parties and add their resulting communications. It should be noted that this counts every bit *twice*: once at the sender, and once at the receiver, however this has no effect on the asymptotic complexity. We conclude the total average communication is $\Omega(ntI)$ bits. ◀

► **Theorem 15.** *There is a (family of) Boolean circuit(s) C with g gates such that any reactive n -party protocol that computes C with perfect malicious security must use total communication $\Omega(ng)$.*

Proof. Follows immediately from Lemmas 13 and 14 since $t = \Theta(n)$. ◀

3.3 Lower bound, malicious security, n parties, submaximal threshold

In this section we consider the case where t is submaximal, i.e. $n = 3t + s$ for some integer $s > 0$.

► **Theorem 16.** *There is a Boolean circuit C with g gates such that any reactive n -party protocol that computes C with perfect malicious security where $n = 3t + s$ for some $s > 0$, and t is the number of corruptions, must use total communication $\Omega(ng/s)$.*

Proof. By Lemma 13 it suffices to show a total communication lower bound of $\Omega(ntI/s)$. Consider any partition of the $2t + s$ honest parties into sets of size s . For simplicity assume s divides $2t + s$ so that any such partition consists of exactly $2t/s + 1$ sets. We may group each set of s honest parties into a single party which we will call P_4 . The remaining $3t$ parties may be arbitrarily grouped together into 3 groups of t parties each. This immediately gives a protocol for $\text{IP}_{tI,4}$ where Theorem 11 applies, meaning P_4 must communicate $\Omega(tI)$. Since each set of k honest parties are disjoint we may add their communications together to get the total communication up to a constant factor. There are $2t/s + 1$ such sets giving a total communication of $(2t/s + 1)\Omega(tI) = \Omega(ntI/s) = \Omega(ng/s)$. ◀

3.4 Lower bound, malicious security, arithmetic circuits

The argument presented in previous sections only considered Boolean circuits, however the same argument applies to arithmetic circuits. Let \mathbb{F} be a finite field whose elements require κ bits to describe. The exact same line of reasoning applies with the difference that $H(X_1) = \kappa I$ instead of $H(X_1) = I$. This increases the bounds by a factor of κ showing the following:

► **Theorem 17.** *There is an arithmetic circuit C with elements of size κ with g gates such that any reactive n -party protocol that securely computes C where $n = 3t + s$ for some $s > 0$, and t is the number of corruptions, must use total communication $\Omega(ng\kappa/s)$.*

3.5 Weakening the assumptions

Instead of assuming UC security we can instead make do with much weaker assumptions in order to show our lower bounds: What we can assume is a two-phase protocol as defined before, but with much weaker demands on the simulator than what we need for UC security, as we now sketch:

- The protocol in question can be split in two phases: we call the first one the input phase and the second the computation phase.
- The simulator first simulates the input phase and then the computation phase. It may rewind the adversary during both phases, but once it has started simulating the computation phase, it is not allowed to rewind back to the input phase.
- Once the simulator starts simulating the computation phase, and the functionality has received all the inputs, the simulator may now ask for the outputs (so this means it cannot ask for the output during the input phase).

It is not hard to see that our lower bound proofs go through, also in this model.

3.6 Lower bound, malicious security, statistical security

The lower bound presented above crucially relied on perfect security of the underlying protocol. In this section we briefly sketch where the lower bound for four parties breaks down in the case of statistical security. We show how the four parties may compute the function $IP_{I,4}$ in a way where P_4 has a communication complexity of $O(\text{poly}(n))$. In particular, the communication complexity of P_4 is independent of I , the input size.

It is well-known that we can compute any circuit with statistical security in an honest majority setting given access to a broadcast channel. We will then let P_1, P_2, P_3 run such a protocol, letting P_4 assist only in the broadcasts (since $t < n/3$ is required for broadcast). Specifically, P_4 produces a VSS of her input and broadcasts to the other parties, who then compute a VSS of P_4 's output and sends back. We use the protocol by [1]. We denote by $X + Y \cdot \mathcal{BC}$ a communication complexity of X bits, and Y bits for broadcast.

► **Theorem 18** (Ben-Sasson, Fehr, Ostrovsky). *Let C be a Boolean circuit with g gates. Then there is a statistically secure MPC protocol (with security parameter κ) for computing C with communication complexity $O((n \lg n)g) + O(n^3 \kappa) \cdot \mathcal{BC}$.*

The communication required for P_4 is dominated by the cost of doing broadcasts, which in particular is independent of I , the input size. This means the lower bound of Theorem 1 does not apply in the statistical setting, even without a broadcast channel. Interestingly, this suggests a “gap” between the two worlds.

We now show various upper bounds and compare them to the corresponding lower bounds. In most cases we are able to match the lower bounds up to a constant factor, however there is a gap of $O(\lg n)$ in the case of “unshaped” Boolean circuits, resulting from the fact that we need $> n$ evaluation points to do secret sharing.

3.7 Upper bound, malicious security, arithmetic circuits

For arithmetic circuits over large fields the parties can secret share their inputs and compute the circuit using Beaver triples. A recent protocol by [14] gives a protocol that is not dependent on the depth of the circuit being computed:

► **Theorem 19** (Goyal, Liu, Song). *If C is an arithmetic circuit with g gates over a field \mathbb{F} with $|\mathbb{F}| > n$, and κ is the size of field element, then there is a perfect maliciously secure protocol for computing C using $O(n g \kappa + n^3 \kappa)$ bits of communication.*

This shows that our lower bound of $\Omega(n g \kappa)$ is tight wrt. the circuit size for arithmetic circuits over fields of sufficient size. It also shows that our lower bound is the best generic lower bound one can hope to prove.

3.8 Upper bound, malicious security, $IP_{I,n}$

The protocol from [14] is based on secret sharings and as a result requires fields with a size greater than the number of players, i.e. it must be the case that $|\mathbb{F}| > n$. This is because n distinct evaluation points are needed for the secret sharing. For smaller fields this is usually remedied by mapping elements into an extension field \mathbb{K} and performing the secret sharings there. This unfortunately incurs an overhead of $O(\lg n)$ compared to our lower bound.

To remedy this for our specific function $IP_{I,n}$ we can use *reverse multiplication friendly embeddings* (RMFE) following the work of [4]. An RMFE allows us to evaluate multiple small circuits in an extension field in parallel with good amortization in the communication.

► **Definition 20.** *Let \mathbb{F} be a finite field. A k -RMFE scheme (ϕ, ψ) consists of two \mathbb{F} -linear mappings, $\phi : \mathbb{F}^k \rightarrow \mathbb{K}$, and $\psi : \mathbb{K} \rightarrow \mathbb{F}^k$ where for any vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^k$ it holds that:*

$$\psi(\phi(\mathbf{a}) \cdot \phi(\mathbf{b})) = \mathbf{a} * \mathbf{b}$$

where $*$ is the coordinate-wise (Schur) product. This allows us to perform k parallel multiplications in \mathbb{F} using a single multiplication in \mathbb{K} . Using an RMFE scheme, [4] construct a protocol for Boolean circuits with an amortized communication complexity of $O(n)$ per multiplication gate:

► **Theorem 21** (Cascudo et al.). *There is a secure n -party protocol for computing $\Omega(\lg n)$ parallel evaluations of a Boolean circuit with an amortized communication complexity of $O(n)$ per multiplication gate.*

We now establish an upper bound for our functionality $IP_{I,n}$:

► **Theorem 22.** *There is a perfect maliciously secure protocol based on secret sharing for computing $IP_{I,n}$ using $O(n^2 I)$ bits of communication.*

Proof. Let C be the circuit described in Lemma 13. Assume for simplicity that $nI = 2^k$ and let $u = \Theta(k)$ be the the number of bits required to describe an element in \mathbb{K} . At a high level our strategy is to compose C into smaller circuits for which we get good amortization. The resulting computation is then computed without embeddings, in the hope that so much work was saved by parallelization that the remaining computation is asymptotically small.

The protocol is parameterized by an integer i that denotes the *depth* at which C is composed into smaller circuits: the parties first invoke the protocol from [4] until all but the last i layers remain, and then ignore the output reconstruction phase. At this point the parties have secret sharings of an element $w \in \mathbb{K}$ that encodes all 2^i wire values. The next step is extracting secret shares of each wire value. To do so, the parties generate sharings of random bits $[r_1], \dots, [r_u]$, encoding an element $[r]$ for some random $r \in \mathbb{K}$. To do this, each party contributes a random bit $[b]$ which are XORed together. To verify that the parties actually input bits, a public opening of $b^2 - b$ is produced and verified to equal 0 (as the only roots are 0 and 1). Next the parties compute $w - r$ and open the result in public. The result is added to $[r]$ to get a sharing $[w]$. Linearity of the secret sharing implies the parties may apply ψ locally to get a secret sharing of each wire value. Finally the parties invoke the protocol [7] on the shares obtained on the rest of the circuit.

Let $i = \Theta(\lg n)$ and let us analyze the communication complexity. It is clear that the cost is dominated by the first phase since the remaining two steps do not depend on I . It is also clear that the size of the circuit is $\Theta(nI)$ since there are nI inputs. By Theorem 21 the complexity of the first phase is $O(n) \cdot nI = O(n^2 I)$ as we wanted to show. ◀

3.9 Upper bound, malicious security, submaximal threshold

Both of the previous upper bounds assumed a maximal threshold of $n = 3t + 1$. In this section we briefly consider the case of submaximal threshold, i.e. where $n = 3t + s$ for some $s > 1$. In this setting we can use *packed secret sharing* to “pack together” s shares into a single element, allowing us to evaluate multiple gates in parallel and saving a factor $O(s)$ in communication. This matches the submaximal lower bound shown in this paper up to a constant factor. This shows that packed secret sharing is the best kind of optimization in terms of communication one could hope to achieve.

4 Conclusion and future work

In this paper we showed two classes of lower bounds for information-theoretic multiparty computation. For the case of active security, we have show the bound for a reactive functionality. It remains open whether a similar bound can be shown for (non-reactive) secure function evaluation.

References

- 1 Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 663–680, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- 2 Carlo Blundo, Alfredo De Santis, Giuseppe Persiano, and Ugo Vaccaro. Randomness complexity of private computation. *Computational Complexity*, 8(2):145–168, 1999.
- 3 R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001. doi:10.1109/SFCS.2001.959888.
- 4 Ignacio Cascudo, Ronald Cramer, Chaoping Xing, and Chen Yuan. Amortized complexity of information-theoretically secure mpc revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 395–426, Cham, 2018. Springer International Publishing.
- 5 Benny Chor and Eyal Kushilevitz. A communication-privacy tradeoff for modular addition. *Inf. Process. Lett.*, 45(4):205–210, 1993.
- 6 Ivan Damgård, Kasper Green Larsen, and Jesper Buus Nielsen. Communication lower bounds for statistically secure mpc, with or without preprocessing. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 61–84. Springer, 2019.
- 7 Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 572–590. Springer, 2007.
- 8 Ivan Damgård, Jesper Buus Nielsen, Rafail Ostrovsky, and Adi Rosén. Unconditionally secure computation with reduced interaction. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 420–447. Springer, 2016.
- 9 Ivan Damgård, Jesper Buus Nielsen, Antigoni Polychroniadou, and Michael A. Raskin. On the communication required for unconditionally secure multiplication. In *CRYPTO (2)*, volume 9815 of *Lecture Notes in Computer Science*, pages 459–488. Springer, 2016.
- 10 Deepesh Data, Manoj M. Prabhakaran, and Vinod M. Prabhakaran. On the communication complexity of secure computation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014*, pages 199–216, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
- 11 Uri Feige, Joe Killian, and Moni Naor. A minimal model for secure computation (extended abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, STOC '94*, page 554–563, New York, NY, USA, 1994. Association for Computing Machinery. doi:10.1145/195058.195408.

- 12 Matthew Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '92, page 699–710, New York, NY, USA, 1992. Association for Computing Machinery. doi:10.1145/129712.129780.
- 13 Anna Gal and Adi Rosen. Lower bounds on the amount of randomness in private computation. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, page 659–666, New York, NY, USA, 2003. Association for Computing Machinery. doi:10.1145/780542.780638.
- 14 Vipul Goyal, Yanyi Liu, and Yifan Song. Communication-efficient unconditional mpc with guaranteed output delivery. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 85–114, Cham, 2019. Springer International Publishing.
- 15 Eyal Kushilevitz and Yishay Mansour. Randomness in private computations. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '96, page 181–190, New York, NY, USA, 1996. Association for Computing Machinery. doi:10.1145/248052.248089.
- 16 Eyal Kushilevitz and Adi Rosén. A randomness-rounds tradeoff in private computation. In Yvo G. Desmedt, editor, *Advances in Cryptology – CRYPTO '94*, pages 397–410, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.
- 17 I. Parberry. The pairwise sorting network. *Parallel Process. Lett.*, 2:205–211, 1992.

On Prover-Efficient Public-Coin Emulation of Interactive Proofs

Gal Arnon ✉

Weizmann Institute of Science, Rehovot, Israel

Guy N. Rothblum ✉

Weizmann Institute of Science, Rehovot, Israel

Abstract

A central question in the study of interactive proofs is the relationship between private-coin proofs, where the verifier is allowed to hide its randomness from the prover, and public-coin proofs, where the verifier's random coins are sent to the prover. The seminal work of Goldwasser and Sipser [STOC 1986] showed how to transform private-coin proofs into public-coin ones. However, their transformation incurs a super-polynomial blowup in the running time of the honest prover.

In this work, we study transformations from private-coin proofs to public-coin proofs that preserve (up to polynomial factors) the running time of the prover. We re-consider this question in light of the emergence of doubly-efficient interactive proofs, where the honest prover is required to run in polynomial time and the verifier should run in near-linear time. Can every private-coin doubly-efficient interactive proof be transformed into a public-coin doubly-efficient proof? Adapting a result of Vadhan [STOC 2000], we show that, assuming one-way functions exist, there is no general-purpose black-box private-coin to public-coin transformation for doubly-efficient interactive proofs.

Our main result is a loose converse: if (auxiliary-input infinitely-often) one-way functions do *not* exist, then there exists a general-purpose efficiency-preserving transformation. To prove this result, we show a general condition that suffices for transforming a doubly-efficient private coin protocol: every such protocol induces an efficiently computable function, such that if this function is efficiently invertible (in the sense of one-way functions), then the proof can be efficiently transformed into a public-coin proof system with a polynomial-time honest prover.

This result motivates a study of other general conditions that allow for efficiency-preserving private to public coin transformations. We identify an additional (incomparable) condition to that used in our main result. This condition allows for transforming any private coin interactive proof where (roughly) it is possible to efficiently approximate the number of verifier coins consistent with a partial transcript. This allows for transforming any *constant-round* interactive proof that has this property (even if it is not doubly-efficient). We demonstrate the applicability of this final result by using it to transform a private-coin protocol of Rothblum, Vadhan and Wigderson [STOC 2013], obtaining a doubly-efficient public-coin protocol for verifying that a given graph is close to bipartite in a setting for which such a protocol was not previously known.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Interactive proof systems

Keywords and phrases Interactive Proofs, Computational complexity, Cryptography

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.3

Related Version *Full Version:* <https://eccc.weizmann.ac.il/report/2019/176/>

Funding This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819702).

Acknowledgements We would like to thank Ron Rothblum for making us aware of Kilian's ideas which in turn developed into our piecemeal emulation protocol. We would also like to thank Zvika Brakerski and Moni Naor for helpful comments on the presentation of this work.



© Gal Arnon and Guy N. Rothblum;
licensed under Creative Commons License CC-BY 4.0
2nd Conference on Information-Theoretic Cryptography (ITC 2021).
Editor: Stefano Tessaro; Article No. 3; pp. 3:1–3:15



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Interactive proofs (IPs), introduced by Goldwasser, Micali and Rackoff [11] in 1985 are an important object in the study of complexity and cryptography. An interactive proof is an interactive protocol between two parties, a “prover” and a “verifier”, where the prover is trying to convince the verifier of the membership of a string in a language. If this claim is true, then the verifier should be convinced with high probability. Otherwise, if the claim is false, then no matter what the prover does, the verifier should reject the claim with high probability. Since their inception, a central question in the study of interactive proofs has been the connection between private-coin proofs, where the verifier is allowed to hide its randomness from the prover, and public-coin proofs, where hiding information is not allowed. Public-coin protocols are especially appealing since they are easier to analyze and manipulate [4, 1, 2, 3]. Goldwasser and Sipser [12] showed that any private-coin interactive proof can be transformed into a public-coin proof while preserving the number of rounds (up to an additive constant).

One issue with this transformation is that of the honest prover’s running time. Vadhan [22] showed that (assuming the existence of one-way functions) there exist protocols that cannot be transformed to be public-coin in a black-box manner while preserving the running time of the prover (up to polynomial factors). While in the classical setting the running time of the prover is considered unbounded, the recent line of works on *doubly-efficient* interactive proofs (deIPs) [10] restricts the honest prover to run in polynomial time. We emphasize that soundness is required to hold against computationally unbounded adversaries. Doubly-efficient interactive proofs apply only to tractable computations, and are therefore of interest when the verifier time can be smaller than the time required to decide the language without the help of a prover. Indeed, the main focus in the literature is on verifiers that run in near-linear time. Goldreich [5] gives a survey on recent work on doubly-efficient interactive proofs.

1.1 This Work

In this work we ask whether transformations of proofs from using private coins to using public coins are applicable to the doubly-efficient setting:

Which private-coin doubly-efficient interactive proofs can be transformed into public-coin doubly-efficient proofs and how can this be done?

We tackle the above question from a number of angles. Some of our results also apply to proofs that are not doubly-efficient.

We extend Vadhan’s impossibility result to show that the existence of one-way functions implies that there are no transformations from private-coin deIPs to public-coin deIPs that work in a natural “black-box” way. Note that since deIPs exist only for problems in \mathcal{BPP} , one can *always* transform such proofs to using “public-coins” by having the verifier solve the problem on its own. This transformation is not black-box, but it is also not interesting, as the motivation for deIPs is to reduce the verifier’s running time to under what is required for it to solve the problem on its own.

Our main result shows that this reliance on one-way functions is essentially tight. Namely, if one-way functions (of a certain type) do not exist, then (essentially) every doubly-efficient proof can be efficiently transformed:

► **Theorem 1** (Efficient emulation of deIPs in Pessiland (Informal)). *Suppose that infinitely-often auxiliary-input one-way functions do not exist. Then every language that has a doubly-efficient private-coin interactive proof with “good enough” soundness has a doubly-efficient public-coin interactive proof with the same number of rounds (up to a constant).*

► **Remark 2.** Theorem 1 mentions “good enough” soundness. This is due to the fact that there is a strong degradation in soundness when applying our technique.¹ One could be tempted to amplify the soundness using parallel or sequential repetition, but in the setting of deIPs, the overhead of repeating the protocol in terms of the verifier might be problematic (e.g. it might degrade the verifier’s running-time from linear to quadratic).

Viewing this result through the prism of Impagliazzo’s worlds [15], it (very) roughly says that in Pessiland (a world where one-way functions do not exist), efficiency-preserving transformation is always possible. We prove Theorem 1 by showing that for every deIP there exists a specific efficiently computable function such that if it is efficiently invertible in the sense of one-way functions², then efficiency-preserving transformation is possible (see Section 2.1 for a discussion on the notion of invertibility). We remark that a straightforward implementation of the Goldwasser-Sipser transformation requires exponential running time from the prover, and even an oracle that inverts *any* given function (on random inputs) does not seem sufficient for making their public-coin prover efficient. Indeed, our results require changing the transformation so that the ability to invert becomes sufficient for constructing a public-coin prover.

Using the technique for proving Theorem 1 (and some additional technical work) we show that in Pessiland’s “one-way function”-less landscape, standard constant-round proofs (i.e. ones where the honest prover is allowed to run in super-polynomial time) can also be transformed to be public-coin with only a polynomial overhead on the honest prover’s running time:

► **Theorem 3** (Efficient emulation of constant-round IPs in Pessiland (Informal)). *Suppose that infinitely-often auxiliary-input one-way functions do not exist. Then every language that has a constant-round private-coin interactive proof has a constant-round public-coin interactive proof where the honest prover’s running time is polynomially related to that of the private-coin prover.*

1.1.1 Sufficient conditions for efficient transformation

Both the impossibility result of [22] and our extension to doubly-efficient proofs are proved by demonstrating a specific (arguably contrived) protocol that is hard to transform. It is very natural, then, to ask: considering interactive proofs on a case-by-case basis, for which protocols (or families of protocols) is efficiency-preserving transformation possible? In other words we wish to identify sufficient conditions that allow for efficiency-preserving transformation of private-coin proofs to public-coin ones.

In particular, Theorem 1 implies one such condition: Every deIP has an efficiently computable function such that if this function is efficiently invertible in the sense of one-way functions, then efficient transformation for this proof system is possible.

We identify an additional, rather natural, sufficient condition for efficient transformation. We show that if it is possible to efficiently count the number of coins that are consistent with transcripts of the protocol, then it is also possible to efficiently emulate the protocol using public-coins. Unlike in Theorems 1 and 3, this result does not preserve the number of rounds, but it applies to general interactive proofs (even when the protocol has an inefficient honest prover and a polynomial number of rounds).

¹ Specifically, in order for the public-coin protocol that we end up with to have constant soundness error, the soundness error of the original (private-coin) protocol should be $O(\text{poly}(n, r, \ell)^{-r})$ where n is the input length, and r and ℓ are the number of rounds and number of random bits used by the verifier in the original private-coin protocol respectively.

² The requirement that a specific function is *distributionally* invertible [16] also suffices.

► **Theorem 4** (Efficient emulation using approximation (Informal)). *Let \mathcal{L} be a language and suppose that \mathcal{L} has an r -round private-coin interactive proof with communication complexity m , and suppose that for every incomplete transcript it is possible to efficiently approximate the number of verifier random coins that are consistent with the transcript. Then \mathcal{L} has a $2rm$ -round public-coin interactive proof with an efficient prover.*

A string of random coins ρ is consistent with an incomplete transcript of an execution of a protocol if for every verifier message α in the transcript, the verifier outputs α when given the transcript prefix leading up to α and using ρ as its random coins. We prove this theorem using a “piecemeal” emulation protocol in which the prover and the verifier together generate a string that is distributed according to the distribution of a random transcript in the private-coin protocol. The soundness error of the resulting protocol is a function of how good the approximation algorithm is. In particular, if one can *exactly* count the number of verifier random coins that are consistent with a transcript efficiently, then soundness is perfectly preserved.

Theorem 4 gives us a condition for efficiently transforming proofs from private-coin to public-coin that is incomparable to the condition implied by Theorems 1 and 3. The condition implied by from Theorems 1 and 3 is efficient distributional inversion for some (efficiently computable) function that depends on the protocol, whereas Theorem 4 uses efficient approximation of the number of verifier coins consistent with a transcript. We demonstrate a natural protocol for which the efficient counting condition of Theorem 4 is satisfied, whereas we don’t know how to efficiently invert the function implied by Theorems 1 and 3.

1.1.2 An application

Rothblum, Vandhan and Wigderson [18] show a private-coin proof of proximity for distinguishing between a graph that is bipartite and graphs that are both far from bipartite and well-mixing. Roughly speaking, an interactive proof of proximity (IPP) is an interactive proof where the verifier has sub-linear query access to the input. The problem of distinguishing between a bipartite graph and a well-mixing graph that is far from bipartite has also been studied extensively in the past in the context of property testing [8, 9]. By applying Theorem 4 to the private-coin protocol of [18] we show a new doubly-efficient public-coin proof system for this problem. We describe this below in more detail.

► **Theorem 5** (Public-coin IPP for bipartiteness (Informal)). *For every $\varepsilon > 0$, there exists a public-coin interactive proof of ε -proximity with an efficient prover for the problem of distinguishing between bipartite graphs and graphs that are ε -far from bipartite and are well-mixing.*

We remark that no such proof was previously known (except for ε close to 1).

The main property of the RVW bipartiteness protocol that we use, is the fact that it that the (private coin) verifier uses only a logarithmic number of coins (it has logarithmic randomness complexity), and this suffices for soundness error $1 - \Omega(\varepsilon)$. Thus, polynomial time suffices for enumerating all possible choices of verifier randomness and for exactly counting how many choices are consistent with the transcript. The transformation of Theorem 4 is soundness preserving when efficient exact counting is possible, and so gives an efficient public-coin protocol with soundness error $1 - \Omega(\varepsilon)$, which can be amplified to obtain constant soundness. See the full paper for further discussion and details.

We note that a similar argument applies to every proof system where the verifier's randomness complexity is $O(\log n)$. We show that *every* such private-coin IP (resp. IPP) can be transformed to a public-coin IP (resp. IPP) while the honest prover's running time remains polynomial.

We further note that the Goldwasser-Sipser (GS) approach to transforming private-coin proofs into public-coin ones, and the result behind Theorem 1, can also preserve the prover's efficiency when the verifier's randomness complexity is $O(\log n)$. However, the GS approach degrades soundness significantly, and hence usually requires parallel repetition before applying the transformation. Here, since the starting (private coin) protocol has large soundness error, repeating it to reduce the soundness error to the point where the GS approach can be applied requires super-logarithmic randomness complexity, which means that the GS transformation will not preserve the prover's efficiency. In comparison, in this setting Theorem 4 preserves soundness (see above), and so it can be used even when the soundness error of the private-coin protocol is large.

1.2 Related Work

A number of works have tackled the question of private versus public coins, including Haitner, Mahmoody and Xiao [13] who showed that if the prover is given an \mathcal{NP} oracle it is possible to transform private-coin protocols into public-coin ones where both the prover and the verifier run in polynomial time. Holenstein and Künzler [14] show a public-coin protocol in which the prover helps the verifier sample from a distribution, where in addition to the sampled element the verifier ends up with an approximation of the probability that the element is sampled from the distribution. They then show this can be used for public-coin emulation. Goldreich and Leshkowitz [6] improved upon the soundness requirement of Goldwasser and Sipser. In all of the results described the prover is inefficient and the running time of the verifier incurs a polynomial overhead. We additionally note that the celebrated $\mathcal{IP} = \mathcal{PSPACE}$ Theorem [17, 19], implies a non-black-box transformation of private-coin protocols to public-coin ones. The protocol used to show that $\mathcal{PSPACE} \subseteq \mathcal{IP}$ is public-coin, and so one can use this result to transform private-coin protocols into public-coin ones as follows: Given an interactive proof, use the transformation for $\mathcal{IP} \subseteq \mathcal{PSPACE}$ to convert it into a \mathcal{PSPACE} problem. Then use the reduction and protocol showing that $\mathcal{PSPACE} \subseteq \mathcal{IP}$ to construct a public-coin proof. While this transformation is not black-box, it blows up the complexity of the honest prover and the number of rounds of the protocol.

2 Technical Overview

2.1 Overview of the Round-Efficient Emulation

Goldwasser and Sipser [12] showed not only that it is possible to transform private-coin proofs into public-coin ones, but also that this can be done without significantly increasing the number of rounds in the protocol. We show that, given a distributional inverter for certain functions, the prover can be made efficient. A distributional inverter for a function f is an efficient randomized algorithm that upon receiving an input y drawn from the distribution $f(U_n)$ returns a random element from the set $f^{-1}(y)$.

► **Remark 6.** Proving Theorems 1 and 3 would be significantly simpler if we were to consider a stronger form of inversion, where the input y to the inverter can be any element in the support of f . That is, y need not be given as a sample from the distribution $f(U_n)$. We consider the weaker and more complicated variant, since it allows us to establish Theorem 1, and through it the tight relationship between one-way functions and the (non-)existence of private-coin to public-coin transformations that preserve the prover's running time.

We give three toy cases for protocols of increasing complexity, and then discuss the general case. For each case we show how Goldwasser and Sipser’s original protocol can be applied in order to transform it to public-coin, and then discuss how we use distributional inversion in order to make the prover efficient. Eventually, this requires changes to the Goldwasser-Sipser transformation. In all three toy cases consider a language \mathcal{L} and a one round private-coin protocol denoted $\langle P, V \rangle$ with perfect completeness and soundness error s . Since it has one round, the protocol is of the following form: On input x the verifier begins with choosing a random $\rho \leftarrow \{0, 1\}^\ell$, then sends $\alpha = V(x, \rho)$ where $\alpha \in \{0, 1\}^m$. After receiving β from the prover, the verifier accepts if $V(x, \alpha, \beta; \rho) = 1$. Henceforth throughout this overview we omit the shared input x from notation of the verifier and prover functions.

2.1.1 Case 1: Equally Likely Messages With *Known* Number of Messages

2.1.1.1 The Protocol

In addition to the protocol being one-round and having perfect completeness, we assume the following properties:

- Equally Likely Messages: If x is in the language \mathcal{L} , then every pair of messages $\alpha_1, \alpha_2 \in \{0, 1\}^m$ that have non-zero probability of being sent by the original verifier V are equally likely: $\Pr_{\rho \leftarrow U_\ell} [V(\rho) = \alpha_1] = \Pr_{\rho \leftarrow U_\ell} [V(\rho) = \alpha_2]$.
- Known Number of Messages For Completeness: If $x \in \mathcal{L}$ then there is a known efficiently computable function $N : \{0, 1\}^* \rightarrow \mathbb{N}$ such that the total number of messages sent by the verifier with non-zero probability is $N(x)$. That is,

$$N(x) = |\{\alpha | \exists \rho \in \{0, 1\}^\ell \text{ s.t. } V(x; \rho) = \alpha\}|$$

- Few Messages For Soundness: If $x \notin \mathcal{L}$ then there are significantly fewer than $N(x)$ verifier messages.

As a running example for this case one is encouraged to think of the classical private-coin protocol for the graph non-isomorphism problem [7]. In this language the input is comprised of two n -vertex graphs G_0 and G_1 which are claimed to be non-isomorphic. The protocol is as follows: the verifier chooses a random bit b , and random permutation π . It sends $\tilde{G} = \pi(G_b)$ to the prover who must return some b' . The verifier accepts if $b' = b$. One can easily verify that this protocol has completeness 1 and soundness error $\frac{1}{2}$. Moreover, assuming for simplicity that the graphs have no automorphisms, every verifier message is equally likely and if the graphs are non-isomorphic the number of possible verifier messages is $N = 2n!$. If the graphs are isomorphic then there are only $n!$ different messages.

2.1.1.2 The Goldwasser-Sipser Transformation

The transformation of $\langle P, V \rangle$ into a public-coin protocol hinges on the observation that, in this toy case, in order distinguish whether x is in the language, the prover need only show that the number of possible verifier messages is at least N (since by assumption if $x \notin \mathcal{L}$ there are significantly fewer such messages). Thus a (public-coin) “set lower-bound” protocol is used, showing that the set of all valid verifier messages (ones that are sent by V with non-zero probability over the choice of its random coins) is large. Letting $\mathcal{H}_{m,k}$ be a family of pairwise independent hash functions from $\{0, 1\}^m$ to $\{0, 1\}^k$, U_k be the uniform distribution over k bits and $k = k(N)$ be a value to be discussed later, the final protocol is as follows:

1. The parties execute a “set lower-bound” protocol proving that the number of verifier messages is at least N :
 - a. The verifier chooses a random $h \leftarrow \mathcal{H}_{m,k}$ and $y \leftarrow U_k$ ³
 - b. The prover returns $\alpha \in \{0, 1\}^m$.
 - c. The verifier tests that $h(\alpha) = y$ and otherwise rejects.
2. The prover sends $\rho \in \{0, 1\}^\ell$.⁴
3. The verifier accepts if $V(\rho) = \alpha$.

In the case of completeness, where there are N verifier messages, if k is “small enough” (i.e. the hash function is very compressing) it is likely that there exists some valid message α that hashes to y . Conversely, in the case of soundness, where there are significantly fewer than N legal verifier messages, if k is “large enough” then it is unlikely that there will exist a valid message that hashes to y . Thus, completeness and soundness of the protocol are governed by setting k to a reasonable value, which depends on the gap between N and the magnitude of the prover’s lie in the case of a false claim. Note that in this protocol the prover did not even need to send its message β - it was sufficient to use the fact that there is a large gap in the number of verifier messages between the cases of completeness and soundness. The sub-protocol executed in Step 1 is known as the “set lower-bound” protocol, and can be generalized to show a lower-bound on the size of any set S for which the verifier can efficiently test membership. Specifically, the protocol begins with a claim that $N \leq |S|$ and ends with both parties holding an element x for which the verifier needs to verify that it belongs to S . If $N \leq |S|$, then $x \in S$ with high probability, and if $|S| \ll N$, $x \notin S$ with high probability regardless of the prover strategy. A protocol inspired by the set lower-bound protocol is presented and analysed in the full paper under the name “prover-efficient sampling protocol”. Going back to the example of graph non-isomorphism, upon receiving h, y from the verifier, the prover would send some graph \tilde{G} , a bit b and a permutation π . The verifier would then accept if $h(\tilde{G}) = y$ and $\tilde{G} = \pi(G_b)$.

2.1.1.3 Prover Efficiency

The prover strategy in the above protocol is inefficient. It receives some h, y and is required to find a legal message α that hashes to y and some choice of randomness ρ that leads to α . However, the prover can be made efficient by giving it oracle access to an inverter for the function $f(h, \rho) = h, h(V(\rho))$. An inverter for a function $f : \{0, 1\}^a \rightarrow \{0, 1\}^b$ is a randomized algorithm that on input y drawn from the distribution $f(U_a)$ returns some element x in the set of preimages of y under f (that is, $x \in f^{-1}(y)$). We stress that the input to the inverter must come from the correct distribution, which in our case is $f(\mathcal{H}_{m,k}, U_\ell) \equiv (\mathcal{H}_{m,k}, \mathcal{H}_{m,k}(V(U_\ell)))$. The hash function h is clearly chosen by the verifier from the correct distribution. The image y is drawn by the verifier from the uniform distribution which, if the hash function is compressing enough, will be statistically close to $h(V(U_\ell))$ by the Leftover Hash Lemma. Preimages of (h, y) with respect to f are of the form (h, ρ) where $h(V(\rho)) = y$. The prover can send ρ and use ρ to calculate α . In all of this the prover only needs to make a single oracle call, and to calculate $\alpha = V(\rho)$, and is therefore efficient.

³ In the classic transformation it suffices to set $y = 0^k$ and is described here thus as it will be required later by our transformation.

⁴ The final prover message can be merged with the previous one to save a round and is described here as a separate round for clarity.

2.1.2 Case 2: Equally Likely Messages With *Unknown* Number of Messages

2.1.2.1 The Protocol

We make the same assumptions on the protocol as in Case 1, except that the verifier in the transformation does not know N , the number of verifier messages.

2.1.2.2 The Goldwasser-Sipser Transformation

The protocol is based on two observations made in the case of a cheating prover:

- Since the soundness error is s and the verifier uses ℓ random coins, the number of verifier messages α for which there exist β and ρ such that $V(\alpha, \beta; \rho)$ accepts is at most $s \cdot 2^\ell$.
- For every fixed α and β , the number of coins ρ such that $V(\rho) = \alpha$ and $V(\alpha, \beta; \rho)$ accepts is also bounded from above by $s \cdot 2^\ell$.

If either of the above were not true, it would mean the soundness error is greater than s . Now notice that since all messages are equally likely, if there are N valid verifier messages, then each message has $\frac{2^\ell}{N}$ different coins that are consistent with it. Importantly, if N is small, $\frac{2^\ell}{N}$ is large. This gives rise to the following protocol:

1. Prover sends N , a claim on the number of verifier messages.
2. Prover and verifier execute the set lower-bound protocol to show that the number of legal verifier messages is at least N . The parties end up with some α claimed to be a verifier message.
3. Prover sends some β .
4. The parties execute the set lower-bound protocol to show that the number of coins that are consistent with α and lead the verifier to accept (α, β) is at least $\frac{2^\ell}{N}$. The parties end up with a ρ which is supposed to be in the set of random coins that lead the verifier to α .
5. Verifier accepts if $V(\rho) = \alpha$ and $V(\alpha, \beta; \rho) = 1$.

The protocol is complete, since if the prover is honest it is likely to succeed in both the set lower-bound protocols, meaning it samples both a valid message α and valid coins ρ such that $V(\rho) = \alpha$ and $V(\alpha, \beta; \rho) = 1$. We now turn towards soundness. Let S be the set of verifier messages for which the prover has an accepting strategy (messages α for which there exist β and ρ such that $V(\alpha, \beta; \rho)$ accepts). For verifier message α and prover message β , let $T_{\alpha, \beta}$ be the number of random coins ρ such that $V(\alpha, \beta; \rho)$ accepts. Recall from the argument above, that $|S| \leq s \cdot 2^\ell$ and for any α, β , $|T_{\alpha, \beta}| \leq s \cdot 2^\ell$. Now note that if the verifier ends up with a verifier message $\alpha \notin S$ it has a message for which no fixing of β and ρ will make the verifier accept. Thus the prover must try to sample in S . Similarly, after fixing α and β if the verifier has $\rho \notin T_{\alpha, \beta}$ it will reject, and so the cheating prover must try to cause the verifier to end up with an element in $T_{\alpha, \beta}$. To see why the protocol is sound consider the prover's choice of N . If $|S| \leq s \cdot 2^\ell \ll N$, then due to the set lower-bound protocol the prover is unlikely to make the verifier sample from S and so the verifier will reject with high probability. If N is small, then $\frac{2^\ell}{N}$ is large. Fixing α and β , if $|T_{\alpha, \beta}| \leq s \cdot 2^\ell \ll \frac{2^\ell}{N}$, then the verifier is unlikely to end up with such coins, meaning it rejects. Note in the above analysis we have that $s \ll \min\{N, \frac{2^\ell}{N}\}$. Thus if s is small enough to begin with, the prover will be forced to lie and be caught with high probability.

2.1.2.3 Prover Efficiency

Inspecting the above protocol there are three things that the honest prover needs to do: Count N , the number of verifier messages, execute the prover's side of the set lower-bound protocol to show that there are at least N verifier messages, and execute the prover's side of

the set lower-bound protocol to prove that the number of coins that are consistent with α is at least $\frac{2^\ell}{N}$. We explain how to compute each of these efficiently in reverse order:

1. **Set Lower-Bound Protocol for Number of Consistent Coins:** In this set lower-bound protocol the parties have already computed a verifier message α . The prover receives some hash function h and an image y from the verifier, and must return some ρ such that $V(\rho) = \alpha$ and $h(\rho) = y$. Naively it seems that this can be solved simply if the prover has access to an inverter for the function $f(h, \rho) = h, V(\rho), h(\rho)$ since preimages of (h, α, y) are exactly of the form h', ρ such that $h', V(\rho), h'(\rho) = h, \alpha, y$. The problem with this idea is that to use this inverter it must be that the message α be drawn from the distribution $V(U_\ell)$. Thus to use this idea it is imperative that α come from the correct distribution.
2. **Set Lower-Bound Protocol for Number of Messages:** In order to complete this stage, the prover must find some valid verifier message α that hashes to y , and this (as we did in Case(1)) can be done using an inverter for the function $f(h, \rho) = h, h(V(\rho))$. Unfortunately as mentioned in point (1), we need it α to be chosen from the real message distribution $V(U_\ell)$. Unfortunately using an inverter as described, the message α might be drawn from a distribution which is far from the real one.⁵ This issue is fixed if we move from using a regular inversion oracle to a *distributional* inversion oracle. Roughly, a distributional inverter for a function $f : \{0, 1\}^m \rightarrow \{0, 1\}^t$ is a randomized algorithm A such if y is drawn from $f(U_m)$, the distribution $A(y)$ is statistically close to a random pre-image of y under f .
3. **Computing N :** We show that given an inversion oracle for the function $f(h, \rho) = h, h(V(\rho))$ it is possible to efficiently approximate N , the number of verifier messages. The way this is done is inspired by the techniques of [20] for approximate counting using an \mathcal{NP} oracle. The prover chooses $h \leftarrow \mathcal{H}_{m,k}$ and $y \leftarrow \{0, 1\}^k$ for increasingly larger values of k , and calls the inversion oracle on input (h, y) . When k is small relative to the number of verifier messages, there will likely exist a message α that hashes to y , and thus the inverter will return a set of coins ρ such that $V(\rho) = \alpha$. Once k is set to a relatively large value this is unlikely and the inverter will fail. Thus, given the smallest size of k for which the inverter fails the prover can estimate the size of the set.

2.1.3 Case 3: A Two-Cluster Protocol

2.1.3.1 The Protocol

As in Case 2, we do not assume that the parties know initially the number of verifier messages. Moreover we replace the assumption that all messages are of equal likelihood with the following one:

- **Two Clusters:** The verifier messages that have non-zero probability can be partitioned into two “clusters” C_0 and C_1 , where every message in C_b has equal likelihood p_b . Furthermore, each message in C_1 is significantly more likely than messages in C_0 : $p_0 \ll p_1$. Both parties know the values p_0 and p_1 but not $|C_0|$ and $|C_1|$.

⁵ Indeed, suppose that the inverter always returns the lexicographically smallest ρ such that $h(V(\rho)) = y$. This will cause a sampling bias towards those messages that have coins that are lexicographically smaller.

2.1.3.2 The Goldwasser-Sipser Transformation

Rather than giving a claim about the number of possible messages, the prover will claim only that the heaviest of the flat clusters is large (the weight of cluster C_b is $p_b \cdot |C_b|$).

1. Prover calculates $|C_0|$ and $|C_1|$ and chooses a bit b such that $p_b \cdot |C_b| \geq p_{1-b} \cdot |C_{1-b}|$. It sends b and $N = |C_b|$.
2. Verifier tests that $p_b \cdot N \geq \frac{1}{2}$ and otherwise rejects.
3. Prover and verifier execute the set lower-bound protocol to show that the size of cluster C_b is at least N . The parties end up with some α claimed to be in cluster b .
4. Prover sends some β .
5. The parties execute the set lower-bound protocol to show that the number of coins that are consistent with α and lead the verifier to accept (α, β) is at least $p_b \cdot 2^\ell$. The parties end up with a ρ which is supposed to be a set of random accepting coins that lead the verifier to α .
6. Verifier accepts $V(\rho) = \alpha$ and $V(\alpha, \beta; \rho) = 1$.

Completeness can be verified by noting that since there are only two clusters, the heaviest cluster must hold at least half of the weight of the distribution and so the verifier's test that $p_b \cdot N \geq \frac{1}{2}$ will pass. Due to the fact that every message α in C_b has likelihood p_b , there are exactly $p_b \cdot 2^\ell$ different coins that would lead the verifier to output α and to accept. For soundness first fix b . Once b is fixed, the smallest the prover can set N to be is $\frac{1}{2p_b}$ because otherwise the verifier will reject in Step 2. If p_b is very small, this value is large. As in the analysis of Case 2, since the total number of verifier messages for which the prover as an accepting strategy is small, if the claim N is large, it will likely not manage to make the verifier accept. If p_b is large, then the value of N can be set to be small, but in this case the value $p_b \cdot 2^\ell$ in Step 5 is large. Noting that for any α and β the number of coins ρ for which $V(\alpha, \beta; \rho)$ accepts is at most $s \cdot 2^\ell$, which we think of as very small, the prover is unlikely to be able to cause the verifier to end up with coins that will make it accept.

2.1.3.3 Prover Efficiency

In the classic transformation as described above the prover must do three things: Calculate the size of each cluster, and take part in both executions of the set lower-bound protocol.

1. Counting $|C_0|$ and $|C_1|$: The approximate counting technique as used in Case 2 to approximate N can be used to approximate the number of coins which would lead to a message. Notice that for $\alpha \in C_b$ there are exactly $p_b \cdot 2^\ell$ coins that lead to α . Thus, by randomly choosing many messages and counting how many are in each cluster, the prover can build a "histogram" of the weights of each cluster - a list of each cluster and its respective weight. This is formally addressed in the full paper where it is shown that using both a sampling and a membership oracle one can build such a histogram. An issue with this approach is that the approximation procedure returns an *approximate* value for the number of coins that lead to a message. That is, for a message in C_b it may claim that the number of coins that lead to the message is anywhere in the range $(1 \pm \varepsilon) \cdot p_b \cdot 2^\ell$ for some (relatively small) ε . Since p_0 and p_1 are far from each other, the ranges $(1 \pm \varepsilon) \cdot p_0 \cdot 2^\ell$ and $(1 \pm \varepsilon) \cdot p_1 \cdot 2^\ell$ do not intersect. Thus it is still easy to recognize to which cluster a message belong.
2. Set Lower-Bound Protocol for Number of Messages in C_b : In this part of the protocol the prover receives a hash function h and image y and needs to return a message α such that $\alpha \in C_b$ and $h(\alpha) = y$. The prover cannot simply use an inverter for a function that samples inside of C_b since there may not be an efficient function for sampling in C_b . We

instead show that given a distributional inverter for $f(h, \rho) = h, h(V(\rho))$ it is possible to find preimages that are in C_b . This is true because the cluster has significant weight with respect to the distribution of messages, and so for a randomly chosen hash function and random image y the weight of elements that belong to the cluster and are preimages to y is unlikely to be very small relative to all other preimages of y .

3. Set Lower-Bound Protocol for Number of Consistent Coins: The prover's strategy can be made efficient in this protocol in exactly the same way as in the same set lower-bound in the previous case: by inverting $f(h, \rho) = h, V(\rho), h(\rho)$. Doing this has the same issue dealt with in Case 2 - the distribution from which the message α is drawn must be close to uniform. In the classical transformation the value of α will be far from random because the protocol always works with the heaviest cluster. Suppose, for example, that $p_0 \cdot |C_0| = p_1 \cdot |C_1| + \varepsilon$ for a very small ε . Then C_0 will always be chosen, even though in the real message distribution the likelihood of being in each cluster is almost identical. In order to fix this skew in distribution we make the choice of b "smoother". Rather than setting b as the index of the heaviest cluster, we let b be random and sampled from the Bernoulli distribution where 0 is drawn with probability $\frac{p_0 \cdot |C_0|}{p_0 \cdot |C_0| + p_1 \cdot |C_1|}$. This presents a minor issue: now it could be that $p_b \cdot N < \frac{1}{2}$. To fix this, we limit the choice of clusters only to ones that have some noticeable probability of appearing, and so the verifier can make sure that the claimed probability is not smaller than this threshold. Similar smoothing techniques were used in [14] and [6] in different contexts.

2.1.4 Towards the General Case

In the general case, the verifier message distribution cannot be split into a small number of flat clusters and the protocol may have multiple rounds. To keep this overview simple we only consider doubly-efficient proofs. Making the transformation work for constant-round proofs with an inefficient prover requires some slight additional technical work.

2.1.4.1 General Message Distribution

General Message Distribution: The issue for working with a general distribution for the verifier messages is solved in the classical transformation by defining clusters of messages as follows: cluster i is the set of all the messages with weight in the range 2^{-i} and 2^{-i+1} . In our case, we have to work harder. Firstly, due to the way we use distributional inverters, we will need that for every cluster, the distribution of messages when restricted only to messages in the cluster be statistically close to uniform. This can be solved by splitting the distribution into more clusters - cluster i will now be all messages in the range $(1 + 1/\text{poly}(n))^{-i}$ and $(1 + 1/\text{poly}(n))^{-i+1}$. Note that the probabilities of messages in neighbouring clusters are similar. Therefore, the observation made in analysis of Case 3 that we can distinguish to which cluster a message belongs even though the approximation procedure does not return exact values for the number of coins that lead to a message, is false. To solve this issue, we work with "approximate clusters" - cluster i consists of all the verifier messages for which *the approximation procedure claims* the weight is between $(1 + 1/\text{poly}(n))^{-i}$ and $(1 + 1/\text{poly}(n))^{-i+1}$.

2.1.4.2 Imperfect Completeness

In order to accommodate protocols with completeness c , recall that the clusters are defined as sets of accepting coins with some weight. In the classical transformation in Case 3 the final prover's claim is that there are $p_b \cdot 2^\ell$ coins which would lead the verifier to accept. Since

now p_b is the probability that these coins are sampled *conditioned* on sampling accepting coins, the end claim is changed to $p_c \cdot c \cdot 2^\ell$. In our case, we will not be able to use inverters to find accepting coins, since we only know how to invert efficient functions, and we do not have an efficient function that returns a random accepting coin. We therefore redefine the clusters to refer to general coins, not just accepting ones. This means that in our protocol, the verifier accepts with almost the same probability as in the original private-coin protocol.

2.1.4.3 Multiple Rounds

The issue of multiple rounds is solved in the original transformation by iteratively emulating each round of the protocol. In the following we ignore the issue of distributions over messages which are not uniform. This is treated as explained under “General Message Distribution”. In the Goldwasser-Sipser protocol round i starts with the prefix of a transcript $\gamma_{i-1} = (\alpha_1, \beta_1, \dots, \alpha_{i-1}, \beta_{i-1})$ and N_{i-1} , a claimed lower bound on the number of coins that are consistent with γ_{i-1} . The prover gives a claim N_i that there are N_i coins consistent with each message possible verifier message conditioned on the transcript prefix γ_{i-1} . The parties next run the set lower-bound protocol. That is, the verifier sends a randomly sampled hash function h and random y . The honest prover sends back α_i such that $h(\alpha_i) = y$ and that α_i is consistent with γ_{i-1} (i.e. there exist ρ such that $\alpha_1 = V(\alpha_1, \beta_1, \dots, \alpha_{i-1}, \beta_{i-1}; \rho)$). The prover then sends β_i . This process is run iteratively until the parties have a full transcript γ along with a claimed lower-bound on the number of coins consistent with this full transcript, at which point the parties execute a final set lower-bound protocol to sample a set of coins ρ .

To follow Goldwasser and Sipser’s formula with an efficient prover, we would like an efficient method such that given a random hash function h , and y find this method outputs a consistent α_i . In the one-round case as explained previously, we noted that $f(h, \rho) = h, h(V(\rho))$ was an efficiently computable function in order to sample $\alpha_1 = V(\rho)$. How can we use the same idea but correlate the output to a transcript? To more easily illustrate, in the following we consider a 2-round protocol so that our goal is to sample α_2 after the transcript (α_1, β_1) has already been set.

We show that if the proof in question is doubly-efficient, it suffices to invert the function f that on inputs h and ρ : Computes $\alpha_1 = V(\rho)$, $\beta_1 = P(\alpha_1)$, $\alpha_2 = V(\alpha_1, \beta_1; \rho)$ and outputs $(\alpha_1, \beta_1, h(\alpha_2))$. Firstly note that since the prover is efficient the function f can be computed in polynomial time. Next, notice that the distribution $f(\mathcal{H}, U_\ell)$ is identical to that of taking α_1, β_1 from a random execution of the protocol and additionally outputting a random hash of the next verifier message. Consider an inverter for f . Given a random pair α_1, β_1 and a random h, y it returns randomness ρ such that $\alpha_1 = V(\rho)$. Given ρ it is easy to compute $\alpha_2 = V(\alpha_1, \beta_1; \rho)$. Since ρ is consistent with α_1, β_1 , we have that α_2 is also consistent with α_1, β_1 . Moreover, α_2 will hash to y . This is exactly what we needed.

2.2 Overview of the Piecemeal Emulation Protocol

In this section we overview the techniques used to prove Theorem 4. We prove the theorem by constructing a protocol which we call the “piecemeal” emulation protocol, which is inspired by ideas described in [21] that are accredited to Joe Kilian.

The protocol hinges on a sampling protocol where the goal of the honest prover is to help the verifier generate a transcript that is distributed similarly to a random transcript of the protocol the parties are trying to emulate. Let \mathcal{L} be a language and $\langle P, V \rangle$ be an r -round interactive proof for \mathcal{L} with ℓ bits of randomness and message length m . Since there are r rounds, there are $2r$ messages sent in the protocol. Each message is of length m , and so

the length of a complete transcript of the protocol is $2rm$ bits. We assume without loss of generality that the protocol ends with the verifier sending its entire randomness to the prover. To reiterate, our goal is to generate a random transcript of an execution of the proof. We do this bit-by-bit in an iterative manner as follows: Round i begins with a partial transcript prefix γ_{i-1} and a claimed lower bound N_{i-1} on the number of random coins which are consistent with this partial transcript, where $\gamma_0 = \emptyset$ is the empty transcript with the claim that all $N_0 = 2^\ell$ coins are consistent with the empty transcript. By consistent with a partial transcript we mean that had the private-coin verifier received these coins at the beginning of the protocol execution, then this partial transcript would have been generated, with respect to the prover messages. The prover sends two values N_i^0 and N_i^1 where N_i^0 is the number of coins that are consistent with extending the transcript with the bit 0, meaning coins consistent with the transcript $(\gamma_{i-1}, 0)$, and similarly N_i^1 is the number of coins consistent with $(\gamma_{i-1}, 1)$. If the prover can exactly count each of these values, then it should be that $N_{i-1} = N_i^0 + N_i^1$. The verifier tests that indeed $N_{i-1} = N_i^0 + N_i^1$ and chooses a bit b with probability $\frac{N_i^b}{N_{i-1}}$. Both parties set the new transcript to be $\gamma_i = (\gamma_{i-1}, b)$ and the new claim on the number of consistent coins to be $N_i = N_i^b$. This continues on until $i = 2rm$ when a full transcript has been generated, where since we assumed that the verifier ends by outputting its randomness, there can only be one random coin that is consistent with the transcript. Therefore, after the last iteration the verifier tests that the final $N_{2rm} = 1$, and that all verifier messages in the transcript are what V would have sent in an actual execution using randomness from the end of the transcript. Finally, if all these tests pass the verifier and accepts if V accepts given the transcript γ_{2rm} .

For completeness, it can be shown that the protocol described above generates a transcript with the exact same distribution as the original one, since in every stage the next bit of the transcript is chosen with probability equal to the probability that it would appear in a real random transcript conditioned on the part of the transcript that has already been fixed. We now would like to show that the protocol is sound, i.e. that for $x \notin \mathcal{L}$ a malicious prover cannot cause the verifier to accept in the new protocol with probability greater than in the original protocol. To show this we look the ratio between the number of claimed consistent coins, N and the number of consistent coins that would make the verifier accept in a given round. For a given partial transcript γ we denote by $Acc(\gamma)$ the set of coins ρ such that there exists a legal full transcript of the real execution γ' which begins with γ and in which the verifier accepts.

We begin our inspection of soundness with the final round and work backwards from there. Let $i = 2mr$, and let N_{i-1} and γ_{i-1} be the claim and the partial transcript at the beginning of the iteration. Since the transcript ends with the verifier sending its entire randomness, the number of accepting coins consistent with a transcript with only one bit missing can be 0, 1 or 2. It can be shown that in every case, the probability that the verifier ends up accepting is at most $\frac{|Acc(\gamma_{i-1})|}{N_{i-1}}$. For conciseness we focus in this overview on what happens if $|Acc(\gamma_{i-1})| = 1$. In this case only one of the two options for the final bit will make the verifier accept. Suppose this bit is 0, then the probability that the verifier accepts reduces to the probability that it chooses $b = 0$, which is $\frac{N_i^0}{N_{i-1}}$. Now, since in the end of the protocol the verifier tests that $N_i = N_{2rm} = 1$ in order for the prover to cause the verifier to accept bit 0 it must set $N_i^0 = 1$. Therefore, the probability that the verifier ends up accepting the transcript is at most $\frac{1}{N_{i-1}} = \frac{|Acc(\gamma_{i-1})|}{N_{i-1}}$.

We now look at other rounds of the protocol. Let γ_{i-1} and N_{i-1} be the inputs to iteration i . Suppose, as our induction hypothesis, that upon entering round $i + 1$ with γ_i and N_i the probability that the verifier ends up accepting is $\frac{|Acc(\gamma_i)|}{N_i}$. Let N_i^0 and N_i^1 be the values sent

by the prover. By the induction hypothesis, if the verifier chooses bit b , which happens with probability $\frac{N_i^b}{N_{i-1}}$, then it will end up accepting with probability $\frac{|Acc(\gamma_{i-1}, b)|}{N_i^b}$. Therefore the probability that the verifier ends up accepting is:

$$\frac{N_i^0}{N_{i-1}} \cdot \frac{|Acc(\gamma_{i-1}, 0)|}{N_i^0} + \frac{N_i^1}{N_{i-1}} \cdot \frac{|Acc(\gamma_{i-1}, 1)|}{N_i^1} = \frac{|Acc(\gamma_{i-1}, 0)| + |Acc(\gamma_{i-1}, 1)|}{N_{i-1}}$$

Noting that $|Acc(\gamma_{i-1}, 0)| + |Acc(\gamma_{i-1}, 1)| = |Acc(\gamma_{i-1})|$ we have that the verifier eventually accepts with probability $\frac{|Acc(\gamma_{i-1})|}{N_{i-1}}$. This inductive argument extends all the way up to γ_0 and N_0 in which case $\frac{|Acc(\gamma_0)|}{N_0}$ is equal to the soundness error of the original protocol.

The actual protocol differs slightly from the one described above. In the real setting, the honest prover cannot exactly calculate N_i^0 and N_i^1 , but rather only ε -approximate them. This will mean that the transcript that is sampled is only close to uniform. A further implication of this change is that since the honest prover can err, the verifier now must relax its test that $N_i^0 + N_i^1 = N_{i-1}$. This relaxation turns out to be to test that $\frac{N_{i-1}}{N_i^0 + N_i^1} \leq 1 + 3\varepsilon$. This in turn gives the cheating prover some additional leeway, specifically in round i the probability that the verifier ends up accepting changes from $\frac{|Acc(\gamma_{i-1})|}{N_{i-1}}$ to $(1 + 3\varepsilon)^{2rm-i} \cdot \frac{|Acc(\gamma_{i-1})|}{N_{i-1}}$ (recall that $2rm$ is the number of bits sent in the protocol). If ε is small enough this leeway is insignificant.

References


- 1 László Babai and Shlomo Moran. Proving properties of interactive proofs by a generalized counting technique. *Inf. Comput.*, 82(2):185–197, 1989. doi:10.1016/0890-5401(89)90053-9.
- 2 Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 37–56, 1988. doi:10.1007/0-387-34799-2_4.
- 3 Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, pages 186–194, 1986. doi:10.1007/3-540-47721-7_12.
- 4 Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research*, 5:429–442, 1989.
- 5 Oded Goldreich. On doubly-efficient interactive proof systems. *Foundations and Trends in Theoretical Computer Science*, 13(3):158–246, 2018. doi:10.1561/04000000084.
- 6 Oded Goldreich and Maya Leshkowitz. On emulating interactive proofs with public coins. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:66, 2016. URL: <http://eccc.hpi-web.de/report/2016/066>.
- 7 Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 174–187, 1986. doi:10.1109/SFCS.1986.47.
- 8 Oded Goldreich and Dana Ron. A sublinear bipartiteness tester for bounded degree graphs. *Comb.*, 19(3):335–373, 1999. doi:10.1007/s004930050060.
- 9 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. doi:10.1007/s00453-001-0078-7.
- 10 Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 113–122, 2008. doi:10.1145/1374376.1374396.

- 11 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 291–304, 1985. doi:10.1145/22145.22178.
- 12 Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 59–68, 1986. doi:10.1145/12130.12137.
- 13 Iftach Haitner, Mohammad Mahmoody, and David Xiao. A new sampling protocol and applications to basing cryptographic primitives on the hardness of NP. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:1, 2010. URL: <http://eccc.hpi-web.de/report/2010/001>.
- 14 Thomas Holenstein and Robin Künzler. A protocol for generating random elements with their probabilities. *CoRR*, abs/1312.2483, 2013. arXiv:1312.2483.
- 15 Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147, 1995. doi:10.1109/SCT.1995.514853.
- 16 Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 230–235, 1989. doi:10.1109/SFCS.1989.63483.
- 17 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992. doi:10.1145/146585.146605.
- 18 Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 793–802, 2013. doi:10.1145/2488608.2488709.
- 19 Adi Shamir. $IP = PSPACE$. *J. ACM*, 39(4):869–877, 1992. doi:10.1145/146585.146609.
- 20 Larry J. Stockmeyer. The complexity of approximate counting. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, pages 118–126, 1983. doi:10.1145/800061.808740.
- 21 Madhu Sudan. Advanced Complexity Theory (Lecture 14): <http://people.csail.mit.edu/madhu/ST07/scribe/lect14.pdf>. Scribe notes by Nate Ince and Krzysztof Onak, 2007.
- 22 Salil P. Vadhan. On transformation of interactive proofs that preserve the prover's complexity. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 200–207, 2000. doi:10.1145/335305.335330.

On the Randomness Complexity of Interactive Proofs and Statistical Zero-Knowledge Proofs

Benny Applebaum   

Tel-Aviv University, Israel

Eyal Golombek 

Tel-Aviv University, Israel

Abstract

We study the randomness complexity of interactive proofs and zero-knowledge proofs. In particular, we ask whether it is possible to reduce the randomness complexity, R , of the verifier to be comparable with the number of bits, C_V , that the verifier sends during the interaction. We show that such *randomness sparsification* is possible in several settings. Specifically, unconditional sparsification can be obtained in the non-uniform setting (where the verifier is modelled as a circuit), and in the uniform setting where the parties have access to a (reusable) common-random-string (CRS). We further show that constant-round uniform protocols can be sparsified without a CRS under a plausible worst-case complexity-theoretic assumption that was used previously in the context of derandomization.

All the above sparsification results preserve statistical-zero knowledge provided that this property holds against a *cheating verifier*. We further show that randomness sparsification can be applied to honest-verifier statistical zero-knowledge (HVSZK) proofs at the expense of increasing the communication from the prover by $R - F$ bits, or, in the case of honest-verifier perfect zero-knowledge (HVPZK) by slowing down the simulation by a factor of 2^{R-F} . Here F is a new measure of *accessible bit complexity* of an HVZK proof system that ranges from 0 to R , where a maximal grade of R is achieved when zero-knowledge holds against a “semi-malicious” verifier that maliciously selects its random tape and then plays honestly. Consequently, we show that some classical HVSZK proof systems, like the one for the complete Statistical-Distance problem (Sahai and Vadhan, JACM 2003) admit randomness sparsification with no penalty.

Along the way we introduce new notions of pseudorandomness against interactive proof systems, and study their relations to existing notions of pseudorandomness.

2012 ACM Subject Classification Theory of computation → Interactive proof systems

Keywords and phrases Interactive proofs, Zero-knowledge proofs, Pseudorandomness

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.4

Related Version *Full Version*: <https://eprint.iacr.org/2021/605>

Funding Supported by the European Union’s Horizon 2020 Programme (ERC-StG-2014-2020) under grant agreement no. 639813 ERC-CLC, and the Check Point Institute for Information Security.

1 Introduction

Randomness is a valuable resource. It allows us to speed-up computation in various settings and it is especially useful, or even essential, at the presence of adversarial behavior. Consequently, an extensive body of research has been devoted to the question of minimizing the randomness complexity in various contexts. Notably, the seminal notion of *pseudorandomness* [8, 41] has been developed as a universal approach for saving randomness or even completely removing the need for random bits. In this paper, we study this general question in the context of (*probabilistic*) *interactive proofs*.



© Benny Applebaum and Eyal Golombek;
licensed under Creative Commons License CC-BY 4.0
2nd Conference on Information-Theoretic Cryptography (ITC 2021).
Editor: Stefano Tessaro; Article No. 4; pp. 4:1–4:23



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Interactive proofs, presented by [24, 4], form a natural extension of non-deterministic polynomial time computation (NP). A computationally-bounded probabilistic verifier V wishes to decide whether an input x is a member of a promise problem¹ $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ with the aid of a computationally-unbounded untrusted prover P who tries to convince V that x is a yes-instance. Towards this end, the two parties exchange messages via a protocol, and at the end the verifier decides whether to accept or to reject the input. The protocol should achieve *completeness* and *soundness*. The former asserts that yes-instances should be accepted except for some small probability (*completeness error*), and the latter asserts that no-instances should be rejected regardless of the prover’s strategy except for some small probability (*soundness error*). (See Definition 10.)

The celebrated result of [30, 37] shows that interactive proofs are as strong as polynomial-space computations (i.e., $\text{IP} = \text{PSPACE}$). Moreover, randomness seems essential for this result: If one limits the verifier to be deterministic then interaction does not really help – the prover can predict the verifier messages and so can send all the answers at once – and the power of such proof systems is limited to NP. Put differently, randomness provides “unpredictability” which is crucial for achieving soundness, i.e., for coping with a cheating prover. In fact, even in cases where soundness can be achieved deterministically (i.e., when the underlying problem is in NP) one may want to use a randomized proof system. This is the case, for example, when the prover wants to hide some information from the verifier like in the case of *zero-knowledge proofs* [24]. Indeed, deterministic proof systems inherently allow the verifier to convince others in the validity of the statement, a property that violates zero-knowledge for non-trivial languages [34]. In this context, randomness is used for *hiding information* similarly to its use in the setting of randomized encryption [23].

How much randomness is needed for interactive proofs?

We would like to understand how randomness complexity scales with other resources. Specifically, we would like to relate it to the communication complexity of the protocol – a measure that was extensively studied in the context of interactive proofs and for which we have better understanding (e.g., [18, 21]). We therefore ask:

Given an interactive proof system $\langle P, V \rangle$ for a problem Π , can we always *sparsify* the randomness complexity R to be comparable with the amount of communication complexity? Can we do this while preserving zero-knowledge?

We use the term *randomness sparsification* to highlight the point that we do not aim for full *de-randomization*, rather we only try to make sure that the randomness complexity is not much larger than the communication complexity.

1.1 Related works

Clearly the question of sparsification becomes trivial for public-coin protocols (aka Arthur-Merlin protocols) in which all the randomness of the verifier is being sent during the protocol. Goldwasser and Sipser [25] showed that any general interactive proof protocol

¹ A promise problem [13] is a partition of the set of all strings into three sets: Π_{yes} the set of *yes instances*, Π_{no} the set of *no instances*, and $\{0, 1\}^* \setminus (\Pi_{\text{yes}} \cup \Pi_{\text{no}})$ the set of “disallowed strings”. The more common notion of a *language* corresponds to the special case where Π_{no} is the complement of Π_{yes} (i.e., there are no disallowed strings). The promise problem formalization is especially adequate for the study of interactive proofs and is therefore adopted for this paper. See [17] for a thorough discussion.

can be transformed into public-coin protocol, however, this transformation increases the randomness complexity of the new system and therefore does not resolve the sparsification question.

Information-theoretically, if the verifier sends at most C_V bits during the whole interaction, it should be possible to emulate it with about C_V bits of randomness (in expectation). Indeed, in the context of two-party communication complexity games, it is well known [31] that randomized protocols that use R random bits can be converted into protocols whose randomness complexity is not much larger than the communication complexity C . While this result can be generalized to the setting of interactive proof systems [2], it does not preserve the computational complexity of the verifier. Specifically, this sparsification is essentially based on an inefficient pseudorandom generator G whose existence follows from the probabilistic method.

The question of *efficient sparsification* in the related context of *information-theoretic secure multiparty computation* (ITMPC) was addressed by Ishai and Dubrov [12]. They introduced the notion of *non-Boolean PRG* (nb-PRG) and showed that such a PRG can be used to sparsify *efficiently-computable* protocols with *passive* security.² The definition of nb-PRG generalizes the standard notion of PRG by considering non-Boolean distinguishers. Formally, a (T, C, ε) nb-PRG $G : \{0, 1\}^S \rightarrow \{0, 1\}^R$ fools any T -time non-Boolean algorithm $D : \{0, 1\}^R \rightarrow \{0, 1\}^C$ with C output bits in the sense that $D(U_R)$ is ε -close (in statistical distance) to $D(G(U_S))$ where U_N denotes the uniform distribution over N -bit strings. For polynomially related parameters, nb-PRGs with an optimal seed length of $O(C)$ bits can be obtained either based on (exponentially strong) cryptographic assumptions [12] or based on standard worst-case complexity-theoretic assumptions [3, 1]. In order to sparsify a passively-secure efficient ITMPC protocol, it suffices to invoke the parties over pseudorandom tapes that are selected according to (T, C, ε) nb-PRG where C upper-bounds the number of bits communicated to the adversary and T is the total computational complexity of the protocol. The main idea is to note that any fixed coalition of corrupted parties receives from the honest parties at most C bits of incoming messages whose distribution can be generated by applying a procedure D to the pseudorandom tapes of the honest parties. The procedure D is obtained by “gluing” together the codes of all parties, and can therefore be implemented with complexity T . Since the underlying nb-PRG fools D , the sparsified protocol remains information-theoretic private: An *external* unbounded environment that examines the view of the adversary “learns” nothing on the honest parties inputs.

The above argument relies on the efficiency of all *internal parties* that participate in the protocol. It is therefore unclear whether it can be extended it to our setting where prover, even when played honestly, may be computationally unbounded.³ Nuida and Hanaoka [33] pointed out to the limitation of the nb-PRG approach in the context of “leaky” distinguishing games with an *internal* computationally-unbounded adversary, and suggested to use exponentially-strong cryptographic pseudorandom generators (whose distinguishing advantage is exponential in the leakage available to the adversary). It should be mentioned, however, that although the original sparsification argument of [12] fails, we do not know

² More precisely, their sparsification applies to protocols with privacy against parties that passively follow the protocol but may select their random tape arbitrarily. (In addition, they used an indistinguishability-based definition which is equivalent to unbounded simulation, however, their result seems to generalize to the case of efficient simulation as well.)

³ In contrast, one can use nb-PRGs (against arbitrary polynomial-time adversaries) to sparsify *efficiently-computable argument systems*. In such systems correctness holds with respect to an efficient prover strategy, and soundness is required to hold only against efficient provers. However, this setting has no information-theoretic flavor and a standard cryptographic pseudorandom generator can be used as well.

whether nb-PRGs suffice for sparsification neither in our context nor in the more general context suggested by [33]. In fact, known concrete constructions of nb-PRGs (e.g., ones that are based on exponential cryptographic-PRGs) seem to suffice for this purpose.

Finally, let us mention that several works have studied other aspects of randomness complexity in the context of public-coin interactive proof systems. This includes randomness-efficient methods for round-reduction [7] and for error-reduction [6].

1.2 Our Results and Techniques

In this paper, we present several sparsification results for interactive proofs and for zero-knowledge proofs. We begin with the former case.

1.2.1 General Interactive Proofs

Before stating our results, we set-up some notation.

► **Notation 1.** *For polynomially-bounded integer-valued functions R, C_V, T_V, C_P and k we consider proof systems that on an n -bit input, the parties exchange $k(n)$ messages, where the verifier V uses $R(n)$ random bits, sends a total number of $C_V(n)$ bits, and runs in time $T_V(n)$, and the prover sends a total number of $C_P(n)$ bits. We refer to such protocols as $\text{IP}_k[R, C_V, T_V, C_P]$ protocols. We also consider non-uniform $\text{IP}_k[R, C_V, T_V, C_P]$ protocols in which the verifier is implemented by a T_V -size circuit. We sometimes omit k and use $\text{IP}[R, C_V, T_V, C_P]$ (or non-uniform $\text{IP}[R, C_V, T_V, C_P]$) to denote a protocol with an unspecified round complexity. (Observe that in any case k is upper-bounded by $C_V + C_P$.) Similarly, we let IP (resp., IP/poly , IP_k) denote the union of $\text{IP}[R, C_V, T_V, C_P]$ (resp., non-uniform $\text{IP}[R, C_V, T_V, C_P]$, $\text{IP}_k[R, C_V, T_V, C_P]$) where R, C_V, T_V, C_P range over all polynomially-bounded functions.*

1.2.1.1 PRGs against interactive proofs

Let us begin by presenting a natural definition for a PRG against an interactive proof. Consider an $\text{IP}[R, C_V, T_V, C_P]$ proof system $\langle P, V \rangle$ for a problem Π with completeness error of δ_c and soundness error of δ_s . For a length-extending function $G : \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)}$ we define the verifier $V^G(x)$ to be the verifier that samples a seed $s \leftarrow \{0, 1\}^{S(n)}$ and invokes V with the random tape $G(s)$ on the input x . We say that G ε -fools the protocol $\langle P, V \rangle$ if $\langle P, V^G \rangle$ forms an interactive proof system for Π with an additive penalty of ε in the completeness and soundness error, i.e., the completeness error and soundness errors are upper-bounded by $\delta_c + \varepsilon$ and by $\delta_s + \varepsilon$, respectively.

We begin by noting that, in the non-uniform setting, one can construct such PRGs *unconditionally* with a seed length that is linear in the verifier's communication complexity and logarithmic in its running time.

► **Theorem 2.** *For every functions $T_V(n), C_V(n), C_P(n), R(n) : \mathbb{N} \rightarrow \mathbb{N}$ and $\varepsilon : \mathbb{N} \rightarrow [0, 1]$, there exists a $G : \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)}$ that can be computed by a non-uniform $\tilde{O}(RT_V)$ -size circuit and ε -fools every non-uniform $\text{IP}[R, C_V, T_V, C_P]$ protocol where $S = 2C_V + 2 \log(1/\varepsilon) + \log T_V + \log \log T_V + O(1)$.*

As an immediate corollary we derive the following result.

► **Theorem 3 (Non-Uniform Randomness Sparsification for IP).** *Suppose that a promise problem Π has a (possibly non-uniform) $\text{IP}[R, C_V, T_V, C_P]$ interactive proof $\langle P, V \rangle$ with completeness error δ_c and soundness error δ_s . Then, for every $\varepsilon(n)$, the promise problem Π*

also has a non-uniform proof system $\langle P, V' \rangle$ whose verifier is a non-uniform algorithm with randomness complexity $R' = O(C_V + \log(1/\varepsilon) + \log T_V)$ and computational complexity of $T'_V = T_V + \tilde{O}(T_V(R + \log(1/\varepsilon)))$ and with identical communication complexity ($C'_V = C_V$ and $C'_P = C_P$), and identical round complexity. The soundness and completeness error of the new system are $\delta'_s \leq \delta_s + \varepsilon$ and $\delta'_c \leq \delta_c + \varepsilon$. Moreover, if the original proof system has a perfect completeness then so is the new system.⁴

The PRG construction (Theorem 2) is based on a family of t -wise independent hash functions. That is, we show that, for a properly chosen parameter t , a randomly chosen t -wise independent hash function is likely to fool $\text{IP}[R, C_V, T_V, C_P]$. Unfortunately, one has to invest too many random bits in order to sample a hash function, and so we use non-uniformity to hard-wire one “good” hash function. (See Section 3 for details.) An alternative solution is to select the hash function via a common-random-string (CRS) that is available to both parties and can be reused among many invocations.⁵ This also leads to uniform sparsification in an amortized setting where many instances are considered together. In such a case one can even remove the CRS and let the verifier sample it once for all the instances. (See Corollary 19.)

1.2.1.2 Single-Instance Sparsification in the uniform setting without CRS?

A natural way for achieving randomness sparsification in the uniform setting is to “sparsify” the process of selecting the hash function. That is, to use a different pseudorandom generator to sample a hash function. Indeed, this approach was taken by [1] to construct nb-PRGs. The idea is to show that given the description of a hash function h_z one can determine with “not-too-large-complexity” (e.g., low in the polynomial hierarchy) whether h_z fools an interactive proof system. If such a decision can be made by some “algorithm” D then we can select the hash function by using a PRG that fools D . Unfortunately, our definition of “fooling interactive proofs” does not seem to be efficiently-decidable. First, the definition implicitly refers to inputs that satisfy the promise of the underlying problem Π , and deciding whether an input x belongs to $\Pi_{\text{yes}} \cup \Pi_{\text{no}}$ may be very hard. Second, as part of the pseudorandomness requirement, the new system $\langle P, V^G \rangle$ should preserve completeness (up to an error of ε). However, this property depends on the behavior of the honest prover P which is an *inefficient* procedure on which we have no “handle”. In particular, even if we try to design an interactive proof system for deciding whether h_z is a good PRG, it is not clear how to make sure that the unbounded prover really uses the honest P when needed.

1.2.1.3 Strong PRGs

We solve both problems by *strengthening* the notion of pseudorandomness against interactive proofs. Specifically, we say that G *strongly* ε -fools the protocol $\langle P, V \rangle$ if for every string $x \in \{0, 1\}^*$ and every possible prover strategy P^* , the gap between the acceptance probability of $V(x)$ when interacting with $P^*(x)$ and the acceptance probability of $V^G(x)$ when interacting with $P^*(x)$ is at most ε . While this definition seems stronger than the previous one, the proof of Theorem 2 actually shows that random hash functions *strongly* fool interactive proofs. Crucially, this new definition makes no reference to the underlying promise problem or to the honest prover P . (Indeed, one may say that G ε -fools the *interactive machine* V .)

⁴ All our transformations preserve perfect completeness. From now on, we omit this point throughout this section.

⁵ In many scenarios such a CRS is available “for free”. Furthermore, the fact that it is re-usable and that it should not be kept private from the prover even before the protocol begins, makes it highly-attractive even compared to *public coins*.

As a result, the above-mentioned obstacles are removed and we can show that the problem of checking whether a given hash function h_z strongly-fools an IP_k proof system admits an IP_{k+1} proof system. For constant k , this puts the language of “bad” hash functions in the class AM and so we can select our hash function by a pseudorandom generator that fools AM – a well-studied object in complexity theory. Specifically, known constructions of such PRGs [32, 27, 28, 36] can be based on the assumption that $\text{E} = \text{DTime}(2^{O(n)})$ is hard for exponential size non-deterministic circuits. (See Theorem 22 for details). In Section 4 we prove the following result.

► **Theorem 4** (Uniform Randomness Sparsification for constant-round proofs). *Suppose that E is hard for exponential size non-deterministic circuits. Then, for every inverse polynomial ε , every constant k and every polynomially-bounded functions R, C_V, T_V, C_P , there exists a PRG computable in uniform polynomial time of $T'_V = \tilde{O}(T_V \cdot (R + \log n))$ that strongly ε -fools non-uniform $\text{IP}_k[R, C_V, T_V, C_P]$ proof systems with seed length of $R' = 2C_V + O(\log n)$. Consequently, every $\text{IP}_k[R, C_V, T_V, C_P]$ proof system can be transformed into a new $\text{IP}_k[R', C_V, T'_V, C_P]$ with an additive penalty of ε in the soundness and completeness errors. Moreover, perfect completeness is preserved.*

The underlying assumption can be viewed as a natural extension of $\text{EXP} \neq \text{NP}$ to the non-uniform settings. Similar assumptions were made in the literature (e.g., [5, 11, 14, 22, 39]).

► **Remark 5.** One should note that when k is constant the underlying assumption suffices for full de-randomization of the protocol (via a sequence of transformations). Still, one may prefer to use the sparsified protocol (that still uses some randomness), either due to its efficiency properties (in terms of computation and communication) or due to its zero-knowledge properties as discussed in Section 1.2.2.

The seed length of our PRGs is dominated by the number of bits, C_V , sent by the verifier. (This is the case both in the uniform and non-uniform settings.) It is not hard to show that such a dependency is essentially optimal even if one considers the weaker variant of IP PRGs.

► **Proposition 6** (Sparsification lower-bound). *For every functions $T_V(n), C_V(n), C_P(n), R(n) : \mathbb{N} \rightarrow \mathbb{N}$ where $C_V < R$ every $G : \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)}$ that 0.1-fools $\text{IP}[R, C_V, T_V, C_P]$ protocols must have a seed length of $\Omega(C_V)$.*

Proof. Assume that $S < \alpha C_V$ for some small constant $\alpha < 1$. A simple information-theoretic argument shows that $y = G(U_S)$ is predicatable in the following sense. There exists an index $i \in [C_V]$ such that given the $(i-1)$ -prefix $y[1 : i-1]$, one can guess (possibly inefficiently) the next bit $y[i]$ with success probability of, say, 0.8. Indeed, letting $p_i := H(y_i | y[1 : i-1])$ denote the conditional entropy of y_i given the prefix, we know that $\sum_{i=1}^{C_V} p_i \leq H(y) = S < \alpha C_V$ and so, by an averaging argument, there exists an index i for which $p_i < \alpha$. For sufficiently small constant α , this implies that y_i is predictable with probability 0.8. Consider the following proof system for the trivial empty language ($\Pi_{\text{yes}} = \emptyset$ and $\Pi_{\text{no}} = \{0, 1\}^*$). The verifier samples $r \in \{0, 1\}^R$ and sends $r[1 : i-1]$ to the prover who responds with a single bit b . The verifier accepts if $b = y[i]$. When r is random the soundness error is $1/2$, but when $r = G(U_S)$, the error grows to 0.8. ◀

1.2.1.4 nb-PRGs are not IP-PRGs

We also show (in the full version) that, under plausible cryptographic assumptions, some nb-PRGs do not fool IP protocols. Roughly, this is done by constructing a nb-PRG which is *malleable*. That is, although the prover cannot tell whether the verifier uses random bits

or bits that were generated via the nb-PRG, she can provide a short hint that allows a computationally-bounded algorithm (the original verifier) to distinguish between the two cases. Our results therefore show that the inapplicability of nb-PRGs to our setting reflects an inherent limitation and it is not just an artifact of the previous proof techniques.

1.2.2 Zero-Knowledge Proofs

We move on and study randomness-sparsification for statistical zero-knowledge proofs. In the following we focus on constant-round zero-knowledge protocols with a uniform verifier and base our results on the assumption from Theorem 4. If one is willing to make the verifier non-uniform (or to allow a public common reference string), then the following results can be proved unconditionally without assumptions for protocols with an arbitrary number of rounds.

Let $\text{SZK}_k[R, C_V, T_V, C_P]$ be an $\text{IP}_k[R, C_V, T_V, C_P]$ statistical zero-knowledge protocol, whose zero-knowledge property holds against an arbitrary, possibly malicious, verifier that may deviate from the protocol. We begin by noting that PRG-based randomness-sparsification trivially preserves such a strong zero knowledge property.

► **Theorem 7** (Uniform Randomness Sparsification for constant-round SZK). *Suppose that E is hard for exponential size non-deterministic circuits. Then, for every inverse polynomial ε , every constant-round $\text{SZK}_k[R, C_V, T_V, C_P]$ proof system can be transformed into a new $\text{SZK}_k[R', C_V, T'_V, C_P]$ with randomness of $R' = 2C_V + O(\log n)$, (uniform) verifier's complexity of $T'_V = \tilde{O}(T_V \cdot (R + \log n))$ and with an additive penalty of ε in the soundness and completeness errors.*

The proof is straightforward: Any malicious verifier strategy that can be played in the original protocol $\langle P, V \rangle$ can be also played in the sparsified protocol $\langle P, V^G \rangle$. Indeed, SZK is a feature of the honest prover that remains unchanged in the sparsified proof system.

1.2.2.1 Sparsifying HVSZK?

We move on and ask whether such a theorem can be proved for the case of *honest-verifier* statistical zero-knowledge protocols (HVSZK). While there are known transformations from HVSZK to SZK (e.g., [40, 20, 26]) these transformations incur a communication complexity overhead that is at least as large as the randomness complexity of the original protocol. Therefore, the problem of sparsifying HVSZK is not known to be reducible to the sparsification of SZK.

It is instructive to see why Theorem 7 does not immediately generalize to the HVSZK setting. Consider for simplicity a 2-message proof system $\langle P, V \rangle$ where V sends a message a and receives a message b . The view of an honest verifier consists of the input x , the random tape r and the incoming message b . In the sparsified system, $\langle P, V^G \rangle$, the view consists of the input x , a PRG seed s and the message a . Suppose that the original verifier admits a simulator that, given x , samples the pair (r, a) . How can we use such a simulator to sample (s, a) ? If we use the original simulator then a random r is unlikely to land in the image of G which is *sparse* in the set of all R -bit strings. Moreover, even if we hit the image, it is not clear how to invert G and find an appropriate seed. We observe that the second problem can be easily solved by exploiting the concrete structure of our PRGs. Specifically, by using algebraic constructions of t -wise independent hash functions we can efficiently invert

the PRGs in polynomial-time.⁶ To handle the sparsity problem we suggest two possible approaches:

- Our first solution exploits the prover. We show that the simulation problem can be avoided by asking the prover to supply R random bits at the beginning of the interaction.
- In the context of honest-verifier *perfect* zero-knowledge proofs, we show that randomness can be traded by a simulation slow-down. Specifically, the sparsified protocol (without any modifications) can be simulated with an overhead of time 2^{R-S} where S is the seed-length of the generator. (See Corollary 37.) Such a simulation implies witness-indistinguishability [15] and can be meaningful when the underlying problem is harder than 2^{R-S} . Specifically, one can tune S , i.e., the level of sparsification, according to the hardness of the problem.

1.2.2.2 How much should we pay?

In the above solutions we pay a communication overhead of R (resp., simulation slow-down of 2^{R-S}) in the sparsification of HVSZK systems (resp., HVPZK) whereas in the case of SZK proof systems (with security against cheating verifier) we pay nothing. It turns out that one can interpolate between these two extremes based on a single measure. Roughly, we say that a proof system is an F -semi-malicious statistical zero-knowledge system (F -SMSZK), for some function $0 < F < R$ if it is possible to simulate every verifier that plays honestly except that it selects the first F -bits of its random tape by some arbitrary (efficiently-computable) distribution (the other $R - F$ coins are chosen uniformly).⁷ We prove the following theorem. (See Corollaries 30 and 33.)

► **Theorem 8** (Trading randomness with prover's communication or simulation slowdown). *Suppose that E is hard for exponential size non-deterministic circuits. Then, every promise problem Π that admits a constant-round F -SMSZK $_k[R, C_V, C_P, T_V]$ proof system $\langle P, V \rangle$ also has:*

- An HVSZK $_{k+1}[R' = 2C_V + O(\log n), C_V, T'_V = \tilde{O}(T_V \cdot (R + \log n)), C'_P = C_P + R - F]$ proof system. Specifically, the new protocol consists of an additional preliminary message from the prover that consists of a random string of length $R - F$ bits.
- In the perfect zero-knowledge setting, where $\langle P, V \rangle$ is F -SMPZK $_k[R, C_V, C_P, T_V]$ system, the problem Π admits an HVPZK $_k[2C_V + O(\log n), C_V, C_P, T_V]$ proof system whose simulator runs in time $\text{poly}(n)2^{R-F}$.

Observe that $0 \leq F \leq R$ and that any HVSZK proof system is also an 0-SMSZK and every SZK proof system is R -SMSZK. Thus Theorem 8 implies Theorem 7. Interestingly, some classical HVSZK proof systems also achieve full accessibility of $F = R$. Most notably, this is the case for the classical protocol for the complete *statistical-distance* problem of [35] as well as the classical proof system for *graph-non-isomorphism* (GNI) of [19]. (See the full version.) In fact, these proof systems have only two messages and therefore they are known to be insecure against a cheating verifier [34, Theorem 8] (unless the underlying problems are in BPP). It follows that even the notion of R -SMSZK proof systems is likely to be weaker than SZK.

⁶ This does not contradict security since our PRG fools verifiers of predetermined fixed polynomial-time (corresponding to the running time of the verifier) but can be inverted in larger polynomial time. This feature of the fixed-polynomial-time setting (that is typically used in the context of derandomization [32]) seems novel to this work.

⁷ One should not be confused with our notion of semi-malicious SZK proof systems and the one suggested by [29] that applies to zero-knowledge PCPs.

We further mention that even when $F = 0$, we can get some non-trivial simulation for HVPZK. Specifically by exploiting the concrete properties of our PRG we can get a simulator whose complexity is $\text{poly}(n)2^{R-S}$ where R is the original randomness complexity and S is the seed length of the simulator. (See Section 5.5.) As an application, one can adjust the seed length (i.e., the level of sparsification) according to a given time-bound on the simulation (that may be dictated by the intractability of the underlying language).

1.2.2.3 Organization

Following some preliminaries (Section 2), we study, in Section 3, randomness sparsification for interactive proofs in the non-uniform setting and in the amortized sparsification in the uniform setting. Section 4 is devoted to randomness sparsification for constant-round uniform interactive proofs, and Section 5 to statistical zero-knowledge proofs.

2 Preliminaries

Probabilistic notation

For every $n \in \mathbb{N}$ we denote by U_n the uniform distribution over the set $\{0,1\}^n$ of binary strings of length n . For a probability distribution \mathcal{D} , we use the notation $x \leftarrow \mathcal{D}$ to denote a value x that is sampled according to \mathcal{D} . When \mathcal{D} is a finite set, the notation $x \leftarrow \mathcal{D}$ denotes a value x that is sampled uniformly from \mathcal{D} . We follow the standard way of defining distance between two distributions:

► **Definition 9** (Statistical Distance). *Given X, Y two probability distributions over some discrete universe Ω the statistical difference between them is defined:*

$$\text{SD}(X, Y) = \max_{S \subseteq \Omega} |\Pr[X \in S] - \Pr[Y \in S]|.$$

► **Definition 10** (Interactive proof system [24]). *A pair of interactive machines $\langle P, V \rangle$ is called an interactive proof system with completeness error of δ_c and soundness error of δ_s for a promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ if the followings hold:*

■ **Completeness:** *For every $x \in \Pi_{\text{yes}}$ we have*

$$\Pr[(P, V)(x) = 1] \geq 1 - \delta_c(|x|)$$

where the probability is taken over the randomness of V and P and we write $(P, V)(x) = 1$ to denote the event that, after interacting with $P(x)$, the verifier $V(x)$ accepts.

■ **Soundness:** *For any cheating strategy for the prover P^* and every $x \in \Pi_{\text{no}}$, it holds that*

$$\Pr[(P^*, V)(x) = 1] \leq \delta_s(|x|).$$

When the parameters δ_c and δ_s are unspecified we assume that they are taken to be $o(1)$.⁸ By default, we assume that V is efficient, i.e., it runs in time $T_V(|x|)$ for some polynomially-bounded function T_V . In the non-uniform setting, we assume that V can be implemented by a non-uniform family of $T_V(|x|)$ -size probabilistic circuits.

⁸ Standard ρ -fold parallel repetition reduces the errors exponentially with ρ at the expense of increasing the communication and computation complexity by a factor of ρ and without affecting the round complexity (see e.g., [16]).

4:10 On the Randomness Complexity of Interactive Proofs & SZK Proofs

Following Notation 1, we let k, R, C_V, C_P denote the number of messages sent in the protocol, the randomness complexity of V , the number of bits sent by V , and the number of bits sent by P .

► **Definition 11** (Statistical Zero-Knowledge). *An interactive proof system $\langle P, V \rangle$ for a promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ is a Statistical Zero-Knowledge proof system (SZK) with a simulation error of δ_z if for every computationally-unbounded verifier V^* there exists a simulator Sim that runs in time polynomial in the complexity of V^* such that for every yes-instance $x \in \Pi_{\text{yes}}$ it holds that*

$$\text{SD}(\text{view}_{V^*}(x), \text{Sim}(x)) \leq \delta_z(|x|),$$

where $\text{view}_{V^*}(x)$ is the random variable that corresponds to the view of $V^*(x)$ when interacting with $P(x)$ which consists of the random tape and all the incoming messages that were sent by P .

The proof system is an Honest-Verifier Statistical Zero-Knowledge proof system (HVSZK) if the above holds for the special case where $V^* = V$. We also denote by HVSZK and SZK the class of all promise problems that possess such an interactive proof system (with error parameters of $o(1)$).

3 Non-uniform randomness sparsification for IP

In this section we study the possibility of reducing the randomness of a general proof system $\langle P, V \rangle$. We begin by defining a strong form of pseudo-random generators against interactive proof systems.

► **Definition 12** (Strongly fooling a protocol). *Let $\langle P, V \rangle$ be a protocol and $R(n)$ denote the randomness complexity of V . For a length-extending function $G : \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)}$ we define the verifier $V^G(x)$ to be the verifier that samples a seed $s \leftarrow \{0, 1\}^{S(n)}$ and invokes $V(x; r)$ with randomness $r = G(s)$.*

We say that G strongly ε -fools the protocol $\langle P, V \rangle$ if for every input x and any possible prover strategy P^* it holds that

$$|\Pr[(V, P^*)(x) = 1] - \Pr[(V^G, P^*)(x) = 1]| \leq \varepsilon.$$

We say that G strongly ε -fools $\text{IP}[R, C_V, T_V, C_P]$ if it strongly ε -fools any interactive proof $\langle P, V \rangle \in \text{IP}[R, C_V, T_V, C_P]$.

Recall that the class $\text{IP}[R, C_V, T_V, C_P]$ is the class of IP protocols in which on an n -bit input the verifier runs in $T_V(n)$ time, uses at most $R(n)$ random bits and sends at most $C_V(n)$ bits to the prover, and the total length of the prover responds is at most $C_P(n)$ bits. Observe that a PRG strongly fools a protocol regardless of the prescribed prover, and it is a trait of the verifier.

► **Observation 13.** *Suppose that $\langle P, V \rangle$ is an interactive proof system for a promise problem Π with completeness error δ_c and soundness error δ_s and G strongly ε -fools $\langle P, V \rangle$. Then $\langle P, V^G \rangle$ is an interactive proof system for a promise problem Π with completeness error $\delta_c + \varepsilon$ and soundness error $\delta_s + \varepsilon$. Moreover, if the original system has perfect completeness then so is the new system.*

Proof. The first part is immediate from Definition 12. The “Moreover” part holds for any G since for any yes instance x a bad (faulty) random string s in $\langle P, V^G \rangle$ for which $(V^G, P)(x)$ rejects translate into a random tape $r = G(s)$ for which $(V, P)(x)$ rejects as well. ◀

We continue by showing that pseudo-random generators against circuits with very small error can be used to fool protocols.

► **Lemma 14** (Fooling protocols via circuit-PRGs). *Let $T_V(n), C_V(n), C_P(n), R(n) : \mathbb{N} \rightarrow \mathbb{N}$ be some integer-valued functions and let $\epsilon : \mathbb{N} \rightarrow [0, 1]$. Every PRG $G : \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)}$ that $\epsilon/2^{C_V(n)}$ -fools $3T_V$ -size circuits also strongly ϵ -fools non-uniform IP $[R, C_V, T_V, C_P]$ protocols.*

Proof. Let $\langle P, V \rangle$ be some (possibly non-uniform) IP $[R, C_V, T_V, C_P]$ proof system and let $G : \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)}$ be a PRG that $\epsilon/2^{C_V(n)}$ -fools $3T_V$ -size circuits. Fix some input $x \in \{0, 1\}^n$ and let $C_V = C_V(n), T_V = T_V(n), C_P = C_P(n)$ and $S = S(n)$. Fix some proof strategy P^* . Let $\text{view}_V(r)$ denote the verifier's view when interacting with P^* on the shared input x with randomness r . This view consists of (x, r) , the concatenation, \vec{a} of all the messages sent from V to P^* during the interaction and the messages \vec{b} that were sent from P^* to V during the interaction.⁹ In the following, we will think of (\vec{a}, \vec{b}) as random variables whose distribution is induced by a random choice of the verifier's random coins.

We will show that (*) $\text{view}_V(U_r)$ is ϵ indistinguishable from $\text{view}_V(G(U_S))$ by T_V -size circuits. Note that (*) implies that $|\Pr[(V, P^*)(x) = 1] - \Pr[(V^G, P^*)(x) = 1]| \leq \epsilon$ since V decides whether to accept its view by applying a predicate which is computable by a circuit of size at most T_V . Let us assume, without loss of generality, that the strategy P^* is deterministic. Indeed, if (*) does not hold for some randomized P^* then, by an averaging argument, there exists a deterministic P^* that violates (*).

We proceed by proving (*). Assume towards contradiction that there exists some distinguisher D of complexity at most T_V that violates (*). Then, we can write

$$\begin{aligned} \epsilon &< \left| \sum_{a \in \{0,1\}^{C_V}} \Pr_{r \leftarrow U_R} [D(x, r, \vec{a}, \vec{b}) = 1 \mid \vec{a} = a] \Pr_{r \leftarrow U_R} [\vec{a} = a] \right. \\ &\quad \left. - \sum_{a \in \{0,1\}^{C_V}} \Pr_{r \leftarrow G(U_S)} [D(x, r, \vec{a}, \vec{b}) = 1 \mid \vec{a} = a] \Pr_{r \leftarrow G(U_S)} [\vec{a} = a] \right| \\ &\leq \sum_{a \in \{0,1\}^{C_V}} \left| \Pr_{r \leftarrow U_R} [D(x, r, \vec{a}, \vec{b}) = 1 \mid \vec{a} = a] \Pr_{r \leftarrow U_R} [\vec{a} = a] \right. \\ &\quad \left. - \Pr_{r \leftarrow G(U_S)} [D(x, r, \vec{a}, \vec{b}) = 1 \mid \vec{a} = a] \Pr_{r \leftarrow G(U_S)} [\vec{a} = a] \right|, \end{aligned}$$

where the inequality is due to the triangle inequality. By an averaging argument, we conclude that there should be at least one element a^* such that

$$\begin{aligned} \frac{\epsilon}{2^{C_V}} &< \left| \Pr_{r \leftarrow U_R} [D(x, r, \vec{a}, \vec{b}) = 1 \mid \vec{a} = a^*] \Pr_{r \leftarrow U_R} [\vec{a} = a^*] \right. \\ &\quad \left. - \Pr_{r \leftarrow G(U_S)} [D(x, r, \vec{a}, \vec{b}) = 1 \mid \vec{a} = a^*] \Pr_{r \leftarrow G(U_S)} [\vec{a} = a^*] \right|. \end{aligned}$$

Recall that the prover is deterministic and therefore once the verifier's messages are fixed to a^* , the prover's messages become fixed as well to some value b^* . We now can define a new distinguisher $D' : \{0, 1\}^{R(n)} \rightarrow \{0, 1\}$ that holds (x, a^*, b^*) as a non-uniform advice

⁹ In the context of this proof, we omit the seed s from the verifier's view. While such an omission will be problematic later when discussing zero-knowledge, it has no consequences in the current proof.

4:12 On the Randomness Complexity of Interactive Proofs & SZK Proofs

and operates as follows. Given an input $r \in \{0, 1\}^{R(n)}$, the distinguisher D' invokes the verifier $V(x)$ using r as the random coins, and emulates the prover P^* by responding according to b^* . If the resulting transcript disagrees with (a^*, b^*) the distinguisher D' rejects. Otherwise, D' return $D(x, r, a^*, b^*)$. Clearly, D' distinguishes between $r \leftarrow U_R$ to $r \leftarrow G(U_S)$ with advantage $\varepsilon/2^{C_V}$. Moreover, D' can be implemented by a circuit of size $T_V + (C_V + C_P) + T_V \leq 3T_V$, and therefore we derive a contradiction to the pseudorandomness of G and (*) follows. \blacktriangleleft

The following claim from [1] shows that good circuit PRGs can be obtained from t -wise independent hash functions. In the following we say that a family of functions $\mathcal{H} = \{h_z : X \rightarrow Y\}$ is *t-wise independent* [10] if for every t distinct inputs $x_1, \dots, x_t \in X$ and uniformly chosen $h_z \leftarrow \mathcal{H}$, the random variable $(h_z(x_1), \dots, h_z(x_t))$ is uniformly distributed over Y^t .

\triangleright **Claim 15** (PRGs from hash functions (Claim 5.2 in [1])). For every T and $\varepsilon, \delta \in [0, 1]$, and every family $\mathcal{H} = \{h_z : \{0, 1\}^s \rightarrow \{0, 1\}^r\}$ of t -wise independent hash functions with $t = 4T \log T + 2 \log(1/\delta)$ and $s = 2 \log(1/\varepsilon) + \log t$ the following holds. With probability $1 - \delta$, a random member $h_z \leftarrow \mathcal{H}$ ε -fools any T -size circuit.

By combining Claim 15 with Lemma 14 we derive the following theorem.

\blacktriangleright **Theorem 16** (Fooling protocols via hashing). Let $T_V(n), C_V(n), C_P(n), R(n) : \mathbb{N} \rightarrow \mathbb{N}$ and $\varepsilon, \delta : \mathbb{N} \rightarrow [0, 1]$ be some arbitrary functions and let $\mathcal{H} = \{h_z : \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)}\}$ be a family of t -wise independent hash functions where $t = O(T_V \log T_V + \log(1/\delta))$ and $S = 2C_V + 2 \log(1/\varepsilon) + \log T_V + \log \log T_V + \log \log(1/\delta) + O(1)$.

Then, $\Pr_{h_z \leftarrow \mathcal{H}}[h_z \text{ strongly } \varepsilon\text{-fools non-uniform IP}[R, C_V, T_V, C_P]] > 1 - \delta$.

\blacktriangleright **Remark 17** (Canonical construction of t -wise independent hash functions). Throughout the paper we use the following standard construction of t -wise independent hash functions $\mathcal{H} = \{h_z : \{0, 1\}^S \rightarrow \{0, 1\}^R\}$ where $t < 2^S < 2^R$. Let $\mathbb{F} = GF(2^R)$ denote the finite field of 2^R elements. We identify field elements with binary strings of length R via some canonical representation that supports arithmetic operations with a computational cost of $\tilde{O}(R)$ bit operations (For instance [38]). It is well known [10] that the family $\mathcal{H}' = \{h'_z : \mathbb{F} \rightarrow \mathbb{F}\}_{z \in \mathbb{F}^t}$ where h'_z denotes the degree- t univariate polynomial whose coefficients are given by the vector $z \in \mathbb{F}^t$ is a family of t -wise independent hash functions from $\{0, 1\}^R$ to $\{0, 1\}^R$. We define \mathcal{H} by restricting the domain of \mathcal{H}' to some fixed 2^S subset. Specifically, Let h_z denote the function that takes an input $x \in \{0, 1\}^S$, maps it to \mathbb{F} by padding it with $R - S$ zeroes, and outputs $h'_z(x)$. Then, $\mathcal{H} = \{h_z\}_{z \in \mathbb{F}^t}$ is a t -wise independent family. Observe that one can sample an index z by sampling a tR random bits, and that given z and $x \in \{0, 1\}^S$ we can evaluate $h_z(x)$ by making $O(t)$ arithmetic operations. Hence the total bit complexity of sampling and evaluating a function in \mathcal{H} is $\tilde{O}(tR)$.

By hard-wiring a “good” hash function as a non-uniform advice to Theorem 16 we derive the following corollary (that strengthens Theorem 2 from the introduction.).

\blacktriangleright **Corollary 18.** For every functions $T_V(n), C_V(n), C_P(n), R(n) : \mathbb{N} \rightarrow \mathbb{N}$ and $\varepsilon : \mathbb{N} \rightarrow [0, 1]$, there exists a PRG : $\{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)}$ that can be computed by a non-uniform $\tilde{O}(RT_V)$ -time and strongly ε -fools non-uniform IP $[R, C_V, T_V, C_P]$ where $S = 2C_V + 2 \log(1/\varepsilon) + \log T_V + \log \log T_V + O(1)$.

Theorem 3 follows immediately.

The amortized setting

For a promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ and a polynomial $k(\cdot)$ define the problem $\Pi^k = (\Pi'_{\text{yes}}, \Pi'_{\text{no}})$ by letting Π'_{yes} denote the set of all tuples $\vec{x} = (x_1, \dots, x_{k(n)}) \in (\{0, 1\}^n)^{k(n)}$ such that $x_i \in \Pi_{\text{yes}}$ for every i and by letting Π'_{no} denote the set of tuples $\vec{x} = (x_1, \dots, x_{k(n)}) \in (\{0, 1\}^n)^{k(n)}$ such that, for every i , $x_i \in \Pi_{\text{yes}} \cup \Pi_{\text{no}}$ and for at least one i , $x_i \in \Pi_{\text{no}}$.

► **Corollary 19** (Uniform Amortized sparsification of many instances). *Let Π be a promise problem that admits a uniform $\text{IP}[R, C_V, T_V, C_P]$ proof system with negligible soundness and correctness errors. Then, for every polynomial $k(\cdot)$, the promise problem Π^k admits a (uniform) $\text{IP}[R', C'_V = kC_V, T'_V, C'_P = kC_P]$ proof system with constant error where $R' = R \cdot \tilde{O}(T_V) + O(k(C_V + \log k + \log T_V))$ and $T'_V = k\tilde{O}(T_V R)$.*

So for sufficiently large k , the amortized randomness complexity R'/k is $O(C_V + \log k + \log T_V)$ per instance.

Proof. Let $\varepsilon = 1/(10k)$ and $\delta = 0.1$. Let \mathcal{H} be a family of t -wise hash function that expand S bits to R bits where $t = O(T_V \log T_V)$ and $S = 2C_V + 2 \log(1/\varepsilon) + \log T_V + \log \log T_V + \log \log(1/\delta) + O(1) \leq 2C_V + 2 \log k + 2 \log T_V + O(1)$. The verifier samples a function $h_z \leftarrow \mathcal{H}$ and given $\vec{x} = (x_1, \dots, x_{k(n)})$ applies, for each i , the original verifier $V(x_i; h_z(s_i))$ where s_i is chosen uniformly and independently from U_S . At the end, we accept if and only if all interactions accepted. The prover simply runs the original protocol k times.

By Theorem 16, with probability $1 - \delta$ the hash function h_z ε -fools the original protocol. Therefore, conditioned on this event, the error in each instance is at most $\varepsilon + n^{-\omega(1)}$, and by a union bound the total error is at most $\delta + k\varepsilon + kn^{-\omega(1)} \leq 0.2$, as required. The communication grows by a factor of k , the randomness complexity is $O(tR)$ for sampling the hash function (Remark 17) plus $O(kS)$ for sampling the seeds. The computational complexity for sampling h_z is $\tilde{O}(tR)$ and each instance has an additional cost of $T_V + \tilde{O}(tR)$ (again see Remark 17). ◀

4 Uniform Randomness Sparsification for Constant-Round Protocols

In this section we extend the randomness reduction seen in the previous section from the non-uniform setting to the uniform setting. Recall that in the previous section we reduced the randomness of general IP proofs by using a non-uniform advice that consisted of a description of a “good” hash function that can be used as a PRG. As explained in Section 1.2 we cannot afford to sample the hash function uniformly since this requires too much randomness (larger than the amount of randomness that is needed for the original protocol). Instead, we describe a randomness-efficient method for sampling a “good” hash function via a uniform algorithm by reducing the problem to a more standard de-randomization problem. We further show that for constant number of rounds, the latter problem can be solved under standard complexity-theoretic assumptions.

We begin by defining a promise problem whose no-instances corresponds to hash functions that “fool a given protocol” and its “yes” instances are hash functions that “fail to fool the protocol”.

► **Definition 20.** *Let $\langle P, V \rangle$ be a k -round (possibly non-uniform) $\text{IP}[R, C_V, C_P, T_V]$ protocol for a promise problem L with a polynomial-time verifier, and let $\varepsilon(n)$ be some inverse polynomial. Fix some efficiently computable family of hash functions*

$$\mathcal{H} = \left\{ h_z : \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)} \right\}_{z \in \{0, 1\}^{Z(n)}}$$

4:14 On the Randomness Complexity of Interactive Proofs & SZK Proofs

that satisfies Theorem 16 with respect to $\text{IP}[R, C_V, C_P, T_V]$ protocols where the underlying parameters ε, δ are taken both to be $\varepsilon(n)$. We define a promise problem $\Pi = \Pi_{P,V,\varepsilon}$ over strings $z \in \{0, 1\}^*$ as follows:

- The set of yes instances, Π_{yes} , consists of all strings z such that h_z does not strongly 2ε -fool $\langle P, V \rangle$.
 - The set of no instances, Π_{no} , consists of all strings z such that h_z strongly ε -fools $\langle P, V \rangle$.
- We prove the following lemma.

► **Lemma 21** ($\Pi_{P,V,\varepsilon} \in \text{IP}_{k+1}$). For any k -message protocol $\langle P, V \rangle$ (resp., non-uniform protocol $\langle P, V \rangle$) and inverse polynomial ε the promise problem $\Pi_{P,V,\varepsilon}$ is in IP_{k+1} (resp., $\text{IP}/\text{poly}_{k+1}$) and the computational complexity of the corresponding verifier is $O(T_V + T_H)$ where T_V is the complexity of V and T_H is the computational complexity of universal evaluation of \mathcal{H} . Consequently, for constant k , $\Pi_{P,V,\varepsilon}$ is in AM (resp., in AM/poly).

Proof. On a shared input $z \in \{0, 1\}^*$, the prover will try to convince the verifier that h_z does **not** strongly 2ε -fool $\langle P, V \rangle$. Recall that this means that one of the following holds for some input x :

- (Case 0:) There exists P^* Strategy such that $\Pr[(V^{h_z}, P^*)(x) = 1] - \Pr[(V, P^*)(x) = 1] > 2\varepsilon$.
- (Case 1:) There exists P^* Strategy such that $\Pr[(V, P^*)(x) = 1] - \Pr[(V^{h_z}, P^*)(x) = 1] > 2\varepsilon$.

Accordingly, the prover first declares x and whether case (0) or case (1) holds and then proceeds to prove its claim via an interactive proof. Specifically, on common input $(1^n, z)$ the parties invoke the following $(k+1)$ -move protocol.

1. The prover finds an input $x \in \{0, 1\}^n$ and a proof strategy P^* such that Case $c \in \{0, 1\}$ holds.
The prover sends x and c .
2. The verifier samples two strings, $r_0 \leftarrow U_R$, $r_1 \leftarrow h_z(U_S)$ and a random bit $b \in \{0, 1\}$.
The two parties invoke an interactive protocol where the prover plays $P^*(x)$ and the verifier plays $V(x; r_b)$.
Let $v \in \{0, 1\}$ denote the output (acceptance bit) of $V(x; r_b)$.
3. The verifier accepts if $b = v \oplus c$.

Completeness: Assume that h_z does not strongly 2ε -fool $\langle P, V \rangle$ and let us assume that case (0) holds. (The other case is proved symmetrically.) Then, the probability that the verifier accepts is

$$\frac{1}{2} \Pr[(P^*, V^{h_z})(x) = 1] + \frac{1}{2} (1 - \Pr[(P^*, V)(x) = 1]) \geq \frac{1}{2} + \varepsilon.$$

Soundness: Fix some no instance z for which h_z strongly ε -fool $\langle P, V \rangle$. We analyze the acceptance probability of the verifier when interacting with a cheating prover. Fix an arbitrary first message (x, c) of the prover and let us denote by P^* the strategy that the prover plays in Step 2 of the protocol. Since h_z strongly ε -fool $\langle P, V \rangle$, it holds that the difference between the quantities

$$q = \Pr[(P^*, V)(x) = 1] \quad \text{and} \quad q_z = \Pr[(P^*, V^{h_z})(x) = 1]$$

is at most ε in absolute value. Suppose that $c = 0$ (the other case is symmetric). Then, the verifier accepts with probability

$$\frac{1}{2} q_z + \frac{1}{2} (1 - q) \leq 1/2 + \varepsilon/2,$$

as required.

Overall, the protocol has completeness of $1/2 + \varepsilon$ and soundness of $1/2 + \varepsilon/2$. Since $\varepsilon = \Omega(1/\text{poly}(n))$, we can use standard parallel amplification theorems to reduce the error (cf. [16, Appendix A]). This completes the proof of the first part of the lemma. The “Consequently” part, follows from the equivalence between constant-round IP protocols and AM proofs [25, 4]. ◀

We will make use of the following result.

► **Theorem 22** (PRGs against AM/poly [27, 28, 36]). *Suppose that $E = \text{DTime}(2^{O(n)})$ is hard for exponential-size non-deterministic circuits¹⁰, i.e., there exists a language L in E and a constant $\beta > 0$, such that for every sufficiently large n , circuits of size $2^{\beta n}$ fail to compute the characteristic function of L on inputs of length n .*

Then for every polynomial $T(\cdot)$ and inverse polynomial $\varepsilon(\cdot)$, there exists a pseudo-random generator G that stretches seeds of length $\rho = O(\log m)$ into a string of length m in time $\text{poly}(m)$ such that G ε -fools every promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ that admits an AM/poly proof system with a T -size verifier in the following sense. For every sufficiently large m and $b \in \{\text{yes}, \text{no}\}$

$$\left| \Pr_{z \leftarrow U_m} [z \in \Pi_b] - \Pr_{z \leftarrow G(U_\rho)} [z \in \Pi_b] \right| \leq \varepsilon(m).$$

By combining the above theorem with Lemma 21, we derive the following corollary.

► **Corollary 23** (uniform PRG against constant-round IP protocols). *Under the assumption of Theorem 22 for every polynomials $T_V(n), C_V(n), C_P(n), R(n) : \mathbb{N} \rightarrow \mathbb{N}$, constant $k \in \mathbb{N}$ and inverse polynomial $\varepsilon : \mathbb{N} \rightarrow [0, 1]$ there exists a polynomial-time computable PRG that strongly ε -fools non-uniform $\text{IP}_k[R, C_V, C_P, T_V]$ with seed length of $2C_V + O(\log n)$.*

Proof. Let $\varepsilon' = \varepsilon/4$. Fix some non-uniform $\langle P, V \rangle$ interactive proof in $\text{IP}_k[R, C_V, C_P, T_V]$ and let $\Pi = \Pi_{P, V, \varepsilon'}$ denote the corresponding promise problem defined in Definition 20. Recall that

$$\mathcal{H} = \left\{ h_z : \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)} \right\}_{z \in \{0, 1\}^{Z(n)}}$$

is a family of t -wise independent hash functions where $t = O(T_V \log T_V + \log(1/\varepsilon'))$ and $S = 2C_V + 2 \log(1/\varepsilon') + \log T_V + \log \log T_V + \log \log(1/\varepsilon') + O(1)$ that can be evaluated by a $\text{poly}(n)$ -time universal evaluation algorithm $H : \{0, 1\}^{Z(n)} \times \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)}$. As shown in Lemma 21, the promise problem Π is in AM/poly. Let us denote by $T(n)$ the time complexity of the verifier in the corresponding proof system (and recall that $T = O(T_V + T_H)$ and so it depends only on ε, R, C_V, C_P and T_V). Let $G' : \{0, 1\}^{\rho(n)} \rightarrow \{0, 1\}^{Z(n)}$ be the PRG that ε' -fools AM/poly problems with T -time verifiers as promised in Theorem 22. Recall that $\rho(n) = O(\log Z(n)) = O(\log n)$.

We define the PRG against non-uniform $\text{IP}_k[R, C_V, C_P, T_V]$ that maps a random seed of length $\rho(n) + S(n)$ into a pseudorandom string of length $R(n)$ as follows. Given a seed (s_1, s_2) where $s_1 \in \{0, 1\}^{\rho(n)}$ and $s_2 \in \{0, 1\}^{S(n)}$, output $H(G'(s_1), s_2) = h_{G'(s_1)}(s_2)$. Note that PRG is indeed efficiently computable and that its definition depends only in the parameters

¹⁰ A non-deterministic circuit C has additional “non-deterministic input wires”. Such a circuit evaluates to 1 on x if and only if there exist an assignment to the non-deterministic input wires that makes C output 1 on x . Non-Deterministic circuits can be therefore viewed as a non-uniform version of the class NP.

4:16 On the Randomness Complexity of Interactive Proofs & SZK Proofs

R, C_V, C_P, T_V and ε . We prove that PRG strongly ε -fools $\langle P, V \rangle$. For this it suffices to show that, except with probability $\varepsilon/2$, over the choice of s_1 , it holds that $h_{G'(s_1)}(s_2)$ strongly ε -fools $\langle P, V \rangle$. Indeed,

$$\Pr_{s_1}[G'(s_1) \in \Pi_{\text{no}}] \geq \Pr_z[z \in \Pi_{\text{no}}] - \varepsilon' \geq 1 - 2\varepsilon' \geq 1 - \varepsilon/2$$

where the first inequality follows from the pseudo-randomness of G' and the second inequality follows from Theorem 16. The corollary follows. \blacktriangleleft

Theorem 4 follows immediately from Corollary 23

5 Zero Knowledge Proofs

In this section we study the problem of randomness sparsification for zero-knowledge proof systems.

5.1 SZK proof systems

We begin by noting that PRG-based sparsification trivially preserves zero-knowledge against malicious verifier.

► **Observation 24.** *If $\langle P, V \rangle$ is a constant-round SZK proof system and G ε -fools $\langle P, V \rangle$ then $\langle P, V^G \rangle$ is an SZK proof system whose soundness error and completeness error increase by ε . Moreover, if $\langle P, V \rangle$ has perfect completeness then so is $\langle P, V^G \rangle$.*

Proof. By Observation 13, the system $\langle P, V^G \rangle$ is an interactive proof system with the desired parameters. Since zero-knowledge against cheating verifier is a property of P the new system is also zero-knowledge. \blacktriangleleft

By combining the above observation with Corollary 23 we derive Theorem 7.

5.2 Semi-Malicious SZK Proof Systems

We move on to study sparsification for semi-malicious SZK proof systems. We begin by introducing this new variant of zero-knowledge.

► **Definition 25** (F semi-malicious SZK). *Let $F : \mathbb{N} \rightarrow \mathbb{N}$ be an integer valued function and let $\langle P, V \rangle$ be a proof system with randomness complexity of R for a promise problem Π . Let $D(1^n; s)$ be an efficiently-computable algorithm that given randomness s outputs $F(n)$ bits. Define the verifier $V_D(x)$ as follows:*

- Sample random coins s for D , and compute the $F(|x|)$ -bit string $f = D(1^{|x|}, s)$.
- Sample $r' \leftarrow \{0, 1\}^{R(|x|) - F(|x|)}$.
- invoke $V(x)$ on the concatenated random tape $f \circ r'$.

Let $\mu(D)$ denote the completeness error of the proof system $\langle P, V_D \rangle$ with respect to Π , and let $\text{view}_{V_D}(x)$ denote the random variable that corresponds to the view of the verifier $V_D(x)$ when interacting with P on a common input x .

We say that $\langle P, V \rangle$ is F semi-malicious zero-knowledge proof system with zero-knowledge error of δ_z , abbreviated (F, δ_z) -SMSZK, if for every efficiently-computable algorithm $D(1^n; s)$ there exists a simulator Sim_D that runs in expected polynomial-time such that for every yes instance x ,

$$\text{SD}(\text{Sim}_D(x), \text{view}_{V_D}(x)) \leq \delta_z + \mu(D). \quad (1)$$

By default, we assume that δ_z is negligible and in this case we refer $\langle P, V \rangle$ as F -SMSZK proof system. The notion of F semi-malicious perfect zero-knowledge proof system (F -SMPZK is short) is defined analogously, except that the simulator's deviation in (1) must be zero. We refer to the F -bit prefix of the verifier's tape as the accessible bits.

► **Remark 26** (On the additive term $\mu(D)$). One could consider a more restrictive definition of F -SMSZK in which the deviation of the simulator Sim_D is bounded by δ_z regardless of the completeness error $\mu(D)$ of D . While our reductions are compatible with this alternative variant as well, we choose to employ the current definition since it is more liberal. Further note that the additive term $\mu(D)$ intuitively allows the simulator to deviate when the protocol outputs non-accepting transcripts. Thus, one can roughly think of our definition as restricting the attention to semi-malicious distributions D that put most of their mass on strategies for which completeness hold.

Observe that $0 \leq F \leq R$ and that any HVSZK proof system is also an 0-SMSZK and every SZK proof system is R -SMSZK. On the other hand, as mentioned in Section 1.2.2, the classical HVSZK proof system for the complete *statistical-distance* problem of [35] can be shown to have maximal accessible bit complexity of $F = R$ too. (See the full version.) Thus, even R -SMSZK complexity is a weaker notion than SZK complexity.

► **Remark 27**. One can use a more general definition in which the “accessible bits” are not necessarily the first ones and can be taken to be any set of $F(|x|)$ indices that can be efficiently computable and possibly depend on the input x itself. However, in this case one can always modify the verifier (by pre-permuting the random tape) and make sure that the accessible bits are located in the first $F(|x|)$ indices.

► **Remark 28**. Typical SMSZK systems (e.g., for statistical-distance [35] or for GNI [19]) satisfy the following stronger definition. There exists a “universal” simulator Sim such that for every yes instance x and every fixing $f \in \{0, 1\}^{F(|x|)}$ of the first $F(|x|)$ bits of the verifier, the distribution $\text{Sim}(x, f)$ is $(\delta_z + \mu(f))$ -close, in statistical-distance, to the view of V_f when interacting with P on the input x , where V_f denotes the verifier that given an input x and a random tape $r' \leftarrow \{0, 1\}^{R(|x|) - F(|x|)}$ invokes $V(x)$ on the concatenated random tape $f \circ r'$.

5.3 SMSZK: Randomness vs. Prover's Communication/CRS

► **Theorem 29**. Let $\langle P, V \rangle$ be an (F, δ_z) -SMSZK $_k[R, C_V, C_P, T_V]$ proof system for the promise problem Π . Suppose that $G : \{0, 1\}^S \rightarrow \{0, 1\}^R$ ε -fools non-uniform $\text{IP}[R, C_V, C_P, T_V]$ protocols. Consider the following proof system $\langle P', V' \rangle$ that on shared input x of length n proceeds as follows:

1. P' sends a random message a of length $R - F$ where $R = R(n)$ and $F = F(n)$.
2. The verifier reads his random tape $s \leftarrow U_{S(n)}$, computes $r_2 = G(s)$, expands a to an R -bit string $r_1 = 0^F \circ a$ and sets $r = r_1 \oplus r_2$. From now on, the prover plays $P(x)$ and verifier plays $V(x; r)$.

Then, $\langle P', V' \rangle$ is an HVSZK $_{k+1}$ proof system with zero-knowledge error of $\delta_z + \varepsilon$ and an ε additive penalty in the correctness and soundness error.

Proof. We begin by showing that $\langle P', V' \rangle$ is an IP_{k+1} proof system for Π . For any fixing of $a \in \{0, 1\}^{R-F}$, define the proof system $\langle P_a, V_a \rangle$ in which the verifier expands a to r_1 like in the above description, samples r_2 uniformly and calls $V(x; r_1 \oplus r_2)$ and the prover operates as before. Clearly, the soundness and correctness of this system is the same as the original one. Next, define the a -residual proof system $\langle P'_a, V'_a \rangle$ which is identical to the *sparsified* system $\langle P', V' \rangle$ except that a is hard-coded into V' who skips the first step of the

above protocol. The proof system $\langle P'_a, V'_a \rangle$ is the G -sparsified version of $\langle P_a, V_a \rangle$, and since G ε -fools non-uniform proof systems, the system $\langle P'_a, V'_a \rangle$ is sound and complete (with an additive error of ε). Since this is true for every choice of a , it follows that $\langle P', V' \rangle$ is an IP_{k+1} proof system for Π .

Let $D(1^n; s)$ be the algorithm that samples $s \leftarrow U_{S(n)}$ and outputs the $F(n)$ -bit prefix of $G(s)$, and let Sim_D denote the simulator of the original F -SMSZK $_k[R, C_V, C_P, T_V]$ proof system with respect to the distribution $D(1^n)$. We define a simulator Sim' for $\langle P', V' \rangle$ that, on an input x of length n , operates as follows:

1. Let $S = S(n)$, $R = R(n)$ and $F = F(n)$. Invoke $\text{Sim}_D(x)$ and sample a view (x, s', α, c') where s' is the S -bit seed sampled for D , $\alpha \in \{0, 1\}^{R-F}$ form the uniform part of the verifier's random tape, and c' is the (simulated) sequence of incoming messages.
2. Compute $r'_2 = G(s)$ and set $a' \in \{0, 1\}^{R-F}$ to be the XOR of α with the $(R - F)$ -bit suffix of r'_2 .
3. Output the tuple (x, s', a', c') .

Fix a yes instance x . We analyze the statistical distance between the simulated tuple (x, s', a', c') and the “real” tuple (x, s, a, c) that corresponds to the distribution of the real view of V' when interacting with P . It suffices to show that if the original simulator is perfect the two distributions are identical. (Indeed, since the new simulator makes a single call to the original simulator, a deviation of $\delta_z + \varepsilon$ of the original simulator can increase the statistical distance of the new one by at most $\delta_z + \varepsilon$.)

First observe that in both experiments s and s' are distributed uniformly. Fix some value for $s = s'$, and consider the conditional distributions $[(a', c')|s']$ and $[(a, c)|s]$. Next observe that a is uniform and that a' is uniform as well (since α is uniform). Finally, conditioned on $(s, a) = (s', a')$ the transcript c is sampled according to the experiment $\langle P, V \rangle(x; r)$ where $r = (0^F \circ a) \oplus G(s)$ and similarly the simulated transcript c' is sampled according to the experiment $\langle P, V \rangle(x; r')$ where $r' = (0^F \circ a') \oplus G(s') = (G(s')[1 : F] \circ \alpha)$ and so the tuples are identically distributed. \blacktriangleleft

By combining Theorem 29 with Corollary 23 we derive the following corollary which implies the first part of Theorem 8 from the introduction.

► **Corollary 30** (Trading randomness with prover's communication for SMSZK). *Suppose that E is hard for exponential size non-deterministic circuits. Then, for every inverse polynomial ε , every constant-round (F, δ_z) -SMSZK $_k[R, C_V, C_P, T_V]$ proof system can be transformed into a new*

$$\text{HVSZK}_{k+1}[R' = 2C_V + O(\log n), C_V, T'_V = \tilde{O}(T_V \cdot (R + \log n)), C'_P = C_P + R - F]$$

system with an additive penalty of ε in the soundness and completeness error and an additive penalty of $\varepsilon + \delta_z$ in the simulation error. Specifically, the new protocol consists of an additional preliminary message from the prover that consists of a random string of length $R - F$ bits. Moreover, the transformation preserves perfect completeness, and if the original proof system is semi-malicious perfect zero-knowledge then the resulting scheme admits a perfect simulation (i.e., it is HVPZK $_{k+1}[R', C_V, T'_V, C'_P]$).

► **Remark 31.** Corollary 30 can be converted to a statement regarding HVSZK in the common reference string model by replacing the first message of the prover with a common reference string ρ . This CRS can be chosen by the prover (a malicious choice does not affect the soundness). However, the CRS is *not* reusable among several invocations.

5.4 SMPZK: Randomness vs. Simulation Complexity

In the perfect setting, SMPZK proof systems can be sparsified at the expense of slowing-down the simulation by a factor of 2^{R-F} .

► **Lemma 32.** *Let $\langle P, V \rangle$ be an F -SMPZK proof system for a promise problem Π . Let $G : \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)}$ be a $\text{poly}(n)$ -time computable function that ε -fools $\langle P, V \rangle$. Then the protocol $\langle P, V^G \rangle$ is a proof system with an additive penalty of ε in soundness and completeness errors that has a perfect honest-verifier simulator Sim' with expected running-time of $(\text{poly}(n))2^{R(n)-F(n)}$.*

Proof. Let $D(1^n; s)$ be the algorithm that samples $s \leftarrow U_{S(n)}$ and outputs the $F(n)$ -bit prefix of $G(s)$, and let Sim_D denote the simulator of the original F -SMPZK proof system with respect to the distribution $D(1^n)$. The view of V^D in interaction with P over a yes-instance $x \in \{0, 1\}^n$ is parsed into (x, s, β, c) where $s \leftarrow U_{S(n)}$, $\beta \leftarrow U_{R(n)-F(n)}$ and c is the vector of incoming messages. We define a new simulator $\text{Sim}'(x)$ as follows: (1) Sample $(x; s', \beta', c')$ by invoking $\text{Sim}_D(x)$ (2) If the last $R(n) - F(n)$ bits of $G(s')$ equal to β' output the transcript $(x; s', \beta', c')$ and halt; otherwise, goto (1).

Since β' is uniformly distributed, at each iteration Sim' halts with probability $2^{F(n)-R(n)}$, and so the expected running time is $\text{poly}(n)2^{R(n)-F(n)}$. Perfect simulation follows by noting that (s', β') are distributed identically to the random tape of V^G and that conditioned on every fixing of these coins, (s, β) , the simulated transcript c' is distributed just like a real interaction between $P(x)$ and $V^G(x; s, \beta)$ (since Sim_D is a perfect simulator). ◀

By combining Lemma 32 with Corollary 23 we derive the following corollary which implies the second part of Theorem 8 from the introduction.

► **Corollary 33.** *Assuming that E is hard for exponential size non-deterministic circuits, let $\varepsilon : \mathbb{N} \rightarrow [0, 1]$ be an inverse polynomial and $R, C_V, C_P, T_V : \mathbb{N} \rightarrow \mathbb{N}$ be polynomially bounded functions where $C_v = \omega(\log n)$. Suppose that the promise problem Π admits a constant-round F -SMPZK $_k[R, C_V, C_P, T_V]$ proof system. Then Π admits an $\text{IP}_k[R' = 2C_V + O(\log n), C_V, C_P, T_V]$ proof system with an honest-verifier perfect simulator that runs in expected time of $\text{poly}(n)2^{R-F}$ and with ε penalty in the soundness and completeness errors.*

5.5 HVPZK: Randomness vs. Simulation Complexity

Corollary 33 shows that F -SMPZK systems can be sparsified with a simulation slow-down of 2^{R-F} . In this section we describe a different simulation strategy that yields a slow-down of 2^{R-S} where S is the seed-length of the PRG. This holds even when $F = 0$, i.e., for HVPZK proof systems. This theorem is based on a PRG that satisfies some additional features (e.g., regularity and the existence of an efficient inversion algorithm). We later show that our PRGs meet these requirements.

► **Definition 34.** *We say that a function $G : \{0, 1\}^S \rightarrow \{0, 1\}^R$ is δ -regular if $G(U_S)$ is δ -close in statistical distance to $U(\text{Image}(G))$, the uniform over the image of G . (In particular, a 0-regular function maps the same number of inputs to each of its outputs.) A uniform inversion algorithm for G is a randomized algorithm that given an input $y \in \{0, 1\}^R$ outputs \perp if y is not in the image of G , and, otherwise, outputs a uniformly chosen preimage of y under G .*

► **Lemma 35.** *Let $\langle P, V \rangle$ be an HVPZK proof system for a promise problem Π whose simulator Sim runs in time T_{Sim} . Let $G : \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)}$ be a $\text{poly}(n)$ -time computable function that ε -fools $\langle P, V \rangle$, can be uniformly inverted in expected time of $T_{G^{-1}}$, and is δ -regular.*

Then the protocol $\langle P, V^G \rangle$ is a proof system with an additive penalty of ε in soundness and completeness errors and with an honest-verifier simulator Sim' with statistical deviation of δ and expected running-time of $(T_{\text{Sim}} + T_{G^{-1}}) \frac{2^R}{|\text{Image}(G)|}$.

Proof. For a given instance x , the view of the original verifier V can be parsed to (x, r, v) where r is the randomness and v is the transcript. Let us parse the view of V^G (in an interaction $(P, V^G)(x)$) as a tuple (x, s, r, v) where s is the seed $r = G(s)$ and v is the transcript v . (While r is redundant it will be useful to keep it as part of the view.) The simulator $\text{Sim}'(x)$ does the following: (1) Sample (r', v') by calling $\text{Sim}(x)$; (2) Call the G -inverter on r' and denote its output by s' . If the output is \perp output \perp ; otherwise, output the tuple (x, s', r', v') .

Let us analyze the statistical deviation of Sim' . Fix some yes instance x and consider the distribution (x, s, r, v) in the real interaction $(P, V^G)(x)$. Observe that, conditioned on $r = r'$ the simulated tuple (x, s', r', v') is distributed identically to the real distribution (x, s, r, v) . Indeed, in both cases s is uniform preimage of r and v is a random transcript that corresponds to an interaction between $P(x)$ and $V(x; G(s))$. Therefore, the statistical distance between the simulated view (conditioned on not outputting \perp) and the real view is exactly the statistical distance between $r = G(U_S)$ and $r' = U(\text{Image}(G))$ which is at most δ since G is δ -regular. Finally, observe that the success probability (that r' hits $\text{Image}(G)$) is exactly $|\text{Image}(G)|/2^R$, and so the expected number of iterations is $2^R/|\text{Image}(G)|$ as required. \blacktriangleleft

We move on and show that our PRGs are invertible and almost-uniform.

► **Proposition 36.** *Let $k \in \mathbb{N}$ be a constant, $R, C_V, C_P, T_V : \mathbb{N} \rightarrow \mathbb{N}$ polynomially-bounded functions and $\varepsilon : \mathbb{N} \rightarrow [0, 1]$ be an inverse polynomial. Let $G : \{0, 1\}^{S(n)} \rightarrow \{0, 1\}^{R(n)}$ be the uniform PRG (resp., non-uniform PRG) that ε -fools non-uniform $\text{IP}_k[R, C_V, C_P, T_V]$ (resp., non-uniform $\text{IP}[R, C_V, C_P, T_V]$) that is promised by Corollary 23 (resp., Corollary 18). Then G is $(\text{poly}(n)2^{-S(n)})$ -regular, the image of G , on n -bit inputs, consists of at least $2^{S(n)}/\text{poly}(n)$ strings and there is an algorithm that, given the description of G , uniformly inverts G in expected $\text{poly}(n)$ time.*

Proof. We begin with the non-uniform version of G (from Corollary 18). As explained in Remark 17, G is defined by some degree- t univariate polynomial $h_z : \mathbb{F} \rightarrow \mathbb{F}$ over the field $\mathbb{F} = GF(2^R)$ and $t = \text{poly}(n)$. To compute G on an input $x \in \{0, 1\}^S$, we map x to a field element (by padding with $R - S$ zeroes) and output the evaluation of h_z on the padded-version of x .

Let y be a string in the image of G . First observe that the number of preimages under G is at most t since the polynomial $h_{z,y} = h_z(x) - y$ is of degree $t = \text{poly}(n)$. Hence, $|\text{Image}(G)| \geq 2^S/\text{poly}(n)$ and $G(U_S)$ samples every element $y \in \text{Image}(G)$ with probability $p_y \in [1/|\text{Image}(G)|, t/|\text{Image}(G)|]$. Since $U(\text{Image}(G))$ samples each element from $\text{Image}(G)$ with weight $1/|\text{Image}(G)|$, it follows that G is δ regular for $\delta = O(t/|\text{Image}(G)|) = \text{poly}(n)/2^{S(n)}$.

Next, observe that there exists a randomized algorithm A that given y lists in expected time of $T_A = \text{poly}(t, R) = \text{poly}(n)$ all the pre-images of y under G . (This can be done, for example, by factoring $h_{z,y}$ to its irreducible components via the algorithm of [9] and by noting that, for each root a of $h_{z,y}$, the polynomial $x - a$ must appear in the factorization.) We can therefore sample a random preimage in expected-polynomial time.

We move on to the uniform setting. Recall that in this setting (Corollary 23), the PRG G is defined as follows: (1) Sample a short seed s_1 of length $O(\log n)$ and a long seed s_2 of length $S - O(\log n)$; (2) Feed the short seed s_1 into a PRG G_1 that fools AM/poly languages

(with properly chosen parameters) and use the resulting string $z = G_1(s_1)$ to select a degree- t univariate polynomial $h_z : \mathbb{F} \rightarrow \mathbb{F}$ over the field $\mathbb{F} = GF(2^R)$ where $t = \text{poly}(n)$ as before; (3) Output $h_z(s_2)$.

It follows that each point in the image of G has at most $t \cdot |\text{Image}(G_1)| \leq \text{poly}(n)$ preimages. Therefore, $|\text{Image}(G)| \geq 2^{S(n)}/\text{poly}(n)$ and G is δ -regular for $\delta = O(\text{poly}(n)/2^S)$. Finally, in order to uniformly invert $y \in \text{Image}(G)$ we compute, for every s_1 , the list $L_{s_1} = \{(s_1, s_2) : h_{G_1(s_1)}(s_2) = y\}$ (using the aforementioned algorithm for h_z where $z = G_1(s_1)$), and then sample a preimage (s_1, s_2) uniformly from the union of all these (polynomially-many) lists. The expected running time is $O(2^{|s_1|}\text{poly}(n)) = \text{poly}(n)$, as required. ◀

By combining Lemma 35 and Proposition 36, we derive the following corollary.

► **Corollary 37.** *Assuming that E is hard for exponential size non-deterministic circuits, let $\varepsilon : \mathbb{N} \rightarrow [0, 1]$ be an inverse polynomial and $R, C_V, C_P, T_V : \mathbb{N} \rightarrow \mathbb{N}$ be polynomially bounded functions where $C_v = \omega(\log n)$. Suppose that the promise problem Π admits a constant-round HVPZK $_k[R, C_V, C_P, T_V]$ proof system. Then Π admits an $\text{IP}_k[R' = 2C_V + O(\log n), C_V, C_P, T_V]$ proof system with an honest-verifier simulator with negligible deviation error and expected running time of $\text{poly}(n)2^{R-S}$.*

References

- 1 Benny Applebaum, Sergei Artemenko, Ronen Shaltiel, and Guang Yang. Incompressible functions, relative-error extractors, and the power of nondeterministic reductions. *computational complexity*, 25(2):349–418, 2016.
- 2 Benny Applebaum and Prashant Nalini Vasudevan. Placing conditional disclosure of secrets in the communication complexity universe. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019*,, pages 4:1–4:14, 2019.
- 3 Sergei Artemenko and Ronen Shaltiel. Pseudorandom generators with optimal seed length for non-boolean poly-size circuits. *ACM Transactions on Computation Theory (TOCT)*, 9(2):1–26, 2017.
- 4 László Babai and Shlomo Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. Syst. Sci.*, 36(2):254–276, 1988. doi:10.1016/0022-0000(88)90028-1.
- 5 Boaz Barak, Shien Jin Ong, and Salil Vadhan. Derandomization in cryptography. In *Annual International Cryptology Conference*, pages 299–315. Springer, 2003.
- 6 Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Randomness in interactive proofs. *Computational Complexity*, 3(4):319–354, 1993.
- 7 Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 276–287. IEEE Computer Society, 1994. doi:10.1109/SFCS.1994.365687.
- 8 Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo random bits. In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 112–117. IEEE Computer Society, 1982. doi:10.1109/SFCS.1982.72.
- 9 David G Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, pages 587–592, 1981.
- 10 J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. *Journal of computer and system sciences*, 18(2):143–154, 1979.
- 11 Andrew Drucker. Nondeterministic direct product reductions and the success probability of sat solvers. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 736–745. IEEE, 2013.

- 12 Bella Dubrov and Yuval Ishai. On the randomness complexity of efficient sampling. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 711–720, 2006.
- 13 Shimon Even, Alan L. Selman, and Yacov Yacobi. The complexity of promise problems with applications to public-key cryptography. *Inf. Control.*, 61(2):159–173, 1984. doi:10.1016/S0019-9958(84)80056-X.
- 14 Uriel Feige and Carsten Lund. On the hardness of computing the permanent of random matrices. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 643–654, 1992.
- 15 Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 416–426. ACM, 1990. doi:10.1145/100216.100272.
- 16 Oded Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, volume 17 of *Algorithms and Combinatorics*. Springer, 1998. URL: <http://link.springer.com/book/10.1007/978-3-662-12521-2/page/1>.
- 17 Oded Goldreich. On promise problems: A survey. In Oded Goldreich, Arnold L. Rosenberg, and Alan L. Selman, editors, *Theoretical Computer Science, Essays in Memory of Shimon Even*, volume 3895 of *Lecture Notes in Computer Science*, pages 254–290. Springer, 2006. doi:10.1007/11685654_12.
- 18 Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4):205–214, 1998.
- 19 Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.
- 20 Oded Goldreich, Amit Sahai, and Salil Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 399–408, 1998.
- 21 Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1-2):1–53, 2002.
- 22 Oded Goldreich and Avi Wigderson. Derandomization that is rarely wrong from short advice that is typically good. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 209–223. Springer, 2002.
- 23 Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. doi:10.1016/0022-0000(84)90070-9.
- 24 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.
- 25 Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 59–68, 1986.
- 26 Pavel Hubáček, Alon Rosen, and Margarita Vald. An efficiency-preserving transformation from honest-verifier statistical zero-knowledge to statistical zero-knowledge. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–87. Springer, 2018.
- 27 Russell Impagliazzo and Avi Wigderson. P= bpp if e requires exponential circuits: Derandomizing the xor lemma. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 220–229, 1997.
- 28 Adam R Klivans and Dieter Van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002.
- 29 Alex Lombardi, Willy Quach, Ron D. Rothblum, Daniel Wichs, and David J. Wu. New constructions of reusable designated-verifier nizks. In Alexandra Boldyreva and Daniele

- Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 670–700. Springer, 2019. doi:10.1007/978-3-030-26954-8_22.
- 30 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992. doi:10.1145/146585.146605.
 - 31 Ilan Newman. Private vs. common random bits in communication complexity. *Information processing letters*, 39(2):67–71, 1991.
 - 32 Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of computer and System Sciences*, 49(2):149–167, 1994.
 - 33 Koji Nuida and Goichiro Hanaoka. On the security of pseudorandomized information-theoretically secure schemes. *IEEE transactions on information theory*, 59(1):635–652, 2012.
 - 34 Yair Oren. On the cunning power of cheating verifiers: Some observations about zero knowledge proofs. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 462–471. IEEE, 1987.
 - 35 Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *Journal of the ACM (JACM)*, 50(2):196–249, 2003.
 - 36 Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *computational complexity*, 15(4):298–341, 2006.
 - 37 Adi Shamir. $IP = PSPACE$. *J. ACM*, 39(4):869–877, 1992. doi:10.1145/146585.146609.
 - 38 Joseph H Silverman. Fast multiplication in finite fields $GF(2^n)$. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 122–134. Springer, 1999.
 - 39 Luca Trevisan and Salil Vadhan. Extracting randomness from samplable distributions. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 32–42. IEEE, 2000.
 - 40 Salil Pravin Vadhan. *A study of statistical zero-knowledge proofs*. PhD thesis, Massachusetts Institute of Technology, 1999.
 - 41 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91. IEEE Computer Society, 1982. doi:10.1109/SFCS.1982.45.

Line-Point Zero Knowledge and Its Applications

Samuel Dittmer ✉

Stealth Software Technologies Inc., Los Angeles, CA, USA

Yuval Ishai ✉

Department of Computer Science, Technion, Haifa, Israel

Rafail Ostrovsky ✉

Department of Computer Science and Mathematics, University of California, Los Angeles, CA, USA

Abstract

We introduce and study a simple kind of proof system called *line-point zero knowledge* (LPZK). In an LPZK proof, the prover encodes the witness as an affine line $\mathbf{v}(t) := \mathbf{a}t + \mathbf{b}$ in a vector space \mathbb{F}^n , and the verifier queries the line at a single random point $t = \alpha$. LPZK is motivated by recent practical protocols for *vector oblivious linear evaluation* (VOLE), which can be used to compile LPZK proof systems into lightweight designated-verifier NIZK protocols.

We construct LPZK systems for proving satisfiability of arithmetic circuits with attractive efficiency features. These give rise to designated-verifier NIZK protocols that require only 2-5 times the computation of evaluating the circuit in the clear (following an input-independent preprocessing phase), and where the prover communicates roughly 2 field elements per multiplication gate, or roughly 1 element in the random oracle model with a modestly higher computation cost. On the theoretical side, our LPZK systems give rise to the first *linear interactive proofs* (Bitansky et al., TCC 2013) that are zero knowledge against a malicious verifier.

We then apply LPZK towards simplifying and improving recent constructions of *reusable non-interactive secure computation* (NISC) from VOLE (Chase et al., Crypto 2019). As an application, we give concretely efficient and reusable NISC protocols over VOLE for *bounded inner product*, where the sender's input vector should have a bounded L_2 -norm.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases Zero-knowledge proofs, NIZK, correlated randomness, vector oblivious linear evaluation, non-interactive secure computation

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.5

Related Version *Full Version*: <https://eprint.iacr.org/2020/1446> [23]

Funding Supported by DARPA Contract No. HR001120C0087. Y. Ishai supported in part by ERC Project NTSC (742754), NSF-BSF grant 2015782, BSF grant 2018393, and ISF grant 2774/20. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA. Distribution Statement “A” (Approved for Public Release, Distribution Unlimited).

1 Introduction

Zero-knowledge proofs, introduced by Goldwasser, Micali, and Rackoff [29] in the 1980s, are commonly viewed as a gem of theoretical computer science. For many years, they were indeed confined to the theory domain. However, in the past few years we have seen explosive growth in research on concretely efficient zero-knowledge proof systems. This research is motivated by a variety of real-world applications. See [49] for relevant pointers.

Designated-verifier NIZK

There are many different kinds of zero-knowledge proof systems. Here we mainly consider the setting of *designated-verifier, non-interactive zero knowledge* (dv-NIZK), where the proof consists of a single message from the prover to the verifier, but verification requires a secret



© Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky;
licensed under Creative Commons License CC-BY 4.0

2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 5; pp. 5:1–5:24



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

5:2 Line-Point Zero Knowledge and Its Applications

verification key that is known only to the verifier and is determined during a (reusable) setup phase. Moreover, we consider by default computationally sound proofs, also known as *arguments*. Designated-verifier NIZK has a rich history starting from [39]; see [43, 40, 19] and references therein for recent works in the area. We will typically consider a more restrictive setting, sometimes referred to as *preprocessing NIZK*, where also the prover needs to hold secret information. In this variant of dv-NIZK the prover and the verifier engage in a (typically inexpensive and reusable) interaction during an offline preprocessing phase, before the inputs are known. In the end of the interaction the prover and the verifier obtain *correlated secret randomness* that is consumed by an online protocol in which the prover can prove multiple statements to the verifier. While this preprocessing model will be our default model for NIZK, our results are relevant to both kinds of dv-NIZK.

Efficiency of proof systems

We are primarily motivated by the goal of improving the *efficiency* of zero-knowledge proofs. There are several metrics for measuring efficiency of proof systems. Much of the research in this area focuses on improving *succinctness*, which refers both to the proof length and to the verifier’s running time. This is highly relevant to the case of publicly verifiable proofs that are generated once and verified many times. However, in the case of a proof that is verified once by a designated verifier, other complexity metrics, such as prover’s running time and space, can become the main performance bottlenecks. Indeed, state-of-the-art succinct proof systems, such as zk-SNARKs based on pairings [30] or IOPs [7], typically incur high concrete prover computation costs when scaled to large verification tasks. Moreover, they require a big amount of space, and are not compatible with a “streaming” mode of operation in which the proof is generated on the fly together with the computation being verified. On the other hand, non-succinct or semi-succinct proof systems based on the “MPC-in-the-head” [35, 27, 18, 37], garbled circuits [24, 31], or interactive proofs [28, 46, 48], scale better to big verification tasks.

Minimizing prover complexity

Our goal is to push the advantages of non-succinct zero-knowledge proof systems to their limits, focusing mainly on optimizing the *prover’s computation*. This can be motivated by settings in which the prover and the verifier are connected via a fast local network. An extreme case is that of physically connected devices, for which the distinction between computation and communication is blurred. Alternatively, one can think of scenarios in which the proofs can be generated and stored *offline* on the prover side and only verified at a later point, or possibly not at all. Another motivating scenario is one where the *statement* is short and simple, but is kept secret from the verifier. In this setting, which comes up in applications such as “commit-and-prove” and NISC on small inputs (which will be discussed later), the concrete overhead of “asymptotically succinct” systems is too high. Finally, if the witness is secret-shared between multiple provers and the proof needs to be generated in a distributed way, the prover’s computation is likely to become a bottleneck. All of the above reasons motivate a systematic study of minimizing the prover’s complexity in zero-knowledge proofs.

Achieving constant computational overhead

We consider the goal of zero-knowledge proofs with *constant computational overhead*, namely where the total computational cost (and in particular the prover’s computation) is only a constant times bigger than the cost of performing the verification in the clear. In the

case of proving the satisfiability of a Boolean circuit, this question is still open, and the best computational overhead is polylogarithmic in a statistical security parameter [20]. However, when considering arithmetic circuits over a big finite field \mathbb{F} and settling for $O(1/|\mathbb{F}|)$ soundness error, this goal becomes much easier. The first such proof system was given by Bootle et al. [11], who also achieved “semi-succinctness.” However, the underlying multiplicative constants are very big, and this system is not considered practical. A more practical approach uses variants of the GKR interactive proofs protocol [46, 48, 47]. Here the concrete computational overhead is smaller, but still quite big: roughly 20x overhead in the best-case scenario of “layered” arithmetic circuits. On top of that, this overhead is only relevant when the verification circuit is much bigger than the witness size. In some of the applications we consider (such as the NISC application discussed below), this will not be the case.

A third approach, which is most relevant to our work, relies on *oblivious linear evaluation* (OLE) [42, 36] and its vector variant (VOLE) [2]. An OLE is an arithmetic variant of oblivious transfer, allowing the receiver, on input α , to learn a linear combination $a\alpha + b$ of two ring elements held by the sender. VOLE is a natural vector analogue of OLE: the receiver learns $\mathbf{a}\alpha + \mathbf{b}$ for a pair of vectors \mathbf{a}, \mathbf{b} held by the sender. The idea of using *random* precomputed instances of OLE and VOLE towards zero-knowledge proofs with constant computational overhead was suggested in [12, 19]. This is motivated by recent techniques for securely realizing pseudorandom instances of (V)OLE with sublinear communication and good concrete overall cost [12, 13, 44, 16, 15]. However, these protocols for zero knowledge from (V)OLE still suffered from a high concrete overhead. For instance, the protocol from [19] requires 44 instances of OLE for each multiplication gate. Recent and concurrent works by Weng et al. [45] and Baum et al. [6] improved this state of affairs. We discuss these works and compare them to our own in the full version of this paper [23].

1.1 Our contribution

Motivated by the goal of minimizing prover complexity in zero-knowledge proofs, we introduce and study a simple kind of proof systems called *line-point zero knowledge*. We then apply this proof system towards obtaining simple, concretely efficient, and reusable protocols for *non-interactive secure computation*. We elaborate on these results below. We defer many proofs to the full version of our paper [23].

Line-point zero knowledge

A recent work of Boyle et al. [12], with improvements in [13, 44], has shown how to securely generate a long, pseudorandom instance of a vector oblivious linear evaluation (VOLE) correlation with low communication complexity (sublinear in the vector length) and good concrete efficiency. Here we show how to use this for implementing simple and efficient dv-NIZK protocols for circuit satisfiability, improving over similar protocols from [12, 19]. In particular, previous protocols involve multiple VOLE instances and have a large (constant) overhead in communication and computation compared to the circuit size.

The goal of reducing NIZK to a single instance of VOLE motivates the key new tool we introduce: a simple kind of information-theoretic proof system that we call *line point zero knowledge* (LPZK). In an LPZK proof, the prover P generates from the witness w (a satisfying assignment) an affine line $\mathbf{v}(t) := \mathbf{a}t + \mathbf{b}$ in an n -dimensional vector space \mathbb{F}^n . The verifier queries a single point $\mathbf{v}(\alpha) = \mathbf{a}\alpha + \mathbf{b}$ on this line, and determines whether to accept or reject. We call this proof system LPZK over \mathbb{F} of length (or dimension) n . We define the LPZK model formally along with more refined cost metrics in Section 2.1.

Information-theoretic LPZK construction

We start by showing the existence of an LPZK for arithmetic circuit satisfiability (an NP-complete problem), where the dimension n and computational costs scale linearly with the circuit size.

► **Theorem 1** (LPZK for arithmetic circuit satisfiability). *For any NP-relation $R(x, y)$ and finite field \mathbb{F} , there exists an LPZK system for R over \mathbb{F} with soundness error $O(1/|\mathbb{F}|)$. Concretely, in the case of proving the satisfiability of an arithmetic circuit C over \mathbb{F} , we have an LPZK over \mathbb{F} with dimension $n = O(|C|)$, soundness error $\varepsilon = O(1/|\mathbb{F}|)$, and where the prover and verifier can be implemented by arithmetic circuits of size $O(|C|)$.*

As an information-theoretic proof system, LPZK can be viewed as a simple instance of a (1-round) zero-knowledge *linear interactive proof* (LIP) [9], in which the verifier sends a single field element to the prover. Theorem 1 implies the first such system that is zero knowledge even against a *malicious verifier*.

From LPZK to NIZK over random VOLE

It is easy to convert an LPZK into an NIZK protocol in the *rVOLE-hybrid model*, namely with a trusted setup in which the prover P receives a *random* pair of vectors \mathbf{a}' , $\mathbf{b}' \in \mathbb{F}^n$, while the verifier V receives a random field element $\alpha \in \mathbb{F}$ and the vector $\mathbf{a}'\alpha + \mathbf{b}'$. This uses a standard reduction from VOLE to rVOLE; see Section 2.2 for details. We refer to the length of the vectors \mathbf{a}' , \mathbf{b}' as the *rVOLE length*.

The rVOLE setup, whose efficient implementation will be discussed later, allows the prover to compress the LPZK proof by eliminating entries that can be picked at random independently of the input. Using this and other optimizations, we obtain an information-theoretic NIZK protocol in the rVOLE-hybrid model with the following concrete efficiency features.

► **Theorem 2** (NIZK over a single random VOLE). *Fix an integer $t \geq 1$. There exists an (unconditional, perfect zero-knowledge) NIZK protocol in the rVOLE-hybrid model that proves the satisfiability of an arithmetic circuit C over a field \mathbb{F} , where C has k inputs, k' outputs and m multiplication gates, with the following security and efficiency features:*

- **Soundness error** $\varepsilon = 2t/|\mathbb{F}|$;
- **Communication** $k + k' + (2 + \frac{1}{t})m$ field elements from P to V ;
- **rVOLE length** $n = k + 2m$ field elements;
- **Computation** Assuming the cost of field additions is negligible compared to multiplications, the computation of the prover is less than 4 times the cost of evaluation in the clear, and the computation of the verifier is less than 5 times the cost of evaluation in the clear.

VOLE instantiations

The random VOLE required by Theorem 2 can be instantiated in a variety of ways. For instance, one could use a 2-message protocol in the CRS model based on Paillier’s encryption scheme, which yields *statistical* dv-NIZK arguments for NP from the DCRA assumption [19]. Other efficient VOLE implementations under different assumptions appear in [2, 22, 5]. In terms of asymptotic efficiency, random VOLE can be implemented with constant multiplicative computational overhead under plausible variants of the learning parity with noise (LPN) assumption over big fields [2, 12]. From a concrete efficiency viewpoint, the most appealing current VOLE implementations rely on pseudorandom correlation generators (PCGs) [12,

13, 44]. A PCG for VOLE enables a “silent” generation of a long random VOLE correlation by locally expanding a pair of short, correlated seeds. This local expansion can be done in near-linear or even linear time, and may be carried out in an offline phase before the statement is known. The secure generation of the correlated seeds can also be done by a concretely efficient, low-communication protocol. Optimized pseudorandom *function* analogs of PCG that enable random access to the outputs of a virtually unbounded VOLE correlation were recently considered in [15]. The above approaches generally lead to a *preprocessing* NIZK, where both the verifier and the prover are fixed in advance. However, using 2-round protocols for VOLE with security against malicious receivers [19, 13], LPZK can be compiled into dv-NIZK protocols in which the same (short) verifier message can be used by different provers.

1.2 Improving proof size in the random oracle model

Inspired by the concurrent¹ work of Weng et al. [45], we can improve the communication cost of our proofs in the random oracle model by a factor of 2 (asymptotically) at the cost of a modest increase of prover and verifier computation, in the form of calls to a cryptographic hash function. Note that other attractive features of LPZK such as space- and streaming-friendliness are maintained. See [23] for a detailed comparison between the results of [45] and our work.

► **Theorem 3** (NIZK over random VOLE in the ROM). *Fix an integer $r \geq 1$. There exists an (unconditional) NIZK protocol in the RO- r VOLE-hybrid model that proves the satisfiability of an arithmetic circuit C over a field \mathbb{F} , where C has k inputs and m multiplication gates and ℓ is the number of oracle calls a malicious prover P^* makes, with the following features:*

- **Soundness error** $\varepsilon = \frac{2}{|\mathbb{F}|} + \frac{\ell}{|\mathbb{F}|^r}$;
- **Communication** $k + k' + m + 2r$ field elements from P to V ;
- **rVOLE length** $n = k + m + r$ field elements;
- **Computation** Computation of $O(r|C|)$ field operations and 1 cryptographic hash call (from \mathbb{F}^m to \mathbb{F}^{mr}) for both the prover and the verifier.

1.3 Reusable NISC from LPZK via certified VOLE

A *non-interactive secure computation* (NISC) protocol [34] is a two-party protocol that securely computes a function $f(x, y)$ using two messages: a message by a *receiver*, encrypting its input x , followed by a message by a *sender*, that depends on its input y . The output $f(x, y)$ is only revealed to the receiver. A major challenge is making such protocols secure even when either party can be malicious. Another challenge is to make such protocols *reusable*, in the sense that the same encrypted input x can be used to perform computations with many sender inputs y_i without violating security. This should hold even when a malicious sender can learn partial information about the honest receiver’s output, such as whether the receiver “aborts” after detecting an inconsistent sender behavior. Existing NISC (or even NIZK) protocols based on parallel calls to oblivious transfer (OT) and symmetric cryptography [39, 34, 1, 41] are *not* fully reusable, and this is in some sense inherent [19].

Chase et al. [19] recently showed how to realize reusable NISC by using parallel instances of VOLE instead of OT. This can be seen as a natural extension of the LPZK model, where the receiver randomly encodes its NISC input x into multiple points α_i and the sender

¹ Most of the present work was done concurrently and independently of [45]. We explicitly point out the improvements that are based on ideas from [45].

randomly encodes its input y into corresponding lines $\mathbf{v}_i(t)$. Here reusability refers to fixing the VOLE inputs (points) α_i generated by an honest receiver on input x and reusing them in multiple interactions with a malicious sender.

On top of the reusability feature, another advantage of the VOLE-based protocol, which is inherited from earlier protocols with security against semi-honest senders [32, 3], is that it “natively” supports simple *arithmetic* computations over the VOLE field. This is contrasted with NISC protocols over OT [34, 1, 41], which apply to Boolean circuits and are expensive to adapt to arithmetic computations.

We provide an alternative construction of reusable NISC over VOLE that uses LPZK to protect against malicious senders. Our approach significantly simplifies the protocol from [19] and results in much better concrete constants.

NISC for bounded inner product

To illustrate the concrete efficiency potential of our NISC technique, we optimize it for a simple application scenario. Consider an “inner product” functionality that measures the level of similarity (or correlation) between receiver feature vector x and a sender feature vector y , where the same x can be reused with multiple sender inputs y_i . Here we view both x and y as integer vectors that are embedded in a sufficiently large finite field. An obvious problem is that the ideal functionality allows a malicious sender to scale its real input by an arbitrary multiplicative factor, thereby increasing the perceived similarity. To prevent this attack, we modify the functionality to bound the L_2 norm of the sender’s input. In this way, the sender’s strategy is effectively restricted to choosing the direction of a unit vector, where the bound on the norm determines the level of precision. For this *bounded inner product* functionality, we obtain a concretely efficient protocol that offers reusable malicious security. Even when considering malicious security alone, without reusability, previous techniques for NISC are much less efficient for such simple arithmetic functionalities. To give just one data point, for vectors of length 1000 over \mathbb{F} , with $|\mathbb{F}| \approx 2^{64}$ and sender L_2 norm bounded by 1024, our protocol requires 1002 instances of VOLE with a total of 21,023 entries and communication of 36,047 field elements (roughly 282 kB) after the offline generation of VOLE instances. Given recent methods for “silent” generation of multiple VOLE instances [13, 44, 16, 15], the amortized cost of setting up the required VOLE instances is small.

1.4 Overview of techniques

From LPZK to NIZK via random VOLE. An LPZK proof system can be directly realized by a single instance of VOLE, where the prover’s line $\mathbf{v}(t) := \mathbf{a}t + \mathbf{b} \in \mathbb{F}^n$ determines the VOLE sender’s input (\mathbf{a}, \mathbf{b}) and the verifier’s point α is used as the VOLE receiver’s input. A further observation is that this single VOLE instance can be easily reduced to a *random* VOLE functionality that assigns to the prover a uniformly random pair of vectors $(\mathbf{a}', \mathbf{b}')$ each in \mathbb{F}^n and to the verifier a uniformly random value $\alpha \in \mathbb{F}$ and $\mathbf{v}' = \mathbf{a}'\alpha + \mathbf{b}'$. Indeed, the prover can send $(\mathbf{a} - \mathbf{a}')$ and $(\mathbf{b} - \mathbf{b}')$ to the verifier, who computes $\mathbf{v}(\alpha) = \mathbf{v}' + (\mathbf{a} - \mathbf{a}')\alpha + (\mathbf{b} - \mathbf{b}')$. This requires communication of $2n$ field elements on top of the pre-processing step required to set up the random VOLE instance. Combined with efficient protocols for generating long instances of random VOLE, this gives rise to dv-NIZK protocol in which the offline phase consists of secure generation of random VOLE and the online phase uses the random VOLE as a “one-time pad” for realizing LPZK.

Constructing information-theoretic LPZK proofs

Our information-theoretic LPZK construction follows the general template of similar kinds of proof systems: the verification circuit is evaluated in two different ways that depend on secret randomness picked by the verifier, and the verifier accepts if the two evaluations are consistent. Zero knowledge is obtained by masking the values revealed to the verifier using randomness picked by the prover. This high level approach was used in previous information-theoretic zero-knowledge proof systems (such as succinct zero-knowledge linear PCPs [4, 33, 26, 9]), actively secure computation protocols (such as the SPDZ line of protocols [8, 21]), and circuits resilient to additive attacks [25]. Our LPZK systems most closely resemble the “homomorphic MAC” approach used for actively secure computation in the preprocessing model [8, 21], but differ in the low-level details.

More concretely, we construct LPZK for proving the satisfiability of an arithmetic circuit C by encoding intermediate wire values in the vector \mathbf{a} and masking these values with randomness in \mathbf{b} . This is an information-theoretic encryption: If the verifier holds $v_1(\alpha) := a_1\alpha + b_1$ and α , where a_1 is sampled from some distribution and b_1 is chosen uniformly at random from \mathbb{F} , the distribution of $v_1(\alpha)$ holds no information about a_1 .

We can “add” two encrypted wires $v_1(t) = a_1t + b_1$ and $v_2(t) = a_2t + b_2$ non-interactively for free; the prover adds to obtain $(a_1 + a_2)t + (b_1 + b_2)$, and the verifier adds $v_1(\alpha) + v_2(\alpha) = (a_1 + a_2)\alpha + (b_1 + b_2)$.

To multiply v_1 and v_2 , the prover seeks to construct the encrypted wire $a_1a_2t + b$, for some value b . When the prover multiplies $v_1(t) \cdot v_2(t)$ they obtain a quadratic in t . By adding and subtracting a masking term b_3t , they can write $v_1(t)v_2(t) = tv_3(t) + v_4(t)$, with $v_3(t) = a_1a_2t + (b_1a_2 + b_2a_1 - b_3)$ and $v_4(t) = b_3t + b_1b_2$, so that $v_3(t)$ is the desired encryption of a_1a_2 and satisfies $v_3(t) = (v_1(t)v_2(t) - v_4(t))/t$. The verifier learns $v_i(\alpha)$, for $1 \leq i \leq 4$ from the LPZK, and accepts if

$$v_3(\alpha) = \frac{v_1(\alpha)v_2(\alpha) - v_4(\alpha)}{\alpha},$$

and rejects otherwise. Finally, to open the value of an encrypted wire $v(t) = at + b$, the prover sends b to the verifier who computes $a = (v(\alpha) - b)/\alpha$.

Certified VOLE

As a building block for NISC, we build a *certified* variant of VOLE. This primitive is useful for invoking several parallel instances of VOLE while assuring the receiver that a given circuit C is satisfied when its inputs are a certain subset of the entries of the VOLEs.

We construct fully general certified VOLE from a weaker construction, *distributional VOLE with equality constraints*. This construction allows us to move all inputs to C to a single VOLE instance. The sender and receiver then prove that C is satisfied using LPZK NIZK.

This weaker variant, which we call eVOLE, is *distributional*, because it requires the VOLE inputs from the receiver to be chosen independently and uniformly at random. In general certified VOLE, which we call cVOLE, we use two additional evaluation points α, β , and perform an affine shift to the receiver’s inputs, replacing $(\alpha_1, \dots, \alpha_n)$ with $(\alpha + \alpha_1, \dots, \alpha + \alpha_n, \alpha, \beta)$.

This forces all receiver inputs to be uniformly random, and every input besides β is independent of β . We move all inputs to C to the VOLE instance with receiver input β , and use the VOLE instance with input α to reverse the affine shift of the receiver’s inputs.

From certified VOLE to NISC

Following [19], our NISC protocol is obtained from certified VOLE in a conceptually straightforward way: we start with existing protocols for arithmetic branching programs [32, 3] that achieve security against a malicious receiver and *semi-honest sender*. We then protect the receiver against a malicious sender by using certified VOLE to enforce honest behavior. This yields a statistically secure reusable NISC protocol for “simple” arithmetic functions represented by polynomial-size arithmetic branching programs. We can bootstrap this to get reusable NISC over VOLE for general Boolean circuits using the approach of [19]; however, this comes at the cost of making a non-black-box use of a pseudorandom generator and losing the concrete efficiency features of the arithmetic variant of the protocol.

2 LPZK and random VOLE

In this section we give a formal definition of our new notion of LPZK proof system and show how to compile such a proof system into a designated-verifier NIZK when given a random VOLE correlation.

2.1 Defining LPZK

While an LPZK proof system can be defined for any NP-relation, we focus here on the case of arithmetic circuit satisfiability that we use for describing our constructions. Our definition can be seen as a simple restriction of the more general notion of (1-round) zero-knowledge *linear interactive proof* [9] that restricts the verifier to sending a single field element.

Here and in the following, we work in an arithmetic model in which probabilistic polynomial time (PPT) algorithms can sample a uniformly random element from a finite field \mathbb{F} and perform field operations at a unit cost. All of the protocols we describe make a black-box use of the underlying field \mathbb{F} .

► **Definition 4 (LPZK).** *A line-point zero-knowledge (LPZK) proof system for arithmetic circuit satisfiability is a pair of algorithms (Prove, Verify) with the following syntax:*

- *Prove(\mathbb{F}, C, w) is a PPT algorithm that given an arithmetic verification circuit $C : \mathbb{F}^k \rightarrow \mathbb{F}^{k'}$ and a witness $w \in \mathbb{F}^k$, outputs a pair of vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$ that specify an affine line $\mathbf{v}(t) := \mathbf{a}t + \mathbf{b}$. We assume that the dimension n is determined by C .*
- *Verify($\mathbb{F}, C, \alpha, \mathbf{v}_\alpha$) is a polynomial-time algorithm that, given an evaluation \mathbf{v}_α of the line $\mathbf{v}(t)$ at some point $\alpha \in \mathbb{F}$, outputs **acc** or **rej**.*

The algorithms (Prove, Verify) should satisfy the following:

- **Completeness.** *For any arithmetic circuit $C : \mathbb{F}^k \rightarrow \mathbb{F}^{k'}$ and witness $w \in \mathbb{F}^k$ such that $C(w) = \mathbf{0}$, and for any fixed $\alpha \in \mathbb{F}$, we have*

$$\Pr[\mathbf{v}(t) \stackrel{R}{\leftarrow} \text{Prove}(\mathbb{F}, C, w) : \text{Verify}(\mathbb{F}, C, \alpha, \mathbf{v}(\alpha)) = \text{acc}] = 1.$$

- **Reusable ε -soundness.** *For every arithmetic circuit $C : \mathbb{F}^k \rightarrow \mathbb{F}^{k'}$ such that $C(w) \neq \mathbf{0}$ for all $w \in \mathbb{F}^k$, and every (adversarially chosen) line $\mathbf{v}^*(t) = \mathbf{a}^*t + \mathbf{b}^*$, where the length n of \mathbf{v}^* depends on C as above, we have $\Pr[\alpha \stackrel{R}{\leftarrow} \mathbb{F} : \text{Verify}(\mathbb{F}, C, \alpha, \mathbf{v}^*(\alpha)) = \text{acc}] \leq \varepsilon$. Moreover, for every $\mathbb{F}, C, \mathbf{v}^*(t)$ the probability of Verify accepting (over the choice of α) is either 1 or $\leq \varepsilon$. Unless otherwise specified, we assume $\varepsilon \leq O(1/|\mathbb{F}|)$.*

- **Perfect zero knowledge.** *There exists a PPT simulator Sim such that, for any arithmetic circuit $C : \mathbb{F}^k \rightarrow \mathbb{F}^{k'}$, any witness $w \in \mathbb{F}^k$ such that $C(w) = \mathbf{0}$, and any $\alpha \in \mathbb{F}$, the output of $\text{Sim}(\mathbb{F}, C, \alpha)$ is distributed identically to $\mathbf{v}(\alpha)$, where $\mathbf{v}(t)$ is the affine line produced by $\text{Prove}(\mathbb{F}, C, w)$.*

The *reusable* soundness requirement guarantees that even by observing the verifier's decision bit on a maliciously chosen circuit C , and line $\mathbf{v}^*(t) = \mathbf{a}^*t + \mathbf{b}^*$, the prover learns essentially nothing about the verifier's secret point α , which allows the same α to be reused without substantially compromising soundness.

Proof of Knowledge

For simplicity, we focus here on (reusable) soundness and ignore the additional *proof of knowledge* property. However, the LPZK systems we construct all satisfy this stronger notion of soundness (see [9] a definition of proofs of knowledge in the context of linear proof systems). More formally, there is an efficient *extractor* that can extract a valid witness from any (maliciously generated) line that makes the verifier accept with $> \varepsilon$ probability.

Computational LPZK

The above definition considers our main *information-theoretic* flavor of LPZK, with statistical soundness and perfect zero knowledge. Computational variants of LPZK can be defined analogously. In particular, we will later consider computationally sound LPZK in the random oracle model, which bounds the number of oracle queries made by a malicious prover.

Complexity measures for LPZK: (n, n', n'') -LPZK

In addition to the dimension/length parameter n , we use two other parameters n' and n'' as complexity measures for LPZK. These will help us obtain a more efficient compiler from LPZK to NIZK that takes advantage of verifier outputs that are either known by the prover (namely, are independent of α) or entries of \mathbf{a}, \mathbf{b} that can be picked at random independently of w . Concretely, the parameter n'' is the number of entries of \mathbf{a} that are always equal to zero; we assume without loss of generality that these are the last n'' entries. The parameter n' measures the total number of entries of the first $n - n''$ entries of \mathbf{a} and \mathbf{b} that functionally depend on w . To take advantage of the random VOLE setup, we assume the remaining $2n - 2n'' - n'$ entries are picked uniformly and independently at random, and then these n' entries are determined by w and the random entries. We will assume that the parameters (n, n', n'') as well as the identity of the entries of each type are determined by the public information C .

2.2 Compiling LPZK to NIZK over random VOLE

We now describe and analyze a simple compiler that takes an LPZK proof system as defined above and converts it into a (designated verifier) NIZK protocol that relies on a *random VOLE* correlation, where the prover gets a *random* pair of vectors $\mathbf{a}', \mathbf{b}' \in \mathbb{F}^n$ specifying an affine line $\mathbf{a}'t + \mathbf{b}'$ in \mathbb{F}^n and the verifier gets the value of the line at a random point $\alpha \in \mathbb{F}$, namely $\mathbf{v}' = \mathbf{a}'\alpha + \mathbf{b}'$. Similarly to previous VOLE-based compilers from [12, 19], we rely on the simple known reduction from VOLE to random VOLE. Our compiler takes advantage of the extra parameters n' and n'' of the LPZK, which help reduce the cost of the NIZK below the $2n$ field elements communicated by the natural generic compiler.

► **Lemma 5** (From LPZK to NIZK). *Given (n, n', n'') -LPZK over \mathbb{F} with soundness error ε , there is an NIZK protocol that uses a single instance of random VOLE of length $n - n''$ and requires communication of $n' + n''$ field elements from the prover to the verifier.*

Proof. Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$ be the vectors for the prover's line $\mathbf{a}t + \mathbf{b}$. The prover and verifier are given a random VOLE of length n , so that the prover holds $(\mathbf{a}', \mathbf{b}')$, and the verifier holds $\mathbf{v}' = \mathbf{a}'\alpha + \mathbf{b}'$ for a random $\alpha \in \mathbb{F}$.

We recall a simple self-reduction property of VOLE (see e.g. [12]) that allows us to replace a random pair $(\mathbf{a}', \mathbf{b}')$ with the pair (\mathbf{a}, \mathbf{b}) as follows. The prover sends vectors $\mathbf{a}' - \mathbf{a}$ and $\mathbf{b}' - \mathbf{b}$ to the verifier, who then computes

$$\mathbf{v} = \mathbf{v}' + \alpha(\mathbf{a}' - \mathbf{a}) + (\mathbf{b}' - \mathbf{b})$$

Finally, the prover sends the final n'' values of \mathbf{b} to the verifier in the clear, and the verifier appends these values to \mathbf{v} .

For any entry of \mathbf{a}, \mathbf{b} that should be chosen randomly for LPZK, the prover sets the corresponding entry of $\mathbf{a}' - \mathbf{a}$ or $\mathbf{b}' - \mathbf{b}$ to zero, and so no communication is required for those entries. The entire reduction requires a random VOLE of length n with communication of $n' + n''$ field elements, as desired. The security completeness, soundness, and zero knowledge properties of the above NIZK protocol are inherited directly from the corresponding properties of the LPZK. ◀

UC security

While we only consider here a standard standalone security definition for NIZK proofs [29, 10], all of our LPZK-based NIZK protocols are in fact *UC-secure* NIZK protocols (e.g., in the sense of [17]) in the rVOLE-hybrid model. This is the typical situation for information-theoretic protocols.

Using a corruptible random VOLE functionality

When using a pseudorandom correlation generator (PCG) for generating the random VOLE correlation with sublinear communication complexity [12, 14, 44], what is actually realized is a so-called “corruptible” random VOLE functionality that allows the malicious party to choose its output, and then samples the honest party's output conditioned on this choice. The transformation of Lemma 5 remains secure even when using this corruptible VOLE functionality. Indeed, it was already observed in [12] that the reduction of VOLE to random VOLE remains secure even when using corruptible random VOLE, and the LPZK to NIZK transformation builds on this reduction.

3 Single gate example

To clarify the exposition, we begin with an example where the prover wishes to convince the verifier that they hold a triple of values x, y, z satisfying $xy = z$. More precisely, the prover and verifier realize a commit-and-prove functionality for the triple (x, y, z) and the relation $R(x, y, z) := xy - z$. We prove that our single gate example satisfies this stronger flavor of ZK, which is meaningful even for finite functions. Note that our LPZK construction is adapted from this single gate example, rather than directly built up from it, so this proof and the proof in Section 4 can be read independently.

A commit-and-prove protocol for the above relation R has the same syntax as LPZK, and should satisfy the following loosely stated properties (see, e.g., [38] for a formal definition).

- **Completeness** If the prover runs honestly on (x, y, z) such that $z = xy$, then the verifier always accepts.
- **Binding** There is a deterministic extractor that given a line picked by a (potentially malicious) prover outputs effective inputs (x^*, y^*, z^*) such that the following holds. Any attempt of the prover to “explain” a different input triple (x', y', z') (by revealing its randomness) would lead to an inconsistent verifier view, except with the binding error probability (over the choice of α).
- **Soundness** For any malicious prover, if the extracted values (x^*, y^*, z^*) satisfy $z^* \neq x^*y^*$, then the verifier rejects except for the soundness error probability (over the choice of α).
- **Zero knowledge** For any choice of α , the verifier’s evaluation on an honestly generated line can be simulated without knowing x, y, z .

Random evaluation of the line picked by a prover (even a malicious one) effectively commits the prover to unique values of x, y, z , in the sense that except for the binding error probability it cannot reveal randomness that consistently explains different (x', y', z') , and moreover the verifier rejects unless $z = xy$ (except with soundness error probability).

3.1 Protocol

We construct our commit-and-prove protocol for the relation $R(x, y, z) := xy - z$ as a $(5, 4, 1)$ -LPZK over \mathbb{F} with binding and soundness error $\leq 2/|\mathbb{F}|$.

The (honest) prover chooses some triple (x, y, z) and constructs a line $\mathbf{a}t + \mathbf{b}$ by setting

$$\mathbf{a} = (a_1, a_2, a_3, a_4, a_5) := (x, y, z, xb_2 + yb_1 - b_3, 0)$$

with b_1, b_2, b_3, b_4 chosen uniformly at random and $b_5 := b_1b_2 - b_4$. We write

$$\mathbf{v}(t) := \mathbf{a}t + \mathbf{b},$$

for the line held by the prover, and $\mathbf{v} = \mathbf{a}\alpha + \mathbf{b}$ for the point received by the verifier, for a random $\alpha \in \mathbb{F}$. We likewise write the prover’s view of the entries as

$$\mathbf{v}(t) = (v_1(t), v_2(t), v_3(t), v_4(t), v_5(t)),$$

and write v_i for $v_i(\alpha)$. The verifier now checks whether

$$v_1v_2 - \alpha v_3 - v_4 - v_5 = 0.$$

We remark that it would be possible to present the same protocol as a $(4, 5, 0)$ -LPZK by dropping the v_5 term and setting $b_4 := b_1b_2$. This variant has the same communication and computation complexity, but we give the $(5, 4, 1)$ -LPZK construction here because it is more similar to the construction in Section 4.

► **Remark 6 (Extension to general arithmetic circuits).** We can convert this protocol to an LPZK for arithmetic circuits by placing all intermediate wire values into \mathbf{a} and running the commit-and-prove protocol for each multiplication gate. The binding property ensures that the wire values match the values x, y, z for which the prover demonstrates $xy = z$. For all multiplication gates whose inputs are intermediate values, the verifier no longer needs to learn the values v_1, v_2 masking the inputs x, y from VOLE, but can instead compute them as a linear combination of previous multiplication gate outputs. This therefore gives a communication cost of 3 field elements per multiplication gate. We improve on this by batching together verification messages into blocks of size t , as we show in the next section.

4 Information-Theoretic LPZK for Arithmetic Circuits

In this section we describe an information-theoretic LPZK for proving the satisfiability of arithmetic circuits. A full proof, and more formal theorem statement, of Theorem 1 are given in the full version of this paper [23].

4.1 Setup

An arithmetic circuit C over a field \mathbb{F} with k input wires, k' output wires, m multiplication gates, and arbitrarily many addition gates can be converted into an ordered triple (\mathbf{a}, Q_C, R_C) , where $\mathbf{a} = (a_0, a_1, \dots, a_{k+k'+4m})$ represent wire values. The input wires correspond to indices $0, 1, \dots, k$, the intermediate wires correspond to indices $k+1, \dots, k+4m$, and the output wires correspond to indices $k+4m+1, \dots, k+k'+4m$. Q_C is a collection of m degree 2 polynomials, with the i th polynomial defined as

$$q_i(\mathbf{a}) := a_{k+4i-1} - a_{k+4i-3}a_{k+4i-2},$$

and R_C is a set of linear relations defining certain a_i 's in terms of previous elements. Formally, we write $\mathbf{r} \cdot \mathbf{a}$ for the standard dot product, and write R_C as $2m+k'$ vectors \mathbf{r}_i corresponding to the relations

$$\mathbf{r}_{2i-j} \cdot \mathbf{a} = a_{k+4i-2-j},$$

for $j \in \{0, 1\}$, and $1 \leq i \leq m$, where the only nonzero entries of \mathbf{r}_{2i-j} occur at indices $\leq k+4i-4$, and

$$\mathbf{r}_{2m+i} \cdot \mathbf{a} = 0,$$

for $1 \leq i \leq k'$.

The wires a_{k+4i} are not needed for the insecure evaluation of the circuit, but we introduce them now to keep indices consistent. We require that each of \mathbf{r}_j have zero at each of their entries in positions $k+4i$, for $1 \leq j \leq 2m+k'$ and $1 \leq i \leq m$, i.e. the relations in R_C cannot depend on the unused a_{k+4i} wires. We set $a_0 = 1$ so that the relations R_C can include addition by constant terms.

We construct a NIZK in this setting. Using a $(k+2m, k+2m, \frac{m}{t}+k')$ -LPZK with soundness error $2t/|\mathbb{F}|$, a prover P will convince a verifier V that they hold a witness $\mathbf{w} = (w_1, \dots, w_k)$ of circuit inputs to C such that the k' entries $a_{k+4m+i} = 0$, for $1 \leq i \leq k'$. The circuit C and associated data k, k', m and Q are public.

4.2 The LPZK construction

To begin, the prover constructs a pair of vectors $(\mathbf{a}, \mathbf{b}) \in \mathbb{F}^{k+(4+\frac{1}{t})m+2}$, with $a_0 = 1$ and $b_0 = 0$. The next k elements of \mathbf{a} are set equal to the witness \mathbf{w} , and the corresponding elements of \mathbf{b} are chosen uniformly at random. Using the relations in R_C , for the i th multiplication gate, and for $j \in \{0, 1\}$, the prover defines

$$a_{k+4i-2-j} := \mathbf{r}_{2i-j} \cdot \mathbf{a}$$

$$b_{k+4i-2-j} := \mathbf{r}_{2i-j} \cdot \mathbf{b}$$

$$a_{k+4i-1} := a_{k+4i-3}a_{k+4i-2}$$

$$a_{k+4i} := a_{k+4i-3}b_{k+4i-2} + a_{k+4i-2}b_{k+4i-3} - b_{k+4i-1},$$

with b_{k+4i-j} chosen uniformly at random, for $j \in \{0, 1\}$. Then, for $1 \leq i \leq k'$, P sets $a_{k+4m+i} = 0$ and

$$b_{k+4m+i} := \mathbf{r}_{2m+i} \cdot \mathbf{b}.$$

Next, P constructs a vector \mathbf{c} of length m and defines

$$c_i := b_{k+4i-3}b_{k+4i-2} - b_{k+4i},$$

if this value is not equal to zero, and $c_i = 1$ otherwise, for $1 \leq i \leq m$. Finally, for $i = 1, \dots, m/t$, P sets $a_{k+k'+4m+i} = 0$ and defines

$$b_{k+k'+4m+i} := \prod_{j=t \cdot i}^{t \cdot i + t - 1} c_j.$$

After constructing (\mathbf{a}, \mathbf{b}) , the prover constructs a shortened pair of vectors $(\hat{\mathbf{a}}, \hat{\mathbf{b}})$ of length $k + k' + (2 + \frac{1}{t})m + 1$ by deleting the zeroth entry and the entries $k + 4i - 2 - j$, for $1 \leq i \leq m$ and $j \in \{0, 1\}$, and performs LPZK with the verifier so that the verifier learns $\hat{\mathbf{v}} = \alpha \hat{\mathbf{a}} + \hat{\mathbf{b}}$.

The verifier then computes from $\hat{\mathbf{v}}$ a vector \mathbf{v} of length $k + k' + (4 + \frac{1}{t})m + 2$ by re-indexing to match the indexing of \mathbf{a} and \mathbf{b} , setting $v_0 = 1$, and computing

$$v_{k+4i-2-j} := \mathbf{r}_{2i-j} \cdot \mathbf{v},$$

for $1 \leq i \leq m$ and $j \in \{0, 1\}$.

Then for $1 \leq i \leq k'$, the verifier checks that $\mathbf{r}_{2m+i} \cdot \mathbf{v} = v_{k+4m+i}$. Finally, the verifier defines, for $1 \leq i \leq m$, the values

$$x_i := v_{k+4i-3}v_{k+4i-2} - \alpha v_{k+4i-1} - v_{k+4i},$$

when this is nonzero, and $x_i := 1$ otherwise, and checks that

$$\prod_{j=t \cdot i}^{t \cdot i + t - 1} x_j = v_{k+k'+4m+i}.$$

5 LPZK in the Random Oracle Model

In the section we present Theorem 3, which gives an improved NIZK over random VOLE in the random oracle model (ROM). This follows by applying the compiler of Lemma 5 to the LPZK in following theorem.

► **Theorem 7** (LPZK in the ROM). *For any positive integer r , there exists an LPZK in the ROM for arithmetic circuit satisfiability, with the following size parameters (n, n', n'') and soundness error. If C has k inputs, k' outputs, and m multiplication gates, we have $n = k + k' + m + 2r$, $n' = k$, $n'' = k' + m + 2r$. For any malicious prover making ℓ calls to a random oracle $H : \mathbb{F}^m \rightarrow \mathbb{F}^{mr}$, the soundness error is $\varepsilon = \frac{2}{|\mathbb{F}|} + \frac{\ell}{|\mathbb{F}|^r}$. Moreover, the computation of both the prover and the verifier consists of $O(r|C|)$ field operations and a single call to H .*

At a high level, the LPZK construction begins by setting \mathbf{a} equal to the wire values in the circuit evaluation, and choosing \mathbf{b} at random, as in § 4.2. To convince the verifier that all multiplication gates have been evaluated correctly, the prover must show that a sequence of

quadratic polynomials whose coefficients are determined by \mathbf{a} and \mathbf{b} each have leading term zero, i.e. that this sequence of quadratics is actually a sequence of linear polynomials. The protocol uses LPZK to reveal to the verifier a vector \mathbf{s} of the evaluations of those quadratics at α and then the prover must show they have vectors \mathbf{y}, \mathbf{z} such that $\mathbf{s} = \mathbf{y}\alpha + \mathbf{z}$. In other words, the prover must show that \mathbf{y}, \mathbf{z} as VOLE inputs give \mathbf{s} as a VOLE output.

To do this, prover and verifier choose a random $r \times m$ matrix $M := H(\mathbf{w})$ by evaluating a random oracle H on the prover messages \mathbf{w} sent during the protocol. Then after adding random masks from the LPZK to $\mathbf{y}, \mathbf{z}, \mathbf{s}$, the verifier checks that $M\mathbf{s} = M\mathbf{y}\alpha + M\mathbf{z}$.

5.1 The LPZK construction

Similar to § 4.2, the prover begins by constructing a line $\mathbf{v}(t) := \mathbf{a}t + \mathbf{b}$ with $\mathbf{v} \in \mathbb{F}^{k+k'+5m+3r+1}$, and then reduces to a shorter $\hat{\mathbf{v}}$ that is used as VOLE input. For $0 \leq i \leq k + k' + 4m$, the prover defines a_i and b_i identically to their definitions in § 4.2, except each entry a_{k+4j} is chosen uniformly at random from \mathbb{F} , for $1 \leq j \leq m$, and each entry b_{k+4j} is chosen so that $b_{k+4j} = b_{k+4j-1}$. The partial redundancy between the $k + 4j - 1$ th and $k + 4j$ th entry is to preserve the indexing of § 4.2 while enabling the reconstruction of v_{k+4i-1} from v_{k+4i} and the value of $a_{k+4j} - a_{k+4j-1}$, as described below.

The next r entries of \mathbf{a} and \mathbf{b} are chosen uniformly at random from \mathbb{F} . The remaining $m+2r$ entries of \mathbf{a} are all set equal to zero, and the remaining $m + 2r$ entries of \mathbf{b} will be given explicitly later. These $m + 2r$ entries, in other words, can be sent from the prover to the verifier directly without require any VOLE overhead.

For $1 \leq i \leq m$, the prover computes

$$y_i := b_{k+4i-1} - a_{k+4i-3}b_{k+4i-2} - a_{k+4i-2}b_{k+4i-3}$$

and

$$z_i := -b_{k+4i-3}b_{k+4i-2},$$

and defines $\mathbf{y} = (y_i)$ and $\mathbf{z} = (z_i)$, where i ranges from 1 to m . For r the positive integer fixed in the statement of the theorem, let $H : \mathbb{F}^m \rightarrow \mathbb{F}^{mr}$ be a random oracle, and treat the output of H as a matrix in $M_{r \times m}(\mathbb{F})$. The prover then defines $\mathbf{w} := (w_i) := (a_{k+4i-1} - a_{k+4i})$, where i ranges from 1 to m . The prover then sets

$$\mathbf{y} := (a_{k+k'+4m+1}, \dots, a_{k+k'+4m+r})^T + H(\mathbf{w})\mathbf{y}^T$$

and

$$\mathbf{z} := (b_{k+k'+4m+1}, \dots, b_{k+k'+4m+r})^T + H(\mathbf{w})\mathbf{z}^T.$$

For $1 \leq i \leq m$, the prover sets

$$b_{k+k'+4m+r+i} := a_{k+4i-1} - a_{k+4i},$$

then the prover sets

$$\mathbf{b}[k + k' + 5m + r + 1 : k + k' + 5m + 2r] = \mathbf{y},$$

and

$$\mathbf{b}[k + k' + 5m + 2r + 1 : k + k' + 5m + 3r] = \mathbf{z},$$

writing $\mathbf{b}[i, j]$ for the projection onto coordinates i through j inclusive.

Next, the prover computes from the pair (\mathbf{a}, \mathbf{b}) a line in a lower-dimensional space $\hat{\mathbf{v}}(t) := \hat{\mathbf{a}}t + \hat{\mathbf{b}} \in \mathbb{F}^{k+k'+2m+3r}$. For $1 \leq i \leq k$, we take $\hat{a}_i = a_i$ and $\hat{b}_i = b_i$. For $1 \leq i \leq m$ we take $\hat{a}_{k+i} = a_{k+4i}$ and $\hat{b}_{k+i} = b_{k+4i}$. For $1 \leq i \leq r$, we take $\hat{a}_{k+m+i} = a_{k+k'+4m+i}$ and $\hat{b}_{k+m+i} = b_{k+k'+4m+i}$. The remaining $k' + m + 2r$ values of \mathbf{a} we set equal to zero. For $1 \leq i \leq k'$, we set $\hat{b}_{k+m+r+i} = \mathbf{r}_{2m+i} \cdot \mathbf{b}$. For $1 \leq i \leq m$, we set $\hat{b}_{k+k'+m+r+i} = w_i = a_{k+4i-1} - a_{k+4i}$. Finally, for $1 \leq i \leq 2r$, we set $\hat{b}_{k+k'+2m+r+i} = b_{k+k'+5m+r+i}$.

Now, having constructed $\hat{\mathbf{v}}(t)$, the prover and verifier run LPZK so that the verifier learns $\hat{\mathbf{v}}(\alpha)$, and, similar to § 4.2, expands $\hat{\mathbf{v}}(\alpha)$ to a vector $\mathbf{v} = \alpha\mathbf{a} + \mathbf{b}$. The verifier reconstructs v_{k+4i-1} as

$$v_{k+4i-1} = v_{k+4i} + \alpha v_{k+k'+m+r+i},$$

and the other missing values as in § 4.2.

The verifier now computes, for $1 \leq i \leq m$,

$$s_i := v_{k+4i-1}\alpha - v_{k+4i-3}v_{k+4i-2},$$

the vector $\mathbf{s} = (s_i)$, and the value

$$s := (v_{k+k'+4m+1}, \dots, v_{k+k'+4m+r})^T + H(\mathbf{w})\mathbf{s}^T,$$

$$y_\alpha := (\mathbf{v}[k+k'+5m+r+1 : k+k'+5m+2r])$$

$$z_\alpha := (\mathbf{v}[k+k'+5m+2r+1 : k+k'+5m+3r])$$

and returns `rej` unless $y_\alpha + z = s$. Then for $1 \leq i \leq k'$, the verifier checks that $\mathbf{r}_{2m+i} \cdot \mathbf{v} = v_{k+4m+i}$ and returns `rej` if any test fails, and `acc` otherwise.

6 Non-Interactive Secure Computation

In this section we apply LPZK towards simplifying and improving the efficiency of the reusable protocol for non-interactive secure computation (NISC) from [19]. Our construction relies on a variant of VOLE called *certified* VOLE, described in more detail in § 6.2.

6.1 NISC definition

We start by giving a simplified definition of reusable NISC over VOLE, which strengthens the definition from [19]. The definition can be seen as a natural extension of the definition of LPZK to the case of secure computation, where both the sender and the receiver have secret inputs. Instead of the prover encoding its witness as a line and the verifier picking a random point, here the sender encodes its input as multiple lines and the receiver encodes its input as multiple points, one for each line. (The lines are the sender's VOLE inputs and the points are the receiver's VOLE inputs.)

At a high level, reusable security is ensured by preventing a malicious sender from making the receiver's output depend on its input beyond the dependence allowed by the ideal functionality. This is contrasted with OT-based NISC protocols, where the sender can learn a receiver's OT input by starting from an honest strategy and replacing one of the sender OT inputs by a random one.

We formulate the NISC definition for arithmetic functions f defined over an arbitrary field \mathbb{F} , where the security error vanishes with the field size. For simplicity we consider a single function f and information-theoretic security. The definition can be naturally generalized to take a function description as input and allow computational security.

► **Definition 8** (Reusable arithmetic NISC). *A reusable non-interactive secure computation (NISC) protocol over VOLE for an arithmetic function $f : \mathbb{F}^n \times \mathbb{F}^m \rightarrow \mathbb{F}^\ell$ is a triple of algorithms $(R1, S, R2)$ with the following syntax:*

- $R1(\mathbb{F}, \mathbf{x})$ is a PPT algorithm that, given an input $\mathbf{x} \in \mathbb{F}^n$, outputs points $(\alpha_1, \dots, \alpha_{n'}) \in \mathbb{F}^{n'}$ and auxiliary information aux .
- $S(\mathbb{F}, \mathbf{y})$ is a PPT algorithm that, given $\mathbf{y} \in \mathbb{F}^m$, outputs n' pairs of vectors $\mathbf{a}_i, \mathbf{b}_i \in \mathbb{F}^s$, each specifying an affine line $\mathbf{v}_i(t) := \mathbf{a}_i t + \mathbf{b}_i$.
- $R2(\mathbb{F}, \text{aux}, (\mathbf{v}_1, \dots, \mathbf{v}_{n'}))$ is a polynomial-time algorithm that, given auxiliary information aux and evaluations \mathbf{v}_i , outputs either $\mathbf{z} \in \mathbb{F}^\ell$ or rej .

The algorithms $(R1, S, R2)$ should satisfy the following security requirements:

- **Completeness.** *When both parties follow the protocol, running the above algorithms in sequence, with $\mathbf{v}_i = \mathbf{v}_i(\alpha_i)$, results in the output $\mathbf{z} = f(\mathbf{x}, \mathbf{y})$.*
- **Reusable ε -security against malicious sender.** *There exists a polynomial-time extractor algorithm Ext such that for any field \mathbb{F} and lines $\mathbf{v}_i^*(t) := \mathbf{a}_i^* t + \mathbf{b}_i^*$, the output of $\text{Ext}(\mathbb{F}, (\mathbf{a}_1^*, \mathbf{b}_1^*), \dots, (\mathbf{a}_{n'}^*, \mathbf{b}_{n'}^*))$ is $\mathbf{y}^* \in \mathbb{F}^m \cup \{\perp\}$ such that the following holds: for every honest receiver's input $\mathbf{x} \in \mathbb{F}^n$, the receiver's output when interacting with malicious sender strategy $\mathbf{v}_i^*(t)$ is equal to $f(\mathbf{x}, \mathbf{y}^*)$ except with $\leq \varepsilon$ probability over the receiver's randomness. Here we assume that the output on \perp is rej . Unless otherwise specified, we assume $\varepsilon \leq O(1/|\mathbb{F}|)$. We will also use a **random-input** variant of the above definition, where the probability is over both the receiver's randomness and a uniformly random choice of $\mathbf{x} \in \mathbb{F}^n$.*
- **Perfect security against malicious receiver.** *There exist a polynomial-time extractor algorithm Ext and PPT simulator algorithm Sim such that, for any field \mathbb{F} and malicious receiver points $\alpha_1^*, \dots, \alpha_{n'}^* \in \mathbb{F}$, the extractor outputs an effective input $\mathbf{x}^* = \text{Ext}(\mathbb{F}, (\alpha_1^*, \dots, \alpha_{n'}^*))$, where $\mathbf{x}^* \in \mathbb{F}^n$, such that the following holds. For every honest sender's input $\mathbf{y} \in \mathbb{F}^m$, the output distribution of $\text{Sim}(\mathbb{F}, f(\mathbf{x}^*, \mathbf{y}))$ is identical to $\{(\mathbf{v}_1(\alpha_1^*), \dots, \mathbf{v}_{n'}(\alpha_{n'}^*)) : (\mathbf{v}_1(t), \dots, \mathbf{v}_{n'}(t)) \stackrel{R}{\leftarrow} S(\mathbb{F}, \mathbf{y})\}$.*

We note that instead of allowing the receiver to output rej , we could instead make the receiver use a default value for the sender input and compute the output of f . However, making the receiver reject whenever it detects cheating makes protocol descriptions more natural.

The definition above does not permit the sender to transmit additional values to the receiver in the clear. In order to simplify the definition and the proofs, we note that we can realize plaintext transmission from sender to receiver as a reusable NISC protocol over VOLE. The function $f(\mathbf{x}, \mathbf{y}) := \mathbf{y}$ prints the sender input, the algorithm $R1(\mathbb{F}, \mathbf{x})$ outputs random points α_1, α_2 , and the sender algorithm $S(\mathbb{F}, \mathbf{y})$ outputs $\mathbf{a}_i := \mathbf{0}$ and $\mathbf{b}_i = \mathbf{y}$ for $i = 1, 2$. Finally, $R2(\mathbb{F}, (\mathbf{v}_1, \mathbf{v}_2))$ rejects if $\mathbf{v}_1 \neq \mathbf{v}_2$, and outputs \mathbf{v}_1 otherwise. The security conditions are straightforward to verify.

In the proofs below, when we refer to “sending values in the clear”, we formally mean the protocol above. In actual applications, of course, we will continue to send the plaintexts directly. We use direct transmission, rather than this more involved NISC protocol, in our analysis of computation and communication complexity.

Throughout this section, whenever we desire to refer to the j th entry of a vector $\mathbf{a}_i, \mathbf{b}_i, \mathbf{v}_i$, etc, we write the entry as a_i^j, b_i^j, v_i^j , etc.

6.2 Certified VOLE

The main building block for NISC is a *certified* variant of VOLE, allowing the sender and the receiver to invoke multiple parallel instances of VOLE while assuring the receiver that the sender's VOLE inputs satisfy some global consistency relation.

6.2.1 Definitions and results

In its general form, *certified VOLE with a general arithmetic relation*, the VOLE consistency requirement is specified by a general arithmetic circuit. We write cVOLE for this form of certified VOLE.

We begin with a more specialized form, distributional certified VOLE with equality constraints, which we write as eVOLE. In this variant of certified VOLE, the arithmetic circuit on the family of VOLEs is restricted to a single equality constraint between two coefficients from \mathbf{a} vectors. In eVOLE, we require additionally that R 's inputs are uniformly distributed over \mathbb{F} and independent. It is straightforward to extend this result to an arbitrary set of equality constraints on terms from \mathbf{a} and \mathbf{b} vectors, and we explain the details below.

Certified VOLE of these flavors can be realized by extending a family of random VOLEs with a NIZK proof that the random VOLEs satisfy the desired constraints. We give more precise definitions of these forms of certified VOLE as ideal functionalities in Figures 1 and 2. We state this result as the following two lemmas.

► **Lemma 9.** *A receiver R and a sender S can realize the functionality $\mathcal{F}_{eVOLE}^{(\mathbb{F})}$ with parameters (ℓ_1, ℓ_2, i, j) in the rVOLE hybrid model with 2 instances of random VOLE of total length $\ell_1 + \ell_2 + 2$ and communication of 3 field elements from sender to receiver, in addition to any communication cost for transforming random VOLEs to the VOLEs with inputs $(\hat{\mathbf{a}}_1, \hat{\mathbf{b}}_1, \hat{\mathbf{a}}_2, \hat{\mathbf{b}}_2)$.*

► **Lemma 10.** *Fix an integer $t \geq 1$. A receiver R and a sender S can realize the functionality $\mathcal{F}_{cVOLE}^{(\mathbb{F})}$, in the rVOLE hybrid model with $k+2$ instances of random VOLE. For a circuit C with $q_{\mathbf{a}}$ inputs from the $\hat{\mathbf{a}}_i$'s, $q_{\mathbf{b}}$ inputs from the $\hat{\mathbf{b}}_i$'s, q' outputs, and m multiplication gates, these VOLE instances have total length*

$$2m + 6q_{\mathbf{a}} + 7q_{\mathbf{b}} + \sum_{i=1}^k \ell_i,$$

and the protocol requires communication of

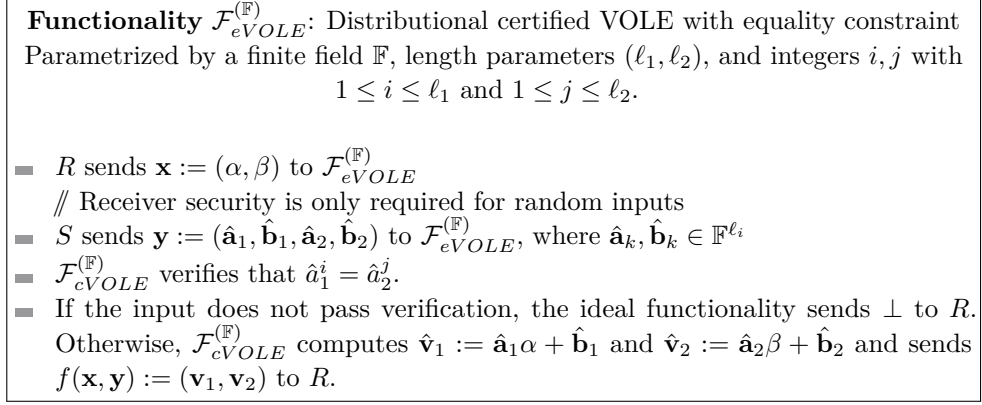
$$(2 + \frac{1}{t})m + q' + 8q_{\mathbf{a}} + 9q_{\mathbf{b}} + 2 \sum_{i=1}^k \ell_i$$

field elements from sender to receiver.

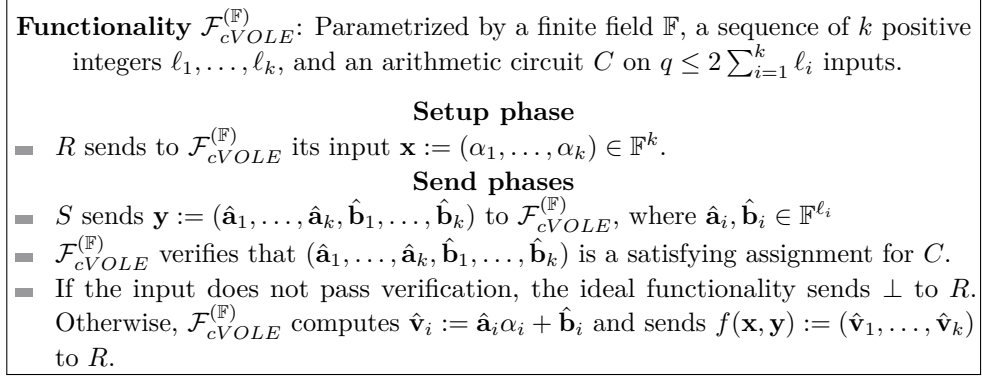
6.2.2 The protocols

eVOLE

eVOLE is a special case of reusable arithmetic NISC where the receiver has no inputs, and $R1(\mathbb{F})$ outputs uniformly random and independent points (α, β) , and stores their values as the auxiliary information $\mathbf{aux} := (\alpha, \beta)$. The sender's input $\mathbf{y} := (\hat{\mathbf{a}}_1, \hat{\mathbf{b}}_1, \hat{\mathbf{a}}_2, \hat{\mathbf{b}}_2)$ is two existing VOLE inputs, and the algorithm $S(\mathbb{F}, \mathbf{y})$ outputs vectors $(\mathbf{a}_1, \mathbf{b}_1, \mathbf{a}_2, \mathbf{b}_2)$ whose first ℓ, ℓ, ℓ, ℓ' ,



■ **Figure 1** Distributional certified VOLE with equality constraints.



■ **Figure 2** Certified VOLE with a general arithmetic relation.

and ℓ' coordinates are equal to $(\hat{\mathbf{a}}_1, \hat{\mathbf{b}}_1, \hat{\mathbf{a}}_2, \hat{\mathbf{b}}_2)$, respectively. The remaining values are defined as $a_1^{\ell+1} := \hat{b}_2^j$, $a_2^{\ell'+1} := \hat{b}_1^i$, with $b_1^{\ell+1}$ and $b_2^{\ell'+1}$ chosen uniformly at random. In addition, the sender sends the value $b_1^{\ell+1} - b_2^{\ell'+1}$ in the clear.

The VOLE protocol evaluates the sender's output on α and β , respectively, so that in an honest run of the protocol, the receiver learns $\mathbf{v}_1 := \mathbf{a}_1\alpha + \mathbf{b}_1$ and $\mathbf{v}_2 := \mathbf{a}_2\beta + \mathbf{b}_2$. In the algorithm $\text{R2}(\mathbb{F}, \text{aux}, (\mathbf{v}_1, \mathbf{v}_2))$, the receiver tests whether

$$\beta v_1^i - \alpha v_2^j + v_1^{\ell+1} - v_2^{\ell'+1} = b_1^{\ell+1} - b_2^{\ell'+1}.$$

The receiver rejects if the test fails, and otherwise outputs the vectors $\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2$ obtained by deleting the last element from $\mathbf{v}_1, \mathbf{v}_2$.

This protocol can be modified to prove constraints of the form $\hat{a}_1^i = \hat{b}_2^j$ or $\hat{b}_1^i = \hat{b}_2^j$ for the same communication cost and one or two additional multiplications, respectively, by the receiver. Indeed, by multiplying \mathbf{v}_1 by α^{-1} or \mathbf{v}_2 by β^{-1} , the receiver can locally obtain the VOLE outputs $\mathbf{w}_1 := \alpha^{-1}\mathbf{b}_1 + \mathbf{a}_1$ and $\mathbf{w}_2 := \beta^{-1}\mathbf{b}_2 + \mathbf{a}_2$, and the same construction above applies to the pair $\mathbf{v}_1, \mathbf{w}_2$ or the pair $\mathbf{w}_1, \mathbf{w}_2$.

Additionally, since the eVOLE protocol transforms VOLE inputs $\hat{\mathbf{a}}_i, \hat{\mathbf{b}}_i$ for the sender into extended VOLE inputs $\mathbf{a}_i, \mathbf{b}_i$ and delivers extended VOLE outputs \mathbf{v}_i to the receiver, this protocol can be implemented repeatedly on the *same two instances* of VOLE, proving c equality constraints with VOLEs of length $\ell + c, \ell' + c$.

cVOLE

We write the receiver's inputs as $\mathbf{x} := (\alpha_1, \dots, \alpha_k)$. The receiver's algorithm $R1(\mathbb{F}, \mathbf{x})$ generates their VOLE inputs by choosing random independent values α, β , and then outputs $(\alpha + \alpha_1, \dots, \alpha + \alpha_k, \alpha, \beta)$.

As in eVOLE, the sender defines $a_i^j := \hat{a}_i^j$ everywhere this is defined. We give the definition of b_i^j later. Then, for each input to C from the $\hat{\mathbf{a}}_i$'s, say $\hat{a}_i^{j_1}$, the sender chooses one entry of \mathbf{a}_{k+1} and one entry of \mathbf{a}_{k+2} , say $a_{k+1}^{j_2}$ and $a_{k+2}^{j_3}$ respectively, and uses eVOLE to prove $\hat{a}_i^{j_1} = a_{k+2}^{j_3}$ and $a_{k+1}^{j_2} = a_{k+2}^{j_3}$. Since each of the pairs $(\alpha + \alpha_i, \beta)$ and (α, β) are uniformly random and independent, the conditions for eVOLE are satisfied.

Similarly, for an input $\hat{b}_i^{j_1}$ to C , the sender chooses entries $b_{k+1}^{j_2}$, $a_{k+2}^{j_3}$ and $a_{k+2}^{j_4}$ and proves $b_i^{j_1} = a_{k+2}^{j_3}$ and $b_{k+1}^{j_2} = a_{k+2}^{j_4}$. We now define $b_i^{j_1} := \hat{b}_i^{j_1} + b_{k+1}^{j_2}$. Upon subtracting $v_{k+1}^{j_2} := a_{k+1}^{j_2} \alpha + b_{k+1}^{j_2}$ from $v_i^{j_1} := a_i^{j_1} (\alpha + \alpha_i) + b_i^{j_1}$, the receiver holds

$$\hat{v}_i^{j_1} := v_i^{j_1} - v_{k+1}^{j_2} = a_i^{j_1} \alpha_i + (b_i^{j_1} - b_{k+1}^{j_2}) = \hat{a}_i^{j_1} \alpha_i + \hat{b}_i^{j_1}.$$

After deleting unneeded entries of the $\hat{\mathbf{v}}_i$'s receiver ends with the VOLE outputs $\hat{\mathbf{v}}_i := \hat{\mathbf{a}}_i \alpha_i + \hat{\mathbf{b}}_i$, as desired. In addition, the elements $a_i^{j_1}, b_i^{j_2}, b_{k+1}^{j_2}$ have all been transferred to entries of \mathbf{a}_{k+2} , so the receiver and sender extend the $(k+2)$ nd instance of VOLE \mathbf{v}_{k+2} with a NIZK proof that C is satisfied by $\hat{\mathbf{a}}_i, \hat{\mathbf{b}}_i$.

6.3 Reusable NISC over VOLE

In this section we build on certified VOLE to compile NISC protocols with security against semi-honest senders into reusable NISC protocols in the fully malicious setting. We follow the same high level approach of [19], but present the compiler at a higher level of generality and with a more refined efficiency analysis.

Consider a two-party sender-receiver functionality $f(\mathbf{x}, \mathbf{y})$ where the receiver R holds $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{F}^n$ and the sender S hold inputs $\mathbf{y} = (y_1, \dots, y_n) \in \mathbb{F}^m$. The function f is arithmetic, in the sense that its outputs are defined by a sequence of ℓ arithmetic branching programs P_1, \dots, P_ℓ over \mathbb{F} , where program P_i has s_i nodes. (Note that such an arithmetic program P_i can simulate any arithmetic formula with s_i additions and multiplication gates.)

The goal is to securely evaluate f using only parallel instances of VOLE. (The ideal VOLE instances can be implemented using the same kind of cryptographic compilers we used in the context of LPZK.) We also require the NISC protocol to be *reusable* in the sense that if the receiver's input is fixed but the sender's input changes, the same VOLE inputs of the receiver can be securely reused, even if the sender can obtain partial information about the receiver's outputs in the different invocations. This feature is impossible to achieve in the information-theoretic setting when we use OT instead of VOLE [19].

To get a reusable NISC for f , we take the following two-step approach:

1. Using a so-called "Decomposable Affine Randomized Encoding" (DARE) for branching programs [32, 3] (an arithmetic variant of information-theoretic garbling), we get a NISC protocol for f with n instances of VOLE, each of length $S_j = \sum_{i \in D(j)} \binom{s_i}{2}$, where $D(j)$ is the set of output entries that depend on x_j .² This protocol is secure against a malicious receiver R and a *semi-honest* sender R .

² In a bit more detail, for a branching program $P(x_1, \dots, x_n)$ of size s , the output can be encoded by the n matrices $Y_j = L \cdot A_j(x_j) \cdot R + Z_j$, where L, R, Z_j , and $A_j(x_j)$ are $(s-1) \times (s-1)$ matrices, A_j is an affine (degree-1) function of x_j , the Z_j are random subject to the constraint $\sum Z_j = 0$, and L, R are

2. To obtain reusable security against a malicious S (while maintaining security against malicious R) we replace the parallel VOLE in the previous protocol by *certified* VOLE, where the circuit C specifying the consistency relation takes the sender’s input \mathbf{y} and randomness in the previous protocol as a witness, and checks that the sender’s VOLE inputs are obtained by applying the honest sender’s algorithm to the witness. Using naive matrix multiplication, this requires a circuit C of size $S = \sum_{j=1}^n S_j + \sum_{i=1}^{\ell} s_i^3$. Applying our protocol for $\mathcal{F}_{cVOLE}^{(\mathbb{F})}$ with the arithmetic relation specified by C , we ensure that whenever a malicious sender does not provide a witness that “explains” its VOLE inputs by an honest sender strategy, the receiver outputs \perp except with $O(1/|\mathbb{F}|)$ probability. In particular, a (reusable) simulator for a malicious sender interacting with the $\mathcal{F}_{cVOLE}^{(\mathbb{F})}$ functionality either outputs the input \mathbf{y} found in the witness, if the consistency check specified by C passes, or \perp if C fails.

Combining the above two steps, we derive the feasibility result from [19] in a simpler way.

► **Theorem 11 (Reusable arithmetic NISC over VOLE).** *Suppose $f : \mathbb{F}^n \times \mathbb{F}^m \rightarrow \mathbb{F}^{\ell}$ is a sender-receiver functionality whose i -th output can be computed by an arithmetic branching programs over \mathbb{F} of size s_i that depends on d_i inputs. Then f admits a reusable NISC protocol over VOLE with the following efficiency and security features:*

- *The protocol uses $n + 2$ parallel VOLE instances.*
- *The total length of the VOLE instances is $15 \sum_{i=1}^{\ell} d_i \binom{s_i}{2} + 2 \sum_{i=1}^{\ell} s_i^3$.*
- *The simulation error (per invocation) is $\varepsilon = O(1/|\mathbb{F}|)$.*

Chase et al. [19] show how to bootstrap Theorem 11 to get reusable NISC over VOLE for general Boolean circuits, by making (a non-black-box) use of any pseudorandom generator, or equivalently a one-way function.

6.4 NISC Example: Bounded Inner Product

In this section we showcase the usefulness of reusable arithmetic NISC by presenting an optimized construction for a natural functionality: an inner product between the receiver’s input vector and the sender’s input vector, where the sender’s vector is restricted to have a bounded L_2 norm. This functionality is useful for measuring similarity between two normalized feature vectors. The bound on the sender’s input is essential for preventing a malicious sender from inflating the level of similarity by scaling its input.

6.4.1 Functionality

Let R hold inputs $\mathbf{x} = (x_1, \dots, x_n)$ and S hold inputs $\mathbf{y} = (y_1, \dots, y_n)$ such that $y_i \in \{0, 1, \dots, K\}$ and

$$\sum_{i=1}^n y_i^2 \leq L^2,$$

for some other constant L , so that the ℓ^2 norm satisfies

$$\|\mathbf{y}\|_2 \leq L.$$

random invertible matrices of a special form. The matrix Y_j contains $\binom{s}{2}$ non-constant entries. See [32] for details. The S_j entries of VOLE j are the concatenation of the (non-constant entries of) matrices Y_j associated with outputs that depend on x_j

R desires to compute the dot product $\mathbf{x} \cdot \mathbf{y}$ (as a measure of the similarity of R and S 's inputs). To simplify the protocol, we restrict to the case where K and L are powers of 2. When R and S do not wish to impose any bound on individual entries beyond what is implied by the ℓ^2 norm, they set $K = L$.

In the above description we assume the inputs to be vectors over non-negative integers. This functionality can be naturally embedded by considering vectors over a finite field \mathbb{F} of prime order p , provided that p is bigger than the square-norm bound L^2 and an upper bound on the output size.

6.4.2 Protocol

S begins with a sequence of n inputs (y_i) , and selects associated random masks z_i .

First S engages in pre-processing of their data by computing the bit decomposition (c_{ij}) of each element y_i and the bit decomposition (c_{sj}) of the sum of squares $\sigma_y := \sum y_i^2$. We use $\lg K$ bits for the bit decompositions (c_{ij}) and $2 \lg L$ bits for bit decomposition (c_{sj}) , which ensures that \mathbf{y} satisfies the desired bounds if the bit decompositions are correct.

We give a slightly modified construction of cVOLE, optimized for this setting. R and S generate $n + 2$ instances of random VOLE. As in cVOLE, R chooses inputs $(x_1 + \alpha, \dots, x_n + \alpha, \alpha, \beta)$, with $\alpha, \beta \in \mathbb{F}$ random and independent. The input of S to the i th instance of VOLE is y_i , for $1 \leq i \leq n$. Then S uses the entire vector \mathbf{y} as inputs to the $(n + 1)$ st and $(n + 2)$ nd instance of VOLE. S also takes as inputs to the $(n + 2)$ nd instance all constant terms from the first n VOLEs, the sum of the constant terms from the $(n + 1)$ st instance, the squares y_i^2 , and the bit decompositions (c_{ij}) and (c_{sj}) . After this initial set up, R learns the following:

- $v_i^1 := y_i(x_i + \alpha) + z_i$, for $1 \leq i \leq n$, where z_i is a random element determined in the initial random VOLE set up, and thus requires no additional communication.
- $v_{n+1}^i := y_i \alpha + w_i$, for $1 \leq i \leq n - 1$, with w_i from the random VOLE.
- $v_{n+1}^n := y_n \alpha + w_n$, where w_n is chosen such that $\sum_{i=1}^n z_i = \sum_{i=1}^n w_i$.
- $v_{n+2}^i := y_i \beta + u_i$, for $1 \leq i \leq n$, with u_i from the random VOLE.
- $v_{n+2}^{n+i} := z_i \beta + u_{n+i}$, for $1 \leq i \leq n$, with u_i from the random VOLE.
- $v_{n+2}^{2n+1} := (\sum_{i=1}^n w_i) \beta + u_{2n+1}$, with u_{2n+1} from the random VOLE.

Additionally, \mathbf{a}_{n+2} holds all of the bit decompositions and associated data mentioned above. To complete the verification step of the protocol, R and S execute eVOLE to ensure that all inputs that occur in multiple VOLE instances are, in fact, equal, and then S uses LPZK-NIZK on the $(n + 2)$ nd instance of VOLE to convince R of the validity of S 's input.

The NIZK proof checks that all values y_i^2 sent by S are actually equal to the squares of the values y_i , and confirms that c_{ij} and c_{si} are in $\{0, 1\}$ by evaluating the quadratic $t^2 - t$ on each entry. The proof then checks that the bit-decompositions are correct by computing and revealing $y_i - \sum_j c_{ij} 2^j$ and $\sum y_i^2 - \sum_j c_{sj} 2^j$, all of which are equal to zero when both parties behave honestly.

Finally, R computes the output value as

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^n (v_i^1 - v_{n+1}^i).$$

We give the proof and the calculation of the communication and computational complexity in the full version of this paper [23].

References

- 1 Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. In *EUROCRYPT 2014*, pages 387–404, 2014.
- 2 Benny Applebaum, Ivan Damgård, Yuval Ishai, Michael Nielsen, and Lior Zichron. Secure arithmetic computation with constant computational overhead. In *CRYPTO 2017, Part I*, pages 223–254, 2017.
- 3 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. In *FOCS 2011*, pages 120–129, 2011.
- 4 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3), 1998.
- 5 Carsten Baum, Daniel Escudero, Alberto Pedrouzo-Ulloa, Peter Scholl, and Juan Ramón Troncoso-Pastoriza. Efficient protocols for oblivious linear function evaluation from Ring-LWE. In *SCN 2020*, pages 130–149, 2020.
- 6 Carsten Baum, Alex J. Malozemoff, Marc Rosen, and Peter Scholl. Mac’n’cheese: Zero-knowledge proofs for arithmetic circuits with nested disjunctions. Cryptology ePrint Archive, Report 2020/1410, 2020. URL: <https://eprint.iacr.org/2020/1410>.
- 7 Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In *CRYPTO 2019, Part III*, Lecture Notes in Computer Science, pages 701–732, 2019.
- 8 Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, 2011.
- 9 Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC 2013*, pages 315–333, 2013.
- 10 Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC 1988*, pages 103–112, 1988.
- 11 Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In *ASIACRYPT 2017, Part III*, pages 336–365, 2017.
- 12 Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In *CCS 2018*, pages 896–912, 2018.
- 13 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In *CCS 2019*, pages 291–308, 2019.
- 14 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO 2019, Part III*, pages 489–518, 2019.
- 15 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *FOCS 2020*, pages 1069–1080, 2020. Full version: <https://eprint.iacr.org/2020/1417>.
- 16 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from Ring-LPN. In *CRYPTO 2020, Part II*, pages 387–416, 2020.
- 17 Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS 2001*, pages 136–145, 2001.
- 18 Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *CCS 2017*, pages 1825–1842, 2017.
- 19 Melissa Chase, Yevgeniy Dodis, Yuval Ishai, Daniel Kraschewski, Tianren Liu, Rafail Ostrovsky, and Vinod Vaikuntanathan. Reusable non-interactive secure computation. In *CRYPTO 2019, Part III*, pages 462–488, 2019.

- 20 Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *EUROCRYPT 2010*, pages 445–465, 2010.
- 21 Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, 2012.
- 22 Leo de Castro, Chiraag Juvekar, and Vinod Vaikuntanathan. Fast vector oblivious linear evaluation from ring learning with errors. *IACR Cryptol. ePrint Arch.*, 2020:685, 2020. URL: <https://eprint.iacr.org/2020/685>.
- 23 Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. Cryptology ePrint Archive, Report 2020/1446, 2020. URL: <https://eprint.iacr.org/2020/1446>.
- 24 Tore Kasper Frederiksen, Jesper Buus Nielsen, and Claudio Orlandi. Privacy-free garbled circuits with applications to efficient zero-knowledge. In *EUROCRYPT 2015, Part II*, pages 191–219, 2015.
- 25 Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In *STOC 2014*, pages 495–504, 2014.
- 26 Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT*, 2013.
- 27 Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. Zkboo: Faster zero-knowledge for boolean circuits. In *USENIX Security 2016*, 2016.
- 28 Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4):27:1–27:64, 2015.
- 29 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
- 30 Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, pages 305–326, 2016.
- 31 David Heath and Vladimir Kolesnikov. Stacked garbling for disjunctive zero-knowledge proofs. In *EUROCRYPT 2020, Part III*, pages 569–598, 2020.
- 32 Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP 2002*, pages 244–256, 2002.
- 33 Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short PCPs. In *CCC*, 2007.
- 34 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In *EUROCRYPT 2011*, pages 406–425, 2011.
- 35 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.
- 36 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC 2009*, pages 294–314, 2009.
- 37 Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *CCS 2018*, pages 525–537. ACM, 2018.
- 38 Dakshita Khurana, Rafail Ostrovsky, and Akshayaram Srinivasan. Round optimal black-box “commit-and-prove”. In *Theory of Cryptography Conference*, pages 286–313, 2018.
- 39 Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proof. In *CRYPTO 1989*, pages 545–546. Springer, 1989.
- 40 Alex Lombardi, Willy Quach, Ron D. Rothblum, Daniel Wichs, and David J. Wu. New constructions of reusable designated-verifier nizks. In *CRYPTO 2019*, pages 670–700, 2019.
- 41 Payman Mohassel and Mike Rosulek. Non-interactive secure 2pc in the offline/online and batch settings. In *EUROCRYPT 2017, Part III*, pages 425–455, 2017.
- 42 Moni Naor and Benny Pinkas. Oblivious polynomial evaluation. *SIAM Journal on Computing*, 35(5):1254–1281, 2006.

- 43 Willy Quach, Ron D. Rothblum, and Daniel Wichs. Reusable designated-verifier nizks for all NP from CDH. In *EUROCRYPT 2019*, pages 593–621, 2019.
- 44 Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In *CCS 2019*, pages 1055–1072, 2019.
- 45 Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *IEEE Symposium on Security and Privacy (S&P)*, 2021. Full version: <https://eprint.iacr.org/2020/925>.
- 46 Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In *CRYPTO 2019, Part III*, pages 733–764, 2019.
- 47 Jiaheng Zhang, Weijie Wang, Yinuo Zhang, and Yupeng Zhang. Doubly efficient interactive proofs for general arithmetic circuits with linear prover time. Cryptology ePrint Archive, Report 2020/1247, 2020. URL: <https://eprint.iacr.org/2020/1247>.
- 48 Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *2020 IEEE Symposium on Security and Privacy*, pages 859–876, 2020.
- 49 ZKProof. *ZKProof Standards*, 2020. URL: <https://zkproof.org>.

ZK-PCPs from Leakage-Resilient Secret Sharing

Carmit Hazay ✉

Bar-Ilan University, Ramat Gan, Israel

Muthuramakrishnan Venkatasubramaniam ✉

University of Rochester, NY, USA

Mor Weiss ✉

Bar-Ilan University, Ramat Gan, Israel

Abstract

Zero-Knowledge PCPs (ZK-PCPs; Kilian, Petrank, and Tardos, STOC ‘97) are PCPs with the additional zero-knowledge guarantee that the view of any (possibly malicious) verifier making a bounded number of queries to the proof can be efficiently simulated up to a small statistical distance. Similarly, ZK-PCPs of Proximity (ZK-PCPPs; Ishai and Weiss, TCC ‘14) are PCPPs in which the view of an adversarial verifier can be efficiently simulated with few queries to the input.

Previous ZK-PCP constructions obtained an exponential gap between the query complexity q of the honest verifier, and the bound q^* on the queries of a malicious verifier (i.e., $q = \text{poly log}(q^*)$), but required either exponential-time simulation, or adaptive *honest* verification. This should be contrasted with standard PCPs, that can be verified non-adaptively (i.e., with a single round of queries to the proof). The problem of constructing such ZK-PCPs, *even when* $q^* = q$, has remained open since they were first introduced more than 2 decades ago. This question is also open for ZK-PCPPs, for which no construction with non-adaptive honest verification is known (not even with exponential-time simulation).

We resolve this question by constructing the *first* ZK-PCPs and ZK-PCPPs which *simultaneously* achieve *efficient* zero-knowledge simulation and *non-adaptive* honest verification. Our schemes have a square-root query gap, namely $q^*/q = O(\sqrt{n})$ where n is the input length.

Our constructions combine the “MPC-in-the-head” technique (Ishai et al., STOC ‘07) with leakage-resilient secret sharing. Specifically, we use the MPC-in-the-head technique to construct a ZK-PCP variant over a large alphabet, then employ leakage-resilient secret sharing to design a new alphabet reduction for ZK-PCPs which preserves zero-knowledge.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques; Theory of computation → Cryptographic protocols; Security and privacy → Cryptography; Theory of computation → Proof complexity

Keywords and phrases Zero Knowledge, Probabilistically Checkable Proofs, PCPs of Proximity, Leakage Resilience, Secret Sharing

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.6

Related Version *Full Version:* <https://eprint.iacr.org/2021/606> [20]

Funding The first and third authors are supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister’s Office. The first author is supported by ISF grant No. 1316/18. The first and second authors are supported by DARPA under Contract No. HR001120C0087. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

Acknowledgements We thank the anonymous ITC’21 reviewers for their helpful comments, in particular for pointing out the connection to RPEs and noting that the ZK code of [14, Theorem 2.2] is equivocal.



© Carmit Hazay, Muthuramakrishnan Venkatasubramaniam, and Mor Weiss;
licensed under Creative Commons License CC-BY 4.0

2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 6; pp. 6:1–6:21



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Probabilistically Checkable Proofs (PCPs) [2, 3] allow a probabilistic verifier to check the validity of a given statement by only querying few proof bits. Zero-Knowledge (ZK) proofs [18] allow a prover to convince a verifier of the validity of a statement, without revealing any additional information to the verifier. This work focusses on *Zero-Knowledge Probabilistically Checkable Proofs (ZK-PCPs)* (and *ZK-PCPs of proximity*), which combine the advantages of these two types of proof systems. Before describing our main results, we first give a short overview of these proof systems.

Probabilistically Checkable Proofs (PCPs)

PCPs [2, 3] allow a randomized efficient verifier \mathcal{V} with oracle access to a purported proof π to verify an NP-statement of the form “ $x \in \mathcal{L}$ ” by reading only few bits of π . The proof can be efficiently generated given the NP witness, and the verifier accepts true claims with probability 1, whereas false claims are accepted with low probability (which is called the soundness error). The celebrated PCP theorem [2, 3, 15] states that any NP language has a PCP system with soundness error $1/2$, in which the verifier reads only $O(1)$ bits from a polynomial-length proof (soundness can be amplified through repetition). An attractive feature of these PCP systems is that the verifier is *non-adaptive*, namely it makes a single round of queries to the proof. *PCPs of Proximity (PCPPs)* [16, 6, 15] are a generalization of PCPs in which the verifier does not read its entire input. Instead, \mathcal{V} has oracle access to x, π , and wishes to check whether x is close to \mathcal{L} (in relative Hamming distance). The best PCPP constructions for NP [7, 27] obtain comparable parameters to the PCP systems described above, where any x which is δ -far from \mathcal{L} in relative Hamming distance is rejected with high probability, and δ is a constant or even inverse polylogarithmic (δ is called the *proximity parameter*).

Zero-Knowledge (ZK) proofs

ZK proofs [18] allow a randomized efficient prover to prove an NP-statement of the form “ $x \in \mathcal{L}$ ” to a randomized efficient verifier, while guaranteeing that true claims are accepted with probability 1, false claims are rejected with high probability, and the verifier learns no information about the corresponding NP-witness. This last property, known as *zero-knowledge*, is formalized by requiring that for any (possibly malicious) efficient verifier \mathcal{V}^* , there exists an efficient *simulator* machine that has access only to the statement x , and can simulate the interaction of \mathcal{V}^* with the honest prover.

Zero-Knowledge PCPs (ZK-PCPs)

ZK-PCPs [26] combine the attractive features of PCPs and ZK proofs. Specifically, ZK-PCPs are PCPs in which the prover \mathcal{P} is randomized, and the proof π has the following zero-knowledge guarantee: the view of every (possibly malicious, possibly unbounded) verifier \mathcal{V}^* that makes an a-priori bounded number of queries to the proof, can be efficiently simulated up to a small statistical distance. We remark that restricting \mathcal{V}^* to making an a-priori bounded number of queries is inherent to obtaining ZK with polynomial-length proofs.

The first ZK-PCP constructions for NP [26, 22] obtain ZK against any verifier \mathcal{V}^* that is restricted to querying at most $q^* = q^*(|x|)$ proof bits, with proofs of length $\text{poly}(q^*)$ that can be verified with $\text{polylog}(q^*)$ queries and have a negligible soundness error. In particular, the *query gap* q^*/q – the ratio between the query complexities of the malicious and honest

verifiers – obtained by these constructions is exponential.¹ Unfortunately, obtaining ZK in [26, 22] did not come without a cost: it required the *honest* verifier to be adaptive, namely to make *several* rounds of queries to the proof (where the queries of each round depend on the answers to previous queries). In cryptographic applications of ZK-PCPs (e.g., in [25]) this blows-up the round complexity of resultant protocols. In particular, every round of queries which the verifier makes to the ZK-PCP induces two communication rounds in the interactive protocols which rely on ZK-PCPs.

Ishai and Weiss [24] introduce the notion of *Zero-Knowledge PCPPs (ZK-PCPPs)*. Similar to ZK-PCPs, the ZK-PCPP prover is randomized, and zero-knowledge means that the view of any verifier \mathcal{V}^* making q^* queries to *the input* and the proof can be efficiently simulated, up to a small statistical distance, *by making only q^* queries to the input*. They use similar techniques to [26, 22] to obtain ZK-PCPPs for NP with comparable parameters to the ZK-PCPs of [26, 22], where the proximity parameter δ is constant or inverse polylogarithmic. These ZK-PCPPs also require adaptive verification, which increases the round complexity in their cryptographic applications [24, 30].

As discussed in Sections 2.1.4 and 2.2.2 below, adaptive verification is in fact inherent to the constructions of [26, 22, 24]. Indeed, these schemes are obtained by combining a PCP or PCPP with a weak zero-knowledge guarantee that only holds against the *honest* verifier, with an information-theoretic commitment primitive called locking schemes [26]. This latter primitive requires adaptive opening, which causes the resultant ZK-PCP verifier to be adaptive.

Can ZK-PCPs be verified non-adaptively?

Motivated by the goal of obtaining ZK for PCPs at no additional cost, Ishai et al. [25] gave a partial answer to this question. Specifically, they construct ZK-PCPs with similar parameters to the schemes of [26, 22] in which the honest verifier is *non-adaptive*, but with a weaker zero-knowledge guarantee compared to standard ZK-PCPs: the zero-knowledge simulator is inefficient (this is also known as *witness-indistinguishability*). Alternatively, they obtain ZK with efficient simulation against *computationally-bounded* verifiers, assuming the existence of one-way functions and a common random string. The techniques of [25] diverge from the standard method of designing ZK-PCPs [26, 22] discussed above. Specifically, the ZK-PCP of [25] is based on a novel connection to *leakage-resilient circuits*, which are circuits that operate over encoded inputs, and resist certain “side channel” attacks in the sense that such attacks reveal nothing about the input other than the output. Unfortunately, the weaker ZK guarantee of the ZK-PCPs of [25] carries over to any application in which these systems are used. Moreover, [25] give evidence that inefficient simulation is inherent to their technique of using leakage-resilient circuits.

Non-adaptive honest vs. malicious verification

It is instructive to note that while having non-adaptive (*honest*) verification is a feature of the system (since it guarantees that the honest verifier can achieve soundness with a single round of queries), having zero-knowledge against *non-adaptive malicious* verifiers is a restriction of the system, since there is no ZK guarantee against *adaptive* malicious verifiers, that make several rounds of queries to the proof.

¹ We stress that a larger gap is preferable to a smaller one, since it means the proof can be verified with few queries, while guaranteeing zero-knowledge even when a malicious verifier makes many more queries (compared to the honest verifier).

We note that in [25], leakage-resilient circuits fall short of yielding ZK-PCPs with full-fledged zero-knowledge not only because the simulation is inefficient, but also because *zero-knowledge holds only against non-adaptive (malicious) verifiers*. Ishai et al. [25] obtain ZK (with inefficient simulation) against adaptive verifiers by combining leakage-resilient circuits with techniques of [9]. These techniques incur an exponential blowup in the complexity of the ZK simulator, but did not pose a problem for [25] since their simulator (even against *non-adaptive malicious* verifiers) was already inefficient.

The current landscape of ZK-PCPs is unsatisfying. Current ZK-PCP constructions either require adaptive verification [26, 22], or guarantee only a weak form of ZK with an inefficient simulator [25]. This holds regardless of the query gap, i.e., even if we restrict malicious verifiers to making *the same* number of queries as the honest verifier. For ZK-PCPPs, the situation is even worse: *no* constructions with non-adaptive verification are known (not even with inefficient simulation). This state of affairs gives rise to the following natural question:

Do there exist ZK-PCPs (and ZK-PCPPs) with non-adaptive verification and efficient simulation?

As we discuss in Sections 2.1.4 and 2.2.2 below, the limitations of existing ZK-PCP and ZK-PCPP constructions seem to be inherent to the respective techniques they employ to obtain ZK. This seems to imply that obtaining both non-adaptive verification and efficient simulation requires new techniques. Or maybe such objects do not even exist?

1.1 Our results

In this work, we answer our research question in the affirmative: we construct ZK-PCPs and ZK-PCPPs that can be verified non-adaptively and have efficient zero-knowledge simulation. Unlike the schemes of [26, 22, 24, 25], which obtain an exponential gap between the query complexities of the malicious and honest verifiers, we are only able to obtain a *polynomial* query gap (q^* vs. $(q^*)^\epsilon$, for some constant $\epsilon \in (0, 1)$).

In the following, we say that a PCP (PCPP, resp.) system is a non-adaptive q -query q^* -ZK-PCP (q^* -ZK-PCPP, resp.) if it is perfectly ZK against a (possibly malicious, possibly adaptive) verifier making q^* queries, and achieves a $\text{negl}(q^*)$ soundness error where the honest verifier makes q non-adaptive queries to the proof.

Specifically, we obtain the following results:

► **Theorem 1 (Non-Adaptive ZK-PCPs with Efficient Simulation).** *There exists a constant $\epsilon \in (0, 1)$ such that for any ZK parameter $q^* \in \mathbb{N}$ there exists a non-adaptive $(q^*)^\epsilon$ -query $\Omega(q^*)$ -ZK-PCP for NP.*

► **Theorem 2 (Non-Adaptive ZK-PCPPs with Efficient Simulation).** *Let $n \in \mathbb{N}$ be an input length parameter. Then there exists a constant $c > 0$ such that for any proximity parameter $\delta \geq 1/\sqrt{n}$, there exists a non-adaptive q -query q^* -ZK-PCPP for NP with proximity parameter δ , $q^* = \Omega(n^{c+1})$, and $q = \tilde{O}(n^{c+1/2})$.*

Our non-adaptive ZK-PCPs and ZK-PCPPs can be plugged-into the applications described in [24, 25, 30], and will reduce the round complexity of the resultant protocols.²

² In this context, we note that if one only requires ZK against the *honest* verifier, then non-adaptive ZK-PCPs and ZK-PCPPs are known. (This is implicit in [26] and [24] for ZK-PCPs and ZK-PCPPs respectively, via standard soundness amplification.) Consequently, our non-adaptive ZK-PCPs and ZK-PCPPs (with ZK against *malicious* verifiers) do not improve the round complexity in applications that only require ZK against the honest-verifier (e.g., the ZK arguments of [22], and the commit-and-prove protocols of [24]).

Our constructions show that leakage-resilience techniques *can* be used to construct ZK-PCPs (and ZK-PCPPs) with both non-adaptive (honest) verification and efficient simulation. Specifically, we circumvent the negative result of [25] on the limitations of using leakage-resilient *circuits*, by relying on leakage-resilient *secret sharing* [17, 12] secure against local leakage [19, 8, 1]. Compared to leakage-resilient circuits, leakage-resilient secret sharing has the weaker guarantee of only protecting *information* from leakage, whereas leakage-resilient circuits also protect *computation*. However, this weaker guarantee suffices for our needs, and admits leaner and more efficient constructions compared to those of leakage-resilient circuits (and applications using them). Specifically, we use leakage resilient secret sharing to design a new alphabet reduction procedure that transforms a ZK-PCP over a large alphabet to a ZK-PCP over bits, while preserving zero-knowledge.

2 Our Techniques

We now give more details about our ZK-PCP and ZK-PCPP constructions.

2.1 ZK-PCPs with Non-Adaptive Verification and Efficient Simulation

Our starting point is a ZK-PCP implicit in the work of [21]. They use secure Multi-Party Computation (MPC) protocols to construct a ZK-PCP variant *over a large (poly-sized) alphabet* with efficient ZK simulation, that can be verified non-adaptively. Their ZK-PCP suffers from two disadvantages. First, strictly speaking it is not a ZK-PCP, since in standard ZK-PCPs the proof is a bit string, whereas the ZK-PCP of [21] is over a large alphabet. Second, their construction has no query gap, namely the proof is ZK against verifiers querying q^* proof symbols, but to get soundness the honest verifier must also make q^* queries.

2.1.1 Amplifying the Query Gap in the ZK-PCP of [21]

To prove that $x \in \mathcal{L}$ for some NP-language \mathcal{L} with a corresponding NP relation $\mathcal{R} = \mathcal{R}(x, w)$, Ishai et al. [21] employ an n -party protocol that computes the function $f_{\mathcal{R}}(x, w_1, \dots, w_n) = \mathcal{R}(x, \oplus w_i)$, where x is a common input, and w_i is the input of the i th party P_i . The prover executes the MPC protocol “in its head”, obtaining the *views* of all parties P_1, \dots, P_n in the execution (the view of party P_i consists of its input, random input, and all messages it received during the execution). The proof consists of all these views, where each view is a symbol in the resultant proof. To verify the proof, the verifier reads several views, and checks that: (1) the output reported in all views is 1; and (2) the views are *consistent*, namely for every pair V_i, V_j of queried views of P_i, P_j (respectively), the incoming messages from P_i reported in V_j are the messages P_i would send in the protocol given its view V_i , and vice versa.

To get q^* -ZK, the protocol should be *private* against q^* (semi-honest) parties, in the sense that they learn nothing from the execution except their inputs and the output. For soundness, [21] rely on a notion of correctness against q^* corrupted parties (known as *robustness*), guaranteeing that even if q^* parties arbitrarily deviate from the protocol, they cannot cause an honest party to output 1 in a protocol execution on $x \notin \mathcal{L}$. We revisit their analysis, and show that general MPC protocols yield a square root query gap. That is, given a q^* -private and q^* -robust MPC protocol, the resultant ZK-PCP over a large alphabet is ZK against a (possibly malicious) verifier querying q^* proof symbols, and can be non-adaptively verified with only $\sqrt{q^*}$ queries, with a negligible soundness error. This already yields a non-trivial ZK-PCP *over a large alphabet*. Our tighter analysis can be found in the full version [20].

2.1.2 Alphabet Reduction for ZK-PCPs

Next, we address the fact that the ZK-PCP of [21] is over a large alphabet. For standard PCPs, one can easily reduce the alphabet Σ over which the proof π is defined to $\{0, 1\}$ by simply replacing each alphabet symbol with a bit string, thus obtaining a new proof π' over $\{0, 1\}$. This would increase the proof length and the query complexity of the honest verifier by a multiplicative $\log |\Sigma|$ factor, but would not otherwise affect the system.³

Unfortunately, applying this transformation to *zero-knowledge* PCPs might render the resultant scheme totally insecure. Indeed, while the system would still be ZK against verifiers making q^* queries, the query gap now reduces since the query complexity of the *honest* verifier (i.e., the number of queries it *must* make to obtain soundness) increases. Specifically, depending on $|\Sigma|$, the honest verifier might now need to make $> q^*$ queries, but π' might not be ZK even against $q^* + 1$ malicious queries. As a result, π' might not be ZK even against *malicious* verifiers that make *fewer* queries than the honest verifier! Indeed, a malicious verifier \mathcal{V}^* with oracle access to π' is not restricted to querying “whole” symbols of π , i.e., reading the entire substring of π' that corresponds to a symbol of π . On the contrary, \mathcal{V}^* might read “*parts*” of symbols, thus potentially gaining (partial) information on $q^* + 1$ symbols of π , and possibly violating the ZK guarantee of the original system.

The trivial alphabet reduction for PCPs described above fails because querying *even a single bit* in the bit string s_σ representing a symbol $\sigma \in \Sigma$ might reveal information about σ . Therefore, to make this alphabet reduction work for *zero-knowledge* PCPs, we must guarantee that querying few bits of s_σ reveals nothing about σ . We do so using leakage-resilient secret sharing.

At a high level, a (t -threshold) *Secret Sharing Scheme (SSS)* is a method of distributing a secret s among n parties by giving each party P_i a share Sh_i , such that any t shares reveal no information about s , but any $t + 1$ shares completely determine s . A *Leakage-Resilient Secret Sharing Scheme (LR-SSS)* against local leakage [19, 8, 1] has the added feature of resisting leakage on the shares, in the following sense. The secret s remains hidden given t shares, *as well as few leakage bits computed separately on every other share*.

Given a ZK-PCP system $(\mathcal{P}, \mathcal{V})$ over a large alphabet Σ , and a LR-SSS for secrets in $\{0, 1\}^{\log |\Sigma|}$, our alphabet reduction works as follows. The prover \mathcal{P}' uses \mathcal{P} to generate a proof $\pi = \sigma_1 \dots \sigma_N$ over Σ , replaces every σ_i with its bit-representation s_{σ_i} , which it secret shares using the LR-SSS. The proof π' consists of the secret sharings of $s_{\sigma_1}, \dots, s_{\sigma_N}$. To verify the proof, the verifier \mathcal{V}' emulates \mathcal{V} , where a query Q of \mathcal{V} to its proof π is answered as follows. \mathcal{V}' uses the secret sharing of s_{σ_Q} (from its own proof oracle π') to reconstruct σ_Q , which it then provides to \mathcal{V} .⁴

The PCP system obtained through the reduction preserves the completeness and soundness of $(\mathcal{P}, \mathcal{V})$, and guarantees ZK against *non-adaptive* (possibly malicious) verifiers that are restricted to making (roughly) $q^{**} = q^* \ell t$ queries to the proof, where $(\mathcal{P}, \mathcal{V})$ is ZK against verifiers querying q^* proof symbols, and the LR-SSS is private against any t shares as well as ℓ leakage bits from every other share.

³ We note that several PCP constructions (e.g. [15]) use more elaborate alphabet reduction techniques for *efficiency reasons* (in particular, their goal is to achieve quasi-linear length proofs with $O(1)$ query complexity and a constant soundness error). A $\log |\Sigma|$ blowup is less significant in the context of *zero-knowledge* PCPs, where the query complexity is anyway $\omega(1)$ since we wish to have a negligible soundness error.

⁴ Due to some technical issues, the construction is actually somewhat more involved, see Section 4 and the full version [20] for the construction and further details.

To see why ZK holds, we split the proof π' into N “sections”, where the i th section contains the secret sharing of s_{σ_i} , and s_{σ_i} is the bit-representation of the i th symbol σ_i of the original proof π . Roughly (and somewhat inaccurately), the queries of any non-adaptive (possibly malicious) verifier \mathcal{V}^* querying at most q^{**} proof bits divide the sections of π' into two groups.

1. “Light” sections, from which \mathcal{V}^* reads at most ℓt bits. In particular, for each such section containing the secret shares $\text{Sh}_1, \dots, \text{Sh}_n$ of the bit-representation s_σ of a symbol σ , there can be at most t shares from which \mathcal{V}^* reads more than ℓ bits, and each other share is queried only ℓ times. Therefore, the leakage-resilience of the LR-SSS guarantees that s_σ (and consequently also σ) remains entirely hidden.
2. “Heavy” sections, from which \mathcal{V}^* queries more than ℓt bits. Notice that there can only be at most q^* such sections, and \mathcal{V}^* obtains no information about the symbols of π encoded in “light” sections, so the queries to the “heavy” sections can be simulated by the ZK of $(\mathcal{P}, \mathcal{V})$.

(The full – and accurate – analysis appears in the proof of Theorem 18, which can be found in the full version [20].)

In summary, combining a ZK-PCP over a large alphabet with a SSS that resists probing leakage, we obtain a ZK-PCP over $\{0, 1\}$. Instantiating the transformation with the ZK-PCP of [21] (with our improved analysis) together with the LR-SSS of [29] yields a ZK-PCP with the parameters of Theorem 1 that is ZK against (possibly malicious and unbounded) verifiers that only make *non-adaptive* queries to their proof oracle.

2.1.3 Amplifying to ZK Against Adaptive Verifiers

The analysis of our alphabet reduction for ZK-PCPs crucially relied on the fact that the malicious verifier \mathcal{V}^* was *non-adaptive*. Indeed, the queries to “light” and “heavy” sections are simulated differently (using the LR-simulator of the LR-SSS, and the ZK-simulator of $(\mathcal{P}, \mathcal{V})$, respectively), meaning the simulator for $(\mathcal{P}', \mathcal{V}')$ needs to know at the onset of the simulation which sections are “heavy” and which are “light”. Obtaining ZK against *adaptive* verifiers seems to require a stronger leakage-resilience guarantee with a two-phase flavor similar to the locking schemes of [26, 22]. Specifically, in the first phase of the simulation, the LR-simulator Sim_{LR} should be able to answer adaptive leakage queries as in a standard (adaptively-secure) LR-SSS. However, unlike standard LR-SSSs, at some point the simulation may move to a second phase. In the second phase the simulator Sim_{LR} is given a secret \mathbf{s} , and should be able to “explain” \mathbf{s} by providing an entire secret sharing of \mathbf{s} which is random subject to being consistent with the previously-simulated answers to leakage queries. We formalize this notion, introducing *equivocal SSSs* as a generalization of standard LR-SSSs, and provide a construction based on codes with leakage-resilience guarantees (see Section 2.3 for further details).

Applying our alphabet reduction to $(\mathcal{P}, \mathcal{V})$ and an *equivocal* SSS now yields a ZK-PCP with ZK against any – possibly malicious *and adaptive* – verifier \mathcal{V}^* making at most q^{**} queries. Indeed, the ZK-PCP simulator Sim starts the simulation by answering all queries using the LR-simulator Sim_{LR} of the equivocal SSS. Whenever the number of queries to a certain proof section passes some threshold, Sim uses the ZK-simulator Sim_{ZK} to simulate the underlying symbol σ of π . Then, Sim provides the bit-representation of σ to Sim_{LR} as the secret-shared secret. The equivocation property of the SSS guarantees that Sim_{LR} can now “explain” this secret by providing an entire secret sharing which is consistent with the previous leakage. These secret shares can be used to answer any further queries to that section of the proof. The construction appears in Section 4.1, and the analysis can be found in the full version [20].

To sum up, combining a ZK-PCP over a large alphabet with an *equivocal* SSS against probing leakage yields a ZK-PCP over $\{0, 1\}$, where zero-knowledge holds against *adaptive* verifiers. Instantiating the transformation with the ZK-PCP of [21] and the equivocal scheme described in section 2.3.2 yields a ZK-PCP with the parameters of Theorem 1.

2.1.4 Why Do Previous Approaches of Constructing Non-Adaptive ZK-PCPs Fail?

It is instructive to discuss why our approach of combining alphabet reduction with LR secret sharing succeeds in simultaneously obtaining non-adaptive verification and efficient simulation, whereas previous approaches [26, 22, 25] could only achieve one of these properties.

As noted above, the ZK-PCPs of [26, 22] are obtained by combining PCPs that are ZK against the honest verifier with locking schemes. In effect, the locking schemes are used to “force” the queries of a malicious verifier to be distributed (almost) as the queries of *the honest verifier*. This transformation causes adaptive verification due to two reasons: first, the original proof is altered in such a way that necessitates adaptive queries to verify it. Second, the locking schemes themselves require adaptive opening. We are faced with a similar challenge, where the queries of the malicious verifier might drastically deviate from those of an honest verifier (namely, \mathcal{V}^* might query “parts” of symbols, whereas the honest verifier always queries whole symbols). However, instead of “forcing” the queries of \mathcal{V}^* to “look” honest, we allow \mathcal{V}^* to make any set of queries, but guarantee that queries to “parts” of symbols reveal no information on the underlying symbol.

The ZK-PCP of [25] uses a different approach. Their starting point is a *non-ZK* PCP, and they use leakage-resilient circuits to protect *the entire PCP generation*. That is, the queries of the verifier are interpreted as leakage on the process of generating the PCP from the NP-witness, and by protecting this entire computation from leakage, they obtain ZK. More specifically, they change the NP statement which is being verified: instead of verifying that (x, w) is a satisfying input for the verification circuit C of the NP-relation \mathcal{R} , the honest verifier checks whether the leakage-resilient version \widehat{C} of C is satisfiable. Therefore, soundness of the resultant ZK-PCP system crucially relies on the fact that if there exists no w such that $C(x, w) = 1$, then there exists no w' such that $\widehat{C}(x, w') = 1$,⁵ a notion which they call *SAT-respecting*. Ishai et al. [25] give evidence that SAT-respecting leakage-resilient circuits with efficient LR-simulation (for the leakage classes needed to construct ZK-PCPs) exist only for languages in BPP. The inefficient LR-simulation is the cause of ZK with inefficient simulation in their ZK-PCPs. We circumvent their negative results by using LR-SSSs to protect *information* – instead of using LR *circuits* to protect *computation* – and apply the LR-SSS to PCPs with ZK guarantees (whereas [25] use standard PCPs).

2.2 ZK-PCPPs with Non-Adaptive Verification and Efficient Simulation

We extend our techniques to apply to PCPs *of Proximity*. Specifically, our alphabet reduction could also be applied to ZK-PCPPs, which reduces the task of designing ZK-PCPPs with non-adaptive verification to designing such ZK-PCPPs over a large alphabet.

⁵ In fact, \widehat{C} operates on encoded inputs, however to simplify the discussion we disregard this at this point, and provide a more accurate discussion in Section 2.2.2 below.

2.2.1 A ZK-PCPP with Non-Adaptive Verification Over Large Alphabets

The first step is to design a ZK-PCPP over a large alphabet. We do so by presenting a variant of the system of [21] in which the verifier does not read its entire input. Recall that the proof in the ZK-PCP of [21] consists of the views of all parties in an execution of an MPC protocol for the function $f_{\mathcal{R}}(x, w_1, \dots, w_n) = \mathcal{R}(x, \oplus w_i)$, where x is a common input, and w_i is the input of the i th party P_i . In particular, a single proof symbol (i.e., a single view) reveals the entire input x . This is problematic in the context of ZK-PCPPs, in which the proof is required to hide not only the NP-witness w , but also most of the input x , in the sense that a verifier making few queries learns only few physical bits of x .

Thus, no single party can hold the entire input x . Instead, following [24] we introduce m additional “input parties” (where $m = |x|$), such that the MPC protocol is over $m + n$ parties P_1, \dots, P_{m+n} . The inputs of parties P_1, \dots, P_m are x_1, \dots, x_m (respectively), and the inputs of parties P_{m+1}, \dots, P_{m+n} are w_1, \dots, w_n (respectively). Proof generation from this revised protocol is similar to the original construction of [21], and verification is also similar, except that whenever the verifier queries a view of $P_i, i \in [m]$, it also queries x_i from its input oracle, and checks that x_i is the input of P_i reported in the view.

If the MPC protocol is q^* -private, then we are guaranteed that q^* queries to the proof reveal at most q^* bits of x . Indeed, the q^* -privacy of the protocol guarantees that the views of q^* parties reveal no information other than their inputs (which is at most q^* bits of x) and their output (which is 1).

As for soundness, we show that our improved analysis (discussed in Section 2.1.1 above) extends to PCPPs. Specifically, we show that if the MPC protocol is q^* -robust then the resultant ZK-PCPP is sound with proximity parameter $\delta \geq 1/\sqrt{|x|}$, namely the verifier rejects (with high probability) inputs that are δ -far from the language. Indeed, let x be δ -far from \mathcal{L} , and notice that any (possibly ill-formed) proof π^* for x determines an effective input x^* for the underlying MPC protocol (x^* is obtained by concatenating the inputs reported in the views of P_1, \dots, P_m). We show that if x^* is δ -far from x then with overwhelming probability the verifier will query a view on which x, x^* differ, in which case it rejects with probability 1. Otherwise, x^* is δ -close to x , implying $x^* \notin \mathcal{L}$, in which case our improved analysis of Section 2.1.1 essentially shows that the PCPP verifier rejects with overwhelming probability. This yields the first ZK-PCPP with non-adaptive verification and efficient simulation, but the proof is over a large alphabet.

Combining this ZK-PCPP over a large alphabet with our alphabet reduction, we obtain *the first* ZK-PCPPs over $\{0, 1\}$ with non-adaptive verification. Moreover, the system has efficient ZK simulation. Specifically, combining the ZK-PCPP over a large alphabet with a LR-SSS gives a ZK-PCPP with ZK against *non-adaptive malicious* verifiers, whereas combining it with an equivocal SSS gives a full-fledged ZK-PCPP. Instantiating the equivocal SSS with the scheme of Section 2.3.2 gives a ZK-PCPP with the properties of Theorem 2. Due to space limitations, the construction and analysis of our ZK-PCPP system is deferred to the full version [20].

2.2.2 Why Do Previous Approaches of Constructing Non-Adaptive ZK-PCPPs Fail?

We now give some intuition as to why LR-SSS is useful to obtaining ZK-PCPPs with non-adaptive verification, whereas ZK-PCPP constructions based on leakage-resilient circuits seem to fail. Recall that a leakage-resilient circuit operates over encoded inputs, where leakage from some function class \mathcal{F} on the internal wire values of the circuit reveals no

information about the input other than the output. In particular, this implies that the input encoding should resist leakage from \mathcal{F} , since leakage can be applied to the input wires (which carry the encoded inputs).

Recall that the verifier wishes to verify that $(x, w) \in \mathcal{R}$, namely that (x, w) satisfies the verification circuit C of \mathcal{R} . When using leakage-resilient circuits to verify this claim, the circuit C is replaced with its leakage-resilient variant \widehat{C} , which operates over encoded inputs, where the queries of the verifier are interpreted as leakage on the wire values of \widehat{C} . This raises the question of how to incorporate x into the computation. Syntactically, \widehat{C} cannot operate directly on the unencoded input x , but if \widehat{C} operates on an encoding of x , the prover can cheat by providing an encoding of some $x^* \neq x$. (The verifier will not be able to tell the difference because the input encoding is resilient against leakage, namely against the verifier queries.) The solution of [25] is to first *hard-wire* x into C , i.e. replace C with $C_x = C(x, \cdot)$, and then generate the leakage-resilient version \widehat{C}_x of C_x . The verifier will now verify that \widehat{C}_x is satisfiable.

While this solves the problem for ZK-PCPs, it cannot be applied in the context of ZK-PCPPs. Indeed, verifying that \widehat{C}_x is satisfiable requires knowing the structure of \widehat{C}_x . This is indeed the case for ZK-PCPs, since x is known to the verifier in its entirety, so the verifier can locally construct \widehat{C}_x . However, the ZK-PCPP verifier does not know all of x , nor do we want it to – the advantage of ZK-PCPPs over ZK-PCPs (which is crucially exploited by cryptographic applications of ZK-PCPPs) is that the verifier can be sublinear in the input length, and verify the claim without learning “too much” about the input. Therefore, we cannot hard-wire x into the verification circuit, and so it is unclear how the verifier would verify consistency of its own input with the one used to evaluate C .

Finally, we note that even if one were to solve this issue of how to handle the input, using leakage-resilient circuits would incur inefficient ZK simulation in the resultant ZK-PCPP, due to the negative results of [25].

2.3 Equivocal Secret Sharing

We generalize the notion of LR-SSS [17, 12] secure against local leakage [19, 8, 1] by introducing *equivocal SSSs*. We then construct a 1-party equivocal SSS based on codes with probing-resilience. We note that while a 1-party SSS is useless as a means of sharing a secret, its equivocation property gives a meaningful way of encoding a secret in a leakage-resilient (in fact, equivocal) manner. In particular, such schemes suffice for constructing ZK-PCPs and ZK-PCPPs as described in Sections 2.1 and 2.2 above. Moreover, since 1-party schemes can be more efficient than multi-party schemes, using 1-party schemes results in more efficient ZK-PCPs and ZK-PCPPs.

2.3.1 Equivocal SSS: Definition

Recall that a standard t -threshold n -party SSS guarantees that the secret remains entirely hidden given any t of the n shares. LR-SSS enhances this privacy property to hold even given leakage on the other shares. We will focus on resisting adaptive (t, ℓ) -local probing leakage, in which the leakage reveals t shares, as well as ℓ bits from every other share. This can be formalized by comparing the real execution with an ideal experiment in which an efficient simulator Sim , that has no knowledge of the secret, answers the leakage queries.

An equivocal SSS generalizes the notion of (adaptive) LR-SSS by considering a two-phase ideal experiment, where the first phase is similar to the ideal experiment of LR-SSS. At the end of the first phase, the adversary can choose whether to continue to the second phase.

In the second phase, the simulator is given a secret \mathbf{s} , and must generate an entire secret sharing of \mathbf{s} , which is given to the adversary. The adversary should have only a negligible advantage in distinguishing the real execution from the ideal experiment, as long as it didn't violate the leakage restrictions. That is, as long as at the onset of the second phase, the adversary obtained at most t shares, and probed at most ℓ bits from every other share. Since the adversary can choose *not* to continue to the second phase of the simulation, this notion strictly generalizes the notion of a LR-SSS. Notice that we make no restriction on the computational power of the adversary. The definition appears in Section 3.3.

2.3.2 Equivocal SSS: Construction

We use a 1-party equivocal SSS that resists $(0, \ell)$ -local probing leakage [19, 8, 1], where ℓ is a constant fraction of the share size. Considering 1-party schemes suffices for the ZK-PCP and ZK-PCPP application, and admit lean constructions that result in more efficient ZK-PCPs and ZK-PCPPs (in terms of query complexity and proof length).

2.3.2.1 Existing leakage-resilient encodings

A 1-party equivocal SSS gives a method of encoding data such that the resultant encoding is equivocal, and consequently also leakage-resilient. We note that leakage-resilient encodings have been considered before under different names. ZK codes [13, 14, 23] are encodings that resists *non adaptive probing* leakage, i.e., these are 0-threshold 1-party SSSs that resist non-adaptive probing leakage. Leakage-resilient storage [17, 12] encodes the data into two parts that resist *adaptive* leakage from *each part separately*. Thus, leakage-resilient storage can be thought of as a ramp 2-party SSS which is private against 0 parties, reconstructible given both shares, and resists adaptive leakage. We note that the schemes of [17, 12] resists *general* local leakage (i.e., leakage which operates on each share separately, and has short output), and not just probing. An Equivocal SSS generalizes these notions – while ZK codes and leakage-resilient storage are only secure as long as the number of leakage bits does not pass an a-priori bound, equivocal schemes guarantee security even beyond the leakage threshold. Another related notion is that of a Reconstructable Probabilistic Encoding (RPE) [10, 5, 11, 4]. Informally, these are leakage resilient encodings that are also equivocal (with perfect leakage resilience and equivocation), with an additional error correction property. RPEs and equivocal SSSs are incomparable: while RPEs are a strengthening of *1-party* equivocal SSS (due to their error correction), (multi-party) equivocal SSSs guarantee leakage resilience even when full shares are leaked. In particular, they can potentially achieve a better leakage rate.

2.3.2.2 A specific 1-party equivocal SSS construction

Our constructions will employ the ZK code of [13, 14]. They construct a linear code in $\{0, 1\}^n$ with constant rate, and leakage resilience against a constant fraction of leaked bits. It is also equivocal, which follows from linearity using its dual distance (see [4, Lemma 2]). Therefore, it is a 1-party equivocal SSS with constant rate and security against probing of a constant fraction of codeword bits.

2.3.2.3 Why do equivocal SSSs yield non-adaptive ZK-PCPs?

We note that the locking schemes used in the constructions of [26, 22, 24] possess an equivocation property which is similar to our equivocal SSS, but applying them towards constructing ZK-PCPs and ZK-PCPPs causes *the honest* verifier to be adaptive. The reason is that

reconstructing (i.e., opening) the secret in a locking scheme requires making several rounds of queries to the locking scheme. This is because one should be able to recover the locked secret *without reading the entire lock*, which is needed since locking schemes are generally much longer than the locked secret. (The blowup is inherent to obtaining equivocation in locking schemes.) On the other hand, in an equivocal SSS the secret is reconstructed by reading all (or most) of the shares, which can be done non-adaptively. The fact that reconstruction requires reading many shares is not problematic in terms of efficiency, since the total length of all shares is usually relatively short compared to the secret.

2.4 Future Directions

Our work still leaves several open questions for future research. First, it is natural to ask whether the query gap (between the query complexity of the malicious and honest verifiers) in our constructions could be improved – to a better polynomial gap, or even exponential? This could potentially be achieved by instantiating the ZK-PCP of [21] with an MPC protocol with stringent communication requirements, in which the communication complexity (more specifically, the size of the views) is sublinear in the number of parties. Another natural research direction is to construct *multi-party* equivocal SSSs, and in particular ones that withstand *general* local leakage (and not just probing leakage). Finally, it would be interesting to find further applications of equivocal SSSs in other contexts, e.g., for adaptively-secure MPC.

3 Preliminaries

We denote the security parameter by κ . We say that a function $\mu : \mathbb{N} \rightarrow \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large κ 's it holds that $\mu(\kappa) < \frac{1}{p(\kappa)}$. We denote the set of all negligible functions by $\text{negl}(\kappa)$. We use the abbreviation PPT to denote probabilistic polynomial-time, and denote by $[n]$ the set of elements $\{1, \dots, n\}$ for some $n \in \mathbb{N}$. For a string s of length n , and a subset $I \subseteq [n]$, we denote by $s|_I$ the restriction of s to the coordinates in I . For an NP relation \mathcal{R} , we denote by \mathcal{R}_x the set of witnesses of x , and by $\mathcal{L}_{\mathcal{R}}$ its associated language. That is, $\mathcal{R}_x = \{w \mid (x, w) \in \mathcal{R}\}$ and $\mathcal{L}_{\mathcal{R}} = \{x \mid \exists w \text{ s.t. } (x, w) \in \mathcal{R}\}$.

Let $\delta \in (0, 1)$, let Σ be an alphabet, and let x, y be strings over Σ^n . We say that x, y are δ -close if $\frac{|\{i : x_i \neq y_i\}|}{n} < \delta$, otherwise we say that x, y are δ -far. We say that x is δ -close to a language \mathcal{L} if there exists $x' \in \mathcal{L}$ such that x, x' are δ -close. Otherwise, we say that x is δ -far from \mathcal{L} .

► **Definition 3.** Let X_κ and Y_κ be random variables accepting values taken from a finite domain Ω . The statistical distance between X_κ and Y_κ is

$$SD(X_\kappa, Y_\kappa) = \frac{1}{2} \sum_{w \in \Omega} |\Pr[X_\kappa = w] - \Pr[Y_\kappa = w]|.$$

We say that X_κ and Y_κ are ε -close if their statistical distance is at most $\varepsilon(\kappa)$. We say that X_κ and Y_κ are statistically close, denoted $X_\kappa \approx_s Y_\kappa$, if $\varepsilon(\kappa)$ is negligible in κ .

We use the asymptotic notation $O(\cdot)$ and $\Omega(\cdot)$. We will sometimes disregard polylogarithmic factors, using $\tilde{O}(n)$ and $\tilde{\Omega}(n)$ to denote $n \cdot \text{poly log } n$ and $n/\text{poly log } n$, respectively.

3.1 Zero-Knowledge Probabilistically Checkable Proofs (PCPs) and PCPs of Proximity

Informally, a Probabilistically Checkable Proof (PCP) system for a language \mathcal{L} consists of a probabilistic polynomial time prover that given $x \in \mathcal{L}$ and a corresponding witness generates a proof for x , and a probabilistic polynomial-time verifier having direct access to individual symbols of the proof. This proof string (called oracle) will be accessed only partially by the verifier. The oracle queries are determined by the verifier's input and coin tosses. Formally,

► **Definition 4** (PCP). *A Probabilistically Checkable Proof (PCP) for a language \mathcal{L} consists of a PPT prover \mathcal{P} and a PPT verifier \mathcal{V} such that the following holds for some negligible function $\text{negl} = \text{negl}(\kappa)$.*

1. SYNTAX. *The prover \mathcal{P} has input $1^\kappa, x, w$, and outputs a proof π_x for x over some alphabet Σ . The verifier \mathcal{V} has input $1^\kappa, x$, and oracle access to π . It makes q queries to π , and outputs either 0 or 1 (representing reject or accept, respectively).*
2. COMPLETENESS: *For every $x \in \mathcal{L}$, every $w \in \mathcal{R}_x$, and every proof $\pi_x \in \mathcal{P}(1^\kappa, x, w)$,*

$$\Pr[\mathcal{V}^{\pi_x}(1^\kappa, x) = 1] \geq 1 - \text{negl}(\kappa)$$

where the probability is over the randomness of \mathcal{V} , and κ is the security parameter.

3. SOUNDNESS: *For every $x \notin \mathcal{L}$ and every oracle π^* ,*

$$\Pr[\mathcal{V}^{\pi^*}(1^\kappa, x) = 1] \leq \text{negl}(\kappa)$$

where the probability is over the coin tosses of the verifier, and κ is a security parameter. negl is called the soundness error of the system.

Efficiency measures of a PCP system

We associate with a PCP system the following efficiency measures: the alphabet size $|\Sigma|$, the query complexity q , and the proof length $|\pi|$. We will call such a system a q -query PCP over alphabet Σ . We are generally interested in obtaining PCPs with $\Sigma = \{0, 1\}$, in which the proof length $|\pi|$ is polynomial in $|x|$, and q is significantly smaller than $|\pi|$. We note that a PCP prover is usually deterministic, but allowing for randomized provers will be useful when discussing *zero-knowledge* PCPs, as we do next.

Next, we define *zero-knowledge* PCPs. Intuitively, these are PCPs in which the witness remains entirely hidden throughout the verification process, even when the verifier is malicious and can potentially make many more queries to the proof compared to the honest verifier. We guarantee zero-knowledge against any, possibly malicious and unbounded, verifier - the only restriction is on the number of queries the verifier makes to the proof (this restriction is inherent to obtaining zero-knowledge). Thus, we first define the notion of a *query bounded* verifier.

► **Definition 5** (Query-bounded verifier). *We say that a (possibly malicious) verifier \mathcal{V}^* with oracle access to a proof π is q^* -query-bounded if it makes only q^* queries to π .*

► **Definition 6** (Non adaptive verifier). *We say that a (possibly malicious) verifier \mathcal{V}^* is non adaptive if its queries are determined solely by its input x and its randomness (in particular, \mathcal{V}^* can make a single round of queries to its proof oracle).*

We will use the following notation. For a PCP system $(\mathcal{P}, \mathcal{V})$ and a (possibly malicious) verifier \mathcal{V}^* , we use $\mathbf{View}_{\mathcal{V}^*, \mathcal{P}}(x, w)$ to denote the view of \mathcal{V}^* when it has input x and oracle access to a proof that was randomly generated by \mathcal{P} on input (x, w) . We are now ready to define zero-knowledge PCPs.

► **Definition 7 (ZK-PCP).** We say that a PCP system $(\mathcal{P}, \mathcal{V})$ for \mathcal{L} is a (q^*, ε) -Zero-Knowledge PCP (or ZK-PCP for short) if for any (possibly malicious and adaptive) q^* -query-bounded verifier \mathcal{V}^* there exists a PPT simulator Sim , such that for any $(x, w) \in \mathcal{R}$, $\text{Sim}(1^\kappa, x)$ is distributed ε -statistically close to $\mathbf{View}_{\mathcal{V}^*, \mathcal{P}}(x, w)$.

If $(\mathcal{P}, \mathcal{V})$ is a (q^*, ε) -ZK-PCP for $\varepsilon = \text{negl}(\kappa)$ then we simply say that $(\mathcal{P}, \mathcal{V})$ is a q^* -ZK-PCP. If $(\mathcal{P}, \mathcal{V})$ is a $(q^*, 0)$ -ZK-PCP then we say it is a perfect q^* -ZK-PCP. If the ZK property of the system only holds against PPT verifiers \mathcal{V}^* then we say the system is a computational (q^*, ε) -ZK-PCP (or CZK-PCP for short). If the zero-knowledge property is only guaranteed against non-adaptive verifiers then we say the system is a ZK-PCP for non-adaptive verifiers. If the honest ZK-PCP verifier is non-adaptive then we say that $(\mathcal{P}, \mathcal{V})$ is a non-adaptive ZK-PCP.

We stress that having a non-adaptive honest verifier is a desirable feature of the system, whereas having ZK against non-adaptive verifiers is a weaker form of ZK (since the system has no guarantee against malicious *adaptive* verifiers).

We remark that although this definition requires a weaker notion with a non-universal simulator, all our constructions obtain the stronger notion with a universal simulator. Furthermore, our constructions will rely on the MPC-in-the-head approach, where the quality of ZK will be inherited from the level of security of the underlying MPC protocol employed by the construction.

3.2 Leakage-Resilient Secret Sharing Schemes (LR-SSS)

A Secret-Sharing Scheme (SSS) allows a dealer to distribute a secret among n parties. Specifically, during a *sharing* phase each party receives a share from the dealer, and the secret can then be recovered from the shares during a *reconstruction* phase. The scheme is associated with an *access structure* which defines subsets of authorized and unauthorized parties, where every authorized set can recover the secret from its shares, whereas unauthorized sets learn nothing about the secret even given all their shares. A *Leakage-Resilient* SSS (LR-SSS) guarantees this latter property holds even if the unauthorized set obtains some leakage on the other shares.

We will mainly be interested in *t-threshold* secret sharing schemes, in which all (and only) subsets of size at least $t + 1$ are authorized to reconstruct the secret. We first define secret sharing schemes.

► **Definition 8 (Secret Sharing Scheme).** An n -party Secret Sharing Scheme (SSS) for secrets in \mathcal{S} consists of the following pair of algorithms.

Sharing algorithm Share: Takes as input a secret $s \in \mathcal{S}$ and outputs shares $(s_1, \dots, s_n) \in \mathcal{S}_1 \times \dots \times \mathcal{S}_n$, where s_i is called the share of party i , and \mathcal{S}_i is the domain of shares of party i .

Reconstruction algorithm Reconst: Takes as input a description \mathcal{G} of an authorized set, and shares $\{s_i : i \in \mathcal{G}\}$ and outputs $s' \in \mathcal{S}$.

The scheme is required to satisfy the following properties:

Correctness: For every $s \in \mathcal{S}$, and every authorized set \mathcal{G} ,

$$\Pr[\text{Reconst}(\mathcal{G}, \text{Share}(s)|_{\mathcal{G}}) = s] = 1$$

where $\text{Share}(s)|_{\mathcal{G}}$ denotes the shares of the parties in the authorized set \mathcal{G} .

Secrecy: For any pair of secrets $s, s' \in \mathcal{S}$, and any unauthorized set \mathcal{G} , $\text{Share}(s)|_{\mathcal{G}}$ and $\text{Share}(s')|_{\mathcal{G}}$ are statistically close.

In our constructions, we will use Shamir’s secret sharing scheme [28], which we review next.

► **Definition 9** (Shamir’s SSS). *Let \mathbb{F} be a field.*

Sharing algorithm: *For any input $s \in \mathbb{F}$, pick a random polynomial $p(\cdot)$ of degree t in the polynomial-field $\mathbb{F}[x]$ with the condition that $p(0) = s$, and output $p(1), \dots, p(n)$.*

Reconstruction algorithm: *For any input $(s'_i)_{i \in S}$ where none of the s'_i are \perp and $|S| > t$, compute a polynomial $g(x)$ such that $g(i) = s'_i$ for every $i \in S$. This is possible using Lagrange interpolation where g is given by*

$$g(x) = \sum_{i \in S} s'_i \prod_{j \in S/\{i\}} \frac{x - j}{i - j}.$$

Finally the reconstruction algorithm outputs $g(0)$.

We will actually require a stronger correctness property for SSSs which, informally, guarantees unique reconstruction for any set of (possibly ill-formed) shares. This can be thought of as a weak form of unique decoding, where we only require error detection (i.e., identifying whether or not an error occurred), and not error correction. Alternatively, this is a weaker form of verifiable secret sharing, which need only be secure against a corrupted dealer (i.e., all the share holders are assumed to be honest). Formally,

► **Definition 10** (Strongly-correct SSS). *We say that an n -party secret sharing scheme (Share, Reconst) for secrets in \mathcal{S} is strongly correct if Reconst is deterministic, and the only authorized set is $[n]$ (the set of all parties).*

We note that for any $t < n$, t -out-of- n Shamir’s scheme (with the access structure in which the only authorized set is $[n]$) is strongly correct. For this, we assume that the shares are numbered in some arbitrary way, and reconstruction always uses the “first” $t + 1$ shares, see Remark 11 below.

► **Remark 11** (Strong correctness implies unique reconstruction in threshold schemes). The strong correctness property of Definition 10 implies unique reconstruction in threshold schemes, when these are thought of as ramp secret sharing schemes which are private for sets of size at most t , and reconstructible for the set of all parties. Indeed, we assume without loss of generality that the shares are numbered (in some arbitrary way). Given all n shares, reconstruction is performed with the first $t + 1$ shares. These $t + 1$ shares determine *some* secret, and the fact that Reconst is deterministic guarantees its uniqueness. In particular, we note that in this case Shamir’s secret sharing scheme has unique reconstruction.

► **Remark 12.** We note that for our alphabet reduction for ZK-PCPs (Construction 17, Section 4) and ZK-PCPPs (which appears in the full version [20]) we can make due with *any* SSS with deterministic reconstruction, by having the verifier use some arbitrary *fixed* minimal authorized set for reconstruction. Notice that if such a set can be efficiently found then the verifier in Construction 17 will be PPT. We further note that for threshold SSSs (such as the one used in our final constructions in Theorems 1 and 2), such a minimal authorized set can be found efficiently.

Next, we define *leakage-resilient* SSS.

► **Definition 13** (Leakage-resilient SSS). *We say that a secret sharing scheme (Share, Reconst) for \mathcal{S} is ε -leakage resilient against a family F of leakage functions if for every $f \in F$, and every pair of secrets $s, s' \in \mathcal{S}$, $f(\text{Share}(s))$ and $f(\text{Share}(s'))$ are ε -statistically close.*

We will be particularly interested in the *local probing* leakage family, which consists of all functions that, given the n shares, output the shares of an unauthorized set in their entirety, as well as ℓ bits from each of the other shares. More specifically, we will only consider the $t + 1$ -threshold access structure mentioned above, in which all (and only) subsets of size $\geq t + 1$ are authorized. Formally:

► **Definition 14** ((t, ℓ) -local probing leakage). *Let $\mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_n$ be the domain of shares for some secret sharing scheme. For a subset $\mathcal{G} \subseteq [n]$ and a sequence $(\mathcal{I}_1, \dots, \mathcal{I}_n)$ of subsets of $[n]$, the function $f_{\mathcal{G}, \mathcal{I}_1, \dots, \mathcal{I}_n}$ on input (s_1, \dots, s_n) outputs s_i for every $i \in \mathcal{G}$, and outputs $s_i|_{\mathcal{I}_i}$ for every $i \notin \mathcal{G}$. The (t, ℓ) -local probing function family corresponding to $\mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_n$ is defined as follows:*

$$F_{t, \ell} = \{f_{\mathcal{G}, \mathcal{I}_1, \dots, \mathcal{I}_n} : \mathcal{G} \subseteq [n], |\mathcal{G}| \leq t, \forall i \notin \mathcal{G}, |\mathcal{I}_i| \leq \ell\}.$$

3.3 Equivocal Secret Sharing

In this section we define the notion of equivocal secret sharing, compare it to leakage-resilient secret sharing, and present a 1-party equivocal SSS based on coding. We start with the definition.

At a high level, an equivocal SSS is a leakage-resilient SSS with the additional guarantee that even after some bits are leaked from the shares, one can still “open” the secret (by providing the entire secret sharing) consistently with the previous leakage. This is formalized (in Definition 15) in the simulation-based paradigm, by comparing the real world experiment with an ideal experiment, which are described in Figure 1.

The real and ideal experiments have two phases: a leakage phase and a guessing phase. This is captured by having the adversary and simulator consist of two separate algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ and $(\text{Sim}_1, \text{Sim}_2)$, respectively. Leakage resilience is guaranteed against a family F of leakage functions and a leakage bound ℓ .

In the real world, the secret s is secret shared into n shares $\text{Sh}_1, \dots, \text{Sh}_n$. The adversary \mathcal{A}_1 is then given oracle access to a *SHARE* oracle, and a *LEAK* oracle. The *Share* oracle, given an index i , returns the i 'th share Sh_i . Each call to *SHARE* updates the set T of secret shares which the adversary has queried so far, by adding i to T (T is initialized to \emptyset). The *LEAK* oracle takes as input a function $g \in F$, which specifies, for each share $\text{Sh}_i, i \notin T$, a leakage function g_i . It applies these leakage functions to the shares $\text{Sh}_i, i \notin T$, and returns the outputs output_i . For each such share $\text{Sh}_i, i \notin T$, it also updates the counter ℓ_i of the number of leakage bits obtained on Sh_i , by increasing it by $|\text{output}_i|$. T and ℓ_1, \dots, ℓ_n are treated as global parameters that can be accessed and updated by all oracles.

At the end of the first phase of the experiment (the adversary \mathcal{A}_1 decides when to end the first phase and move to the second phase), \mathcal{A}_1 outputs a bit b_R , which specifies whether it wishes to learn the entire secret sharing of s . If $b_R = 1$, i.e., the adversary chose to proceed to the second phase, then it learns the entire secret sharing of s (this is done by calling the *REVEAL* oracle). Otherwise, the adversary obtains no further information beyond what it obtained during the leakage phase.

At the second phase of the game, the adversary \mathcal{A}_2 outputs a guess b'_R as to whether it is in the real or ideal experiments. This guess depends on the leakage in the first phase of the game, and can either depend on the secret shares of s (if $b_R = 1$) or not (if $b_R = 0$). The adversarial guess is only taken into account if the adversary did not violate the leakage restrictions, i.e., only if the following two conditions are satisfied. First, the set T of shares which the adversary received throughout the experiments (through *SHARE* queries) is an unauthorized set. Second, for every share i , the number of bits ℓ_i leaked from Sh_i (through

\mathcal{LEAK} queries) does not exceed the leakage bound ℓ . These checks are performed by calling the \mathcal{VALID} oracle, where if the tests fail then the adversary automatically loses the game (by setting its “guess” to 0).

The ideal experiment is similar to the real experiment, except that the \mathcal{SHARE} , \mathcal{LEAK} , and \mathcal{REVEAL} oracles are emulated by the simulator. In particular, the simulator needs to simulate shares and leakage on shares (through the \mathcal{SHARE} and \mathcal{LEAK} oracles). Additionally, if $b_I = 1$ (i.e., the adversary chose to learn the entire secret sharing in the ideal experiment) then the simulator is given the secret s , and needs to emulate the entire secret sharing of s consistently with the previous leakage (this is done in the \mathcal{REVEAL} oracle).

We note that by allowing the adversary to choose *not* to receive the entire secret sharing of s at the end of the first phase, we capture leakage-resilient secret sharing as a special case of equivocal secret sharing. Indeed, if the adversary chooses not to learn the secret sharing, then the simulator is only required to adaptively simulate the leakage (with no knowledge of the secret). We elaborate more on this in Remark 16 below.

► **Definition 15** (Equivocal SSS). *We say that an n -party secret sharing scheme (Share, Reconst) for secrets in \mathcal{S} is $\varepsilon(n)$ -equivocal for leakage class F , leakage bound ℓ and access structure Acc if for every adversary $(\mathcal{A}_1, \mathcal{A}_2)$ there exists an efficient simulator $(\text{Sim}_1, \text{Sim}_2)$ and a negligible function $\varepsilon(n)$ such that for every $s \in \mathcal{S}$,*

$$|\Pr[\mathcal{REAL}_{F,\ell,\text{Acc}}(s) = 1] - \Pr[\mathcal{IDEAL}_{F,\ell,\text{Acc}}(s) = 1]| \leq \varepsilon(n)$$

where $\mathcal{REAL}_{F,\ell,\text{Acc}}(s)$, $\mathcal{IDEAL}_{F,\ell,\text{Acc}}(s)$ are defined in Figure 1, and the probability is over the random coin tosses of $\mathcal{SETUP}^{\mathcal{R}}$, $(\mathcal{A}_1, \mathcal{A}_2)$ and $(\text{Sim}_1, \text{Sim}_2)$.

We say that the scheme is perfectly equivocal, if it is ε -equivocal with $\varepsilon = 0$.

► **Remark 16** (On the connection between equivocal and LR secret sharing). We note that equivocal secret sharing captures LR secret sharing as a special case, in the following sense. If a t -threshold secret sharing scheme is ε -equivocal with leakage bound ℓ , then it is also 2ε -leakage resilient against (t, ℓ) -local probing leakage. Indeed, if the \mathcal{REVEAL} oracle is not called in Figure 1 (which happens when $b = 0$) then the simulator never receives the secret s , in which case the simulated answers to the leakage queries are required to be distributed ε -statistically close to the real execution. As this holds for any secret, the real leakage on the shares of two different secrets must be 2ε -statistically close.

4 Alphabet Reduction for ZK-PCPs

In this section we describe a reduction for ZK-PCPs over any alphabet Σ into a ZK-PCP over $\{0, 1\}$. In particular, this reduction preserves the zero-knowledge property. We note that for standard (non-ZK) PCPs, one can easily transform a PCP over any alphabet Σ into a PCP over $\{0, 1\}$ by simply representing every symbol of Σ as a binary string. However, this reduction *does not* preserve zero-knowledge since a malicious verifier given access to the binary proof can read “parts” of symbols of the original proof, and thus potentially violate the zero-knowledge guarantee of the underlying ZK-PCP over Σ (which only guarantees zero-knowledge when most symbols are not accessed at all).

We begin by describing a general reduction, then instantiate it to obtain ZK-PCPs over $\{0, 1\}$ with a square root gap.

<p><u>$SETUP^{\mathcal{R}}(s)$</u>: pick a uniformly random string r for Share $(Sh_1, \dots, Sh_n) \leftarrow \text{Share}(s; r)$ output (Sh_1, \dots, Sh_n)</p>	<p><u>$SETUP^{\mathcal{I}}()$</u>: initialize St to the empty string $St \leftarrow \text{Sim}_1(St)$ output St</p>
<p><u>$SHARE^{\mathcal{R}}(s, r, i)$</u>: $T_1 \leftarrow T_1 \cup \{i\}$ output Sh_i</p>	<p><u>$SHARE^{\mathcal{I}}(i)$</u>: $Sh_i \leftarrow \text{Sim}_1(St, i)$ $T_1 \leftarrow T_1 \cup \{i\}$ output Sh_i</p>
<p><u>$LEAK^{\mathcal{R}}(s, r, g, T)$</u>: if $g \notin F$ then return $(\text{output}_i)_{i \notin T} \leftarrow g((Sh_i)_{i \in T})$ $T_1 \leftarrow T_1 \cup T$ for every $i \notin T$ $\ell_i \leftarrow \ell_i + \text{output}_i$ output $((\text{output}_i)_{i \notin T}, (Sh_i)_{i \in T})$</p>	<p><u>$LEAK^{\mathcal{I}}(g, T)$</u>: if $g \notin F$ then return run $\text{Sim}(St, g, T)$ to obtain $((\text{output}_i)_{i \notin T}, (Sh_i)_{i \in T}, St)$ $T_1 \leftarrow T_1 \cup T$ for every $i \notin T$ $\ell_i \leftarrow \ell_i + \text{output}_i$ output $((\text{output}_i)_{i \notin T}, (Sh_i)_{i \in T})$</p>
<p><u>$REVEAL^{\mathcal{R}}(s, r)$</u>: output (Sh_1, \dots, Sh_n)</p>	<p><u>$REVEAL^{\mathcal{I}}(s)$</u>: $rev \leftarrow \text{Sim}_2(St, s)$ output rev</p>
<p><u>$VALID(\ell, \text{Acc})$</u>: if $T_1 \notin \text{Acc} \wedge \ell_i \leq \ell$ for every $i \in [n]$ then output true else output false</p>	
<p><u>$REAL_{F, \ell, \text{Acc}}^{\mathcal{R}}(s)$</u>: $\ell_1, \dots, \ell_n \leftarrow 0$ $T_1 \leftarrow \emptyset$ $r \leftarrow SETUP^{\mathcal{R}}(s)$ $(St_A, b_R) \leftarrow \mathcal{A}_1^{SHARE^{\mathcal{R}}(s, r, \cdot), LEAK^{\mathcal{R}}(s, r, \cdot)}$ if $b_R = 1$ then $(Sh'_1, \dots, Sh'_n) \leftarrow REVEAL^{\mathcal{R}}(s, r)$ $St_A \leftarrow St_A \circ (Sh'_1, \dots, Sh'_n)$ $b'_R \leftarrow \mathcal{A}_2(St_A)$ if $VALID(\ell, \text{Acc})$ then output b'_R else output 0</p>	<p><u>$IDREAL_{F, \ell, \text{Acc}}^{\mathcal{I}}(s)$</u>: $\ell_1, \dots, \ell_n \leftarrow 0$ $T_1 \leftarrow \emptyset$ $St \leftarrow SETUP^{\mathcal{I}}()$ $(St_A, b_I) \leftarrow \mathcal{A}_1^{SHARE^{\mathcal{I}}(\cdot), LEAK^{\mathcal{I}}(\cdot, \cdot)}$ if $b_I = 1$ then $\text{output} \leftarrow REVEAL^{\mathcal{I}}(s)$ $St_A \leftarrow St_A \circ \text{output}$ $b'_I \leftarrow \mathcal{A}_2(St_A)$ if $VALID(\ell, \text{Acc})$ then output b'_I else output 0</p>

■ **Figure 1** The Security Experiments of Equivocal SSS.

A General Transformation

Our starting point is the trivial transformation described above, in which every proof symbol is replaced with a corresponding bit-string. As discussed above, this alone does not guarantee zero-knowledge since a malicious verifier may read parts of symbols of the original proof. The high-level idea of preventing such malicious strategies from leaking additional information is to “protect” each bit-string by secret-sharing it (equivalently, encoding it) using a *leakage-resilient* secret sharing scheme (equivalently, leakage-resilient encoding). Recall that, very roughly, a probing-resilient secret sharing scheme hides the secret from an adversary that sees several secret shares, and can probe few bits in each of the other shares. The zero-knowledge property of the new PCP system now follows from a combination of leakage-resilience and the zero-knowledge property of the original ZK-PCP. To see why, given a malicious query-bounded verifier \mathcal{V}^* , we partition the symbols of the original proof into two groups, based on the number of bits \mathcal{V}^* reads from the secret-sharing of the bit-string representing the symbol. Since \mathcal{V}^* is query-bounded, there are only few symbols from whose secret shares \mathcal{V}^* can read many bits (having many such symbols would have violated the query bound). The zero-knowledge property of the original ZK-PCP system guarantees that \mathcal{V}^* learns nothing about the witness even if it is given all these symbols in their entirety. For the rest of the symbols, since \mathcal{V}^* reads only few bits from their secret shares, the leakage-resilience of the secret sharing scheme guarantees that the secret shared symbol remains entirely hidden. The actual analysis is slightly more involved, see the proof of Theorem 18 below for details.

We now formally describe the transformation.

► **Construction 17** (Alphabet reduction for ZK-PCPs). *Let κ be a security parameter. The system $(\mathcal{P}', \mathcal{V}')$ is over alphabet $\{0, 1\}$.*

Building blocks:

- A PCP system $(\mathcal{P}, \mathcal{V})$ over alphabet Σ of size $|\Sigma| = 2^m$.
- A strongly-correct secret sharing scheme $(\text{Share}, \text{Reconst})$ for secrets in $\{0, 1\}^m$.

Prover algorithm. \mathcal{P}' has input $1^\kappa, x, w$. It runs \mathcal{P} with input $1^\kappa, x, w$ to obtain a proof π over Σ . For every proof symbol σ , it uses Share to secret-share the bit-representation of σ . (That is, the length- m bit representation of σ is treated as the secret.) Then, \mathcal{P}' outputs the concatenation of all secret shares. We denote the proof generated by \mathcal{P}' by π' .

Verifier algorithm. \mathcal{V}' is given input $1^\kappa, x$ and oracle access to π' . It runs \mathcal{V} with input $1^\kappa, x$, and emulates the oracle π for \mathcal{V} as follows. Whenever \mathcal{V} reads a symbol σ from π , \mathcal{V}' reads the entire secret sharing of σ from π' . Then, it uses Reconst to recover the symbol σ , and provides σ to \mathcal{V} as the answer of the oracle.

The following theorem summarizes the properties of Construction 17. Its proof, and several relevant corollaries, can be found in the full version [20].

► **Theorem 18** (Non-adaptive ZK-PCPs for non-adaptive verifiers). *Assume Construction 17 is instantiated with:*

- A non-adaptive q -query (q^*, ϵ) -ZK-PCP $(\mathcal{P}, \mathcal{V})$ over alphabet Σ for a language \mathcal{L} , with proofs of length N .⁶
- A strongly-correct k -party secret sharing scheme $(\text{Share}, \text{Reconst})$ for secrets in $\{0, 1\}^m$ with secret shares in $\{0, 1\}^M$ which is ϵ' -leakage-resilient against (t, ℓ) -local probing leakage.

⁶ In fact, as will be evident from the proof, it suffices that $(\mathcal{P}, \mathcal{V})$ is ZK against *non-adaptive* malicious verifiers.

Then Construction 17 is a non-adaptive q' -query (q^{**}, ϵ'') -ZK-PCP for non-adaptive verifiers, where:

$$q' = q \cdot M \cdot k \quad q^{**} = (q^* + 1)(\ell + 1)(t + 1) - 1 \quad \epsilon'' = \epsilon + \epsilon' \cdot (N - q^*).$$

Moreover, the transformation preserves the soundness and completeness of $(\mathcal{P}, \mathcal{V})$.

4.1 Upgrading to ZK Against Adaptive Verifiers

Our ZK-PCP (Theorem 18) obtained through the alphabet reduction of Construction 17 can be verified non-adaptively, but guarantee ZK only against *non-adaptive* verifiers. Ideally, we would like a ZK-PCP which can be verified non-adaptively, but guarantees ZK even against *adaptive* malicious verifiers.

In this section, we show that when Construction 17 is instantiated with an equivocal SSS (see Definition 15) instead of a leakage-resilient SSS then the resultant ZK-PCP retains its ZK even when the malicious verifier is adaptive. Concretely, we prove the following:

► **Theorem 19.** *Assume Construction 17 is instantiated with:*

- *A non-adaptive q -query (q^*, ϵ) -ZK-PCP $(\mathcal{P}, \mathcal{V})$ over alphabet Σ for a language \mathcal{L} , with proofs of length N .⁷*
- *A strongly-correct k -party ϵ' -equivocal (ramp) secret sharing scheme against (t, ℓ) -local probing leakage, for secrets in $\{0, 1\}^m$ with secret shares in $\{0, 1\}^M$.*

Then Construction 17 is a non-adaptive q' -query (q^{**}, ϵ'') -ZK-PCP, where

$$q' = q \cdot M \cdot k \quad q^{**} = (q^* + 1)(\ell + 1)(t + 1) - 1 \quad \epsilon'' = \epsilon + \epsilon' \cdot N.$$

Moreover, the transformation preserves the soundness and completeness of $(\mathcal{P}, \mathcal{V})$.

Theorem 1 now follows from Theorem 19 by an appropriate instantiation of the building blocks. See the full version [20] for details.



References

- 1 Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, João L. Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret sharing schemes for general access structures. In *CRYPTO, Proceedings, Part II*, pages 510–539, 2019.
- 2 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *FOCS, Proceedings*, pages 14–23, 1992.
- 3 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. In *FOCS, Proceedings*, pages 2–13, 1992.
- 4 Marshall Ball, Dana Dachman-Soled, Siyao Guo, Tal Malkin, and Li-Yang Tan. Non-malleable codes for small-depth circuits. In *FOCS*, pages 826–837, 2018.
- 5 Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In *EUROCRYPT*, pages 881–908, 2016.
- 6 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. In *STOC, Proceedings*, pages 1–10, 2004.



⁷ We stress that $(\mathcal{P}, \mathcal{V})$ is non-adaptive in the sense that the *honest* verifier is non-adaptive, but ZK holds against possibly *adaptive* verifiers.

- 7 Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.
- 8 Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In *CRYPTO, Proceedings*, pages 531–561, 2018.
- 9 Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. On adaptive vs. non-adaptive security of multiparty protocols. In *EUROCRYPT, Proceedings*, pages 262–279, 2001.
- 10 Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Black-box construction of a non-malleable encryption scheme from any semantically secure one. In *TCC*, pages 427–444, 2008.
- 11 Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. A black-box construction of non-malleable encryption from semantically secure encryption. *J. Cryptol.*, 31(1):172–201, 2018.
- 12 Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In *SCN, Proceedings*, pages 121–137, 2010.
- 13 Scott E. Decatur, Oded Goldreich, and Dana Ron. A probabilistic error-correcting scheme. *IACR Cryptol. ePrint Arch.*, 1997:5, 1997.
- 14 Scott E. Decatur, Oded Goldreich, and Dana Ron. Computational sample complexity. *SIAM J. Comput.*, 29(3):854–879, 1999.
- 15 Irit Dinur. The PCP theorem by gap amplification. In *STOC, Proceedings*, pages 241–250, 2006.
- 16 Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP-theorem. In *FOCS, Proceedings*, pages 155–164, 2004.
- 17 Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *FOCS, Proceedings*, pages 227–237, 2007.
- 18 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC, Proceedings*, pages 291–304, 1985.
- 19 Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In *STOC, Proceedings*, pages 685–698, 2018.
- 20 Carmit Hazay, Muthuramakrishnan Venkatasubramanian, and Mor Weiss. ZK-PCPs from leakage-resilient secret sharing. *IACR Cryptol. ePrint Arch.*, 2021:606, 2021. URL: <https://eprint.iacr.org/2021/606>.
- 21 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC, Proceedings*, pages 21–30, 2007.
- 22 Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. On efficient zero-knowledge PCPs. In *TCC, Proceedings*, pages 151–168, 2012.
- 23 Yuval Ishai, Amit Sahai, Michael Viderman, and Mor Weiss. Zero knowledge LTCs and their applications. In *RANDOM, Proceedings*, pages 607–622, 2013.
- 24 Yuval Ishai and Mor Weiss. Probabilistically checkable proofs of proximity with zero-knowledge. In *TCC, Proceedings*, pages 121–145, 2014.
- 25 Yuval Ishai, Mor Weiss, and Guang Yang. Making the best of a leaky situation: Zero-knowledge PCPs from leakage-resilient circuits. In *TCC, Proceedings*, pages 3–32, 2016.
- 26 Joe Kilian, Erez Petrank, and Gábor Tardos. Probabilistically checkable proofs with zero knowledge. In *STOC, Proceedings*, pages 496–505, 1997.
- 27 Thilo Mie. Short PCPPs verifiable in polylogarithmic time with $O(1)$ queries. *Ann. Math. Artif. Intell.*, 56(3-4):313–338, 2009.
- 28 Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
- 29 Akshayaram Srinivasan and Prashant Nalini Vasudevan. Leakage resilient secret sharing and applications. In *CRYPTO, Proceedings*, pages 480–509, 2019.
- 30 Mor Weiss. Secure computation and probabilistic checking. *PhD Thesis*, 2016.

Secure Merge with $O(n \log \log n)$ Secure Operations

Brett Hemenway Falk  

University of Pennsylvania, Philadelphia, PA, USA

Rafail Ostrovsky  

University of California, Los Angeles, CA, USA

Abstract

Data-oblivious algorithms are a key component of many secure computation protocols.

In this work, we show that advances in secure multiparty shuffling algorithms can be used to increase the efficiency of several key cryptographic tools.

The key observation is that many secure computation protocols rely heavily on secure shuffles. The best data-oblivious shuffling algorithms require $O(n \log n)$, operations, but in the two-party or multiparty setting, secure shuffling can be achieved with only $O(n)$ communication.

Leveraging the efficiency of secure multiparty shuffling, we give novel, information-theoretic algorithms that improve the efficiency of securely sorting sparse lists, secure stable compaction, and securely merging two sorted lists.

Securely sorting private lists is a key component of many larger secure computation protocols. The best data-oblivious sorting algorithms for sorting a list of n elements require $O(n \log n)$ comparisons. Using black-box access to a linear-communication secure shuffle, we give a secure algorithm for sorting a list of length n with $t \ll n$ nonzero elements with communication $O(t \log^2 n + n)$, which beats the best oblivious algorithms when the number of nonzero elements, t , satisfies $t < n / \log^2 n$.

Secure compaction is the problem of removing dummy elements from a list, and is essentially equivalent to sorting on 1-bit keys. The best oblivious compaction algorithms run in $O(n)$ -time, but they are unstable, i.e., the order of the remaining elements is not preserved. Using black-box access to a linear-communication secure shuffle, we give an information-theoretic stable compaction algorithm with only $O(n)$ communication.

Our main result is a novel secure merge protocol. The best previous algorithms for securely merging two sorted lists into a sorted whole required $O(n \log n)$ secure operations. Using black-box access to an $O(n)$ -communication secure shuffle, we give the first multi-party secure merge algorithm that requires only $O(n \log \log n)$ communication. Our algorithm takes as input n secret-shared values, and outputs a secret-sharing of the sorted list.

All our algorithms are generic, i.e., they can be implemented using generic secure computations techniques and make black-box access to a secure shuffle. Our techniques extend naturally to the multiparty situation (with a constant number of parties) as well as to handle malicious adversaries without changing the asymptotic efficiency.

These algorithms have applications to securely computing database joins and order statistics on private data as well as multiparty Oblivious RAM protocols.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases Secure computation, Data-oblivious algorithms, Sorting, Merging, Shuffling, Compaction

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.7

Related Version *Full Version:* <https://eprint.iacr.org/2020/807>

Funding Work done while consulting at Stealth Software Inc.



© Brett Hemenway Falk and Rafail Ostrovsky;
licensed under Creative Commons License CC-BY 4.0
2nd Conference on Information-Theoretic Cryptography (ITC 2021).
Editor: Stefano Tessaro; Article No. 7; pp. 7:1–7:29



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Secure sorting protocols allow two (or more) participants to privately sort a list of n encrypted or secret-shared [41] values without revealing any data about the underlying values to any of the participants. Secure sorting is an important building block for many more complex secure multiparty computations (MPCs), including Private Set Intersection (PSI) [32], secure database joins [33, 10, 43], secure de-duplication and securely computing order statistics as well as Oblivious RAMs [38, 24].

Secure sorting algorithms, and secure computations in general, must have control flows that are input-independent, and most secure sorting algorithms are built by instantiating a data-oblivious sorting algorithm using a generic secure computation framework (e.g. garbled circuits [47, 48], GMW [23], BGW [13]). This method is particularly appealing because it is composable – the sorted list can be computed as secret shares, and used in a further (secure) computations.

Most existing secure sorting algorithms make use of *sorting networks*. Sorting networks are inherently data oblivious because the sequence of comparisons in a sorting network is fixed and thus independent of the input values. The AKS sorting network [2] requires $O(n \log n)$ comparators to sort n elements. The AKS network matches the lower bound on the number of comparisons needed for any (not necessarily data independent) comparison-based sorting algorithm. Unfortunately, the constants hidden by the big- O notation are extremely large, and the AKS sorting network is never efficient enough for practical applications [3]. In practice, some variant of Batcher’s sort [9] is often used¹. The MPC compilers Obliv-c [49], ABY [19] and EMP-toolkit [45] provide Batcher’s bitonic sort. Batcher’s sorting network requires $O(n \log^2 n)$ comparisons, but the hidden constant is approximately $1/2$, and the network itself is simple enough to be easily implementable.

Although Batcher’s sorting network is fairly simple and widely used, the most efficient oblivious sorting algorithms make use of the *shuffle-then-sort* paradigm [31, 30] which builds on the observation that many traditional sorting algorithms (e.g. quicksort, mergesort, radixsort) can be made oblivious by obliviously shuffling the inputs before running the sorting algorithm. Since oblivious shuffling and (non-oblivious) sorting can be done in $O(n \log n)$ -time these oblivious sorting algorithms run in $O(n \log n)$ (but unlike AKS the hidden constants are small).

Although the shuffle-then-sort paradigm is extremely powerful, improvements in shuffling (below $O(n \log n)$) are unlikely to improve these protocols because of the $O(n \log n)$ lower-bound on comparison-based sorting.

In the context of secure multiparty computation, however, sorting can often be reduced to the simpler problem of *merging* two sorted lists into a single sorted whole. Each participant in the computation, sorts their list locally, before beginning the computation, and the secure computation itself need only implement a data-oblivious merge.

Merging is an easier problem than sorting, and even in the insecure setting it is known that any comparison-based sorting method requires $O(n \log n)$ comparisons, whereas (non-oblivious) linear-time merging algorithms are straightforward. Unfortunately, no data-oblivious merge algorithms are known with complexity better than simply performing a data-oblivious sort, and the best merging networks require $O(n \log n)$ comparisons.

Our main result is a secure multiparty *merge* algorithm, for merging two (or more) sorted lists (into a single, sorted whole) that requires only $O(n \log \log n)$ secure operations. This is the first secure multiparty merge algorithm requiring fewer than $O(n \log n)$ secure operations.

¹ For example, hierarchical ORAM [38, 24] uses Batcher’s sort.

The crucial building block of our algorithm is a linear-communication secure multiparty shuffle. Although no single-party, comparison-based shuffle exists using $O(n)$ comparisons, such shuffles exist in the two-party and multiparty setting (see Section 3), and this allows us to avoid the $O(n \log n)$ lower bound for comparison-based merging networks that exists in the single-party setting.

Our secure multiparty merge algorithm makes use of several novel data-oblivious algorithms whose efficiency can be improved through the use of a linear-communication secure multiparty shuffle.

These include

- **Securely sorting with large payloads:** In Section 4 we show how to securely sort t elements (with payloads of size w) using $O(t \log t + tw)$ communication. Previous sorting algorithms required $O(tw \log(tw))$ communication.
- **Securely sorting sparse lists:** In Section 5 we show how to securely sort a list of size n with only t nonzero elements in $O(t \log^2 n + n)$ communication. This beats naïvely sorting the entire list whenever $t < n / \log^2 n$.
- **Secure stable compaction:** In Section 6.1 we show how to securely *compact* a list (*i.e.*, extract nonzero elements) in linear time, while preserving the order of the extracted elements. Previous linear-time oblivious compaction algorithms (e.g. [6]) are *unstable* *i.e.*, they do not preserve the order of the extracted values.
- **Secure merge:** In Section 7 we give our main algorithm for securely merging two lists with $O(n \log \log n)$ communication complexity. Previous works all required $O(n \log n)$ complexity.

All the results above crucially rely on a linear-communication secure multiparty shuffle. Outside of the shuffle, all the algorithms are simple, deterministic and data-oblivious and thus can be implemented using any secure multiparty computation protocol.

In the two-party setting, we give a protocol for a linear-communication secure shuffle using any additively homomorphic public-key encryption algorithm with constant ciphertext expansion (Section 3.3). In the multiparty setting, a linear-communication secure shuffle can be built from any one-way function [34].

By making black-box use of a secure shuffle, our protocols can easily extend to different security models. If the shuffle is secure against malicious adversaries, then the entire protocol can achieve malicious security simply by instantiating the surrounding (data oblivious) algorithm with an MPC protocol that supports malicious security. One benefit of this is that our protocols can be made secure against malicious adversaries without changing the asymptotic communication complexity. Similarly, as two-party and multi-party linear-communication shuffles exist, all our algorithms can run in the two-party or multi-party settings simply by instantiating the surrounding protocol with a two-party or multi-party secure computation protocol (e.g. Garbled Circuits or GMW).

2 Preliminaries

2.1 Secure multiparty computation

Secure multiparty computation (MPC) protocols allow a group of participants to securely compute arbitrary functions of their joint inputs, without revealing their private inputs to each other or any external party. Secure computation has been widely studied in both theory and practice.

7:4 Secure Merge with $O(n \log \log n)$ Secure Operations

Different MPC protocols provide security in different settings, depending on parameters like the number of participants (e.g. two-party or multiparty), the amount of collusion (e.g. honest majority vs. dishonest majority), and whether the participants are semi-honest, covert [15, 7] or malicious.

In this work, we focus on creating data-oblivious algorithms that can be easily implemented using a variety of MPC protocols.

2.2 Oblivious algorithms

Secure and oblivious algorithms have been widely studied, it is instructive to differentiate between three types of data oblivious algorithms [37].

1. **Deterministically data independent:** In these algorithms, the control flow is deterministic and dependent only on public data. Most sorting networks are deterministically data independent.
2. **Data independent:** In these algorithms, the control flow is determined completely by the public data as well as additional (data-independent) randomness.
3. **Data oblivious:** In these algorithms, data can be “declassified” during the computation, and the control flow can depend on public data, as well as on previously declassified data. To ensure privacy, we require that the distribution of all declassified data (and the point at which it was declassified) is independent of the secret (input) data. The sorting algorithms of [31] are data oblivious, as are many ORAM constructions [38, 24].

All three of these types of algorithms can be easily implemented using generic MPC protocols.

2.3 Secure sorting

One common technique for secure sorting is to implement a *sorting network* under a generic MPC protocol. Since the sequence of comparisons in a sorting network is data-independent, if each comparison is done securely, the entire sorting procedure is secure.

In practice, many secure sorting algorithms are built on Batcher’s sorting network [9]. Batcher sorting networks require $O(n \log^2 n)$ comparisons to sort n entries, and is straightforward to implement, and is provided by MPC compilers like EMP-toolkit [45] and Obliv-c [49]. In the two-party setting, when each individual’s list is pre-sorted, then the final round of the Batcher sort can be omitted, and Batcher’s Bitonic sort provides an efficient *merge* algorithm with $O(n \log n)$ complexity. The AKS sorting network [2] and its improvements [39, 40] are asymptotically better than a Batcher’s, and requires only $O(n \log n)$ comparisons, but the hidden constants are enormous and the AKS network is not efficient for practical applications [3].

Zig-zag sort [27] is a deterministic data-independent sorting method, requiring $O(n \log n)$ comparisons, but the hidden constants are much smaller than those in AKS. Unfortunately, Zig-zag sort has a *depth* of $O(n \log n)$ (instead of $O(\log^2 n)$ for Batcher’s sorting network), and this high depth makes it less appealing for some applications.

A randomized version of the Shellsort algorithm can be made data-oblivious, and gives an $O(n \log n)$ randomized algorithm that can be made either Monte Carlo or Las Vegas [25].

In a 2-party computation, when both parties hold their data in the clear, each party can locally sort his or her data, and then apply Batcher’s bitonic sorting network to merge the two sorted lists. This results in an algorithm that runs in $O(n \log n)$ time (with small constants). This trick was used, for instance, in private set intersection [32]. Unfortunately, this trick does not apply when the two halves of the list cannot be pre-sorted, e.g. when the list is the (secret-shared) output of a prior computation.

Although sorting networks of size $O(n \log n)$ with a small hidden constant are unknown, secure sorting can be achieved in $O(n \log n)$ time (with a small constant) by combining secure shuffles and a generic sorting algorithm [31]. The core idea is that if the underlying data are randomly shuffled, then the sequence of comparisons in any sorting algorithm (e.g. mergesort, quicksort) are independent of the underlying data.

More concretely, to securely sort a list, data owners can first securely shuffle their lists, then apply an $O(n \log n)$ sorting algorithm (e.g. merge sort) to their shuffled list. Each comparison in the sorting algorithm will be computed under MPC, but the result of the comparison is then revealed, and the players can order the (secret) data based on the output of this public comparison. The Waksman permutation network [44] requires $O(n \log n)$ swaps, to implement a shuffle, so the entire shuffle-then-sort procedure only requires $O(n \log n)$ operations (with small constants). This idea has been implemented using the Sharemind platform [14] and to build efficient mix-nets [4]. These protocols are not data-independent (since the exact sequence of comparisons depends on the underlying data), but instead they are *data-oblivious* which is sufficient for security.

Building on this shuffle-then-sort paradigm, oblivious radix sort [30] requires $O(n \log n)$ communication, but only a constant number of rounds, and is efficient in both theory and practice. This was later improved (in the multiparty setting) [17] by incorporating the linear-time multiparty shuffle algorithm of [34] we review this shuffle in Section 3.2.

See [21] for a survey of data-oblivious sorting methods.

Sorting provides a method for computing all the *order statistics* of the joint list. If, however, only a single order statistic (e.g. the k th largest element) is needed, there are more efficient secure protocols that only require $O(\log n)$ secure comparisons to compute the k th order statistic [1]. The protocols of [1] reveal the order statistics *in the clear*, and it is not clear how to modify them to reveal only *secret shares* of the relevant order statistic, Thus they are not applicable in scenarios where computing order statistics is merely the first step in a larger secure computation. Another way of viewing this distinction is that the algorithms presented in [1] are not data-oblivious – the sequence of comparisons depends on the *output* – but since the output is revealed by the protocol the entire sequence of comparisons could be simulated by a simulator who only sees the protocol’s output.

Merging two sorted lists is potentially easier than sorting, and when data-obliviousness is not needed merging can be done in linear-time using a single-scan over each list.

In the deterministic data independent setting, Batcher’s merging networks are known to be optimal when one list is small [5]. In the (probabilistic) data independent setting, [35] gives a randomized variant of Batcher’s odd-even mergesort using $O(n \log n)$ comparisons (with hidden constant less than one).

In the *three-party* setting, there is a linear-communication secure merge protocol [16], but no similar result is known in the two-party setting.

The main contribution of this work is to provide a new, multiparty secure merge algorithm that only requires $O(n \log \log n)$ secure operations (with small constants). Our construction avoids the lower bound of [35] by using an efficient secure shuffle (see Section 3) that is not comparison-based. Our construction immediately yields efficient, secure algorithms for sorting and obviously computing order statistics in both the two-party and multiparty settings, and these constructions can easily be made secure against malicious adversaries using standard techniques.

3 Shuffling secret shares

3.1 $O(n \log(n))$ -oblivious shuffles

Secure shuffles can be done in $O(n \log n)$ -time using a Waksman permutation network [44, 12]. Waksman permutation networks are built using “controlled-swap-gates” which take two inputs and a “control bit” that determines whether to swap the two inputs. Although the Waksman network guarantees that every permutation can be realized through a choice of control bits, a *uniformly* random choice of control bits does not result in a uniformly random permutation [12]. On the other hand, given a permutation, the specific control bits required to realize this permutation can be calculated efficiently.

Waksman networks can be used to facilitate a secure m -party shuffle by simply having each player separately input their control bits and performing m (sequential) shuffles. The resulting shuffle will be random as long as one player was honest, and the entire cost of the protocol is $O(mn \log n)$. Alternatively, the control bits can be set *within* the MPC [42], but this requires $O(n^2)$ secure multiplications, and is thus *less* efficient than simply repeated executing a Waksman permutation with different control bits provided by each party when the number of players, m , is constant.

Asymptotically efficient oblivious shuffles can also be performed using more complex ORAM-based techniques [6, 20], but these are not nearly as efficient as Waksman shuffles in practice.

3.2 Multiparty secure shuffles

In this section, we review the linear-communication secure multiparty shuffle of [34]. A similar, multiparty secure shuffle was used for efficient multiparty ORAM [16]. The protocol is an *information-theoretic* protocol for executing a *pseudo-random* shuffle. An overview of the multiparty shuffle is given in Figure 1.

The group of participants, C , generates a permutation, $\sigma^{(C)}$. Since $\sigma^{(C)}$ is hidden from players outside C , and every coalition of size t is outside some subset, the final permutation (which is the composition of all the permutations $\sigma^{(C)}$) is hidden from all players [34, Section 4.3].

As noted in [34], simply sharing the (public) permutation among members of C requires $O(n \log n)$ communication. If, however, the players share a *pseudorandom* permutation, this communication cost is essentially eliminated and the total communication complexity becomes $O(n)$ as claimed. This can also be made secure against malicious adversaries, while retaining its $O(n)$ communication complexity [34, Section 4.4].

► **Lemma 1** (Multiparty secure shuffle [34]). *If there exists a Pseudorandom permutation (PRP) with λ -bit keys, then for any $m \geq 3$ and any $t < m - 1$, then there is an m -party secure shuffling protocol that remains secure against t corrupted players, that can shuffle vectors of length m , where each player’s communication is*

$$\binom{m}{m-t} n(m-t) + \binom{m-1}{m-t-1} (nm + \lambda).$$

In particular, if the number of players, m , is constant, the total communication per player is linear in the database size, n .

Although the communication complexity of this re-sharing based protocol is linear in the database size, n , repeating the resharing procedure for every subset of size t makes the overall communication *exponential* in the threshold size, t . Thus if $t = \Theta(m)$, the communication

MultiPartyShuffle

Input: m parties hold secret shares of a vector \vec{v} of length n . Let $t < m - 1$ be the desired corruption threshold.

Output: Secret shares of the shuffled vector \vec{v} .

1. For every subset, C , of $m - t$ players, the protocol does the following:
 - a. The players re-share the secret shares of \vec{v} to members of C .
 - b. The members of C reconstruct shares of \vec{v} from the shares of shares of \vec{v} using the linearity of the secret sharing scheme.
 - c. The members of C choose a (pseudorandom) permutation $\sigma^{(C)} : [n] \rightarrow [n]$, known to all members of C .
 - d. The members of C shuffle the shares of \vec{v} according to this public permutation.
 - e. The members of C re-share the shares of $\sigma^{(C)}(\vec{v})$ to the entire group of players.
 - f. The players reconstruct shares of $\sigma^{(C)}(\vec{v})$ from the shares of shares of $\sigma^{(C)}(\vec{v})$ using the linearity of the secret sharing scheme.

■ **Figure 1** The secure m -party shuffle of [34]. This shuffle provides security against semi-honest adversaries when the corruption threshold is $t < m - 1$.

will be exponential in the number of players, m . Thus it only retains asymptotic efficiency for small (constant) m . From an asymptotic standpoint, this is not a restriction, because if m is super-constant, simply secret-sharing the input data among all the participants requires $\omega(n)$ communication per party, so we can't hope to get $O(n)$ communication whenever $m = \omega(1)$.

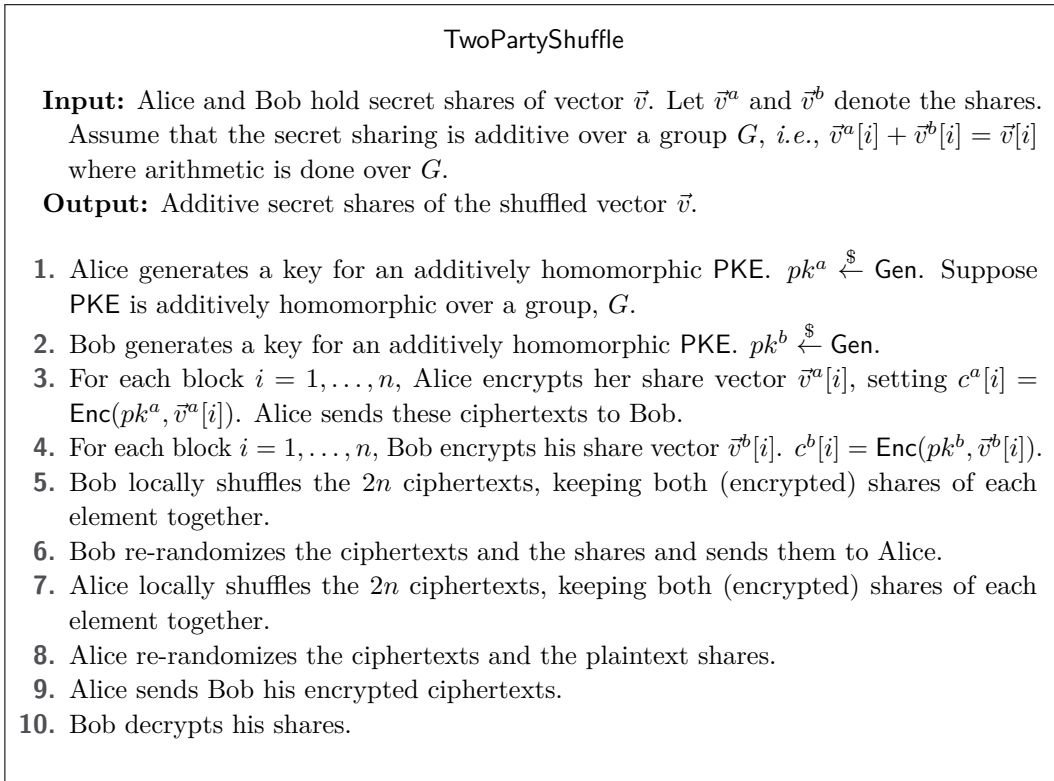
3.3 2-party secure shuffles

In this section, we give a simple two-party shuffle that relies on an additively homomorphic cryptosystem. Such a cryptosystem is not *information-theoretically* secure, and currently there is no known, linear-communication information-theoretic secure shuffle. We note, however, that all our protocols use only *black-box* access to the underlying shuffle, and thus if the underlying shuffle could be made information-theoretically secure, then the entire protocol would inherit this security.

If the cryptosystem has constant ciphertext expansion, then the resulting shuffle requires only $O(n)$ communication. This is essentially the two-party variant of the linear-communication multiparty shuffle [34] described in Section 3.2. A similar 2-party shuffle was described in [22].

Using a lattice-based scheme with ciphertext packing, this can be made extremely efficient in practice. To demonstrate the practical efficiency of this scheme, we implemented it using the PALISADE [18] FHE library, to show that it is dramatically more communication efficient than a simple Waksman shuffle (implemented in EMP [45]). We chose to implement our scheme using lattice-based FHE because ciphertext packing makes these schemes extremely efficient (in terms of ciphertext expansion, and the cost of additively homomorphic operations) when used to encrypt *blocks* of data. See Appendix C for details.

► **Lemma 2** (2-Party secure shuffle). *If PKE is an additively homomorphic, semantically secure cryptosystem with constant ciphertext expansion, then the shuffle TwoPartyShuffle outlined in Figure 2 is secure against passive adversaries, and requires $O(n)$ communication.*



■ **Figure 2** A 2-party shuffle based on additively homomorphic encryption, secure against semi-honest adversaries.

The proof is straightforward, but for completeness we provide it in Appendix B.

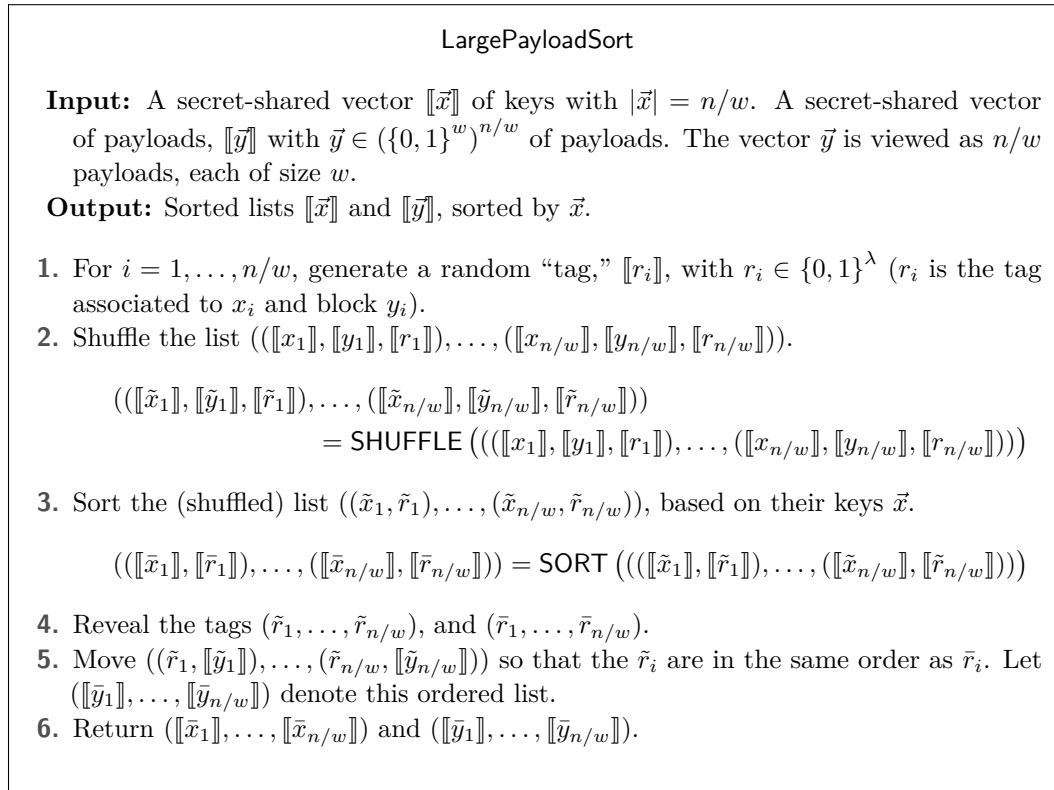
Two-party shuffles of this type can be made secure against malicious adversaries, while retaining their asymptotic efficiency [29, 11].

The linear-communication multiparty secure shuffle in Section 3.2 has been used to create extremely efficient sorting algorithms in the multiparty setting [17]. Using our linear-communication secure 2-party shuffle, TwoPartyShuffle described in Figure 2, the shuffle-then-sort construction of [17] can be extended to the 2-party setting.

4 Securely sorting with large payloads

In this section, we give a simple, linear-communication algorithm for sorting keys with large payloads that makes black-box use of a linear-communication secure shuffle. In large-payload sorting, we have a collection of *blocks* data (payloads), and each block is tagged with a *key*. Each payload must be put into the position determined by its key, but the position of elements *within* each payload remains unchanged. Like all our constructions, this algorithm crucially relies on a black-box access to a linear-communication shuffle (Section 3).

Oblivious sorting algorithms [31] and sorting networks [2] can sort n elements using $O(n \log n)$ comparisons. Now, imagine that instead of n elements, we have n/w blocks, each of size w , and the n/w blocks need to be (obviously) sorted based on n/w (short) keys. In the insecure setting, this requires $O(n/w \log(n/w))$ comparisons. In the secure setting, using an existing oblivious sorting algorithm, requires $O(n/w \log(n/w))$ secure comparisons.



■ **Figure 3** Securely sorting keys with large payloads.

Unfortunately, obviously swapping two blocks (based on the result of the secure comparison) requires $O(w)$ controlled swap gates. Thus the entire process requires $O(n \log(n/w))$ secure operations.

Note that since a secure comparison of λ -bit keys requires λ secure AND gates to implement as a circuit, whereas a controlled-swap gate only requires one, sorting n elements (based on λ -bit keys) requires $O(n\lambda \log n)$ secure AND gates, whereas sorting n/w blocks, requires $O(n(\frac{\lambda}{w} + 1) \log(n/w))$ secure AND gates, so sorting on blocks is actually somewhat faster (although still not linear).

Given a linear-communication secure shuffle, the problem of sorting with large payloads can be reduced to the problem of sorting with small payloads as follows. Each key and its corresponding payload (“block”) are tagged with a random tag. Then the keys are sorted together with their (short) tags, and the (sorted) tags are revealed. The blocks are shuffled together with their tags, and the tags are revealed. Finally, the blocks are moved into the ordering given by the tags. The key observation is that shuffle ensures that this final data-movement is independent of the underlying data. The full algorithm is given by LargePayloadSort in Figure 3.

► **Lemma 3** (Securely sorting with large payloads). *The sorting algorithm, LargePayloadSort, outlined in Figure 3 can be instantiated using $O(n/w \log(n/w) + n)$ communication, and is (t, m) -secure against semi-honest adversaries if $m = 2$, or $t < m - 1$.*

Proof. First, note that the probability that r_i collides with another r_j is at most $\frac{n}{w2^\lambda}$, so a union bound shows that with probability at least $1 - \frac{n^2}{w^2 2^\lambda}$, all the r_i will be distinct. Note that if the r_i are *not* distinct, correctness may fail, but privacy will still be preserved.

7:10 Secure Merge with $O(n \log \log n)$ Secure Operations

If we choose $w = \omega(\log n)$, then $\frac{n^2}{w^2 2^\lambda}$ will be negligible, and for the rest of the argument, we assume we are in the case where all the r_i are distinct.

First, note that the vectors \vec{x} and \vec{y} can be tagged using a single linear pass (requiring $O(n\lambda/w)$ secure operations). Sorting the vector \vec{x} requires $O(n/w \log(n/w))$ operations, using a standard oblivious sorting algorithm (e.g. [31]). The shuffling algorithm requires $O(n)$ secure operations, and the final step of moving the data can be done in linear time, since it does not need to be done obliviously.

To see that this protocol is secure, note that each player's view consists of the $\{r_i\}$ associated with the sorted \vec{x} , and the $\{r_i\}$ associated with the shuffled \vec{y} . These distributions can be simulated as follows: the simulator chooses n/w r_i uniformly from $\{0, 1\}^\lambda$. The simulator reveals $\{r_i\}$ as associated with \vec{x} , then the simulator shuffles the $\{r_i\}$ and reveals the shuffled set as associated with \vec{y} . Since the protocol chooses the $\{r_i\}$ uniformly, their distribution is unchanged after sorting them based on \vec{x} . Since the shuffle is secure, the $\{r_i\}$ associated with the shuffled \vec{y} are simply a random permutation of the $\{r_i\}$ associated with \vec{x} . ◀

5 Sorting sparse lists

The algorithm `LargePayloadSort` provides a method for sorting *sparse* lists with linear communication. The idea is to divide the list into blocks. Then, with a single pass, we can count the number of nonzero elements in each block. Using `LargePayloadSort`, we can sort the blocks based on the number of nonzero elements. If the list is sparse enough (relative to the blocksize), we can be sure that only a small fraction of blocks have nonzero entries. These blocks will appear first (after sorting blocks based on the number of nonzero entries), thus it only remains to sort these “top” blocks (using an $O(n \log n)$ -sorting algorithm). The complete algorithm is outlined in Figure 4.

► **Lemma 4.** *If \vec{V} is a list of length n with t nonzero entries, then \vec{V} can be securely sorted using $O(t \log^2(n) + n)$ secure operations, which is linear in n when $t < n/\log^2(n)$.*

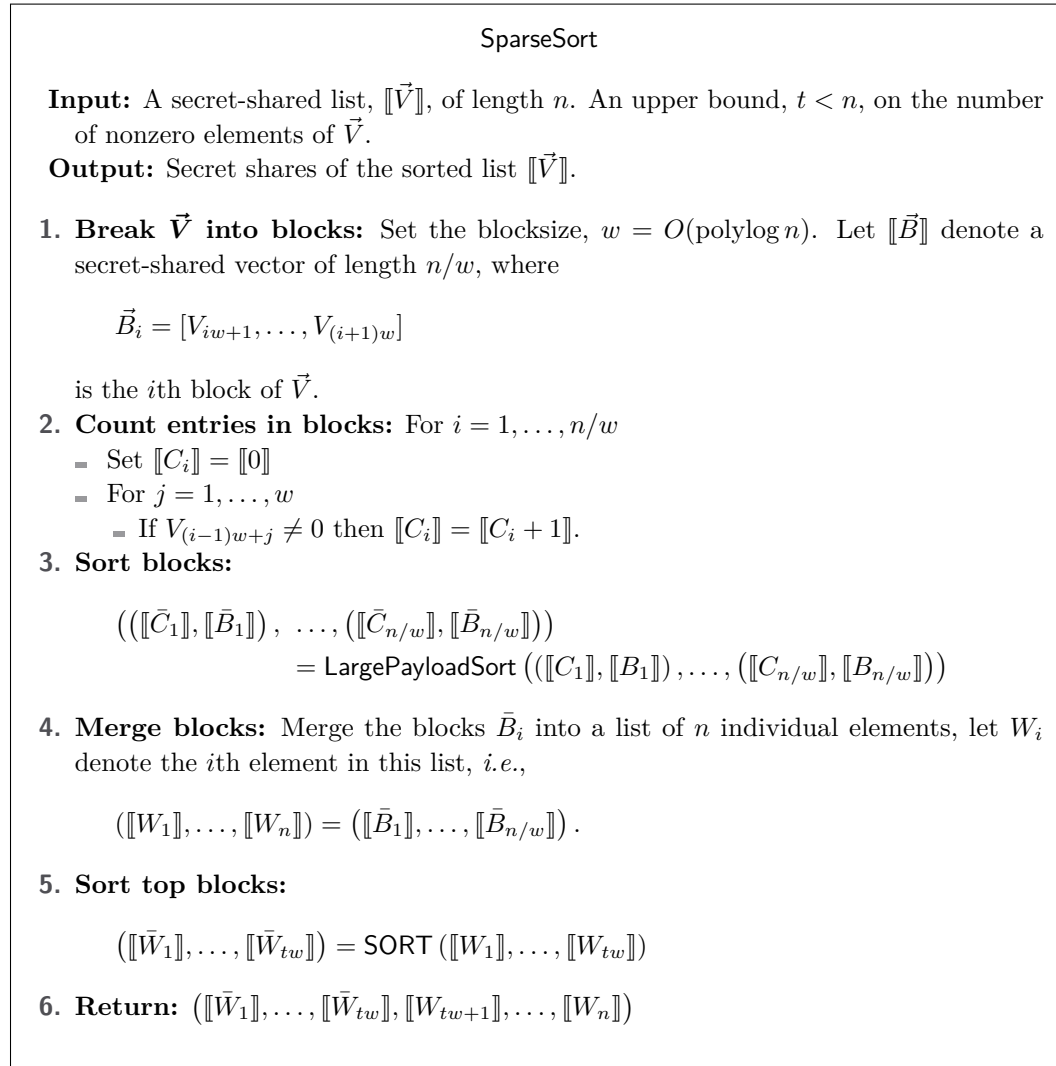
Proof. The algorithm, `SparseSort` is provided in Figure 4.

First, we note that this algorithm is correct. Since \vec{V} has at most t nonzero elements, at most t blocks of \vec{B} contain nonzero elements. Thus after sorting \vec{B} (Step 3) all the nonzero elements are in the top t blocks, and after sorting the top t blocks (Step 5) the entire list is sorted.

Next, we analyze the running efficiency. Step 2 requires a linear pass over the list, and requires $O(n)$ communication. Step 3 calls `LargePayloadSort` which requires $O(n/w \log(n/w) + n)$ communication to sort blocks of size w . Step 5 requires sorting a list of length tw which can be done in time $O(tw \log(tw))$. If $w = \log(n)$, then Step 3 requires $O(n)$ secure operations, and Step 5 requires $O(t \log^2(n)) = O(n)$ secure operations. ◀

6 Oblivious Compaction

In this section, we review the notion of oblivious compaction. The goal of compaction is to remove a set of marked element from a list. Given a secret shared list, where each element is tagged with secret share of 0 or 1, an oblivious compaction procedure removes all elements tagged with a 0, and returns the new (secret shared) list containing only those elements tagged with a 1.



■ **Figure 4** Securely sorting a sparse list of length n with t nonzero entries using $O(t \log^2(n) + n)$ communication.

The first oblivious compaction algorithm was probabilistic and ran in $O(n \log \log \lambda)$ time with failure probability that was negligible as a function of λ [35]. Follow-up works [37, 36] also gave probabilistic algorithms for solving the problem of oblivious compaction with running time $O(n \log \log n)$. The first deterministic, $O(n)$ -time compaction algorithm appeared in [6].

The compaction algorithms of [35, 6, 20] use expander graphs, and while they are asymptotically efficient, the hidden constants in the big- O are large,² and the algorithms are likely to be inefficient for lists of reasonable size. The compaction algorithms of [37] and [36] are data independent and run in time $O(n \log \log n)$, (with reasonable constants) and thus are suitable for our purposes. In Appendix D, we review the algorithm of [37] and give a tight analysis of its error probability and running time.

When the list is sparse (*i.e.*, it has $O(n/\text{polylog}(n))$ nonzero elements), the problem of compaction is much simpler, and in Appendix E we give a simple algorithm for compacting sparse lists.

A sorting algorithm is called *stable* if the order of elements with equal keys is retained. In general, 0-1 principle [8] for sorting networks tells us that any deterministic, data-independent stable compaction algorithm is in fact a sorting algorithm. Thus the lower bounds on the size of comparison-based sorting algorithms tell us that any deterministic, comparison-based compaction algorithm with $o(n \log n)$ complexity must be *unstable*.

In Section 6.1, we show that, given black-box access to a linear-communication shuffle, stable compaction with complexity $O(n)$ is achievable. This does not violate the sorting lower bounds since the underlying shuffle is a multiparty protocol.

6.1 Stable compaction

Using the a linear-communication secure shuffle (see Section 3), we give a simple, linear-time *stable* compaction algorithm. Our stable compaction algorithm takes three arguments, a public bound, t , a secret-shared vector of “tags,” \vec{s} , and a secret shared vector of “payloads,” \vec{x} .

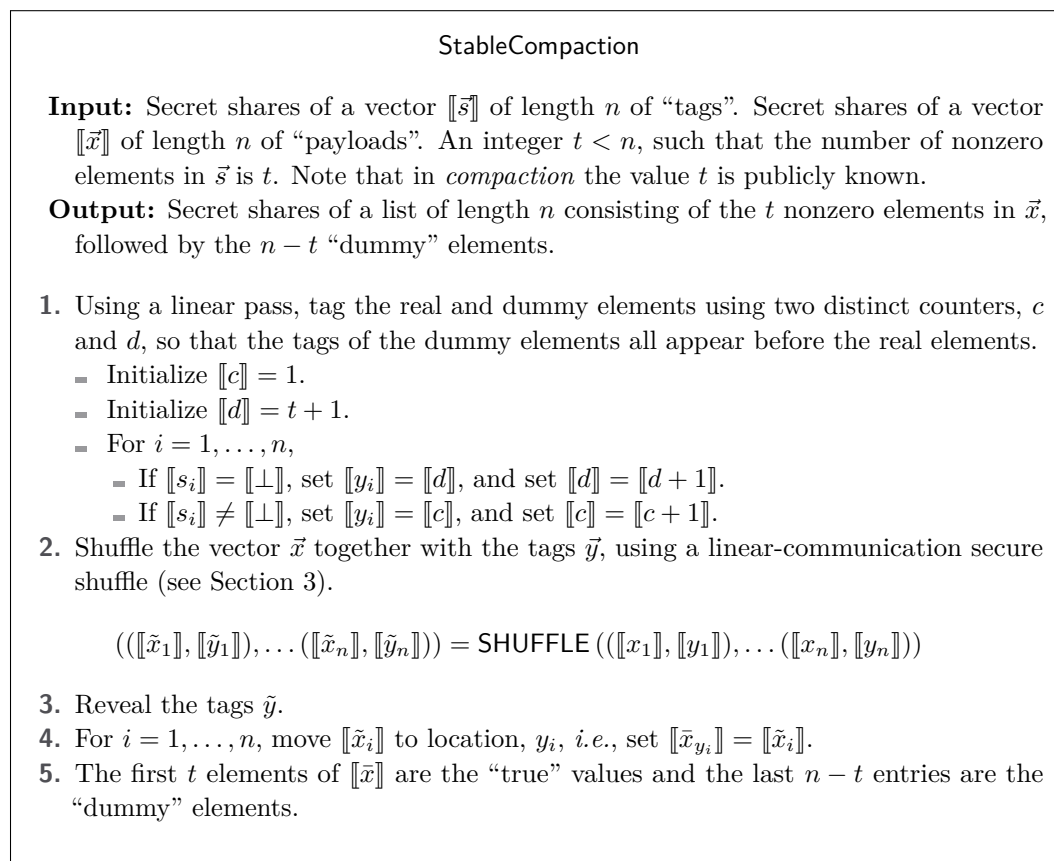
$$\llbracket \vec{y} \rrbracket = \text{StableCompaction}(t, \llbracket \vec{s} \rrbracket, \llbracket \vec{x} \rrbracket).$$

► **Lemma 5** (Stable compaction). *Algorithm, `StableCompaction`, outlined in Figure 5 is stable, secure against passive adversaries, and requires $O(n)$ communication.*

Proof. It is straightforward to see that if the shuffle can be done with linear time and communication, the entire protocol can be done with linear time and communication.

To see that the protocol is secure, we construct a simulator that simulates the players’ views. First, note that, essentially, the players’ views consist of the revealed vector \vec{y} . Consider the following simulator, S . On inputs n, t , the simulator, S , generates a vector \vec{z} such that $z_i = i$ for $i = 1, \dots, n - t$, and $z_i = 0$ for $i > n - t$. Then S shuffles \vec{z} , and outputs the shuffled vector \vec{y} . It is straightforward to check that this has the same distribution as in the real protocol. ◀

² The smallest constant being $\sim 16,000$ in [20].



■ **Figure 5** Stable compaction.

7 Securely merging private lists

7.1 Construction overview

In this section, we describe our novel data-oblivious merge algorithm. Our algorithm requires a linear-communication algorithm for shuffling secret shares (see Section 3), an oblivious sorting algorithm, SORT (e.g. [2, 27, 31, 30]) that requires $\text{sort}(\cdot)$ secure operations, and an oblivious, stable compaction algorithm (see Section 6) The rest of the operations are standard operations (e.g. equality test, comparison) that can be easily implemented in any secure computation framework.

At a high-level, the merging algorithm proceeds as follows:

- The input is two (locally) sorted lists, which are then concatenated.
- The players divide the list into blocks of size $w = O(\text{polylog}(n))$. We call the first element of each block a “pivot” element. Then the players sort these blocks based on their pivots using `LargePayloadSort`. (For efficiency, this step requires the linear-communication shuffle).
- At this point, because the initial lists were sorted, most elements are “close” to their true location in the list. In fact, we can concretely bound the number of “strays” (*i.e.*, the number of elements that may be far from their true location).
- After extracting the strays, every w th element is declared to be a pivot for some parameter $w = O(\text{polylog } n)$.

7:14 Secure Merge with $O(n \log \log n)$ Secure Operations

- The players obliviously extract these strays, and match them to their “true” pivots. Since the number of strays and pivots is not too large, this can be done using a linear number of secure operations using the sparse sorting algorithm `SparseSort` (described in Figure 4).
- The players reinsert the strays next to their true pivot. To avoid revealing the number of strays associated with each pivot, the number of strays associated to each pivot must be padded with “dummy” elements.
- The players use a linear-communication *stable* compaction algorithm to remove the dummy elements that were inserted with the strays.
- The players sort using $\text{polylog}(N)$ -sized sliding windows again. At this point, all the elements will be in sorted order, but there will be many dummy elements.

The details of this construction are described in Section 7.2.

7.2 Oblivious merge with $O(n \log \log n)$ secure operations

In this section, we provide the details of our oblivious-merge algorithm.

1. **Public parameters:** A length, $n \in \mathbb{Z}$. A blocksize $w \in \mathbb{Z}$, such that $w \mid n$ (we will set $w = O(\text{polylog}(n))$). A parameter δ , with $0 < \delta < 1$.
2. **Inputs:** Sorted, secret-shared lists $(\llbracket a_1 \rrbracket, \dots, \llbracket a_\ell \rrbracket)$, and $(\llbracket b_1 \rrbracket, \dots, \llbracket b_{n-\ell} \rrbracket)$. We let \vec{v} denote this list,

$$\llbracket v_1 \rrbracket, \dots, \llbracket v_n \rrbracket \stackrel{\text{def}}{=} \llbracket a_1 \rrbracket, \dots, \llbracket a_\ell \rrbracket, \llbracket b_1 \rrbracket, \dots, \llbracket b_{n-\ell} \rrbracket.$$

3. **Creating pivot tags:** For every pivot, assign a random identifier r from the set $1, \dots, n/w$ as follows

$$(\llbracket r_1 \rrbracket, \dots, \llbracket r_{n/w} \rrbracket) = \text{SHUFFLE}(\llbracket 1 \rrbracket, \dots, \llbracket n/w \rrbracket).$$

At this point, the “identifier” or “tag” r_i remains hidden (secret-shared), and will be assigned to the i th pivot in the next step.

Secure Operations: $O(n/w)$

4. **Sorting based on pivots:** This step uses `LargePayloadSort` to sort blocks of size w based on their leading entry as follows. Define B_i to be the i th block of size w ,

$$B_i \stackrel{\text{def}}{=} (v_{(i-1)w+1} \dots, v_{i \cdot w}),$$

and define $p_i \stackrel{\text{def}}{=} v_{(i-1)w+1}$ for $i = 1, \dots, n/w$ to be the leading element of each block.

$$\begin{aligned} & (\llbracket \vec{p} \rrbracket, ((\llbracket \tilde{r}_1 \rrbracket, \llbracket \tilde{B}_1 \rrbracket), \dots, (\llbracket \tilde{r}_{n/w} \rrbracket, \llbracket \tilde{B}_{n/w} \rrbracket))) \\ &= \text{LargePayloadSort}(\llbracket \vec{p} \rrbracket, ((\llbracket r_1 \rrbracket, \llbracket B_1 \rrbracket), \dots, (\llbracket r_{n/w} \rrbracket, \llbracket B_{n/w} \rrbracket))) \end{aligned}$$

At this point, the blocks of the vector \vec{v} are sorted according to the leading element in each block,

$$(v_1, \dots, v_n) \stackrel{\text{def}}{=} \tilde{B}_1 \cdots \tilde{B}_{n/w}.$$

Secure Operations: $O(n/w \log(n/w) + n)$

5. **Revealing pivot Tags:** For $i = 1, \dots, n/w$, reveal \tilde{r}_i . Note that since each pivot, p_i , was assigned a random tag r_i (which remains hidden), revealing $\{\tilde{r}_i\}$, which are sorted based on the p_i reveals no information about the set of pivots, $\{p_i\}$.

Secure Operations: $O(n/w)$

6. **Tagging:** Using a linear pass, tag each element with its initial index, *i.e.*, the i th element in the list is tagged with a (secret-shared) value i . For $i = 1, \dots, n$ set $\llbracket e_i \rrbracket = \llbracket i \rrbracket$. Note that since the tags are publicly known, this step can be done without communication.

Secure Operations: $O(n)$

7. **Sorting sliding windows:** Fix a threshold, $\delta > 0$ (the exact value of δ is calculated in Lemma 11). Sort the list $\llbracket \vec{v} \rrbracket$ together with the tags \vec{e} , based on windows of size $4\delta^{-1}w$ as follows: For $i = 1, \dots, n/(2\delta^{-1}w) - 1$,

$$\begin{aligned} & ((\llbracket v_{2(i-1)\delta^{-1}w+1} \rrbracket, \llbracket e_{2(i-1)\delta^{-1}w+1} \rrbracket), \dots, (\llbracket v_{2(i+1)\delta^{-1}w} \rrbracket, \llbracket e_{2(i+1)\delta^{-1}w} \rrbracket)) \\ &= \text{SORT}(((\llbracket v_{2(i-1)\delta^{-1}w+1} \rrbracket, \llbracket e_{2(i-1)\delta^{-1}w+1} \rrbracket), \dots, (\llbracket v_{2(i+1)\delta^{-1}w} \rrbracket, \llbracket e_{2(i+1)\delta^{-1}w} \rrbracket))) \end{aligned}$$

Secure Operations: $O((n/(2\delta^{-1}w)) \text{ sort}(4\delta^{-1}w))$

8. **Identifying “strays”** For each element in \vec{v} , if its initial index (stored in its tag e) differs from its current position by more than $\delta^{-1}w$, then mark the element with a (secret-shared) tag “stray”.

For $i = 1, \dots, n$,

$$\llbracket s_i \rrbracket = \begin{cases} \llbracket 1 \rrbracket & \text{if } |e_i - i| > \delta^{-1}w \\ \llbracket 0 \rrbracket & \text{otherwise.} \end{cases}$$

and

$$\llbracket v_i \rrbracket = \begin{cases} \llbracket \perp \rrbracket & \text{if } |e_i - i| > \delta^{-1}w \\ \llbracket v_i \rrbracket & \text{otherwise.} \end{cases}$$

Secure Operations: $O(n)$

9. **Extracting strays** At this point, each stray is tagged with the (secret-shared) tag $\llbracket s_i \rrbracket = \llbracket 1 \rrbracket$ and we can extract these strays using a compaction algorithm.

$$(\llbracket z_1 \rrbracket, \dots, \llbracket z_b \rrbracket) = \text{StableCompaction}(\llbracket \vec{s} \rrbracket, \llbracket \vec{v} \rrbracket).$$

For an appropriate choice of parameters, δ, w , Lemma 6 shows that the number of strays will be less than \mathfrak{b} .

Secure Operations: $O(n)$

10. **Sorting pivots and strays:** Sort pivots together with strays, using SORT. There are n/w pivots, and the list of strays has \mathfrak{b} elements, so this list has $\mathfrak{b} + n/w$ elements. Pivot i , \tilde{p}_i is tagged with its tag, r (from Step 4), and each stray is tagged with 0.

$$\begin{aligned} & ((\llbracket z_1 \rrbracket, \llbracket \rho_1 \rrbracket), \dots, (\llbracket z_{\mathfrak{b}+n/w} \rrbracket, \llbracket \rho_{\mathfrak{b}+n/w} \rrbracket)) \\ &= \text{SORT}(((\llbracket z_1 \rrbracket, \llbracket 0 \rrbracket), \dots, (\llbracket z_{\mathfrak{b}} \rrbracket, \llbracket 0 \rrbracket)) \parallel ((\llbracket \tilde{p}_1 \rrbracket, \llbracket \tilde{r}_1 \rrbracket), \dots, (\llbracket \tilde{p}_{n/w} \rrbracket, \llbracket \tilde{r}_{n/w} \rrbracket))) \end{aligned}$$

Secure Operations: $\text{sort}(\mathfrak{b} + n/w)$

11. **Adding pivot IDs to strays** After step 10, the players hold a (sorted) list, $\llbracket \vec{z} \rrbracket$, of pivots and strays, and a list of “tags” $\llbracket \vec{p} \rrbracket$, where pivot p_i is tagged with r_i and each stray is tagged with 0. Both lists are of length $\mathfrak{b} + n/w$. In this step, they will tag each stray in this list with its corresponding pivot ID as follows. Initialize $\llbracket c \rrbracket = \llbracket \tilde{\rho}_1 \rrbracket$. For $i = 1, \dots, \mathfrak{b} + n/w$.

a. If $\rho_i \neq 0$ (*i.e.*, z_i is a pivot), then $\llbracket c \rrbracket = \llbracket \rho_i \rrbracket$, $\llbracket \rho_i \rrbracket = \llbracket 0 \rrbracket$.

b. If $\rho_i = 0$ (*i.e.*, z_i is not a pivot), then $\llbracket \rho_i \rrbracket = \llbracket c \rrbracket$.

At the end of this process, each of the strays is tagged with a (secret-shared) ID of the nearest pivot above it. To make this step oblivious, the conditional can be implemented with a simple mux. **Secure Operations:** $O(\mathfrak{b} + n/w)$

12. **Counting number of strays associated to each pivot:** Initialize $\llbracket c \rrbracket = \llbracket 0 \rrbracket$. Define the share vector $\llbracket \vec{s} \rrbracket$ as follows. For $i = \mathbf{b} + n/w, \dots, 1$

- a. If $\rho_i \neq 0$ (i.e., z_i is not a pivot), then set $\llbracket c \rrbracket = \llbracket c + 1 \rrbracket$, and $\llbracket s_i \rrbracket = \llbracket 0 \rrbracket$.
- b. If $\rho_i = 0$ (i.e., z_i is a pivot), then set $\llbracket s_i \rrbracket = \llbracket c \rrbracket$, $\llbracket c \rrbracket = \llbracket 0 \rrbracket$.

At the end of this process, if z_i is a pivot, then s_i stores the number of strays associated with that pivot.

Secure Operations: $O(n)$

13. **Removing pivots from stray list:** Using the sparse compaction algorithm `SparseCompaction` (described in Figure 8), extract a list of \mathbf{b} strays, together with their tags (recall the “tag” ρ_i gives the pivot ID r_j of the nearest pivot preceding the i th stray). Note that this compaction does not need to be stable.

$$\begin{aligned} & (((\llbracket z_1 \rrbracket, \llbracket \rho_1 \rrbracket)), \dots, (\llbracket z_{\mathbf{b}} \rrbracket, \llbracket \rho_{\mathbf{b}} \rrbracket)) \\ &= \text{SparseCompaction}(\mathbf{b}, \llbracket \vec{\rho} \rrbracket, (((\llbracket z_1 \rrbracket, \llbracket \rho_1 \rrbracket)), \dots, (\llbracket z_{\mathbf{b}+n/w} \rrbracket, \llbracket \rho_{\mathbf{b}+n/w} \rrbracket))) \end{aligned}$$

Secure Operations: $O(\mathbf{b} \log^2(n))$

14. **Extracting pivot counts:** After Step 12 the \vec{s} is a vector of length $\mathbf{b} + n/w$ containing the number of strays associated with each of the n/w pivots, and 0s in the locations corresponding to strays. Set

$$\llbracket \vec{s} \rrbracket = \text{StableCompaction}(n/w, \llbracket \vec{s} \rrbracket, \llbracket \vec{s} \rrbracket).$$

At this point, \vec{s} is a vector of length n/w , and for $i = 1, \dots, n/w$, s_i is the number of strays associated with pivot i .

Secure Operations: $O(\mathbf{b} + n/w)$

15. **Padding lists of strays:** Although the total number of strays, \mathbf{b} , is known, revealing the number of strays associated with each pivot would leak information. Thus the number of strays associated with each pivot must be padded to a uniform size. Note that every w th element in the *sorted* inputs $\llbracket \vec{a} \rrbracket$ and $\llbracket \vec{b} \rrbracket$ was defined to be a pivot, thus if the list were completely sorted, there could be at most $2(w-1)$ elements between any two adjacent pivots.

- a. For $i = 1, \dots, n/w$, for $j = 1, \dots, w$,

$$B_{(i-1) \cdot w + j} = \begin{cases} (1, (\perp, r_i)) & \text{if } j \leq \llbracket s_i \rrbracket \\ (0, (\perp, r_i)) & \text{otherwise.} \end{cases}$$

The elements tagged with 1 are the “dummy” elements. Note that among all the B_i , there are at most \mathbf{b} elements tagged with a 0. The elements tagged with a 0 will be removed in the next step.

- b. Using the algorithm `SparseSort` (described in Figure 4), sort the B_i .

$$\begin{aligned} & (((\llbracket \tilde{B}_{1,1} \rrbracket, (\llbracket \tilde{B}_{1,2} \rrbracket, \llbracket \tilde{B}_{1,3} \rrbracket)), \dots, (\llbracket \tilde{B}_{2n(w-1)/w,1} \rrbracket, (\llbracket \tilde{B}_{2n(w-1)/w,2} \rrbracket, \llbracket \tilde{B}_{2n(w-1)/w,3} \rrbracket)))) \\ &= \text{SparseSort}(((\llbracket B_{1,1} \rrbracket, (\llbracket B_{1,2} \rrbracket, \llbracket B_{1,3} \rrbracket)), \dots, (\llbracket B_{2n(w-1)/w,1} \rrbracket, (\llbracket B_{2n(w-1)/w,2} \rrbracket, \llbracket B_{2n(w-1)/w,3} \rrbracket)))) \end{aligned}$$

- c. We remove the first components, $\tilde{B}_{i,1}$, and set

$$\begin{aligned} & (((\llbracket C_{1,1} \rrbracket, \llbracket C_{1,2} \rrbracket), \dots, (\llbracket C_{2(w-1)n/w-b,1} \rrbracket, \llbracket C_{2(w-1)n/w-b,2} \rrbracket))) \\ &= (((\llbracket \tilde{B}_{1,2} \rrbracket, \llbracket \tilde{B}_{1,3} \rrbracket), \dots, (\llbracket \tilde{B}_{2n(w-1)/w-b,2} \rrbracket, \llbracket \tilde{B}_{2n(w-1)/w-b,3} \rrbracket))) \end{aligned}$$

Secure Operations: $O(n)$

16. **Merging strays and pads** Concatenate the list of \mathbf{b} strays, $(\llbracket \vec{z} \rrbracket, \llbracket \vec{\rho} \rrbracket)$ (from Step 13) along with the $2(w-1)n/w - \mathbf{b}$ pads $\llbracket \vec{C} \rrbracket$ from the previous step. Shuffle this list, keeping the associated tags, then, reveal the tags and move strays and pads to the positions given by their tags. This is accomplished as follows.

a.

$$((\llbracket \vec{C}_{1,1} \rrbracket, \llbracket \vec{C}_{1,2} \rrbracket), \dots, (\llbracket \vec{C}_{(2w-1)n/w,1} \rrbracket, \llbracket \vec{C}_{(2w-1)n/w,2} \rrbracket)) = \text{SHUFFLE} \left((\llbracket \vec{z} \rrbracket, \llbracket \vec{\rho} \rrbracket) \parallel \llbracket \vec{C} \rrbracket \right)$$

- b. For each element in this shuffled list, reveal the associated tag, $\tilde{C}_{i,2}$. Note that by Step 15 exactly $2(w-1)$ (secret-shared) elements will have each tag.
- c. For each i , move the block $\tilde{C}_{i,1}$ of size $2(w-1)$ to the location where $\tilde{C}_{i,2} = \tilde{r}_j$ (revealed in Step 5). At the end of Step 8, $\llbracket v_1 \rrbracket, \dots, \llbracket v_n \rrbracket$ was the list of elements with the strays set to \perp . To accomplish this, define the function $f(\tilde{r}_j) \stackrel{\text{def}}{=} j$ for $j = 1, \dots, n/w$, for the public \tilde{r}_j (revealed in Step 5).

```

for  $i = 0, \dots, n/w - 1$  do
  Define  $d_i = 1$ .
  for  $j = 1, \dots, w$  do
    set  $\llbracket \tilde{v}_{i(3w-1)+j} \rrbracket = \llbracket v_{iw+j} \rrbracket$ .
  end for
end for
for  $i = 1, \dots, (2w-1)$  do
  Let  $j = f(\tilde{C}_{i,2})$ .
  Set  $\llbracket \tilde{v}_{(j-1)(3w-1)+w+d_j} \rrbracket = \llbracket \tilde{C}_{i,1} \rrbracket$ .
  Set  $d_j = d_j + 1$ .
end for

```

Secure Operations: $O(n)$

17. **Compacting:** Now, we need to remove the $2(w-1)n/w$ dummy elements. We cannot use an off-the-shelf compaction algorithm [37, 6, 36] because these algorithms are not *stable*. Instead, we use the stable compaction algorithm `StableCompaction` (described in Figure 5).

For $i = 1, \dots, (3w-1)n/w$, if $\llbracket \tilde{v}_i \rrbracket = \llbracket \perp \rrbracket$, then set $\llbracket z_i \rrbracket = \llbracket 0 \rrbracket$, otherwise set $\llbracket z_i \rrbracket = \llbracket 1 \rrbracket$

$$\llbracket \vec{v} \rrbracket = \text{StableCompaction} \left(n, \llbracket \vec{z} \rrbracket, \llbracket \vec{v} \rrbracket \right).$$

Secure Operations: $O(n)$

18. **Sorting sliding windows** At this point, the players have a (secret-shared) list, $\llbracket \vec{v} \rrbracket$, consisting of n elements, and all elements are in approximately their correct positions. In this step, sort overlapping blocks of size $4((\delta^{-1} + 4)w + 2)$ using a secure sorting algorithm `SORT`.

For $i = 1, \dots, \left\lceil \frac{n}{2(\delta^{-1}+4)w+2} \right\rceil$, set

$$\begin{aligned} & \left(\llbracket \tilde{v}_{(i-1)2((\delta^{-1}+4)w+2)+1} \rrbracket, \dots, \llbracket \tilde{v}_{(i+1)2((\delta^{-1}+4)w+2)+1} \rrbracket \right) \\ &= \text{SORT} \left(\llbracket \tilde{v}_{(i-1)2((\delta^{-1}+4)w+2)+1} \rrbracket, \dots, \llbracket \tilde{v}_{(i+1)2((\delta^{-1}+4)w+2)+1} \rrbracket \right) \end{aligned}$$

At this point all the elements will be sorted.

Secure Operations: $O \left(\left\lceil \frac{n}{2(\delta^{-1}+4)w+2} \right\rceil \text{sort} \left(4((\delta^{-1} + 4)w + 2) \right) \right)$

19. **Return:** The sorted, secret shared list, $\llbracket \vec{v} \rrbracket$.

See Appendix A for a concrete calculation of the communication cost. See Appendix B for a proof of security.

8 Correctness

8.1 Bounding the number of strays

In order to analyze the running time of our algorithm, we need to bound the number of “strays” that appear in Step 8.

► **Lemma 6** (Bounding the number of strays). *Suppose a list, L , is created as follows*

1. L is composed of two sorted sublists $L = \vec{a} \parallel \vec{b}$ with $|\vec{a}| = \ell$, and $|\vec{b}| = n - \ell$. We assume $w \mid \ell$ and $w \mid n - \ell$.
2. Break the sorted list \vec{a} into blocks of size w . Call the first element (i.e., the smallest element) in each block a “pivot.”
3. Break the sorted list \vec{b} into blocks of size w . Call the first element (i.e., the smallest element) in each block a “pivot.”
4. Alice and Bob sort their joint list of blocks based on their pivots.

We call an element a “stray” if it is more than tw positions above its “true” position (i.e., its position in the fully sorted list of Alice and Bob’s entries). Then there at most $\frac{n}{t}$ strays.

Proof. Call the elements with indices $[iw + 1, \dots, (i + 1)w]$ in L a “block.” Let B_i denote the i th block for $i = 1, \dots, n/w$. Notice that

1. The elements within each block are sorted i.e., $L[iw + j] \leq L[iw + k]$ for each $0 \leq j \leq k \leq w$ and all i .
2. The lead elements in each block are sorted i.e., $L[iw] \leq L[jw]$ for $i \leq j$.
3. Each element is less than or equal to all pivots above it i.e., $L[iw + j] \leq L[kw]$ for all $j < w, k > i$.
4. All entries provided by a single party are in sorted order.

With these facts, notice that the only way an element’s index in L can be greater than its true position is if it was in a block where the preceding block was provided by the other party. Similarly, for an element to be more than tw from its true position, it must be in a block preceded by t consecutive blocks provided by the other party. If we label blocks provided by Alice with an a , and blocks provided by Bob with a b , then in order for w elements to be more than tw out position, we need a sequence of $\underbrace{a, \dots, a}_t, b$ or $b, \dots, b, \underbrace{a}_t$. There can only be $\frac{n}{tw}$ such sequences, so at most $\frac{n}{t}$ elements can be strays. ◀

Note that the sequence of operations described in Lemma 6 exactly corresponds to the process in the merging algorithm. In Step 2, the initial list is created as the concatenation of \vec{a} and \vec{b} . In Step 3, every w th element is tagged as a pivot, and in Step 4, the blocks are sorted based on their pivots. Lemma 6 gives a bound on the number of elements that can be more than tw positions away from their “true” location at the end of this process. In Step 7 (Sorting sliding windows), every element that is more than tw from its true location will move at least tw positions, and thus will be tagged as a “stray” in Step 8. Conversely, every element that moves more than tw positions in Step 7 must have been at least tw positions from its true location, and thus the set of “strays” found in Step 8 will exactly correspond to the set of elements that were tw positions from their true location, and this number is exactly what is bounded in Lemma 6.

► **Theorem 7** (Correctness of the merge). *If the input lists $\llbracket \vec{a} \rrbracket$ and $\llbracket \vec{b} \rrbracket$ in Step 2 are locally sorted, then the output list $\llbracket \vec{v} \rrbracket$ in Step 19 is globally sorted.*

Proof. ■ At the end of Step 2, the two parts of the list \vec{v} , (v_1, \dots, v_ℓ) and $(v_{\ell+1}, \dots, v_n)$ are locally sorted.

- At the end of Step 4 blocks of size w are sorted according to their leading (smallest) elements. Note that if these blocks were non-overlapping (*i.e.*, $v_{iw} < v_{i(w+1)}$ for $i = 1, \dots, n/w - 1$), the entire list would already be sorted at this point. In general, however, there may be considerable overlap in the blocks provided from the \vec{a} and those from \vec{b} .
- At the end of Step 8, Lemma 10 tells us that all elements that are more than $\delta^{-1}w$ from their true (final) location will be tagged as “stray.”
- Corollary 9 shows that after Step 8, no pivot will be tagged as “stray,” so no strays will be extracted in Step 9, and thus concatenating the lists of pivots and strays in Step 10 will not introduce any duplications.
- At the beginning of Step 18, Lemma 10 shows that every non-stray will be within $\delta^{-1}w$ of its true location. Lemma 8 shows that at the end of Step 4, every pivot is within w of its true location. By Step 16, every stray is within $3w + 2$ of its true pivot (based on the pivot’s location after Step 4. Thus at the beginning of 18, every stray is within $4w + 2$ of its true location. Putting this together, every element is within $(\delta^{-1} + 4)w + 2$ of its true location. Since the sorting windows are chosen so that every element is sorted along with all elements within a distance of $(\delta^{-1} + 4)w + 2$ on either side, at the end of Step 18 all the elements are sorted. ◀

► **Lemma 8.** *Let $v_{(i-1)w+1}$ denote the i th pivot at the end of Step 4. The true index, j^* , of $v_{(i-1)w+1}$ (in the completely sorted list) satisfies*

$$(i - 2)w < j^* < (i - 1)w + 1$$

Proof. At the end of Step 4 the pivots are all in sorted order relative to one another, and all the blocks between the pivots are locally sorted.

First, notice that if $(i - 1)w + 1 < j < w$, the $v_j \geq v_{(i-1)w+1}$, since the i th block is locally sorted. Next, notice that if $(i - 1)w + 1 \leq j$, then

$$v_{(i-1)w+1} \leq v_{(\lceil \frac{j}{w} \rceil - 1)w+1} \leq v_j \tag{1}$$

where the first inequality holds because the pivots are sorted, and the second inequality holds because v_j is in the $\lceil \frac{j}{w} \rceil$ th block which is locally sorted. Thus the true index j^* of $v_{(i-1)w+1}$ satisfies $j^* \leq (i - 1)w + 1$.

To see the other side of Equation 1, recall that the list \vec{v} was composed of blocks from two sources \vec{a} , and \vec{b} which were locally sorted. Without loss of generality, assume block i came from source \vec{a} . Now, consider the i' th block for $i' < i$. If the i' th block came from the same source as the i th block (\vec{a}), then since the original lists \vec{a} was sorted, all elements of the i' th block are less than or equal to $v_{(i-1)w+1}$. If the i' th block came from the *other source*, \vec{b} , then the elements $v_{(i'-1)w+2}, \dots, v_{i'w}$ could be out of order relative to $v_{(i-1)w+1}$. On the other hand, if there exists an i'' with $i' < i'' < i$, with i'' also from the source \vec{b} , then since \vec{b} was locally sorted, all elements of the i' th block are less than or equal to those of the i'' th block, in particular, they are less than or equal to the i'' th pivot which is less than or equal to the i th pivot $v_{(i-1)w+1}$. Thus only *one* block from \vec{b} can be out of order relative to $v_{(i-1)w+1}$. Thus at most $w - 1$ elements v_j with $j < (i - 1)w + 1$ can satisfy $v_j > v_{(i-1)w+1}$. ◀

► **Corollary 9.** *In Step 8, no pivot will be tagged as a “stray.”*

Proof. In Step 8, an element will be tagged as a stray if it is more than $\delta^{-1}w$ from its true location. Lemma 8 shows that a pivot is at most w from its true location, and thus can move at most w positions when we sort on sliding windows in Step 7. ◀

► **Lemma 10.** *After Step 8 every element that was more than $\delta^{-1}w$ from its true location before Step 7 will be tagged as a “stray.”*

Proof. To show this, it suffices to show that at the beginning of Step 7, if an element is more than $\delta^{-1}w$ from its true location (in the globally sorted list) then it will move at least $\delta^{-1}w$ during the sorting procedure of Step 7.

First, note that (as in the proof of Lemma 6) the only way an element can be more than $\delta^{-1}w$ from its true position is if $\lfloor \delta^{-1} \rfloor$ consecutive, adjacent blocks were provided by the other party. By the choice of sliding windows, every element will be sorted within a window containing at least $\delta^{-1}w$ elements on either side of it. Thus any element that is directly preceded or followed by $\delta^{-1}w$ “out-of-order” elements will move at least $\delta^{-1}w$ and thus be tagged as a stray. ◀

9 Extensions

Malicious adversaries: Our secure-merge algorithm outlined in Section 7 is “MPC-friendly,” and aside from the $O(n)$ -communication shuffle (discussed in Section 3), the entire algorithm can be naturally represented as an $O(n \log \log n)$ -sized circuit. For this reason, extending our merge protocol to provide malicious security requires (1) a linear-communication shuffle and (2) a generic MPC protocol that both provide security against malicious adversaries.

The multiparty shuffle of [34] can be modified to provide security against malicious adversaries, and several generic MPC protocols (e.g. [46, 28]) provide security against malicious adversaries. In the two-party setting, the literature on efficient, verifiable shuffles (e.g. [29, 11]) provide methods for making homomorphic encryption-based shuffles (like that of Section 3.3) secure against malicious adversaries without affecting its asymptotic communication complexity.

Merging more than two lists. Our protocol can also be modified in a straightforward manner to support more than two parties, by merging multiple lists recursively.

References

- 1 Gagan Aggarwal, Nina Mishra, and Benny Pinkas. Secure computation of the median (and other elements of specified ranks). *Journal of cryptology*, 23(3):373–401, 2010.
- 2 M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \log(n)$ steps. *Combinatorica*, 3:1–19, 1983.
- 3 S.W. Al-Haj Baddar and K.E. Batcher. The AKS sorting network. In *Designing sorting networks*. Springer, 2011. doi:10.1007/978-1-4614-1851-1_11.
- 4 Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. MCMix: Anonymous messaging via secure multiparty computation. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1217–1234, 2017.
- 5 Kazuyuki Amano and Akira Maruoka. On optimal merging networks. In *International Symposium on Mathematical Foundations of Computer Science*, pages 152–161. Springer, 2003.
- 6 Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, and Elaine Shi. OptORAMa: Optimal oblivious RAM. In *EUROCRYPT*, 2020.
- 7 Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *Theory of Cryptography Conference*, pages 137–156. Springer, 2007.
- 8 Sherenaz W Al-Haj Baddar and Kenneth E Batcher. The 0/1-principle. In *Designing Sorting Networks*, pages 19–25. Springer, 2011.
- 9 Kenneth E. Batcher. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 307–314. ACM, 1968.

- 10 Johes Bater, Gregory Elliott, Craig Eggen, Satyender Goel, Abel Kho, and Jennie Rogers. SMCQL: secure querying for federated databases. *Proceedings of the VLDB Endowment*, 10(6):673–684, 2017.
- 11 Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *EUROCRYPT*, pages 263–280. Springer, 2012.
- 12 Bruno Beauquier and Eric Darrot. On arbitrary waksman networks and their vulnerability. Technical Report 3788, INRIA, 1999.
- 13 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, pages 1–10, New York, NY, USA, 1988. ACM. doi:10.1145/62212.62213.
- 14 Dan Bogdanov, Sven Laur, and Riivo Talviste. A practical analysis of oblivious sorting algorithms for secure multi-party computation. In *Nordic Conference on Secure IT Systems*, pages 59–74. Springer, 2014.
- 15 Ran Canetti and Rafail Ostrovsky. Secure computation with honest-looking parties (extended abstract) what if nobody is truly honest? In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 255–264, 1999.
- 16 T-H Hubert Chan, Jonathan Katz, Kartik Nayak, Antigoni Polychroniadou, and Elaine Shi. More is less: Perfectly secure oblivious algorithms in the multi-server setting. In *ASIACRYPT*, pages 158–188. Springer, 2018.
- 17 Koji Chida, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Naoto Kiribuchi, and Benny Pinkas. An efficient secure three-party sorting protocol with an honest majority. IACR ePrint 2019/695, 2019.
- 18 David B. Cousins, Gerard Ryan, Yuriy Polyakov, and Kurt Rohloff. PALISADE. <https://gitlab.com/palisade>, 2019.
- 19 Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.
- 20 Sam Dittmer and Rafail Ostrovsky. Oblivious tight compaction in $O(n)$ time with smaller constant. In *SCN*, 2020.
- 21 Kris Vestergaard Ebbesen. On the practicality of data-oblivious sorting. Master’s thesis, Aarhus university, 2015.
- 22 Craig Gentry, Shai Halevi, Charanjit Jutla, and Mariana Raykova. Private database access with HE-over-ORAM architecture. In *International Conference on Applied Cryptography and Network Security*, pages 172–191. Springer, 2015.
- 23 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *STOC*, pages 218–229, 1987.
- 24 Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *Journal of the ACM (JACM)*, 43(3):431–473, 1996.
- 25 Michael T Goodrich. Randomized shellsort: A simple oblivious sorting algorithm. In *SODA*, pages 1262–1277. Society for Industrial and Applied Mathematics, 2010.
- 26 Michael T Goodrich. Data-oblivious external-memory algorithms for the compaction, selection, and sorting of outsourced data. In *SPAA*, pages 379–388. ACM, 2011.
- 27 Michael T Goodrich. Zig-zag sort: A simple deterministic data-oblivious sorting algorithm running in $o(n \log n)$ time. In *STOC*, pages 684–693. ACM, 2014.
- 28 Vipul Goyal, Rafail Ostrovsky, and Yifan Song. ATLAS: Efficient and scalable MPC in the honest majority setting. Manuscript, 2021.
- 29 Jens Groth. A verifiable secret shuffle of homomorphic encryptions. *Journal of Cryptology*, 23(4):546–579, 2010.
- 30 Koki Hamada, Dai Ikarashi, Koji Chida, and Katsumi Takahashi. Oblivious radix sort: An efficient sorting algorithm for practical secure multi-party computation. *IACR Cryptology ePrint Archive*, 2014:121, 2014.

- 31 Koki Hamada, Ryo Kikuchi, Dai Ikarashi, Koji Chida, and Katsumi Takahashi. Practically efficient multi-party sorting protocols from comparison sort algorithms. In *ICISC*, pages 202–216. Springer, 2012.
- 32 Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS*, 2012.
- 33 Peeter Laud and Alisa Pankova. Privacy-preserving record linkage in large databases using secure multiparty computation. *BMC medical genomics*, 11(4):84, 2018.
- 34 Sven Laur, Jan Willemson, and Bingsheng Zhang. Round-efficient oblivious database manipulation. In *ISC*, pages 262–277. Springer, 2011.
- 35 Tom Leighton, Yuan Ma, and Torsten Suel. On probabilistic networks for selection, merging, and sorting. *Theory of Computing Systems*, 30(6):559–582, 1997.
- 36 Wei-Kai Lin, Elaine Shi, and Tiancheng Xie. Can we overcome the $n \log n$ barrier for oblivious sorting? In *SODA*, pages 2419–2438. Society for Industrial and Applied Mathematics, 2019.
- 37 John C Mitchell and Joe Zimmerman. Data-oblivious data structures. In *STACS*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.
- 38 Rafail Ostrovsky. Efficient computation on oblivious RAMs. In *STOC*, pages 514–523, 1990.
- 39 Michael S Paterson. Improved sorting networks with $O(\log N)$ depth. *Algorithmica*, 5(1-4):75–92, 1990.
- 40 Joel Seiferas. Sorting networks of logarithmic depth, further simplified. *Algorithmica*, 53(3):374–384, 2009.
- 41 Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
- 42 Nigel P Smart and Younes Talibi Alaoui. Distributing any elliptic curve based protocol. In *IMACC*, pages 342–366. Springer, 2019.
- 43 Nikolaj Volgushev, Malte Schwarzkopf, Ben Getchell, Mayank Varia, Andrei Lapets, and Azer Bestavros. Conclave: secure multi-party computation on big data. In *EuroSys*, page 3. ACM, 2019.
- 44 Abraham Waksman. A permutation network. *Journal of the ACM (JACM)*, 15(1):159–163, 1968.
- 45 Xiao Wang, Alex J Malozemoff, and Jonathan Katz. EMP-toolkit: Efficient multiparty computation toolkit. <https://github.com/emp-toolkit/emp-sh2pc>, 2016.
- 46 Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-scale secure multiparty computation. In *CCS*, pages 39–56. ACM, 2017.
- 47 Andrew Yao. Protocols for Secure Computations (Extended Abstract). In *FOCS*, pages 160–164, 1982.
- 48 Andrew Yao. How to Generate and Exchange Secrets. In *FOCS*, pages 162–167, 1986.
- 49 Samee Zahur and David Evans. Obliv-c: A language for extensible data-oblivious computation. IACR Cryptology ePrint Archive 2015/1153, 2015.

Appendix

A Protocol analysis

A.1 Efficiency analysis

In this section, we examine the efficiency of our construction.

Our algorithm requires a data-oblivious sorting routine, SORT. Sorting networks like Batcher’s and the AKS network are deterministic data-independent sorting algorithms. Batcher’s sorting network requires $O(n \log^2(n))$ comparisons to securely sort n elements, and the AKS sorting network [2] uses only $O(n \log(n))$ comparisons, the hidden constants are so large that it only begins to beat Batcher’s sort for $n > 10^{52}$ [3] and is hence impractical.

The work of [31, 30] provide a data-oblivious sorting routines that requires $O(n \log n)$ comparisons (with small constants) by combining a permutation network, and (public) sorting algorithm. To the best of our knowledge, this is the fastest data-oblivious sort in practice, and has optimal asymptotic guarantees.

Throughout the rest of the analysis, we assume that the subroutine SORT is a data-oblivious sorting algorithm that requires $\text{sort}(n) = O(n \log(n))$ secure operations.

► **Lemma 11.** *The merging algorithm in described in Section 7.2 requires $O(n \log \log(n))$ secure operations.*

Proof. Lemma 6 tells us that the maximum number of strays, \mathbf{b} , is δn . In order for Step 15 to run in linear time, we set $\delta = O(\log^{-2}(n))$. Note, however, that if we use the asymptotically efficient linear-time compaction algorithm from [6], we can choose a larger value for δ , (i.e., $\delta = O(1)$). With this choice of δ , the runtime is dominated by Steps 10 and 18. Step 10 takes time $\text{sort}(\mathbf{b} + n/w)$. Setting $t = \delta^{-1}$, Lemma 6 gives $\mathbf{b} = \delta n$, so $\mathbf{b} + n/w = O(n/w)$. Since $\text{sort}(n) = O(n \log(n))$ operations, setting $w = O(\log^2(n))$, step 10 takes $O(n)$ secure operations. Step 18 takes $O\left(\left\lceil \frac{n}{2(\delta^{-1}+4)w+2} \right\rceil \text{sort}\left(4\left((\delta^{-1}+4)w+2\right)\right)\right)$. With our choices of $\delta = O(\log^{-2}(n))$, and $w = O(\log^2(n))$, $\left\lceil \frac{n}{2(\delta^{-1}+4)w+2} \right\rceil = O(n \log^{-4}(n))$, and $4\left((\delta^{-1}+4)w+2\right) = O(\log^4(n))$. Since $\text{sort}(n) = O(n \log(n))$ operations, Step 18 takes time $O(n \log \log(n))$. ◀

B Obliviousness

In this section, we show that the algorithm given in Section 7 is *data oblivious*.

► **Lemma 12** (Obliviousness). *The merge algorithm given in Section 7 is data oblivious.*

Proof. Showing data-obliviousness requires showing

1. All values that affect the control flow are independent of the inputs
2. The value, and time of revelation of all revealed values are independent of the inputs

It is straightforward to check that all revealed values are uniformly and independently chosen, and that the time of their revelation is deterministic (and hence input-independent).

Data are revealed at Steps 4 and 16. At Step 4, the pivot-IDs that are revealed are uniformly random and independent of the input. The pivot locations are deterministic (every w th element).

At Step 16, the same number of pivot IDs of each type are revealed ($2w - 1$) because of the padding, and their locations are data-independent because of the secure shuffle.

Thus the entire algorithm is data-oblivious as long as the secure shuffle is data oblivious. ◀

With the exception of an asymptotically efficient secure shuffling algorithm, all the steps of our sorting algorithm can be implemented with generic secure computation techniques, and hence can easily be made secure against malicious parties or extended to the multiparty setting.

Note that three steps require the asymptotically efficient secure shuffle. These are Steps 4 (“Sorting based on pivots”), 16 (“Merging strays and pads”) and 17 (“Compacting”).

In Section 3 we give standard algorithms for instantiating a two-party data-oblivious shuffle (using additively homomorphic encryption) and a multi-party data-oblivious shuffle (using one-way functions).

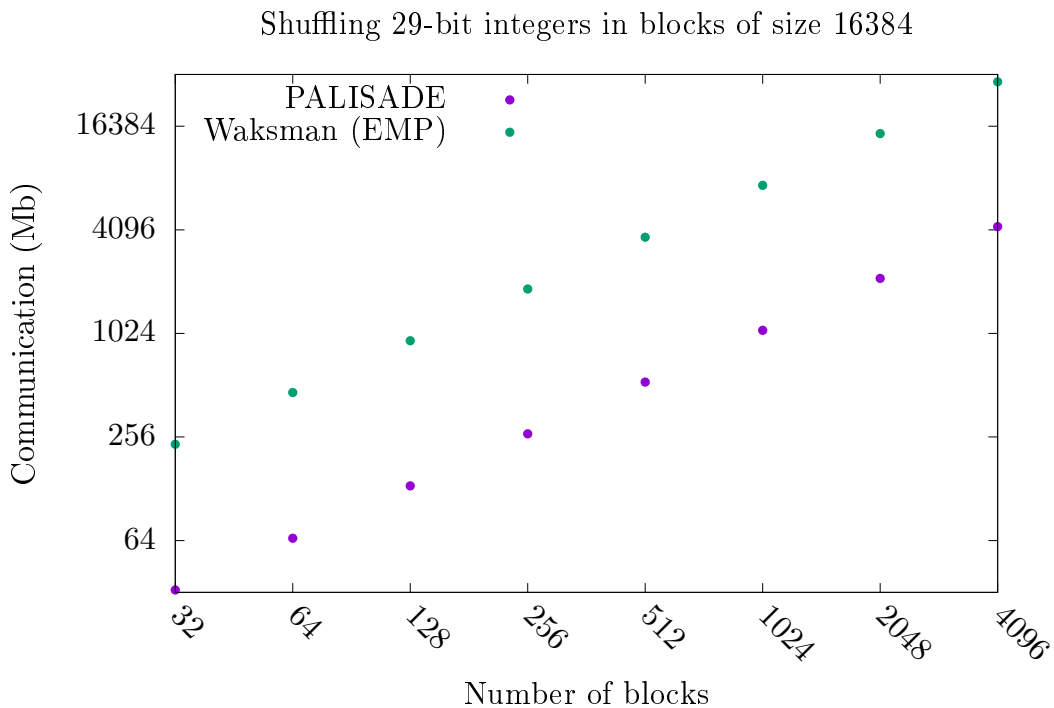
C Shuffling times

Our secure sorting algorithm requires an efficient method for secure *shuffling* with payloads. We give a simple, linear-time algorithm for this in Figure 2 based on additively homomorphic encryption. To demonstrate the *practical* performance of this shuffle, we implemented and benchmarked it using the PALISADE FHE library [18].

We used PALISADE version 1.6, with the “BFVrns” cryptosystem with a security level set to “HEstd_128_classic.” This scheme uses the plaintext modulus 536903681, which can encode plaintexts of length 29 bits. In this scheme, each ciphertext can be “packed” with 16384 plaintexts, so each ciphertext holds $29 \cdot 16384 = 475136$ plaintext bits. Each ciphertext required 1053480 bytes to store, so the ciphertext expansion with these parameters is approximately 17.7.

We benchmarked the running time and communication cost of this scheme, and the results are presented in Figure 6.

For comparison, we also implement the Waksman permutation network [44] using the semi-honest 2pc provided by EMP [45]. The Waksman permutation network has complexity $O(n \log n)$, where n is the number of *bits* being shuffled, rather than the number of *blocks*. Because a uniform setting of control bits in the Waksman network does not yield a uniform permutation, in practice, the Waksman network would usually be run twice (where each player inputs control bits for one of the shuffles). These benchmarks only show a single run of Waksman network.



■ **Figure 6** The communication cost of the FHE-based secure shuffling protocol in Section 3. Note that both the x and y axes are on a log scale, and in such a scale, the function $y = x \log x$, will appear as $y = x + \log x$, which is why the $O(n \log n)$ Waksman shuffle appears linear.

D The [37] compaction algorithm

In this section, we review a data independent compaction algorithm described in [37, Theorem 14]. The algorithm runs in $O(n \log \log n)$ time, and works by recursively peeling off 1/6th of the remaining elements (depending on whether the majority of the remaining elements are zero or one).

Algorithm 3 describes a simple randomized procedure that takes an array, \vec{v} , of length n with the promise that at least $n/2$ of the elements in \vec{v} are 0. Algorithm 3 reorganizes \vec{v} such that the first $n/6$ elements of \vec{v} are 0 with high probability. The algorithm makes use of a deterministic $O(n \log(n))$ partitioning algorithm, `partition`, e.g. that of [26] that requires exactly $n \log(n)$ comparisons.

The full algorithm is described in Algorithm 2.

Algorithm 1 The [37] data-independent partitioning algorithm that runs in $O(n \log \log n)$ time.

```

Private input: A list  $\vec{v} \in \{0, 1\}^n$ 
Initialize  $a_0 = 0$ 
for  $i = 0, \dots, n - 1$  do                                ▷ Count the number of 1s in  $\vec{v}$ 
     $a_0 = a_0 + v[i]$ 
end for
return  $MZ(\vec{v}, a_0)$ 
    
```

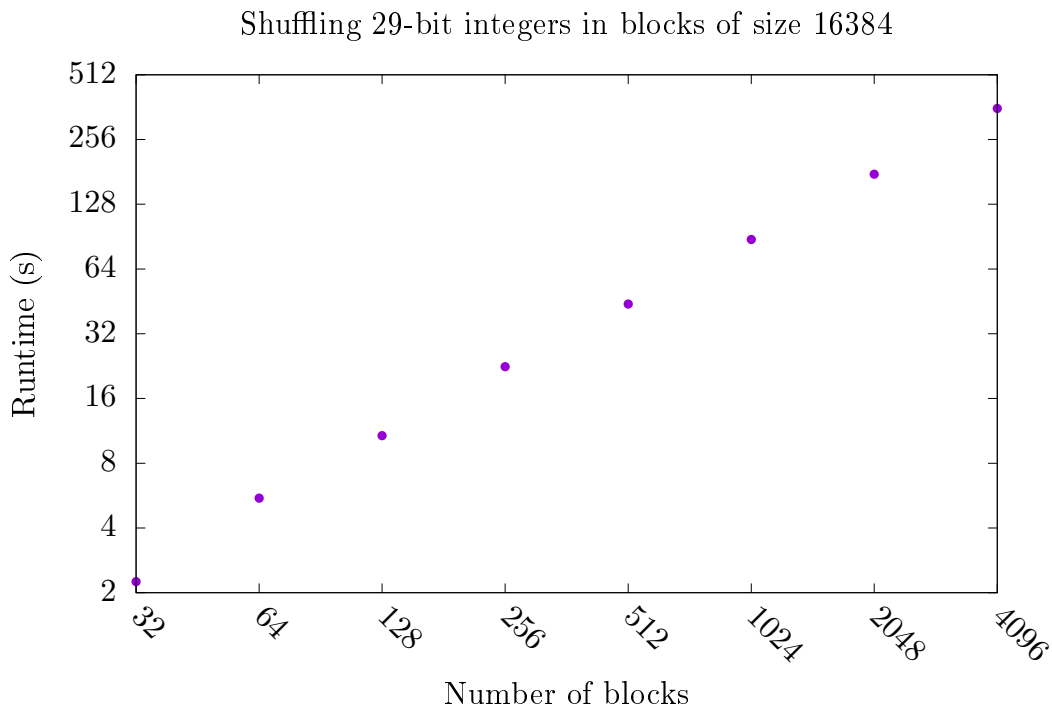


Figure 7 The running time of the FHE-based secure shuffling protocol. Both parties were run on the same machine, so networking costs were minimized.

■ **Algorithm 2** The [37] data-independent partitioning algorithm that runs in $O(n \log \log n)$ time.

Private input: A list $\vec{v} \in \{0, 1\}^n$
 Private input: a , the number of 1's in \vec{v}

```

if  $n < 4s$  then
    return partition( $\vec{v}$ )
end if
    Initialize flipped = 0
if  $a > n/2$  then                                     ▷ If majority ones, flip the bits of  $\vec{v}$ 
    flipped = 1
end if
     $\vec{v} = \text{flipped} \cdot (\vec{v} \oplus 1^n) + (1 - \text{flipped}) \cdot \vec{v}$                                      ▷ If flipped = 1, invert bits of  $\vec{v}$ 
     $\vec{v} = \text{MZInner}(\vec{v}, a)$                                                                                                        ▷ First  $n/6$  bits are now 0 w.h.p.
    Define  $\vec{v}_l = (v[0], \dots, v[\lfloor n/6 \rfloor])$                                                                                        ▷  $\vec{v}_l$  is sorted portion of list
    Define  $\vec{v}_r = (v[\lfloor n/6 \rfloor + 1], \dots, v[n - 1])$                                                                          ▷  $\vec{v}_r$  is unsorted portion of list
     $a = \text{flipped} \cdot (n - a) + (1 - \text{flipped}) \cdot a$                                                                        ▷ Number of 1s remaining in  $\vec{v}_r$ 
     $\vec{v}_r = \text{MZ}(\vec{v}_r, a)$                                                                                                        ▷ Recurse
     $\vec{v}_u = \vec{v}_l \parallel \vec{v}_r$ 
     $\vec{v}_f = \text{reverse}(1^n \oplus \vec{v}_u)$                                                                                                ▷ Flip bits and reverse list
return flipped  $\cdot \vec{v}_f + (1 - \text{flipped}) \cdot \vec{v}_u$ 
    
```

■ **Algorithm 3** The inner step of the [37] algorithm that moves $n'/6$ elements of type 0 to the beginning of the list.

Input: A list $\vec{v} \in \{0, 1\}^{n'}$ with the promise that $\text{majority}(\vec{v}) = 0$

```

for  $i$  from 0 to  $n'/3 - 1$  do                                                                                               ▷ Boost probability that  $\vec{v}[i] = 0$ 
    for  $j$  from 0 to  $c - 1$  do
         $r \xleftarrow{\$} [n'/3, n' - 1]$ 
        if  $\vec{v}[r] = 0$  then
            swap( $\vec{v}[i], \vec{v}[r]$ )
        end if
    end for                                                                                                               ▷ At this point,  $\Pr[\vec{v}[i] = 0] > 1 - (\frac{1}{2})^{c+1}$ 
end for
for  $i$  from 0 to  $n'/(3s) - 1$  do
    partition( $\vec{v}[i \cdot s, \dots, (i + 1) \cdot s - 1]$ )                                                                       ▷ Sort blocks of size  $s$ 
    for  $j$  from 0 to  $s/2$  do
        swap( $\vec{v}[i \cdot s/2 + j], \vec{v}[(2 \cdot i) \cdot s/2 + j]$ )                                                                   ▷ Move first half of each block to the
    end for                                                                                                               beginning of the list
end for
end for
    
```

► **Lemma 13** (Algorithm 1 correctness). *The probability that Algorithm 1 fails to correctly compact a list is*

$$\frac{6n - 20s}{3s} e^{-2\left(\frac{1}{2} - \left(\frac{3}{4}\right)^c\right)^2 s}$$

A straightforward calculation shows that for $c = 6$, setting $s = \log(n)^2$, gives a failure probability of less than 2^{-40} for all $n > 2^{12}$.

Proof. At each iteration through the loop, the size of the remaining list drops by a factor of $5/6$. The loop terminates when the list size reaches $4s$. If we let t denote the number of iteration of the algorithm, we have $4s = \left(\frac{5}{6}\right)^t n$, which means

$$t = \frac{\log\left(\frac{n}{4s}\right)}{\log\frac{6}{5}}$$

and $\left(\frac{5}{6}\right)^t = \frac{4s}{n}$.

First, notice that

$$\begin{aligned} \sum_{i=0}^t \left(\frac{5}{6}\right)^i &= \left(\frac{1 - \left(\frac{5}{6}\right)^{t+1}}{1 - \frac{5}{6}}\right) \\ &= 6 \left(1 - \frac{5}{6} \cdot \left(\frac{5}{6}\right)^t\right) \\ &= 6 \left(1 - \frac{5}{6} \cdot \frac{4s}{n}\right) \\ &= \frac{6n - 20s}{n}. \end{aligned}$$

A given block of size s will fail if it has more than $\frac{s}{2}$ zeros. The right half is guaranteed to have at least $\frac{n'}{2} - \frac{n'}{3} = \frac{n'}{6}$ zeros, so at least $\frac{1}{4}$ of the elements on the right hand side are zero. For a given a_i , after making c attempted swaps, the probability that a_i is zero is at least $1 - \left(\frac{3}{4}\right)^c$. Thus by the Hoeffding bound, the probability that a given block fails is at most

$$e^{-2\left(\frac{1}{2} - \left(\frac{3}{4}\right)^c\right)^2 s}$$

Taking a union bound over the $\frac{n'}{3s}$ blocks of size s , and then summing over the n' , we have the total failure probability is bounded by

$$\frac{n}{3s} e^{-2\left(\frac{1}{2} - \left(\frac{3}{4}\right)^c\right)^2 s}$$

◀

► **Lemma 14** (Algorithm 1 runtime). *Algorithm 1 obviously compacts a list using*

$$\frac{6n - 14s}{3} \log s + \frac{6n - 20s}{3} \cdot (c + 6)$$

comparisons.

Proof. Algorithm 1 uses n comparisons to compute a before calling Algorithm 2.

Each iteration of the loop (Algorithm 3) requires $\frac{n'}{3s}$ calls to `partition` (on sets of size s). At iteration i , $n' = \left(\frac{5}{6}\right)^i n$, which gives

$$\frac{n}{3s} \sum_{i=0}^t \left(\frac{5}{6}\right)^i = \frac{6n - 20s}{3s}.$$

7:28 Secure Merge with $O(n \log \log n)$ Secure Operations

So there are a total of $\frac{6n-20s}{3s}$ calls to partition of size s .

At each iteration of the loop (Algorithm 3) there are also $\frac{n' \cdot c}{3}$ controlled swaps.

Thus the total number of controlled swaps in is

$$\frac{n \cdot c}{3} \sum_{i=0}^t \left(\frac{5}{6}\right)^i = \frac{6n - 20s}{3} \cdot c$$

At every call to Algorithm 2 there are $2n + \log(n)$ swaps. Finally, there is one call to partition of size $4s$.

Thus total runtime is

$$6n - 20s \cdot \left(\frac{1}{3s} \text{partitionTime}(s) + \frac{c}{3} \frac{6n - 20s}{3} + 2 \right) + \text{partitionTime}(4s)$$

Where $\text{partitionTime}(n)$ denotes the number of swaps needed to partition a set of size s . There are many options for the partition algorithm used here. In our implementation, we use the simple, deterministic, data independent partitioning algorithm from [26], which requires $n \log n$ controlled swaps.

Thus the total number of comparisons is

$$\frac{6n - 14s}{3} \log s + \frac{6n - 20s}{3} \cdot (c + 6).$$

As noted above, setting $c = 6$, and $s = \log^2 n$ gives a failure probability below 2^{-40} , for all $n > 2^{12}$, so with these parameters, the overall number of comparisons is

$$4n \log \log n + 14n. \quad \blacktriangleleft$$

The deterministic data independent partitioning algorithm from [26] runs in time requires exactly $n \log n$, secure comparisons, so the [37] compaction algorithm will start to beat the deterministic [26] solution when $4n \log \log n < n \log n$, *i.e.*, when $n > 2^{16}$.

E Sparse compaction

The sparse sorting algorithm of Figure 4 can also be used for extracting a small number of nonzero values from a list. The resulting *sparse compaction* algorithm takes three arguments, a public bound, t , a secret-shared vector of “tags,” \vec{s} , and a secret shared vector of “payloads,” \vec{x} .

$$[\vec{y}] = \text{SparseCompaction}(t, [\vec{s}], [\vec{x}]).$$

We outline our sparse compaction algorithm in Figure 8.

► **Corollary 15** (Sparse compaction). *Given a secret-shared list of length n with at most t nonzero elements, the non-zero elements can be extracted in $O(t \log^2(n) + n)$ time using the algorithm given in Figure 8.*

F Security proofs

► **Lemma 16** (Secure shuffling). *If $\text{PKE} = (\text{Gen}, \text{Enc}, \text{Dec})$ is a CPA-secure additively-homomorphic cryptosystem, over the group \mathbb{G} , and the scheme is rerandomizable, then secure shuffling algorithm in Figure 2 securely implements a 2-party shuffle in the honest-but-curious setting.*

SparseCompaction

Public parameters: An integer, n , and a bound $t < n$.

Input: A secret-shared list, $[[\vec{x}]]$, of length n , each element of $[[\vec{x}]]$ is tagged with a (secret-shared) tag $[[y_i]]$ with $y_i \in \{0, 1\}$. The guarantee is that at most t tags are 1.

Output: A secret shared list, $[[\vec{w}]]$ of length t containing all the nonzero elements of \vec{x} .

1. **Sorting:** Use the sparse sorting algorithm, `SparseSort` to sort the n elements of \vec{V} based on their tags.

$$(([[\bar{x}_1]], [[\bar{y}_1]]), \dots, ([[\bar{x}_n]], [[\bar{y}_n]])) = \text{SparseSort}((([x_1], [y_1]), \dots, (\bar{x}_n, \bar{y}_n)))$$

2. **Extraction:** Return the top t elements of the sorted list $([[\bar{x}_1]], \dots, [[\bar{x}_t]])$.

■ **Figure 8** Sparse compaction.

Proof. First, we note that if Alice *or* Bob, generates a random permutation, the resulting permutation will be random. Second, note that since Alice and Bob are honest-but-curious, at every step the ciphertexts they provide correctly encode some ordering of the secret shares.

Consider a series of Bob's views. Let view_0^b denote Bob's view in the real protocol. Let view_1^b be the protocol where, instead of encrypting her shares under pk^a , Alice encrypts the 0 vector. The semantic security of PKE ensures that view_0^b and view_1^b are indistinguishable. Let view_2^b be the protocol where, instead of shuffling the pairs of ciphertexts, Alice simply re-randomizes Bob's shares (through the homomorphic encryption). Since the encryption is re-randomizable, and the plaintext shares are re-randomized over the group \mathbb{G} , both the ciphertexts and the decrypted plaintexts are indistinguishable from in view_1^b . Thus, Alice's security is preserved.

The proof of security from Bob's side is essentially identical.



Perfectly Oblivious (Parallel) RAM Revisited, and Improved Constructions

T-H. Hubert Chan ✉🏠

Department of Computer Science, University of Hong Kong, Hong Kong

Elaine Shi ✉🏠

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Wei-Kai Lin ✉🏠

Department of Computer Science, Cornell University, Ithaca, NY, USA

Kartik Nayak ✉🏠

Department of Computer Sciences, Duke University, Durham, NC, USA

Abstract

Oblivious RAM (ORAM) is a technique for compiling any RAM program to an oblivious counterpart, i.e., one whose access patterns do not leak information about the secret inputs. Similarly, Oblivious Parallel RAM (OPRAM) compiles a *parallel* RAM program to an oblivious counterpart. In this paper, we care about ORAM/OPRAM with *perfect security*, i.e., the access patterns must be *identically distributed* no matter what the program's memory request sequence is. In the past, two types of perfect ORAMs/OPRAMs have been considered: constructions whose performance bounds hold *in expectation* (but may occasionally run more slowly); and constructions whose performance bounds hold *deterministically* (even though the algorithms themselves are randomized).

In this paper, we revisit the performance metrics for perfect ORAM/OPRAM, and show novel constructions that achieve asymptotical improvements for all performance metrics. Our first result is a new perfectly secure OPRAM scheme with $O(\log^3 N / \log \log N)$ *expected* overhead. In comparison, prior literature has been stuck at $O(\log^3 N)$ for more than a decade.

Next, we show how to construct a perfect ORAM with $O(\log^3 N / \log \log N)$ *deterministic* simulation overhead. We further show how to make the scheme parallel, resulting in a perfect OPRAM with $O(\log^4 N / \log \log N)$ *deterministic* simulation overhead. For perfect ORAMs/OPRAMs with deterministic performance bounds, our results achieve *subexponential* improvement over the state-of-the-art. Specifically, the best known prior scheme incurs more than \sqrt{N} deterministic simulation overhead (Raskin and Simkin, Asiacrypt'19); moreover, their scheme works only for the sequential setting and is *not* amenable to parallelization.

Finally, we additionally consider perfect ORAMs/OPRAMs whose performance bounds hold with high probability. For this new performance metric, we show new constructions whose simulation overhead is upper bounded by $O(\log^3 / \log \log N)$ except with negligible in N probability, i.e., we prove high-probability performance bounds that match the expected bounds mentioned earlier.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols

Keywords and phrases perfect oblivious RAM, oblivious PRAM

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.8

Related Version *Full Version*: <https://ia.cr/2020/604>

Funding *T-H. Hubert Chan*: This work was partially supported by the Hong Kong RGC under the grants 17200418 and 17201220.

Elaine Shi: This work was partially supported by NSF CNS-1601879, an ONR YIP award, and a Packard Fellowship.

Wei-Kai Lin: This work was supported by a DARPA Brandeis award.

Kartik Nayak: This work was partially supported by NSF Award 2016393.

Acknowledgements Author ordering is randomized.



© T-H. Hubert Chan, Elaine Shi, Wei-Kai Lin, and Kartik Nayak;
licensed under Creative Commons License CC-BY 4.0

2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 8; pp. 8:1–8:23



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Oblivious RAM (ORAM) is an algorithmic construction that provably obfuscates a (parallel) program’s access patterns. It was first proposed in the ground-breaking work by Goldreich and Ostrovsky [22, 21], and its parallel counterpart Oblivious Parallel ORAM (OPRAM) was proposed by Boyle et al. [9]. ORAM and OPRAM are fundamental building blocks for enabling various forms of secure computation on sensitive data, e.g., either through trusted-hardware [35, 19, 30, 28] or relying on cryptographic multi-party computation [24, 29]. Since the initial proposal, ORAM and OPRAM have attracted much interest from various communities, and there has been a line of work dedicated to understanding their asymptotic and concrete efficiencies. It is well-known [22, 21, 27] that any O(P)RAM scheme must incur at least a logarithmic *overhead* (also known as *simulation overhead*) in (parallel) runtime relative to the insecure counterpart. On the other hand, ORAM/OPRAM schemes with poly-logarithmic overhead have been known [22, 21, 23, 26, 36, 37, 38, 32], and the very recent exciting work of Asharov et al. [5] showed how to match the logarithmic lower bound in the sequential ORAM setting, assuming the existence of one-way functions and a computationally bounded adversary.¹ Throughout this paper, we use the standard notion of *simulation overhead* originally suggested by Goldreich and Ostrovsky [22, 21]: if the original RAM/PRAM’s (parallel) runtime is T and the corresponding ORAM/OPRAM’s (parallel) runtime is χT , we say that the ORAM/OPRAM has simulation overhead χ .

Motivation for perfectly secure ORAMs/OPRAMs. With the exception of very few works, most of the literature has focused on either *computationally* secure [22, 21, 23, 26, 11, 32, 5] or *statistically* secure [2, 36, 37, 15, 38] ORAMs. A computationally secure (or statistically secure, resp.) ORAM guarantees that for any two request sequences of the same length, the access patterns incurred are computationally (or statistically resp.) indistinguishable. Most known computationally secure or statistically secure schemes [22, 21, 36, 37, 38, 9, 13] suffer from a small failure probability that is *negligible in the ORAM’s size henceforth denoted N* while achieving $\text{poly log } N$ overhead. If the ORAM/OPRAM’s size is large, say, $N \geq \lambda$ for some desired security parameter λ , then the failure probability would also be negligible in the security parameter. Unfortunately, for small choices of N (e.g., $N = \text{poly log } \lambda$), these schemes actually give polylogarithmic overhead in security parameter λ (and not in N) to achieve a $\text{negl}(\lambda)$ security failure probability – note that a $\text{poly log } \lambda$ overhead equals to $N^{\Theta(1)}$ for this parameter regime, and thus the dependence on N is undesirable. Even though at first sight, it seems like we might not care about the parameter regime when N is much smaller than λ ; as it turns out, such a small- N ORAM/OPRAM (with polylogarithmic in N overhead) was needed in many scenarios, such as in the construction of searchable encryption schemes [18], oblivious algorithms [36, 32, 4, 5] including notably, the recent OptORAMa work [5] that constructed an optimal ORAM.

The study of *perfectly* secure ORAMs/OPRAMs is partly motivated by the aforementioned mismatch. Moreover, recall also that *perfect security* has long been a topic of interest in the multi-party computation and zero-knowledge proof literature [25, 20], and its theoretical importance widely-accepted. Historically, perfect security is viewed as attractive since 1) the security holds in any computational model even if quantum computers or other forms of computers can be built; and 2) perfectly secure schemes often have clean compositional properties. Therefore, another natural application of perfectly secure ORAM/OPRAM is to construct efficient perfectly secure, RAM-model MPC.

¹ For the parallel setting, how to achieve optimality remains open.

Does there exist a perfectly secure ORAM/OPRAM with $o(\log^3 N)$ overhead? Despite the sustained and lively progress in understanding the asymptotic overhead of computationally and statistically secure ORAMs/OPRAMs, our understanding of perfectly secure ORAMs/OPRAMs has been somewhat stuck. In general, few results are known in the perfect security regime: in 2011, Damgård et al. [17] first showed a perfectly secure ORAM scheme with $O(\log^3 N)$ expected simulation overhead and $O(\log N)$ server space *blowup*, where the space blowup is the ratio between the server space consumed by the scheme compared to the insecure space N . Recently, Chan et al. [12] show an improved and simplified construction that removed the $\log N$ server space blowup; and moreover, they showed how to extend the approach to the parallel setting resulting in a perfectly secure OPRAM scheme with $O(\log^3 N)$ expected overhead. There is no known super-logarithmic lower bound for perfect security, and thus we do not understand yet whether the requirement of perfect security would inherently incur more overhead than computationally secure ORAMs. Therefore, an exciting and extremely challenging open direction is to understand the exact asymptotic complexity of perfectly secure ORAMs and OPRAMs, that is, to seek a matching upper- and lower-bound. This is a very ambitious goal and in this paper, we aim to take the next natural step forward. Since all prior upper bounds seem stuck at $O(\log^3 N)$, we ask the following natural question: *does there exist an ORAM/OPRAM with $o(\log^3 N)$ asymptotic overhead?*

The large gap between expected and deterministic performance bounds. To achieve perfect security, the prior perfect ORAM/OPRAM constructions pay a price: specifically their algorithms are Las Vegas, and the stated $O(\log^3 N)$ overhead is in an *expected* sense. Their ORAMs can occasionally run longer than $O(\log^3 N)$ time if certain unlucky events happen (where the unlucky events are identically distributed for all inputs so that the scheme remains perfectly secure). Moreover, the smaller the choice of N , the more likely that the ORAM can run much longer than the expectation.

Raskin et al. [34] (Asiacrypt'19) recently pointed out that this issue was somewhat shoved under the rug in prior works on perfect ORAMs/OPRAMs, and they were the first to ask how to construct perfect ORAMs with *deterministic* performance bounds. To avoid confusion, note that all ORAM schemes with non-trivial efficiency must be randomized algorithms; however, their performance bounds can be made deterministic (i.e., deterministic performance bounds does not mean that the algorithm is deterministic). Raskin et al. showed a perfectly secure ORAM with a deterministic simulation overhead $O(\sqrt{N} \frac{\log N}{\log \log N})$ (assuming $O(1)$ client-side storage²). While conceptually interesting, in comparison with the $O(\log^3 N)$ schemes [17, 12], the price to obtain deterministic bounds seems high. Therefore, another natural question is, *does there exist perfectly secure ORAMs/OPRAMs with deterministic polylogarithmic overhead?*

1.1 Our Results and Contributions

Our contributions are two-fold. First, following Raskin et al., we make another effort at systematizing the performance metrics for perfect ORAM/OPRAMs. Besides expected and deterministic performance bounds, we additionally consider the notion of *high-probability* performance bounds (explained below). Second, we show novel perfect ORAM/OPRAM constructions with asymptotical performance improvements for all three types of performance metrics: expected, high-probability, and deterministic.

² Their overhead can be improved to $O(\sqrt{N})$ if we allowed $O(\sqrt{N})$ client-side storage.

Revisiting the performance metrics for perfectly secure ORAMs/OPRAMs. We consider the following performance bounds for ORAM/OPRAMs:

1. *Expected performance bounds.* Suppose that the original RAM/PRAM runs in (parallel) time T , and the corresponding ORAM/OPRAM runs in *expected* (parallel) time $\chi \cdot T$, then we say that the ORAM/OPRAM satisfies expected simulation overhead (or expected overhead) χ .
2. *High-probability performance bounds.* Suppose that the original RAM/PRAM runs in (parallel) time T , and the corresponding ORAM/OPRAM runs in (parallel) time $\chi \cdot T$ with $1 - \delta$ probability where δ is suitably small (e.g., negligibly small in some security parameter), then, we say that ORAM/OPRAM satisfies simulation overhead (or overhead) χ with probability $1 - \delta$. We stress that the failure probability δ describes the probability that the ORAM/OPRAM exceeds the performance bounds, it does *not* describe security failure since the ORAM/OPRAM is perfectly secure. This is a natural, intermediate notion that is not as stringent as deterministic performance bounds and yet gives a strong guarantee. This notion may permit asymptotically better schemes than insisting on deterministic performance bounds.
3. *Deterministic performance bounds.* Suppose that the original RAM/PRAM runs in (parallel) time T , and the corresponding ORAM/OPRAM runs in (parallel) time $\chi \cdot T$ with probability 1, then we say that the ORAM/OPRAM satisfies deterministic simulation overhead χ .

Asymptotically better constructions for all performance metrics. We show novel perfect ORAM/OPRAM constructions that achieve asymptotical performance improvements across the board.

- First, for expected performance, we overcome the $\log^3 N$ barrier that the literature has been stuck at for the past decade. Our perfect ORAM/OPRAM scheme has $O(\log^3 N / \log \log N)$ expected overhead.
- Second, for high-probability performance bounds, previously, there were no documented schemes with this type of performance bounds to the best of our knowledge. We show new perfectly secure ORAM/OPRAMs that achieve $O(\log^3 N / \log \log N)$ simulation overhead with probability $1 - \text{negl}(N)$.³
- Finally, we construct perfect ORAM/OPRAMs with deterministic, poly-logarithmic simulation overhead. Our result achieves a sub-exponential performance improvement relative to prior art [34].

Our results are summarized in the following theorems, and moreover, Table 1 gives an explicit comparison of our results with prior work.

► **Theorem 1** (Informal: perfect OPRAM with expected performance bounds). *There exists a perfectly secure OPRAM scheme that consumes only $O(1)$ blocks of client private cache and $O(N)$ blocks of server-space; moreover the scheme achieves $O(\log^3 N / \log \log N)$ expected simulation overhead.*

► **Theorem 2** (Informal: perfect OPRAM with high-probability performance bounds). *There exists a perfectly secure OPRAM scheme that consumes only $O(1)$ blocks of client private cache and $O(N)$ blocks of server-space; moreover the scheme achieves $O\left(\frac{1}{\log \log N} \cdot (\log^3 N + \log^2 N \cdot \log^2 \log(1/\delta) + \log N \cdot \log^3 \log(1/\delta))\right)$ simulation overhead with probability $1 - \delta$.*

³ Note that our formal theorem statement gives a parametrized version where the performance failure probability may be any free parameter.

■ **Table 1 Comparison of our results with prior work.** For simplicity, the high-probability bounds are parameterized to $1 - \text{negl}(N)$ failure probability.

		Space	ORAM overhead	OPRAM overhead
Schemes with expected performance bounds	Damgård et al. [17]	$O(N \log N)$	$O(\log^3 N)$	N/A
	Chan et al. [12]	$O(N)$	$O(\log^3 N)$	$O(\log^3 N)$
	This work	$O(N)$	$O(\log^3 N / \log \log N)$	$O(\log^3 N / \log \log N)$
Schemes with high-probability performance bounds	This work	$O(N)$	$O(\log^3 N / \log \log N)$	$O(\log^3 N / \log \log N)$
Schemes with deterministic performance bounds	Raskin et al. [34]	$O(N)$	$O(\sqrt{N} \cdot \frac{\log N}{\log \log N})$	N/A
	This work	$O(N)$	$O(\log^3 N / \log \log N)$	$O(\log^4 N / \log \log N)$

► **Theorem 3** (Informal: perfect OPRAM with deterministic performance bounds). *There exists a perfectly secure ORAM scheme that achieves $O(\log^3 N / \log \log N)$ simulation overhead with probability 1. For the parallel setting: there exists a perfectly secure OPRAM scheme that achieves $O(\log^4 N / \log \log N)$ simulation overhead with probability 1.*

For both the ORAM/OPRAM schemes above, we need only $O(1)$ blocks of client private cache and $O(N)$ blocks of server-space.

1.2 Technical Highlight

Getting an $O(\log^3 N / \log \log N)$ deterministic overhead ORAM. To improve the overhead of perfectly secure ORAMs to $O(\log^3 N / \log \log N)$, our techniques are inspired by the rebalancing trick of Kushilevitz et al. [26] (SODA'12), and yet departs significantly from Kushilevitz et al. Namely, the existing perfect ORAM/OPRAM constructions of Chan et al. [12] consist of an “online fetch phase” and an “offline maintain phase,” the fetch phase takes a logical request (as an input to the ORAM) and answers to the request using a data structure, and then the maintain phase reshuffles the data structure. We observe that the maintain and fetch phases suffer from an imbalance; specifically, the offline maintain phase costs $O(\log^3 N)$ per request whereas the online fetch phase costs only $O(\log^2 N)$. A natural idea is to modify the scheme and rebalance the costs of the offline maintain phase and the online fetch phase, such that both phases would cost only $O(\log^3 N / \log \log N)$. Unfortunately, existing techniques such as Kushilevitz et al. completely fail for rebalancing *perfect* ORAMs/OPRAMs – we defer the technical reasons to the full version [14].

Our starting point is the perfect ORAM construction by Chan et al. [12] in which the maintain phase costs $O(\log^3 N)$ and the fetch phase costs only $O(\log^2 N)$. Specifically, their construction consists of $D = O(\log N)$ number of ORAMs such that except for the last ORAM which stores the actual data blocks, every other ORAM serves as a (recursive) index structure into the next ORAM – for this reason, these D ORAMs are also called D recursion depths; and all of the recursion depths jointly realize an implicit logical index structure that is in fact isomorphic to a binary tree (which has a branching factor of 2).

First, we show how to use a *fat-block* trick to increase the branching factor and hence reduce the number of recursion depths by a $\log \log N$ factor. Specifically, we increase the implicit index structure’s branching factor from 2 (i.e., storing two pointers or position labels in the next recursion depth) to $\log N$. Thus a fat-block is a bundle of *logarithmically* many normal blocks and hence each fat-block can store logarithmically many pointers. While this reduces the recursion depth by a $\log \log N$ factor, the fetch phase now costs a logarithmic factor more per recursion depth (since obviously accessing a fat-block is a logarithmic factor more costly than accessing a normal block).

The primary challenge is to realize the maintain phase such that the amortized *per-depth* maintain-phase cost preserves the asymptotics, despite the fat-block now being logarithmically fatter. To accomplish this we rely on the following two key insights:

1. *Exploit residual randomness.* First, we rely on an elegant observation first made in the PanORAMa work [32] in the context of computationally secure ORAMs. Here we make the same observation for perfectly secure ORAMs. At the core of Chan et al.’s ORAM construction is a data structure called an oblivious “one-time-memory” (OTM). When an OTM is initialized, all elements in it are randomly permuted (and the randomness concealed from the adversary) – note that in our setting, each element is a fat-block. The critical observation is that after accessing a subset of the elements in this OTM data structure, the remaining unvisited elements still appear in a random order. By exploiting such residual randomness, when we would like to build a new OTM consisting of the remaining unvisited elements, we can avoid performing expensive oblivious permutation (which would take time $O(n \log n)$ to sort n elements) and instead rely on linear-time operations.
2. *Exploit sparsity.* In the construction of Chan et al., the D ORAMs at all recursion depths must perform a “coordinated shuffle” operation during the maintain phase. An important step in this coordinated shuffle is (for each recursion depth) to inform the parent depth the locations of its fat-blocks after the reshuffle. In Chan et al., two adjacent recursion depths perform such “communication” through oblivious sorting, thus incurring $O(n \log n)$ cost per-depth to rebuild a data structure of size n .

Our key observation is that the fat-blocks contained in each OTM data structure in each recursion depth are sparsely populated. In fact, most entries in the fat-blocks are irrelevant and only a $1/\log N$ fraction of them are populated. Thus, we employ an oblivious tight compaction [5] to compress away the wasted space, where tight compaction is a degenerated sorting that sorts elements tagged with 0/1 keys. After this compression, the OTM becomes logarithmically smaller and we can apply an oblivious sort.

Finally, we stress that to get an ORAM with *deterministic* performance bounds, we will need to instantiate our ORAM with building blocks that give deterministic performance. We defer the details to later technical sections.

Parallelizing the scheme. To parallelize our ORAM scheme, we encounter several additional technicalities. Some of the core algorithmic building blocks can be parallelized; however, to preserve the asymptotical total work of the sequential versions, the only known parallel counterparts give Las Vegas-type performance bounds. This would be fine if we only wanted an OPRAM with *expected* $O(\log^3 N / \log \log N)$ overhead. However, more work is needed to get an OPRAM whose simulation overhead is $O(\log^3 N / \log \log N)$ *with high probability*. Finally, to get an OPRAM whose simulation overhead holds deterministically, we need to replace some of the oblivious parallel building blocks with ones with deterministic performance bounds – here we lose an extra logarithmic factor in total work. We defer a more detailed exposition of these technicalities to later sections.

2 Technical Overview

We start with an informal and intuitive exposition of our main technical ideas. For simplicity, most of the section describes how to get the sequential ORAM result with *deterministic* $O(\log^3 N / \log \log N)$ simulation overhead. Then, in Section 2.4, we sketch the additional steps and technicalities needed to parallelize the scheme to get the expected, high-probability, and deterministic performance bounds for OPRAM. The full formal details will be deferred to the technical sections later.

2.1 Background on Perfect ORAM

The goal of an ORAM scheme is to simulate to the client a memory array of N blocks, where each block consists of $\Omega(\log N)$ bits. In the simulated memory, each block is indexed by a *logically address* in $\{0, 1, \dots, N - 1\}$, and the client can read or write a block using a logical address. The ORAM is allowed to use client-side storage of only $O(1)$ number of blocks as well as server storage, where the server supports only fetch or store the content of blocks (but no computation). By perfect security, we require that the sequence of accessed blocks on the server (also called the *access pattern*) be identically distributed for any sequence of read/write accesses to the memory simulated by ORAM. Such settings are standard in previous works [17, 12, 34].

In a recent work, Chan et al. [12] propose a perfectly secure ORAM with $O(\log^3 N)$ simulation overhead. At a high level, their scheme is inspired by the hierarchical ORAM paradigm by Goldreich and Ostrovsky [22, 21], but Chan et al. replace the “oblivious hashing” (which has a negligible statistical imperfectness) with perfectly secure data structures (including a “one-time-memory” as well as a “position map”). In this way, they also remove the pseudo-random function (PRF) in Goldreich and Ostrovsky’s construction [22, 21].

2.1.1 Position-based Hierarchical ORAM

First, imagine that the client can store per-block metadata and we will later remove this strong assumption through a non-blackbox recursion technique. Specifically, imagine that the client remembers where exactly each block is residing on the server. In this case, we can construct a perfect ORAM as follows – henceforth this building block is called “position-based ORAM” since we assume that the correct position label for every requested block is known to the client.

Hierarchical levels. The server-side data structure is organized into $\log N + 1$ levels numbered $0, 1, \dots, \log N$ where level i is either 1. *empty*, in which case it stores no blocks; or 2. *full*, in which case the level stores 2^i real blocks plus 2^i dummy blocks in a randomly permuted fashion (we also say that the level has *capacity* 2^i). Each block, whose logical addresses range from 0 to $N - 1$, resides in exactly one of the levels at a random position within the level.

Fetch phase. Every time a block with address `addr` is requested, the client looks up the block’s position. Suppose that the block resides in the j -th position of level ℓ . The client now visits for one block per full level from the server – note that the levels are visited in a fixed order from 0 to $\log N$:

- for level ℓ (i.e., where the desired block resides), the client reads precisely the j -th position to fetch the real block; it then marks the visited position as *visited*;
- for every other level $\ell' \neq \ell$, the client reads a random unvisited dummy block (and marks the corresponding block on the server as visited for obliviousness).

Maintain phase. Every time a block B has been fetched by the client, it updates the block B ’s contents if this is a write request, and then it puts B back to level 0 as an *unvisited* block so that level 0 becomes full. Now, suppose levels $0, 1, \dots, \ell^*$ are all full and either level $\ell^* + 1$ is empty or $\ell^* = \log N$. The client will now merge levels $0, 1, \dots, \ell^*$ into the “target” level $\ell_{\text{tgt}} := \min(\ell^* + 1, \log N)$. This procedure is called “rebuilding” level ℓ_{tgt} . At the end of the rebuild, it marks level ℓ_{tgt} as full and every smaller level as empty.

To merge consecutively full levels into the next empty level (or the largest level), the goal is to implement the following ideal functionality *obliviously*:

1. extract all unvisited *real* blocks to be merged and place them in an array called A ;
2. pad A with dummy blocks to a length of $2 \cdot 2^{\ell_{\text{tgt}}}$ blocks and randomly permute the resulting array.

Chan et al. show how to achieve the above obliviously – even though the client has only $O(1)$ blocks of client storage – through oblivious sorting (using the AKS sorting network [1]). The cost of rebuilding a level of capacity n is dominated by the oblivious sorting on $O(n)$ blocks, which has a cost of $O(n \log n)$.

Note that the above construction guarantees that whenever a real block is accessed, it is moved into a smaller level. Thus, in every level, each real or dummy block is accessed at most once before the level is rebuilt; and this is important for obliviousness. For this reason, later in our technical sections, we name each level in this hierarchy an oblivious “one-time memory”. Note also that *the number of dummies in a level must match the total number of accesses the level can support before it is rebuilt again*.

Additional details about dummy positions. The above description implicitly assumed that for a level the desired block does not reside in, the client is informed of the position of a random unvisited dummy block. If the client does not store this information locally, it can construct a (per-level) metadata array M on the server every time a level is rebuilt. When a block is being requested, the client can sequentially scan the metadata array at *every* level (including the level where the desired block resides) to discover the location of the next unvisited dummy (residing at a random unvisited location in the level).

As Chan et al. show, such a dummy metadata array can be constructed with $O(n \log n)$ overhead using oblivious sorting too, at the same time when a level of capacity n is rebuilt.

Overhead. Summarizing, in the position-based ORAM, after every 2^ℓ requests, the level ℓ will be rebuilt, paying $O(2^\ell \cdot \log(2^\ell))$ cost. Amortizing the total cost over the sequence of requests, it is not difficult to see that the average cost per request is $O(\log^2 N)$.

2.1.2 Recursive Position Map

So far we have assumed that the client *magically* remembers where exactly each block is residing on the server. To remove this assumption, Chan et al. propose to recursively store the blocks’ position labels in smaller ORAMs until the ORAM’s size becomes constant, resulting in $D = O(\log N)$ ORAMs henceforth denoted $\text{ORAM}_0, \text{ORAM}_1, \dots, \text{ORAM}_D$ respectively, where ORAM_i stores the position labels of all blocks in ORAM_{i+1} for $i \in \{0, 1, \dots, D\}$. We often call ORAM_D the “data ORAM” and every other ORAM a “metadata ORAM”; we also refer to the index i as the *depth* of ORAM_i . Now, suppose that each block can store $\Omega(\log N)$ bits of information, such that we can pack the position labels of at least 2 blocks into a single block. In this case, each ORAM_i is twice smaller in capacity than ORAM_{i+1} and thus ORAM_0 would be of $O(1)$ size – thus operations to ORAM_0 can be supported trivially by scanning through the whole ORAM_0 consuming only constant cost, and the total space is still $O(N)$.

As Chan et al. show, in the hierarchical ORAM context such a recursion idea does not work in a straightforward blackbox manner,⁴ but needs a special “coordinated rebuild” technique which we now explain. Henceforth, suppose that each block’s logical address addr

⁴ Roughly speaking, it is because each logical access on ORAM_{i+1} would have incurred too many accesses on ORAM_i , and then the cost of such recursion would have been too expensive.

is $\log N$ bits long, and we use the notation $\text{addr}^{(d)}$ to denote the address addr , written in binary format, truncated to the most significant d bits.

- *Fetch phase (straightforward)*: To fetch a block at some logical address addr , the client looks up logical address $\text{addr}^{(d)}$ in each ORAM_d for $d = 0, 1, \dots, D$ sequentially. Since the block at logical address $\text{addr}^{(d)}$ in ORAM_d stores the position labels for the two blocks at logical addresses $\text{addr}^{(d)} \parallel 0$ and $\text{addr}^{(d)} \parallel 1$ in ORAM_{d+1} , the client is always able to find out the position of the block in the next recursion depth before it performs a lookup there.
- *Maintain phase (coordinated rebuild)*: The maintain phase needs special treatment such that the rebuilds at all recursion depths are coordinated. Specifically, whenever the data ORAM_D is rebuilding the level ℓ , each other recursion depth ORAM_d would be rebuilding level $\min(\ell, d)$ in a coordinated fashion – note that each ORAM_d has only d levels. The main goal of the coordination is for each ORAM_d to pass the blocks' updated position labels back to the parent depth ORAM_{d-1} . More specifically, recall that when ORAM_d rebuilds a level ℓ , all real blocks in the level would now be placed at a new random position. When these new positions have been decided, ORAM_d must inform the corresponding metadata blocks in ORAM_{d-1} the new position labels. The coordinated rebuild is possible due to the following invariant which is not hard to observe (recall that $\text{addr}^{(d)}$ is the block that stores the position labels for the block $\text{addr}^{(d+1)}$ in ORAM_{d+1}):
 - For every addr , the block at address $\text{addr}^{(d)}$ in ORAM_d is always stored at a smaller or equal level relative to the level of the block at address $\text{addr}^{(d+1)}$ in ORAM_{d+1} .
 Chan et al. show how to use oblivious sorting to perform a coordinated rebuild, paying $O(n \log n)$ to pass the new position labels of level- ℓ in ORAM_d to the parent ORAM_{d-1} where $n = 2^\ell$ is the level's capacity.

2.1.3 Analysis

It is not hard to see that the entire fetch phase consumes $O(\log^2 N)$ overhead where one $\log N$ factor comes from the number of levels within each recursion depth, and another comes from the number of recursion depths. The maintain phase, on the other hand, consumes $O(\log^3 N)$ amortized cost where one logarithmic factor arises from the number of recursion depths, one arises from the number of levels within each depth, and the last one stems from the additional logarithmic factor in oblivious sorting.

To asymptotically improve the overhead, one promising idea is to somehow balance the fetch and maintain phases. This idea has been explored in computationally secure ORAMs first by Kushilevitz et al. [26] and later improved in subsequent works [11]. Unfortunately as we explain the full version [14], Kushilevitz et al.'s rebalancing trick is not compatible with known perfect ORAMs. Thus we need fundamentally new techniques for realizing such a rebalancing idea.

2.2 Building Blocks

Before we introduce our new algorithms, we describe two important oblivious algorithms as building blocks that were discovered in very recent works [33, 5].

Tight compaction. Tight compaction is the following task: given an input array containing m balls where each ball is tagged with a bit indicating whether it is real or dummy, produce an output array containing also m balls such that all real balls in the input appear in the front and all dummies appear at the end.

In a very recent work called OptORAMa [5], the authors show how to accomplish tight compaction obliviously in $O(m)$ time. Their algorithm can be expressed as a linear-sized circuit (of constant fan-in and fan-out), consisting only of boolean gates and swap gates, where a boolean gate can perform boolean computations on two input bits; and a swap gate takes in a bit and two balls, and decides to either swap or not swap the two balls.

Intersperse. The same work OptORAMa described another randomized oblivious algorithm called “intersperse”, which accomplishes the following task in deterministic linear time: given two randomly shuffled input arrays \mathbf{I} and \mathbf{I}' (where the permutations used in the shuffles are hidden from the adversary), create an output array of length $|\mathbf{I}| + |\mathbf{I}'|$ that contains all elements from the two input arrays, and moreover, all elements in the output array are randomly shuffled in the view of the adversary.

2.3 A New Rebalancing Trick for Perfectly Secure ORAMs

We propose new techniques for instantiating such a rebalancing trick. Our idea is to introduce a notion called a fat-block. A fat-block is a bundle of $\chi := \log N$ normal blocks; thus to access a fat-block requires paying $\chi = \log N$ cost.

Imagine that in each metadata ORAM, the atomic unit of storage is a fat-block (rather than a normal block). Since each fat-block can pack $\chi = \log N$ position labels, the depth of the recursion is now $\log_\chi N = \log N / \log \log N$, i.e., a $\log \log N$ factor smaller than before (see Section 2.1.2). More concretely, a metadata ORAM ORAM_d at depth d stores a total of χ^d metadata fat-blocks – for the time being we assume that N is a power of χ for simplicity, and let $D := \log_\chi N + 1$ be the number of recursion depths such that the total storage is still $O(N)$ blocks (our idea can be generalized to the case when N is not a power of χ). Within each ORAM_d , as before, we have a total of $d \log \chi + 1$ levels where each level ℓ can store 2^ℓ fat-blocks.

It is not hard to see that the fetch phase would now incur $O(\log^3 N / \log \log N)$ cost across all recursion depths – in comparison with before, the extra $\log N$ factor arises from the cost of reading a fat-block, and the $\log \log N$ factor saving comes from the $\log \log N$ saving in recursion depth.

Our hope is that now with the smaller recursion depth, we can accomplish the maintain phase in amortized $O(\log^3 N / \log \log N)$ cost. Recall that each level ℓ in a metadata ORAM_d now contains 2^ℓ fat-blocks. The crux is to rebuild a level containing 2^ℓ fat-blocks in cost that is linear in the level’s total size, that is, $2^\ell \cdot \chi$. Note that if we naively used oblivious sorting on fat-blocks (like in Section 2.1.1) to accomplish this, the cost would have been $2^\ell \cdot \chi \cdot \log(2^\ell)$ which is more expensive than previous scheme and undesirable.

To resolve this challenge, the following two insights are critical:

- *Sparsity:* First, observe that each level in a metadata ORAM is *sparsely populated*: although the entire level, say, level ℓ , has the capacity to store $2^\ell \cdot \chi$ position labels, the level is rebuilt after every 2^ℓ requests. Thus in fact only 2^ℓ of these position label entries are populated.
- *Residual randomness:* The second important observation is that the unvisited fat-blocks contained in any level appear in a random order where the randomness of the permutation is hidden from the adversary – note that a similar observation was first made in the PanORAMa work [32] by Patel et al.

More specifically, suppose that to start with, a level contains n fat-blocks including some reals and some dummies, and all of these n fat-blocks have been randomly permuted (where the randomness of the permutation is hidden from the adversary). As the client

visits fat-blocks in a level, the adversary learns which blocks are visited. Now, among all the unvisited blocks, there are both real and dummy blocks and all these blocks are equally likely to appear in any order w.r.t. the adversary's view.

We now explain how to rely on the above insights to rebuild a level containing $n = 2^\ell$ fat-blocks in $O(n \cdot \chi)$ time – note that at most half of these fat-blocks are real, and the remaining are dummy. From Section 2.1.2, we learned that to rebuild a level containing n fat-blocks, it suffices to realize the following functionality obliviously:

1. *Merge*. The first step of the rebuild is to merge consecutively full levels into the next empty level (or the largest level). After this merge step, this new level is marked full and every smaller level is marked empty.
2. *Permute*. After the above merge step, the resulting array containing n fat-blocks must be randomly permuted (and their positions after the permutation will then be passed to the parent depth next).
3. *Update*. After the permutation step, each real fat-block in the level (at a recursion depth d) whose logical address is `addr` must receive up to χ updated positions from the child recursion depth, i.e., the fat-block at logical address `addr` wants to learn where the fat-blocks at logical addresses `addr||0`, `addr||1`, \dots , `addr||(\chi - 1)` newly reside in the child depth $d + 1$.
4. *Create dummy metadata*. Finally, create a dummy metadata array to accompany this level: the dummy metadata array containing n entries where each entry is $O(\log N)$ bits (note that an entry is a normal block, not a fat-block). This array should store the positions of all *dummy* fat-blocks contained in the level in a randomly permuted order.

Realizing “merge + permute”. We first explain how to accomplish the “merge + permute” steps. For simplicity we focus on explaining the case where consecutive full levels are merged into the next empty level (since it would be fine if the merging into the largest level *alone* is done naïvely using oblivious sort on all fat-blocks). Here it is important to rely on the residual randomness property mentioned earlier. Suppose the levels to be merged contain $1, 2, 4, 8, \dots, n/2$ fat-blocks respectively. Recall that in all of these levels to be merged, the unvisited blocks appear in a random order w.r.t. the adversary's view. Thus, we can simply do $O(\log n)$ cascading merges using **Intersperse** (see Section 2.2), every time merging two arrays each containing 2^i fat-blocks into an array containing 2^{i+1} fat-blocks, and the overall cost is $O(n)$.

Realizing “update”. At this moment, let us not view the level as an array of n fat-blocks any more, but as an array of $O(n \cdot \chi)$ position entries. For realizing the “update” step in $O(n \cdot \chi)$ overhead, the key insight is to exploit the sparsity.

Recall that the problem we need to solve boils down to the following. There is a destination array D consisting $O(n \cdot \chi)$ position entries among which $O(n)$ entries are going to be updated, and we override terminologies “real” and “dummy” (opposed to previously denoted real or dummy fat-blocks) and say the to-be-updated $O(n)$ entries are *real* while all remaining entries are *dummy*. Additionally, there is a source array S consisting of $O(n)$ entries (which can be real or dummy). In both the source S and the destination D , each real entry is of the form (k, v) where k denotes a key and v denotes a payload value; further, in each array D or S , every real entry must have a distinct key. Now, we would like to route each real entry $(k, v) \in S$ to the corresponding entry with the same key in the destination array D .

Exploiting the sparsity in the problem definition, we devise the following algorithm where an important building block is linear-time *oblivious tight compaction* (see Section 2.2).

First, we rely on oblivious tight compaction to compact the destination array D , resulting in a compacted array \tilde{D} consisting of only $O(n)$ entries. Moreover, recall that oblivious tight compaction can be viewed as a *circuit* consisting of boolean gates and swap gates. When we compact the destination array D , each swap gate remembers the routing decision since later it will be useful to run this circuit in the reverse direction. After the compaction, we can now afford to pay the cost of oblivious sorting. Hence, each entry in the source S can route itself to each entry in the compacted destination \tilde{D} – this can be accomplished through a standard technique called oblivious routing [9, 13], which has a cost of $O(n \log n)$. Now, by running the aforementioned tight compaction circuit in the reverse direction, we can route each element of the compacted destination \tilde{D} back into the original destination array D .

It is not difficult to see that the above steps require only $O(n \cdot (\chi + \log n)) = O(n \log N)$ cost.

Obliviously create dummy metadata array. Finally, obliviously creating the dummy metadata array is easy: this can be accomplished by writing down $O(\log N)$ bits of metadata per fat-block, and then performing a combination of oblivious random permutation and oblivious sort on the resulting metadata array. To get deterministic simulation overhead for our ORAM, we will need to use an oblivious random permutation algorithm with deterministic performance bounds – fortunately, this is known due to Asharov et al. [5].

Summary. In the above, we took care to make sure that all oblivious building blocks used give deterministic performance bounds. At this moment, we derive a perfect ORAM scheme with $O(\log^3 N / \log \log N)$ *deterministic* overhead. This warmup result already improves upon Chan et al. [12] who showed $O(\log^3 N)$ *expected* simulation overhead, as well as Raskin [34] who showed roughly \sqrt{N} *deterministic* simulation overhead.

2.4 Parallelizing the Scheme

So far, for simplicity we have focused on the sequential case. To obtain our OPRAM result, we need to make the above scheme parallel. To this end, we will rely on the OPRAM techniques by Chan et al. [12], that is, the fetch phase is still performed sequentially, but the maintain phase is realized using parallel and oblivious sorting, tight compaction, and random permutation. One main challenge here is that we will need a parallel counterpart for the **Intersperse** algorithm. Note that Asharov et al. [5]’s **Intersperse** algorithm gives deterministic performance bounds and perfect security, but is inherently sequential. We devise a new parallel **Intersperse** algorithm that preserves the same asymptotical total work as the sequential version of Asharov et al. [5] (for fat-blocks); however, the algorithm gives Las-Vegas-type performance. This is fine if we only aim for an $O(\log^3 / \log \log N)$ -*expected*-overhead OPRAM, but it will not work if we want matching high probability performance bounds. Observe that our parallel **Intersperse** algorithm’s performance bounds are more concentrated around the mean for larger input sizes. In our OPRAM, the **Intersperse** algorithm needs to be applied to input arrays containing $1, 2, 4, \dots, N / \log N$ fat blocks. Therefore, to get $O(\log^3 / \log \log N)$ overhead with $1 - \text{negl}(N)$ probability, our idea is to apply the Las-Vegas **Intersperse** algorithm only to sufficiently large instances, and for small instances, we apply a variant which gives deterministic performance bounds but is a logarithmic factor more expensive. We prove that this bi-modal approach gives an OPRAM scheme whose simulation overhead is $O(\log^3 / \log \log N)$ with $1 - \text{negl}(N)$ probability.

Finally, to get an OPRAM with deterministic performance bounds, we need to replace the **Intersperse** building block entirely with one that gives deterministic performance bounds, but is a logarithmic factor more expensive. This explains why our OPRAM with deterministic performance bounds has an extra logarithmic factor.

2.5 Open Questions

Our paper raises several interesting open questions. First, for constructions with deterministic performance bounds, currently our OPRAM scheme has a logarithmic factor higher simulation overhead than the sequential ORAM – this extra logarithmic factor stems from the parallel perfect oblivious permutation building block we use [3]. One open question is whether we can get rid of this extra logarithmic factor.

In our paper, we adopt the standard notion of simulation overhead originally defined by Goldreich and Ostrovsky [22, 21]. This standard notion is naturally *amortized* over the multiple steps of the RAM/PRAM, since it takes the ratio of the total runtime of the ORAM/OPRAM and that of the original RAM/PRAM. In comparison, some earlier works consider an even stronger notion, often referred to as *worst-case* overhead [31, 36, 26, 11, 34]: a worst-case overhead of χ requires that *every* (parallel) step of the original RAM/PRAM is simulated by at most χ steps in the compiled ORAM/OPRAM. While some previous ORAM/OPRAM constructions are amenable to a standard deamortization trick [31, 26] to achieve worst-case overhead that match the amortized, our constructions are not compatible with standard deamortization techniques [26] for a similar reason why PanORAMa [32] and OptORAMa [5] are also not compatible with standard deamortization. It is due to the “residual randomness” technique: after the residual randomness of an element is used to facilitate a random permutation, if the same element is then accessed due to the deamortization, then such residual randomness is revealed and the random permutation is no longer random in the adversarial view, which is insecure. An interesting future direction is whether we can achieve worst-case overhead that match the amortized bounds claimed in our paper.

Our paper focuses on the *theoretical* understanding of the asymptotic complexity of perfectly secure ORAMs/OPRAMs. Our asymptotic constant is huge due to using AKS sorting network [1] and the linear tight compaction [6]. The constants are similar to earlier works [17, 12] that also use AKS sorting. A standard way to replace the huge constant with another logarithmic factor is to replace both AKS sorting and tight compaction with bitonic sorter [8] (notice that the standard bitonic sort takes $O(n \log^2 n)$ work, but to sort only 0/1 elements, i.e. tight compaction, a bitonic sort augmented with counting takes only $O(n \log n)$ work). An interesting question is whether we can achieve the performance bounds claimed in this paper, but without the use of expander graphs with large constants.

Last but not the least, for perfectly secure ORAM/OPRAM (and in fact even for statistically secure ORAM/OPRAM), we still do not have matching upper- and lower-bounds. Therefore, a more challenging direction is to close this obvious gap in our understanding.

2.6 Roadmap of Subsequent Formal Sections

In the technical sections, we formalize the blueprint described in this section. Our formal description is modularized which will facilitate formal analysis and proofs. Moreover, in our formal sections we will directly present the OPRAM result (since the sequential ORAM is a special case of the more general OPRAM result).

3 Preliminaries

3.1 Definitions

3.1.1 Parallel Random-Access Machines

We review the concepts of a parallel random-access machine (PRAM) and an oblivious parallel random-access machine (OPRAM). The definitions in this section are borrowed from Chan et al. [12]. Although we give definitions only for the parallel case, we point out that this is without loss of generality, since a sequential RAM can be thought of as a special case PRAM with one CPU.

Parallel Random-Access Machine (PRAM). A *parallel random-access machine* consists of a set of CPUs and a shared memory denoted by `mem` indexed by the address space $\{0, 1, \dots, N - 1\}$, where N is a power of 2. In this paper, we refer to each memory word also as a *block*, which is at least $w = \Omega(\log N)$ bits long.

We consider a PRAM model where the number of CPUs is fixed to be some parameter $m \leq N^5$. Each CPU has a state that stores $O(1)$ blocks. In each step, each CPU executes a next instruction circuit denoted Π , and then interacts with memory. Circuit Π can perform word-level operations including addition, subtraction, and bit-wise boolean operations in unit time and then updates the CPU state. Further, CPUs interact with memory through request instructions $\vec{I}^{(t)} := (I_i^{(t)} : i \in [m])$. Specifically, at time step t , CPU i 's instruction is of the form $I_i^{(t)} := (\text{read}, \text{addr})$, or $I_i^{(t)} := (\text{write}, \text{addr}, \text{data})$ where the operation is performed on the memory block with address `addr` and the block content `data`.

If $I_i^{(t)} = (\text{read}, \text{addr})$ then the CPU i should receive the contents of `mem[addr]` at the beginning of time step t . Else if $I_i^{(t)} = (\text{write}, \text{addr}, \text{data})$, CPU i should still receive the contents of `mem[addr]` at the beginning of time step t ; further, at the end of step t , the contents of `mem[addr]` should be updated to `data`.

Write conflict resolution. By definition, multiple read operations can be executed concurrently with other operations even if they visit the same address. However, if multiple concurrent write operations visit the same address, a conflict resolution rule will be necessary for our PRAM to be well-defined. In this paper, we assume the following:

- The original PRAM supports concurrent reads and concurrent writes (CRCW) with an arbitrary, parametrizable rule for write conflict resolution. In other words, there exists some priority rule to determine which write operation takes effect if there are multiple concurrent writes in some time step t .
- Our compiled, oblivious PRAM (defined below) is a “concurrent read, exclusive write” PRAM (CREW). In other words, our OPRAM algorithm must ensure that there are no concurrent writes at any time.

CPU-to-CPU communication. In the remainder of the paper, we sometimes describe our algorithms using CPU-to-CPU communication. For our OPRAM algorithm to be oblivious, the inter-CPU communication pattern must be oblivious too. We stress that such inter-CPU communication can be emulated using shared memory reads and writes. Therefore, when we express our performance metrics, we assume that all inter-CPU communication is implemented with shared memory reads and writes.

⁵ If $N < m$, the oblivious simulation can be achieved by assigning at most one address to each CPU and then performing oblivious routing [9], which takes only $O(\log m)$ overhead.

Additional assumptions and notations. Henceforth, we assume that *each CPU can only store $O(1)$ memory blocks*. Further, we assume for simplicity that the runtime T of the PRAM is *fixed a priori and publicly known*. Therefore, we can consider a PRAM to be parametrized by the following tuple

$$\text{PRAM} := (\Pi, N, T, m),$$

where Π denotes the next instruction circuit, N denotes the total memory size (in terms of number of blocks), T denotes the PRAM's total runtime, and m denotes the number of CPUs.

Finally, in this paper, we consider PRAMs that are *stateful* and can evaluate a sequence of inputs, carrying states in between, where each input can be stored in a single memory block.

3.1.2 Oblivious Parallel Random-Access Machines

An OPRAM is a (randomized) PRAM with certain security properties, i.e., its access patterns leak no information about the inputs to the PRAM.

Randomized PRAM. A *randomized PRAM* is a PRAM where the CPUs are allowed to generate private random numbers. Concretely, we assume that the next instruction circuit Π can sample a uniform random number from $[a]$ for any positive integer $a \leq 2^w$ in unit time (recall that w is the memory word size in bits), where the assumption is needed by the oblivious random permutation (e.g., [3]). For simplicity, we assume that a randomized PRAM has a priori known, deterministic runtime, and that the CPU activation pattern in each time step is also fixed a priori and publicly known.

Memory access patterns. Given a PRAM program denoted PRAM and a sequence inp of inputs, we define the notation $\text{Addresses}[\text{PRAM}](\text{inp})$ as follows:

- Let T be the total number of parallel steps that PRAM takes to evaluate inputs inp .
- Let $A_t := (\text{addr}_1^t, \text{addr}_2^t, \dots, \text{addr}_m^t)$ be the list of addresses such that the i -th CPU accesses memory address addr_i^t in time step t .
- We define $\text{Addresses}[\text{PRAM}](\text{inp})$ to be the random object $[A_t]_{t \in [T]}$.

Oblivious PRAM (OPRAM). We say that a PRAM is *perfectly oblivious*, iff for any two input sequences inp_0 and inp_1 of equal length, it holds that the following distributions are identically distributed (where \equiv denotes identically distributed):

$$\text{Addresses}[\text{PRAM}](\text{inp}_0) \equiv \text{Addresses}[\text{PRAM}](\text{inp}_1)$$

We remark that for statistical and computational security, some earlier works [11, 13] presented an adaptive, composable security notion. The perfectly oblivious counterpart of their adaptive, composable notion is equivalent to our notion defined above. In particular, our notion implies security against an adaptive adversary who might choose the input sequence inp adaptively over time after having observed partial access patterns of PRAM.

We say that OPRAM is a *perfectly oblivious simulation* of PRAM iff OPRAM is perfectly oblivious, and moreover $\text{OPRAM}(\text{inp})$ is identically distributed as $\text{PRAM}(\text{inp})$ for any input inp .

Metrics. We will use the standard notion of *simulation overhead* to characterize an OPRAM’s performance [22, 21, 9]. If a PRAM that consumes m CPUs and completes in T parallel steps can be obliviously simulated by an OPRAM that completes in $\gamma \cdot T$ steps also with m CPUs, then we say that the simulation overhead is γ . Moreover, supposing that the OPRAM is randomized (by a random tape that is independent from the PRAM), and letting the OPRAM completes in T' steps with m CPUs where T' is a random variable, we say that the *expected* simulation overhead is γ if $\mathbb{E}[T'] = \gamma T$, and we say that the simulation overhead is γ *with probability* $1 - \delta$ if $\Pr[T' \leq \gamma T] \geq 1 - \delta$. We additionally say the simulation overhead γ is *deterministic* if it is γ with probability 1, which coincides with the standard simulation overhead.

More generally, suppose that an ample (i.e., unbounded) number of CPUs are available: in this case if algorithm can be completed in T parallel steps consuming m_1, m_2, \dots, m_T CPUs in each step respectively, then we say that the algorithm can be completed in T *depth* and $W := \sum_{t \in [T]} m_t$ *total work*. Similar to that of simulation overhead, when the total work and depth are random variables, we quantify the total work and depth using expected, with probability, or deterministic, where deterministic is sometimes omitted.

Therefore, for an OPRAM, if the original PRAM (taking T parallel steps and using m CPUs) can be obliviously simulated in W' total work and $T' = O(W'/m)$ depth then the OPRAM has simulation overhead W'/Tm .

Oblivious simulation of a non-reactive functionality. For defining the security of intermediate building blocks, we now define what it means to *obliviously realize* a non-reactive functionality. Let $\mathcal{F} : \{0, 1\}^* \rightarrow \{0, 1\}^*$ be a possibly randomized functionality. We say that $M_{\mathcal{F}}$ is a perfect oblivious simulation (or oblivious simulation for short) of \mathcal{F} with leakage \mathcal{L} , iff there exists a simulator Sim , such that for every input $x \in \{0, 1\}^*$, the following real-world and ideal-world distributions are identical:

- *Real world:* execute $M_{\mathcal{F}}(x)$ and let y be the output and Addr be the memory access patterns; output (y, Addr) ;
- *Ideal world:* output $(\mathcal{F}(x), \text{Sim}(\mathcal{L}(x)))$.

For simplicity, if the leakage function $\mathcal{L}(x) = |x|$, we often say that $M_{\mathcal{F}}$ is a perfect oblivious simulation of \mathcal{F} (omitting the leakage function) for short.

Modeling input assumptions. Some of our building blocks provide perfect obliviousness only if the input array is randomly shuffled and the corresponding randomness concealed. More formally, suppose that a machine $M(A, x)$ and a functionality $\mathcal{F}(A, x)$ both take in an array $A \in D^n$ where $D \in \{0, 1\}^\ell$ as input and possibly an additional input $x \in \{0, 1\}^*$. Formally, we say that “the machine M is a perfectly oblivious simulation of the functionality \mathcal{F} with leakage \mathcal{L} *assuming that the input array A is randomly shuffled*”, iff for every $A \in D^n$ and every $x \in \{0, 1\}^*$, the following real-world and ideal-world distributions are identical:

- *Real world:* randomly shuffle the array A and obtain A' , execute $M_{\mathcal{F}}(A', x)$ and let y be the output and Addr be the memory access patterns; output (y, Addr) ;
- *Ideal world:* output $(\mathcal{F}(A, x), \text{Sim}(\ell, \mathcal{L}(A, x)))$.

Note that the above definition considers only a single input array A , but there is a natural generalization for algorithms that take two or more input arrays – in this case we may require that some or all of these input arrays be randomly shuffled to achieve perfect obliviousness.

3.2 Oblivious Algorithm Building Blocks

We describe some algorithmic building blocks. Unless otherwise noted, for algorithms that operate on arrays of n elements, we always assume that a single memory word is wide enough to store the index of each element within the array, i.e., $w \geq \log n$ where w is the bit-width of each PRAM word. We typically use the following notation: let B denote the bit-width of each element, and let $\beta := \lceil B/w \rceil$ denote the number of memory words it takes to store each element.

3.2.1 Oblivious Sort

Oblivious sorting can be accomplished through a sorting network such as the famous construction by Ajtai, Komlós, and Szemerédi [1]. We restate this result in the context of PRAM algorithms:

► **Theorem 4** (Oblivious sorting [1]). *There exists a deterministic, oblivious algorithm that sorts an array of n elements consuming $O(\beta \cdot n \log n)$ total work and $O(\log n)$ depth where $\beta \geq 1$ denotes the number of memory words it takes to represent each element.*

3.2.2 Oblivious Random Permutation

Let ORP be an algorithm that upon receiving an input array X , outputs a permutation of X . Let $\mathcal{F}_{\text{perm}}$ denote an ideal functionality that upon receiving the input array X , outputs a perfectly random permutation of X . We say that ORP is a *perfectly oblivious* random permutation, iff it is a perfect oblivious simulation of the functionality $\mathcal{F}_{\text{perm}}$. Recall that for any integer $m \in [n]$, each CPU of the PRAM can sample an integer uniformly at random from $[m]$ in unit time.

Sequential ORP algorithm with deterministic performance bounds. Recently, a sequential oblivious algorithm is developed to perform such permutation in $O(n \log n)$ total work [5, Section 6.4].

► **Theorem 5** (A sequential ORP algorithm [5]). *Let $\beta \geq 1$ denote the number of memory words it takes to represent each element. There exists an oblivious random permutation construction that completes in deterministic $O(\beta \cdot n \log n)$ total work.*

Parallel ORP algorithm with deterministic performance bounds. Alonso and Schott [3] construct a parallel random permutation algorithm that takes $O(n \log^2 n)$ total work and $O(\log^2 n)$ depth to randomly permute n elements. Although achieving obliviousness was not a goal of their paper, it turns out that their algorithm is also perfectly oblivious, giving rise to the following theorem:

► **Theorem 6** (Alonso-Schott ORP). *There is a perfectly oblivious algorithm that permutes an array of n elements in deterministic $O(\beta \cdot n \log n + n \log^2 n)$ total work and $O(\log^2 n)$ depth where $\beta \geq 1$ denotes the number of memory words for representing each element.*

Parallel, Las Vegas ORP algorithm. A few recent works [10, 5] describe another perfectly oblivious random permutation algorithm which is asymptotically more efficient but the algorithm is Las Vegas, i.e., the algorithm satisfies perfect obliviousness and correctness, but

with a small probability the algorithm may run longer than the stated bound.⁶ Below, we restate this result in the form that we desire in this paper – the specific theorem stated below arises from the improved analysis of Asharov et al. [5, Theorem 4.3] where we replace the “quadratic oblivious random permutation” with Alonso-Schott ORP; for the performance bounds, we state an *expected* version and a *high-probability* version. Notice that the replaced ORP incurs an $O(\log^2 n)$ depth with probability $o(1)$ but not in expectation.

► **Theorem 7** (A Las Vegas ORP algorithm). *Let $\beta \geq 1$ denote the number of memory words it takes to represent each element. There exists a Las Vegas perfectly oblivious random permutation construction that completes in expected $O(\beta \cdot n \log n)$ total work and expected $O(\log n)$ depth. Furthermore, except with $n^{-\Omega(\sqrt{n})}$ probability, the algorithm completes in $O(\beta \cdot n \log n)$ total work and $O(\log^2 n)$ depth.*

Note that the above theorem gives a high-probability performance bound for sufficiently large n . Later in our OPRAM construction, we will adopt ORP for problems of different sizes – we will use Theorem 7 for sufficiently large instances and use Theorem 6 for small instances.

3.2.3 Oblivious Routing

Oblivious routing [9] is the following primitive where n source CPUs wish to route data to n' destination CPUs based on the key.

- *Inputs:* The inputs contain two arrays: 1) a source array $\text{src} := \{(k_i, v_i)\}_{i \in [n]}$ where each element is a (key, value) pair or a dummy element denoted (\perp, \perp) ; and 2) a destination array $\text{dst} := \{k'_i\}_{i \in [n']}$ containing a list of (possibly dummy) keys. We assume that each (non-dummy) key appears no more than C times in the src array where $C = O(1)$ is a known constant; however, each (non-dummy) key can appear any number of times in dst .
- *Outputs:* We would like to output an array $\text{Out} := \{v'_{i,j}\}_{i \in [n'], j \in [C]}$ where $(v'_{i,1}, \dots, v'_{i,C})$ contains all the values contained in src whose keys match k'_i (padded with \perp to length C).

► **Theorem 8** (Oblivious routing [9, 13, 10]). *There exists a perfectly oblivious routing algorithm that accomplishes the above task in $O(\log(n + n'))$ depth and $O(\beta \cdot (n + n') \log(n + n'))$ total work where $\beta \geq 1$ denotes the number of words it takes to represent each element.*

3.2.4 Oblivious Tight Compaction

As mentioned in Section 2.2, tight compaction is the following task: given an input array containing n elements where each element is tagged with a bit indicating whether it is real or dummy, produce an output array containing also n elements such that all real elements in the input appear in the front and all dummies appear at the end. We will use the parallel oblivious tight compaction of Asharov et al. [6] running in linear work and logarithmic depth.

► **Theorem 9** (Oblivious tight compaction [6]). *There exists a deterministic, oblivious tight compaction algorithm that compacts an array of n elements in total work $O(\beta \cdot n)$ and depth $O(\log n)$ where $\beta \geq 1$ denotes the number of words it takes to represent each element.*

⁶ Using more depth but only unbiased random bits, Czumaj [16] shows a Las Vegas switching network to achieve the same abstraction.

We point out that Asharov et al.’s oblivious parallel compaction algorithm [6] works in the so-called *indivisibility* model, that is, the payload of the elements are moved around as opaque strings.

3.3 Parallel Intersperse

Oblivious intersperse is an abstraction that which can be used to mix two input arrays such that the mixing is uniformly at random in the adversarial view. The abstraction was originated in PanORAMa [32] and then formally defined and realized in OptORAMa [5]. In this section, we define intersperse for completeness and then state the sequential and parallel realizations that run in expected, high probability, or deterministic performance bounds.

3.3.1 Definition

Informally, in the definition of OptORAMa, the **Intersperse** algorithm receives the concatenation of the two input arrays and only the sum of their lengths is public but not each array’s individual length where each input array is shuffled uniformly at random, and then **Intersperse** is required to output a uniformly shuffled array consisting of all input elements. More specifically, **Intersperse** has the following syntax.

- **Input.** The concatenated array $\mathbf{I}_0\|\mathbf{I}_1$, and two integers $n_0 := |\mathbf{I}_0|$ and $n_1 := |\mathbf{I}_1|$.
- **Output.** An array \mathbf{B} of size $n = n_0 + n_1$ that contains all elements of \mathbf{I}_0 and \mathbf{I}_1 . Each position in \mathbf{B} will hold an element from either \mathbf{I}_0 or \mathbf{I}_1 , chosen uniformly at random and the choices are concealed from the adversary.

We now define the security notion required for **Intersperse**. We require that when we run **Intersperse** on two input arrays \mathbf{I}_0 and \mathbf{I}_1 that are both randomly shuffled (based on a secret permutation), the resulting array will be randomly shuffled (based on a secret permutation) too. More formally stated, we require that **Intersperse** is a perfect oblivious simulation of the following $\mathcal{F}_{\text{shuffle}}(\mathbf{I}_0, \mathbf{I}_1)$ functionality provided that the two input arrays are randomly shuffled. Henceforth we assume that the bit-width of each element in the input arrays is a publicly known parameter that the scheme is implicitly parametrized with.

$\mathcal{F}_{\text{shuffle}}(\mathbf{I}_0\|\mathbf{I}_1, n_0, n_1)$:

1. Choose a permutation $\pi : [n] \rightarrow [n]$ uniformly at random where $n := |\mathbf{I}_0| + |\mathbf{I}_1|$.
2. Let \mathbf{I} be the concatenation of $\mathbf{I}_0\|\mathbf{I}_1$.
3. Initialize an array \mathbf{B} of size n . Assign $\mathbf{B}[i] := \mathbf{I}[\pi(i)]$ for every $i = 1, \dots, n$.
4. **Output:** The array \mathbf{B} .

The recent work OptORAMa [5, Claim 6.3] showed how to construct an **Intersperse** algorithm in linear time, i.e., $O(n)$; however, their algorithm is inherently sequential (see the following warmup). A manuscript by Asharov et al. [7] considered how to devise a parallel version of **Intersperse** in an attempt to make OptORAMa parallel; but their parallel **Intersperse** algorithm achieves only statistical security.⁷ Later in this section we will describe a variant of the parallel intersperse that is perfectly secure but consumes more cost.

⁷ The algorithm of Asharov et al. [7] may abort and fail with a negligible probability, and such negligible event reveals some information about the input (n, n_0) so that it is only statistically secure.

3.3.2 Warmup: A Sequential, Linear-Work Intersperse Algorithm

Asharov et al. [5] used the following method to construct a sequential **Intersperse** algorithm:

1. First, initialize an array Aux of size n that has n_0 zeros and n_1 ones, where the zeros' positions are chosen uniformly at random (and the remaining positions are ones). More formally, the algorithm must obviously simulate the following $\mathcal{F}_{\text{SampleAux}}(n, n_0)$ functionality with leakage (n, n_0) .

$\mathcal{F}_{\text{SampleAux}}(n, n_0)$ – **Sample Auxiliary Array**

- **Input:** Two numbers $n, n_0 \in \mathbb{N}$ such that $n_0 \leq n$.
- **The functionality:** Sample an array Aux of n bits uniformly at random conditioned on having n_0 zeros and $n - n_0$ ones. Output Aux .

2. Next, we route elements 1-to-1 from \mathbf{I}_0 to zeros in Aux and 1-to-1 route elements from \mathbf{I}_1 to ones in Aux . This can be accomplished by running oblivious tight compaction circuit (Theorem 9) to pack all the 0s in Aux in the front. During the process, all swap gates remember their routing decisions. Now, we can run the oblivious tight compaction circuit in reverse and on the input array $\mathbf{I}_0 \parallel \mathbf{I}_1$. It is not hard to see that in the outcome, every 0 position in Aux would receive an element from \mathbf{I}_0 and every 1 position in Aux would receive an element from \mathbf{I}_1 .

Asharov et al. [5, Claim 6.3] proved that the above algorithm indeed realizes the **Intersperse** abstraction as defined above. Moreover, they show how to implement the above idea obliviously in linear-time, resulting in the following theorem:

► **Theorem 10** (Sequential, linear-time **Intersperse** [5]). *There exists an algorithm that perfectly obliviously simulates $\mathcal{F}_{\text{shuffle}}$ for two randomly shuffled input arrays. Moreover, the algorithm completes in deterministic $O(\beta n)$ total work where n denotes the sum of the lengths of the two input arrays, and $\beta \geq 1$ denotes the number of memory words required to represent each element.*

3.3.3 Parallel Intersperse Algorithms

We need a parallel version of the **Intersperse** algorithm. In Asharov et al. [5]'s **Intersperse** construction, while the oblivious tight compaction building block can be replaced with a parallel realization of tight compaction (Theorem 9), unfortunately they adopt a highly sequential procedure for generating the Aux array. To get a parallel algorithm, it suffices to devise a parallel procedure for generating such an Aux array. More formally, we would like to devise an algorithm that obliviously simulates the functionality $\mathcal{F}_{\text{SampleAux}}(n, n_0)$.

A naïve algorithm with deterministic performance. A naïve algorithm is the following: simply write down exactly n_0 number of 0s and $n - n_0$ number of 1s, apply an oblivious random permutation to permute the array, and output the result. If we use Theorem 6 to instantiate this naïve algorithm, we obtain the following theorem:

► **Theorem 11** (Naïve parallel algorithm for sampling Aux). *For any $n_0 \leq n$, there exists an algorithm that perfectly obliviously simulates $\mathcal{F}_{\text{SampleAux}}(n, n_0)$; moreover, for sampling an Aux array of length n , the algorithm completes in deterministic $O(n \log^2 n)$ total work and $O(\log^2 n)$ depth.*

This immediately gives rise to the following corollary for **Intersperse** due to the result of Asharov et al. [5] and parallel tight compaction (Theorem 9):

► **Corollary 12** (Naïve parallel Intersperse). *There exists an algorithm that perfectly obliviously simulates $\mathcal{F}_{\text{shuffle}}$ for two randomly shuffled input arrays. Moreover, the algorithm completes in deterministic $O(\beta n + n \log^2 n)$ total work and $O(\log^2 n)$ depth where n denotes the sum of the lengths of the two input arrays, and $\beta \geq 1$ denotes the number of memory words required to represent each element.*

A more efficient Las Vegas algorithm. To obliviously simulate the functionality $\mathcal{F}_{\text{SampleAux}}(n, n_0)$ with better performance, we use the Las Vegas version of oblivious random permutation, Theorem 7, which gives the following theorem:

► **Theorem 13** (Las Vegas parallel algorithm for sampling Aux). *For any $n_0 \leq n$, there exists a Las Vegas algorithm that perfectly obliviously simulates $\mathcal{F}_{\text{SampleAux}}(n, n_0)$. Except with probability $n^{-\Omega(\sqrt{n})}$, the algorithm completes in $O(n \log n)$ total work and $O(\log^2 n)$ depth. Furthermore, the above stated performance bounds also apply in expectation.*

Now due to the work of Asharov et al. [5] and parallel tight compaction (Theorem 9), we have the following corollary.

► **Corollary 14** (Parallel Intersperse). *Let $\beta \geq 1$ be the number of words used to represent an element. There is an Intersperse algorithm that is a perfectly oblivious simulation of $\mathcal{F}_{\text{shuffle}}$ on two randomly shuffled input arrays; moreover, except with $n^{-\Omega(\sqrt{n})}$ probability, the algorithm completes in $O(\beta n + n \log n)$ total work and $O(\log^2 n)$ depth. Moreover, the stated performance bounds also apply in expectation.*

We defer the formal construction and proof to the full version [14]

References

- 1 M. Ajtai, J. Komlós, and E. Szemerédi. An $O(N \log N)$ sorting network. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 1–9, New York, NY, USA, 1983. ACM.
- 2 Miklós Ajtai. Oblivious RAMs without cryptographic assumptions. In *STOC*, 2010.
- 3 Laurent Alonso and René Schott. A parallel algorithm for the generation of a permutation and applications. *Theoretical Computer Science*, 159(1):15–28, 1996.
- 4 Gilad Asharov, Hubert Chan, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Locality-preserving oblivious RAM. In *Eurocrypt*, 2019.
- 5 Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, Enoch Peserico, and Elaine Shi. Optorama: Optimal oblivious RAM. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 403–432, Cham, 2020. Springer International Publishing.
- 6 Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Enoch Peserico, and Elaine Shi. Oblivious Parallel Tight Compaction. In *1st Conference on Information-Theoretic Cryptography (ITC 2020)*, volume 163 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:23, 2020.
- 7 Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Enoch Peserico, and Elaine Shi. Optimal oblivious parallel RAM. Cryptology ePrint Archive, Report 2020/1292, 2020. URL: <https://eprint.iacr.org/2020/1292>.
- 8 K. E. Batcher. Sorting Networks and Their Applications. In *AFIPS '68 (Spring)*, 1968.
- 9 Elette Boyle, Kai-Min Chung, and Rafael Pass. Oblivious parallel RAM and applications. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 175–204, 2016.
- 10 T-H. Hubert Chan, Kai-Min Chung, and Elaine Shi. On the depth of oblivious parallel RAM. In *Asiacrypt*, 2017.

- 11 T-H. Hubert Chan, Yue Guo, Wei-Kai Lin, and Elaine Shi. Oblivious hashing revisited, and applications to asymptotically efficient ORAM and OPRAM. In *Asiacrypt*, 2017.
- 12 T-H. Hubert Chan, Kartik Nayak, and Elaine Shi. Perfectly secure oblivious parallel RAM. In *TCC*, 2018.
- 13 T-H. Hubert Chan and Elaine Shi. Circuit OPRAM: A unifying framework for computationally and statistically secure ORAMs and OPRAMs. In *TCC*, 2017.
- 14 T-H. Hubert Chan, Elaine Shi, Wei-Kai Lin, and Kartik Nayak. Perfectly oblivious (parallel) RAM revisited, and improved constructions. Cryptology ePrint Archive, Report 2020/604, 2020. URL: <https://eprint.iacr.org/2020/604>.
- 15 Kai-Min Chung, Zhenming Liu, and Rafael Pass. Statistically-secure ORAM with $\tilde{O}(\log^2 n)$ overhead. In *Asiacrypt*, 2014.
- 16 Artur Czumaj. Random Permutations Using Switching Networks. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 703–712, New York, NY, USA, 2015. ACM.
- 17 Ivan Damgård, Sigurd Meldgaard, and Jesper Buus Nielsen. Perfectly secure oblivious RAM without random oracles. In *TCC*, pages 144–163, 2011.
- 18 Ioannis Demertzis, Dimitrios Papadopoulos, and Charalampos Papamanthou. Searchable encryption with optimal locality: Achieving sublogarithmic read efficiency. In *Advances in Cryptology – CRYPTO 2018*, 2018.
- 19 Christopher W. Fletcher, Ling Ren, Xiangyao Yu, Marten van Dijk, Omer Khan, and Srinivas Devadas. Suppressing the oblivious RAM timing channel while making information leakage and program efficiency trade-offs. In *HPCA*, pages 213–224, 2014.
- 20 Daniel Genkin, Yuval Ishai, and Mor Weiss. Binary AMD circuits from secure multiparty computation. In *Theory of Cryptography Conference*, pages 336–366. Springer, 2016.
- 21 O. Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In *STOC*, 1987.
- 22 Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 1996.
- 23 Michael T. Goodrich and Michael Mitzenmacher. Privacy-preserving access of outsourced data via oblivious RAM simulation. In *ICALP*, pages 576–587, 2011.
- 24 S. Dov Gordon, Jonathan Katz, Vladimir Kolesnikov, Fernando Krell, Tal Malkin, Mariana Raykova, and Yevgeniy Vahlis. Secure two-party computation in sublinear (amortized) time. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.
- 25 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 406–425. Springer, 2011.
- 26 Eyal Kushilevitz, Steve Lu, and Rafail Ostrovsky. On the (in)security of hash-based oblivious RAM and a new balancing scheme. In *SODA*, 2012.
- 27 Kasper Green Larsen and Jesper Buus Nielsen. Yes, there is an oblivious RAM lower bound! In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 523–542, 2018.
- 28 Chang Liu, Austin Harris, Martin Maas, Michael Hicks, Mohit Tiwari, and Elaine Shi. Ghost rider: A hardware-software system for memory trace oblivious computation. *SIGPLAN Not.*, 50(4):87–101, 2015.
- 29 Chang Liu, Xiao Shaun Wang, Kartik Nayak, Yan Huang, and Elaine Shi. Oblivm: A programming framework for secure computation. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 359–376, 2015.
- 30 Martin Maas, Eric Love, Emil Stefanov, Mohit Tiwari, Elaine Shi, Kriste Asanovic, John Kubiatowicz, and Dawn Song. Phantom: Practical oblivious computation in a secure processor. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.
- 31 Rafail Ostrovsky and Victor Shoup. Private information storage (extended abstract). In *ACM Symposium on Theory of Computing (STOC)*, pages 294–303, 1997.

- 32 S. Patel, G. Persiano, M. Raykova, and K. Yeo. Panorama: Oblivious ram with logarithmic overhead. In *IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 871–882, 2018.
- 33 Enoch Peserico. Deterministic oblivious distribution (and tight compaction) in linear time. *CoRR*, abs/1807.06719, 2018. [arXiv:1807.06719](https://arxiv.org/abs/1807.06719).
- 34 Michael Raskin and Mark Simkin. Perfectly secure oblivious ram with sublinear bandwidth overhead. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 537–563, 2019.
- 35 Ling Ren, Xiangyao Yu, Christopher W. Fletcher, Marten van Dijk, and Srinivas Devadas. Design space exploration and optimization of path oblivious RAM in secure processors. In *ISCA*, pages 571–582, 2013.
- 36 Elaine Shi, T.-H. Hubert Chan, Emil Stefanov, and Mingfei Li. Oblivious RAM with $O((\log N)^3)$ worst-case cost. In *ASIACRYPT*, pages 197–214, 2011.
- 37 Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM – an extremely simple oblivious ram protocol. In *CCS*, 2013.
- 38 Xiao Shaun Wang, T-H. Hubert Chan, and Elaine Shi. Circuit ORAM: On Tightness of the Goldreich-Ostrovsky Lower Bound. In *ACM CCS*, 2015.

On the Complexity of Anonymous Communication Through Public Networks

Megumi Ando  

MITRE, Bedford, MA, USA

Anna Lysyanskaya 

Brown University, Providence, RI, USA

Eli Upfal  

Brown University, Providence, RI, USA

Abstract

Onion routing is the most widely used approach to anonymous communication online. The idea is that Alice wraps her message to Bob in layers of encryption to form an “onion” and routes it through a series of intermediaries. Each intermediary’s job is to decrypt (“peel”) the onion it receives to obtain instructions for where to send it next. The intuition is that, by the time it gets to Bob, the onion will have mixed with so many other onions that its origin will be hard to trace even for an adversary that observes the entire network and controls a fraction of the participants, possibly including Bob. Despite its widespread use in practice, until now no onion routing protocol was known that simultaneously achieved, in the presence of an active adversary that observes all network traffic and controls a constant fraction of the participants, (a) anonymity; (b) fault-tolerance, where even if a few of the onions are dropped, the protocol still delivers the rest; and (c) reasonable communication and computational complexity as a function of the security parameter and the number of participants.

In this paper, we give the first onion routing protocol that meets these goals: our protocol (a) achieves anonymity; (b) tolerates a polylogarithmic (in the security parameter) number of dropped onions and still delivers the rest; and (c) requires a polylogarithmic number of rounds and a polylogarithmic number of onions sent per participant per round. We also show that to achieve anonymity in a fault-tolerant fashion via onion routing, this number of onions and rounds is necessary. Of independent interest, our analysis introduces two new security properties of onion routing – mixing and equalizing – and we show that together they imply anonymity.

2012 ACM Subject Classification Security and privacy → Security protocols

Keywords and phrases Anonymity, privacy, onion routing

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.9

Related Version *Full Version:* <https://arxiv.org/abs/1902.06306>

Funding Part of this work was funded by NSF award IIS-1247581 while Megumi was a graduate student at Brown University.

1 Introduction

Suppose that Alice wishes to send a message anonymously to Bob. Informally, by *anonymously*, we mean that no one (not even Bob) can distinguish the scenario in which Alice sends a message to Bob from an alternative scenario in which it is Allison who sends a message to Bob. To begin with, Alice can encrypt the message and send the encrypted message to Bob so that only Bob can read the message. However, an eavesdropper observing the sequence of bits coming out of Alice’s computer and the sequence of bits going into Bob’s computer can still determine that Alice and Bob are communicating with each other if the sequences of bits match. Thus, encryption is not enough.



© Megumi Ando, Anna Lysyanskaya, and Eli Upfal;
licensed under Creative Commons License CC-BY 4.0
2nd Conference on Information-Theoretic Cryptography (ITC 2021).
Editor: Stefano Tessaro; Article No. 9; pp. 9:1–9:25



Leibniz International Proceedings in Informatics
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Onion routing [11] is the most promising approach to anonymous channels to date. In onion routing, messages are sent via intermediaries and wrapped in layers of encryption, resulting in so-called onions; each intermediary’s task is to “peel off” a layer of encryption and send the resulting onion to the next intermediary or its final destination. The onion’s layers are unlinkable to each other, and so its route through the network cannot be traced from merely observing the sequences of bits that Alice transmits and Bob receives. However, even with Alice sending her message to Bob encoded as an onion, her communication can still be tracked by a resourceful eavesdropper with an extensive view of the network traffic (e.g., an ISP-level or an AS-level adversary) who can observe all Internet traffic.

The adversary who can observe all network traffic is called the *network adversary*. The adversary who, in addition to observing all network traffic, controls a subset of the participants is called the *passive adversary* if it follows the prescribed protocol, or *active* if it does not. The three adversary models – the network adversary, the passive adversary, and the active adversary – are standard for analyzing cryptographic protocols such as multi-party computation (MPC) [21]. The most desirable goal is to achieve security in the presence of the most powerful of these, i.e., the active adversary corrupting as large a fraction of the participants as possible.

It was known how to construct an onion routing protocol that is both efficient and anonymous from the passive adversary who corrupts a constant fraction of the parties. In Π_p [3], each user forms an onion bearing his message to its recipient; the users’ onions are routed randomly through a network of servers. Π_p is anonymous from the passive adversary provided that the onions travel for a superlogarithmic (in the security parameter) number of rounds, and the average number of onions per server per round is also superlogarithmic.

However, Π_p isn’t anonymous from the active adversary. To see why this is the case, consider the following attack. Suppose that the adversary \mathcal{A} suspects that Alice is communicating with Bob. Because \mathcal{A} is active, he can disrupt Alice’s communication by dropping Alice’s outgoing onion in the event that Alice’s first intermediary is corrupt (the probability of this event is identical to the fraction of parties that are under the adversary’s control). If Bob doesn’t receive an onion at the end of the protocol, then \mathcal{A} can infer that her suspicion was correct: Alice’s interlocutor is Bob!

So what can we do instead? Of course, we could use general-purpose MPC [21]. Every party will receive as input a message and its destination, and every party will receive as output the messages that were meant for him/her. In addition to perfect anonymity, this approach provides fault tolerance: in MPC that is secure against the active adversary, the honest parties are guaranteed to receive their output no matter how much the adversary deviates from the protocol. But the problem with this approach is that relying on *general-purpose* MPC makes this approach too inefficient: the most efficient general MPC protocol still requires that at least some of the participants send and receive $\Omega(N)$ bits, where N is the number of participants. (See Cramer, Damgård, and Nielsen [16].)

Recently proposed protocols, Stadium [30] and Atom [25], are more efficient. However, they are not fault-tolerant: honest parties will abort the protocol run whenever even a single message packet is dropped. Thus, while this approach provides anonymity from the active adversary, it is also extremely fragile: if just one message is dropped (which could be the result of an innocuous fault), the entire network suffers a catastrophic failure. In contrast, we would like to design onion routing protocols that can tolerate faults. Thus, compared to MPC and Stadium-Atom-type protocols, onion routing appears attractive from the efficiency and fault tolerance points of view.

In this paper, we answer these fundamental questions: Can an onion routing protocol be simultaneously anonymous, fault-tolerant, and efficient? What is the communication complexity of anonymous and fault-tolerant onion routing?

1.1 Problem setting

Before describing our results in detail, let us first define our problem setting. Let $\mathcal{P} \stackrel{\text{def}}{=} \{P_1, P_2, \dots, P_N\}$ denote the set of N parties, participating in an onion routing protocol. We assume that the protocol progresses in global rounds and that an onion sent at round r arrives at its destination prior to round $r + 1$. Moreover, the adversary is modelled with *rushing*, i.e., the adversary receives onions sent in round r instantaneously in round r .¹ We assume that the number N of participants and every other quantity in the protocol is polynomially bounded in the security parameter λ .

We define an *onion routing protocol* to be a protocol in which the honest parties form and process only message packets that are cryptographic onions. To do this, the honest parties use a secure onion encryption scheme which is a triple of algorithms: $(\text{Gen}, \text{FormOnion}, \text{ProcOnion})$.

See Section 2.1 for more details. During setup of an onion routing protocol, each honest party P generates a public-key pair $(\text{pk}(P), \text{sk}(P)) \leftarrow \text{Gen}(1^\lambda)$ using the onion encryption scheme's key generation algorithm Gen . Each party P publishes his/her public key $\text{pk}(P)$ to a public directory so that everyone knows everyone else's public keys.

Let \mathcal{M} be the space of fixed-length messages. An input $\sigma = (\sigma_1, \dots, \sigma_N)$ to the protocol is a vector of inputs, where σ_i is a set of message-recipient pairs for party P_i . For $m \in \mathcal{M}$ and $P_j \in \mathcal{P}$, the inclusion of a message-recipient pair (m, P_j) in input σ_i means that party P_i is instructed to send message m to recipient P_j . In this paper, we consider the following “benchmark” input space, dubbed the *simple input/output setting (I/O)*. An input $\sigma = (\sigma_1, \dots, \sigma_N)$ is in the simple I/O setting if there exists a permutation $\pi : \mathcal{P} \mapsto \mathcal{P}$ such that each party $P \in \mathcal{P}$ is instructed to send a message to party $\pi(P)$ and no other message, i.e., $\forall P \in \mathcal{P}, \exists m_P \in \mathcal{M}$ such that $\sigma_P = \{(m_P, \pi(P))\}$. The simple I/O setting is a superset of the spaces considered in prior works [3, 25, 30, 31].

Unless stated otherwise, the adversary is active and can observe the traffic on all communication channels and, additionally, can non-adaptively corrupt and control a constant fraction of the parties. By *non-adaptively*, we mean that the corruptions are made independently of any protocol run.² Without loss of generality, this type of corruption is captured by allowing the adversary to select the set Bad of corrupted parties prior to the beginning of the protocol run. Once the adversary corrupts a party, the adversary can observe the internal state and computations of the corrupted party and arbitrarily alter the behavior of the party.

By $\mathbb{V}^{\Pi, \mathcal{A}}(1^\lambda, \sigma)$, we denote the view of the adversary \mathcal{A} when it interacts with protocol Π on input: the security parameter 1^λ and the instructions σ . The view consists of all the observations that \mathcal{A} makes during the run: the values and positions of every onion at every round, the states and computations of every corrupted party between every pair of consecutive rounds, the randomness used by \mathcal{A} , and the numbers of messages received by the honest parties. The view does not include the honest parties' randomness. $\mathbb{V}^{\Pi, \mathcal{A}, \text{Bad}}(1^\lambda, \sigma)$ denotes \mathcal{A} 's view given its choice Bad for the corrupted parties. At the end of the protocol run, each honest party P_i outputs the set $\mathbb{O}_i^{\Pi, \mathcal{A}}(1^\lambda, \sigma)$ of (non-empty) messages from the message space \mathcal{M} that P_i receives from interacting with adversary \mathcal{A} in a run of protocol Π on input σ . We define the *output* $\mathbb{O}^{\Pi, \mathcal{A}}(1^\lambda, \sigma)$ of protocol Π in an interaction with adversary \mathcal{A} on input σ as the N parties' outputs:³ $\mathbb{O}^{\Pi, \mathcal{A}}(1^\lambda, \sigma) \stackrel{\text{def}}{=} (\mathbb{O}_1^{\Pi, \mathcal{A}}(1^\lambda, \sigma), \mathbb{O}_2^{\Pi, \mathcal{A}}(1^\lambda, \sigma), \dots, \mathbb{O}_N^{\Pi, \mathcal{A}}(1^\lambda, \sigma))$.

¹ We do not consider the asynchronous communication model [9] in which Alice's outgoing onions (including her onion to her recipient Bob) can be delayed indefinitely. In such a case, we cannot even guarantee correctness (i.e., message delivery when no party deviates from the protocol).

² If we were to allow the adversary to adaptively corrupt parties, then the adversary could easily block all of Alice's onions. For every onion O sent by Alice, the adversary can corrupt the party P who receives O in time to direct P to drop the onion obtained from processing O before the next round.

³ Technically, the view and the output may depend on other parameters, such as the public parameters

1.2 Our results

We now describe our results. Our construction pertains to the problem setting described in Section 1.1. Our lower bound applies more generally to any arbitrary input set (not necessarily constrained to the simple I/O setting).

Following prior work [3, 25, 30, 31], we use a natural game-based definition of anonymity: a protocol is *anonymous* if the adversary cannot distinguish the scenario in which Alice sends a message to Bob while Carol sends one to David, from one in which Alice’s message goes to David while Carol’s goes to Bob; (see Definition 4). More precisely, for any pair of inputs (σ^0, σ^1) that agree on the inputs and outputs for the adversarial participants, $O^{\Pi, A}(1^\lambda, \sigma^0) \approx O^{\Pi, A}(1^\lambda, \sigma^1)$, where “ \approx ” denotes indistinguishability.

We relate anonymity of an onion routing protocol to two new concepts: An onion routing protocol *mixes* if it sufficiently shuffles the honest users’ onions making it infeasible for the adversary to trace a received message back to its sender. A protocol *equalizes* if the adversary cannot determine the input from the numbers of messages received by the parties; in other words, the number of messages output by each participant – or the fact that a participant did not receive an output at all – are random variables that are computationally unrelated to the input vector σ . (See Definitions 5 and 7.)

We show that in many cases, mixing and equalizing implies anonymity, i.e., an onion routing protocol that mixes and equalizes is anonymous. (See Theorem 3 for the formal theorem statement.) We use this to prove that our protocol is anonymous. Anonymity also implies equalizing; this observation is useful for proving a lower bound that (almost) matches our protocol.

1.2.1 Anonymous, “robust,” and efficient onion routing

As we just explained, our strategy is to construct a protocol that mixes and equalizes.

Intuitively, mixing is the easier one to achieve: the onions need to sufficiently shuffle with other onions traveling over the network to ensure that each of them is hard to trace. This intuition is essentially correct with the caveat that an active adversary can strategically interfere with this process by dropping onions. To ensure that each onion shuffles with a sufficiently large number of onions (formed by an honest party) a sufficiently large number of times, our protocol uses *checkpoint onions* [3] that each intermediary expects to receive, and if a constant fraction (e.g., one-third) of them don’t arrive because the adversary dropped them, the protocol aborts.

An active adversary who controls a fraction of the participants can try to “isolate” an honest party Alice from the rest of the network by dropping all of the messages/onions received directly from Alice. In a fault-tolerant network protocol, the remaining participants may still be able to get their messages through to their destinations. In this case, based on who received an output, an adversary can infer who Alice’s intended recipient was. This attack explains why equalizing is difficult to achieve.

To overcome this attack, we introduce a new type of onions, called *merging onions*. When two merging onions belonging to the same pair arrive at some intermediary I , I recognizes that they are from the same pair (although, other than their next layer and destination, I does not learn anything else about them). The protocol directs I to discard one of them

(denoted, \mathbf{pp}) and the parties’ states (denoted, \mathbf{states}). Thus, we could be more precise by denoting the view and the output as $V^{\Pi, A}(1^\lambda, \mathbf{pp}, \mathbf{state}, \sigma)$ and $O^{\Pi, A}(1^\lambda, \mathbf{pp}, \mathbf{states}, \sigma)$, but we will use the simpler notation for better readability.

(chosen at random) while sending its mate along. If only one onion of the pair arrived at I while its mate is missing (i.e. the adversary dropped it some time earlier in the protocol run), then I simply sends along the mate that survived, and there is nothing to discard.

Why does this help? Suppose that both Alice and Allison created 2^h merging onions; at rounds r_1, r_2, \dots, r_h each of these onions (if it hasn't been deleted yet) will meet a mate. Say, exactly one of Alice's onions is dropped by the adversary at some point prior to round r_1 , so its mate (the onion it was supposed to pair with at round r_1) was not dropped. Also, suppose that none of Allison's onions were dropped. Then at round r_1 all but one of Alice's remaining merging onions will meet a mate, and half of them will be dropped, so exactly 2^{h-1} of Alice's onions will remain in the system – which is exactly how many of Allison's onions remain. Additional $h - 1$ opportunities to merge account for the possibility that the adversary has dropped a larger number of Alice's onions. Merging onions ensure that the number of Alice's onions that remain in the system at the end of the protocol is the same as the number of Allison's onions, i.e., that the protocol equalizes. The fact that Alice was targeted and many of her onions had been dropped upfront doesn't matter because the protocol discards all but one of them anyway! (See Section 4 for a more in-depth description of merging onions and how to construct them.)

Positive result. We construct an onion routing protocol Π_{\bowtie} , pronounced “Pi-butterfly,” because it uses a butterfly network. Π_{\bowtie} takes advantage of the merging onions technique described above. It is (a) anonymous from the active adversary who can corrupt up to a constant fraction $\kappa < \frac{1}{2}$ of the parties and (b) *robust*, i.e. whenever the adversary drops at most logarithmic (in the security parameter) number of message packets (i.e. onions), Π_{\bowtie} delivers the messages from honest senders with overwhelming probability. Moreover, (c) during the execution of the protocol, every honest party transmits up to a polylog (in the security parameter) number of onions: specifically $\gamma_1 \log N \log^{3+\gamma_2} \lambda$ onions, where N is the number of participants, and λ is the security parameter. γ_1 and γ_2 are parameters that can be set as desired: increasing them increases the rate at which the maximum distance in the adversarial views for any two inputs shrinks. (See Theorem 12 for the precise relationship.)

1.2.2 Matching negative result

Our protocol is essentially optimal as far as both the round complexity and the number of onions each participant sends out are concerned. In Section 7, we explain why a protocol that is anonymous and robust in the presence of an active adversary that corrupts a constant fraction of participants requires a polylogarithmic number of onions sent out per participant.

1.3 Related work

Our work is inspired by the fact that Tor [18], the most widely adopted anonymous communication system, is also known to have numerous security flaws [23, 27, 29, 32]: Tor is based on a highly efficient design that favors practicality over security and is not secure even from the passive adversary [17]. Moreover, it has been shown to be vulnerable to network traffic correlation attacks [23, 27, 29, 32]. Thus, our goal was to design a protocol that was as close to Tor's efficiency and fault tolerance as possible, while also being provably anonymous. We consider a very specific and narrow problem in the much larger field of anonymous messaging systems. Although our definition of anonymity and adversary models are standard in cryptography, other definitions have been considered [1, 5, 6, 10, 19] and positive results for alternative models are known [4–6].

Other provably anonymous systems exist [12, 14, 15, 28], but they are not nearly as efficient. Achieving anonymous channels using heavier cryptographic machinery has been considered also. One of the earliest examples is Chaum’s dining cryptographer’s protocol [12]. Rackoff and Simon [28] use secure multiparty computation for providing security from active adversaries. Other cryptographic tools used in constructing anonymity protocols include oblivious RAM (ORAM) and private information retrieval (PIR) [14, 15]. Corrigan-Gibbs et al.’s Riposte solution makes use of a global bulletin board with a latency of days [15].

We are not the first to look into lower bounds on the complexity of anonymous messaging protocols (e.g., [13, 17]). However, all other lower bounds are for the setting where every participant is guaranteed to receive an output, and don’t apply to protocols that allow aborts or that allow some participants to receive an output while others’ output doesn’t make it through.

2 Preliminaries

For a set \mathcal{S} , we denote the cardinality of \mathcal{S} by $|\mathcal{S}|$, and $s \leftarrow \mathcal{S}$ denotes that s is chosen from \mathcal{S} uniformly at random. For an algorithm $A(x)$, $y \leftarrow A(x)$ is the (possibly probabilistic) output y from running A on the input x . In this paper, $\log(x)$ is the logarithm of x base 2.

We say that a function $f : \mathbb{N} \mapsto \mathbb{R}$ is negligible in the parameter λ , written $f(\lambda) = \text{negl}(\lambda)$, if for a sufficiently large λ , $f(\lambda)$ decays faster than any inverse polynomial in λ . When λ is the security parameter, an event E_λ is said to occur with (non-)negligible probability if the probability of E_λ can(not) be bounded above by a function negligible in λ . An event occurs with overwhelming probability (abbreviated, w.o.p.) if its complement occurs with negligible probability. We use the standard notion of a pseudorandom function [20, Chapter 3.6].

2.1 Onion encryption schemes

Our work on onion routing builds upon a secure onion encryption scheme [2, 7, 24]. Recall that an onion encryption scheme is a triple: $(\text{Gen}, \text{FormOnion}, \text{ProcOnion})$. The algorithm Gen generates a participant key pair, i.e., a public key and a secret key. The algorithm FormOnion forms onions, and the algorithm ProcOnion processes onions.

Let \mathcal{P} be a set of participants, and let $\text{Bad} \subseteq \mathcal{P}$ be the set of corrupt parties. For every honest $P \in \mathcal{P} \setminus \text{Bad}$, let $(\text{pk}(P), \text{sk}(P)) \leftarrow \text{Gen}(1^\lambda, \text{pp}, P)$ be the key pair generated for party P , where λ is the security parameter, and pp , the public parameters. For every corrupt party $P \in \text{Bad}$, let $\text{pk}(P)$ denote P ’s public key. Let \mathcal{M} be the message space consisting of messages of the same fixed length, and let the nonce space \mathcal{S} consist of nonces of the same fixed length. These lengths may be a function of the security parameter λ . Here, a *nonce* is really any metadata associated with an onion layer.

FormOnion takes as input a message $m \in \mathcal{M}$, an ordered list (Q_1, \dots, Q_{d-1}, R) of parties from \mathcal{P} , the public keys $(\text{pk}(Q_1), \dots, \text{pk}(Q_{d-1}), \text{pk}(R))$ associated with these parties, and a list (s_1, \dots, s_{d-1}) of (possibly empty) nonces from \mathcal{S} associated with the layers of the onion.⁴ The party R is interpreted as the *recipient* of the message, and the list (Q_1, \dots, Q_{d-1}, R) is the *routing path*. The output of FormOnion is a sequence (O_1, \dots, O_d) of onions. Such a sequence is referred to as an *evolution*, but every O_i in the sequence is an onion. Because it is

⁴ Technically, the input/output syntax and constructions of [2, 7] do not include the sequence (s_1, \dots, s_d) of nonces but can easily be extended to do so; if we use layered CCA2-secure encryption instead of onion encryption – which is fine for this application – then incorporating the nonces is trivial.

convenient to think of an onion as a layered encryption object where processing an onion O_i produces the next onion O_{i+1} , we sometimes refer to the process of revealing the next onion as “decrypting the onion” or “peeling the onion.”

For every $i \in [d - 1]$, only intermediary party Q_i can peel onion O_i to reveal the next layer, $(O_{i+1}, Q_{i+1}, s_{i+1}) \leftarrow \text{ProcOnion}(\text{sk}(Q_i), O_i, Q_i)$, which contains the *peeled* onion O_{i+1} , the *next destination* Q_{i+1} of the onion, and the nonce s_{i+1} . Only the recipient R can peel the innermost onion O_d to reveal the message, $m \leftarrow \text{ProcOnion}(\text{sk}(R), O_d, R)$.

In our constructions, a sender of a message m to a recipient R “forms an onion” by generating nonces and running the **FormOnion** algorithm on the message m , a routing path (Q_1, \dots, Q_{d-1}, R) , the keys $(\text{pk}(Q_1), \dots, \text{pk}(Q_{d-1}), \text{pk}(R))$ associated with the parties on the routing path, and the generated nonces; the *formed onion* is the first onion O_1 from the list of outputted onions.

Secure onion encryption. Suppose that (honest) Alice generates an onion carrying a message m for Bob. That is, she generates a string of nonces and runs the algorithm **FormOnion** on the inputs: the message m , the routing path $(Q_1, \dots, Q_{i-1}, I, Q_{i+1}, \dots, Q_{d-1}, \text{Bob})$, the public keys associated with the routing path, and the nonces. Let O denote the onion for intermediary party I , i.e., O is the i^{th} onion in the outputted evolution. Suppose that (honest) Carol runs the algorithm **FormOnion** on the inputs: the message m' , the routing path $(Q'_1, \dots, Q'_{j-1}, I, Q'_{j+1}, \dots, Q'_{e-1}, \text{David})$, the public keys associated with the routing path, and some nonces. Let O' denote the onion for intermediary party I . Provided that the onion encryption scheme is secure, if party I receives onions O and O' in the same round and consequently processes the two onions in the same batch, then the adversary cannot tell which processed onion resulted from processing O and which resulted from processing O' . In other words, onions formed by honest parties “mix” at honest parties. For a precise, cryptographic definition of secure onion encryption, see the recent paper by Ando and Lysyanskaya [2].

2.2 Onion routing protocols

In an onion routing protocol, all the packets sent between protocol participants are treated as onions; i.e., upon receipt, they are fed to **ProcOnion**. Moreover, internally, there are type checks that ensure that these onions are processed properly. There are two cases for processing an honestly formed onion properly: the case where peeling the onion reveals its next layer and destination and the case where it reveals a message for the processing party.

If Q_i runs **ProcOnion** and outputs the next layer of the onion O_{i+1} (together with its destination Q_{i+1} and nonce s_{i+1}), then the only two options for what an onion routing protocol permits Q_i to do with O_{i+1} is either send it to Q_{i+1} or drop it (if $Q_{i+1} = Q_i$ then this send step is internal to Q_i). Which of these actions are taken depends on the specifics of the algorithm and also on the values (Q_{i+1}, s_{i+1}) . In other words, an onion routing protocol cannot have an onion sent to incorrect destinations or fed as input to another algorithm.

Further, if Q_i runs **ProcOnion** and outputs a message $m \neq \perp$, then this message becomes (ultimately, at the end of the protocol) part of Q_i 's output, i.e., it will be on the list of messages that have been sent to Q_i . In other words, m cannot be internal to the protocol; it must be a message that someone sent to Q_i via the protocol. Conversely, the only way that a message m can be on the list of messages received by Q_i is if Q_i obtained it by peeling one of the onions it received.

These restrictions on protocol design are natural. Indeed, any implementation of onion routing would ensure that it is adhered to by using type checking of the objects created, sent, and processed by the algorithm. Without such a restriction, any protocol can be thought of

as an instance of onion routing protocol, so limiting our attention in this way is meaningful. Note that this places restrictions just on the protocol that the honest parties are executing; the adversary is still free to do anything he wishes: to mismatch types, to route onions incorrectly, to try to rewrap onions, to form and process onions adversarially, etc.

Onion routing serves a purpose: to route messages from senders to recipients. Therefore, it needs to satisfy correctness:

► **Definition 1 (Correctness).** *A messaging protocol Π is correct if in an interaction with a passive adversary, it delivers all the messages with overwhelming probability.*

In this paper, we will consider only correct onion routing protocols, but we will analyze their interactions with active adversaries.

Further, the protocols we design in this paper have an additional attractive property of being indifferent:

► **Definition 2 (Indifference).** *An onion routing protocol Π is indifferent if two properties hold: (i) The routing path corresponding to each honestly formed onion is of a fixed length. (ii) The sequence of intermediaries, including the recipients of dummy onions, and the sequence of nonces corresponding to each honestly formed onion do not depend on the input.*

The intuition behind this notion is that the contents of the messages sent and received between parties have no bearing on how the messages are routed and transmitted. For protocol design, indifference is an attractive property that allows components of an onion to be in place (and possibly the bulk of the cryptographic computation finished) before the message contents even becomes known. Another attractive feature of indifferent protocols is that their security properties are easier to analyze, as we will explore in the next section. Our negative results apply to all onion routing schemes, indifferent or not.

3 Security definitions: anonymity, equalizing, and mixing

A motivating example. Consider Ando, Lysyanskaya, and Upfal’s very simple protocol Π_p (p , for passive) in the passive adversary setting [3]. Recall that corrupted parties also follow the protocol in this setting. Let $\text{Servers} \subseteq \mathcal{P}$ be the set of servers which is a subset of \mathcal{P} . During the *onion-forming phase*, every party P generates an onion from the message-recipient pair (m, R) in P ’s input by first choosing $d - 1$ servers (S_1, \dots, S_{d-1}) , each chosen independently and uniformly at random from Servers . Next, P forms an onion O by running FormOnion on the message m , the routing path $P^\rightarrow = (S_1, \dots, S_{d-1}, R)$, the public keys associated with P^\rightarrow , and the sequence of empty nonces. At the first round of the *execution phase*, each party P sends its formed onion O to the first server S_1 on the routing path. For every round $r \in [d]$, each server S does the following: Between the r^{th} and $(r + 1)^{\text{st}}$ rounds, S processes all the onions it received at the r^{th} round. At the $(r + 1)^{\text{st}}$ round, S sends the processed onions to their respective next destinations. At the d^{th} round, each party receives an onion that, once processed, reveals a message m for the party.

Π_p is anonymous if the protocol sufficiently shuffles the onions during the execution phase. In prior work [3], Ando, Lysyanskaya, and Upfal showed that sufficient shuffling occurs when the server load (i.e., the average number of onions received by a server at a round: $\frac{N}{|\text{Servers}|}$) and the number of rounds (i.e., d) are both superlogarithmic in the security parameter. However, there is no parameter setting for which Π_p can be anonymous from the *active* adversary. If κN out of N participants are corrupted, then with probability κ , the adversary can determine the recipient of any honest party, say Alice: Suppose that during

the onion-forming phase, Alice picks a routing path that begins with an adversarial party S_1 . During the execution phase, the adversary can direct S_1 to drop Alice's onion before the second round. In this case, the adversary can figure out who Alice's recipient is (say, it's Bob) by observing who does not receive an onion at the end of the protocol run.

The motivating example illustrates that while *mixing* (i.e. sufficiently shuffling onions) is helpful for achieving anonymity, it is not enough. To be anonymous, the protocol must also guarantee that the numbers of messages received by the parties don't reveal the input. We call this property, *equalizing*.

Here, we provide formal game-based definitions of anonymity (Section 3.1), equalizing (Section 3.2), and mixing (Section 3.3). Given these definitions, it can be shown that for indifferent onion routing protocols, equalizing and mixing imply anonymity:

► **Theorem 3.** *For any adversary class \mathbb{A} , an indifferent (Definition 2) onion routing protocol that mixes and equalizes for \mathbb{A} in the simple I/O setting is anonymous for \mathbb{A} in the simple I/O setting, provided that the underlying onion encryption is secure (i.e., UC-realizes the ideal functionality for onion encryption [2]).*

We omit the proof for brevity. We will use Theorem 3 to prove our upper bound in Section 6.3.

3.1 Anonymity

Anonymity is a property of a messaging protocol Π (i.e., Π doesn't have to be an onion routing protocol).

In the anonymity game (for defining anonymity), the adversary necessarily learns the corrupt parties' inputs and received messages. For example, let $N = 4$, and let P_3 be a corrupt party. Suppose that the adversary chooses as inputs $\sigma^0 = (\sigma_1^0, \sigma_2^0, \sigma_3^0, \sigma_4^0)$ and $\sigma^1 = (\sigma_1^1, \sigma_2^1, \sigma_3^1, \sigma_4^1)$ such that $\sigma_3^0 \neq \sigma_3^1$. Then, the adversary can determine the input from P_3 's input. Suppose that the adversary chooses as inputs σ^0 and σ^1 such that σ^0 contains an instruction to send message m^0 to P_3 , whereas σ^1 contains an instruction to send message $m^1 \neq m^0$ to P_3 . Then, the adversary can determine the input from P_3 's received message. Thus, the adversary's choice for (σ^0, σ^1) is constrained to pairs of inputs that differ only in the honest parties' inputs and "outputs."

We define this formally by first defining equivalence classes for inputs as follows. Let Σ be a set of input vectors. Let \mathcal{A} be the adversary, and let Bad be the set of parties controlled by \mathcal{A} . Fixing Bad imposes an equivalence class on Σ . Each equivalence class is defined by a vector (e_1, e_2, \dots, e_N) . For each corrupted party $P_i \in \text{Bad}$, $e_i = (\sigma_i, \mathcal{M}_i)$ "fixes" the input σ_i for P_i and also, the set \mathcal{M}_i of messages instructed to be sent from honest parties to P_i . For each honest party $P_i \in \mathcal{P} \setminus \text{Bad}$, $e_i = V_i$ "fixes" the number V_i of messages instructed to be sent from honest parties to P_i . An input vector belongs to the equivalence class (e_1, e_2, \dots, e_N) if for every $P_i \in \text{Bad}$, the input for P_i is σ_i , the set of messages from honest parties to P_i is \mathcal{M}_i , and $e_i = (\sigma_i, \mathcal{M}_i)$; and if for every $P_i \in \mathcal{P} \setminus \text{Bad}$, the number of messages from honest parties to P_i is V_i , and $e_i = V_i$. Two input vectors σ^0 and σ^1 are equivalent w.r.t. the adversary's choice Bad for the corrupted parties, denoted $\sigma^0 \equiv_{\text{Bad}} \sigma^1$, if they belong to the same equivalence class imposed by Bad .

We now describe the anonymity game (below); the protocol Π is anonymous if this induces indistinguishable adversarial views.

The anonymity game. $\text{AnonymityGame}(1^\lambda, \Pi, \mathcal{A}, \Sigma)$ is parametrized by the security parameter 1^λ , a protocol Π , an adversary \mathcal{A} , and a set Σ of input vectors.

First, the adversary \mathcal{A} and the challenger \mathcal{C} set up the parties' keys: \mathcal{A} chooses a subset $\text{Bad} \subseteq \mathcal{P}$ of the parties to corrupt and sends Bad to the challenger \mathcal{C} . For each honest party in $\mathcal{P} \setminus \text{Bad}$, \mathcal{C} generates a key pair for the party; the public keys $\text{pk}(\mathcal{P} \setminus \text{Bad})$ of the honest parties are sent to \mathcal{A} . \mathcal{A} picks the keys for the corrupted parties and sends the corrupted parties' public keys $\text{pk}(\text{Bad})$ to \mathcal{C} .

Next, the input is selected: \mathcal{A} picks two input vectors $\sigma^0, \sigma^1 \in \Sigma$ such that $\sigma^0 \equiv_{\text{Bad}} \sigma^1$ and sends them to \mathcal{C} . \mathcal{C} chooses a random bit $b \leftarrow_{\$} \{0, 1\}$ and interacts with \mathcal{A} in an execution of protocol Π on input σ^b with \mathcal{C} acting as the honest parties adhering to the protocol and \mathcal{A} controlling the corrupted parties.

At the end of the execution, \mathcal{A} computes a guess b' for b from its view $\mathcal{V}^{\Pi, \mathcal{A}, \text{Bad}}(1^\lambda, \sigma^b)$ and wins the anonymity game if $b' = b$.

The standard notion of anonymity is defined as follows:

► **Definition 4 (Anonymity).** *A messaging protocol $\Pi(1^\lambda, \text{pp}, \text{states}, \$, \sigma)$ is anonymous from the adversary class \mathbb{A} w.r.t. the input set Σ if every adversary $\mathcal{A} \in \mathbb{A}$ wins the anonymity game $\text{AnonymityGame}(1^\lambda, \Pi, \mathcal{A}, \Sigma)$ with only negligible advantage, i.e., $|\Pr[\mathcal{A} \text{ wins } \text{AnonymityGame}(1^\lambda, \Pi, \mathcal{A}, \Sigma)] - \frac{1}{2}| = \text{negl}(\lambda)$.*

The protocol is computationally (resp. statistically) anonymous if the adversaries in \mathbb{A} are computationally bounded (resp. unbounded).

3.2 Equalizing

Here, we introduce a new concept called equalizing, which is closely related to anonymity. Like anonymity, equalizing is a property of a messaging protocol Π .

Informally, Π equalizes if observing how many messages each party received during the protocol run does not reveal whether the protocol ran on σ^0 or σ^1 . In Π_p (in our motivating example), whether Bob receives a message or not exposes who was sending Bob the message: Alice or another party, Allison; so Π_p does not equalize. Instead, in an equalizing protocol, the probability that Bob receives a message doesn't depend on the sender's identity. Put another way, Bob is expected to receive the same number of messages in the scenario where Alice is the sender as the one where it is Allison. Formally, equalizing is defined with respect to the equalizing game (below).

The equalizing game. $\text{EqualizingGame}(1^\lambda, \Pi, \mathcal{A}, \mathcal{D}, \Sigma)$ is parametrized by the security parameter 1^λ , a protocol Π , an adversary \mathcal{A} , a distinguisher \mathcal{D} , and a set Σ of input vectors.

The challenger for the equalizing game first interacts with the adversary exactly the same way as the challenger for the anonymity game. (See the previous section, Section 3.1, for the description of the anonymity game.) Recall that at the end of the anonymity game, each honest party P_r outputs the set $\mathcal{O}_r^{\Pi, \mathcal{A}}(1^\lambda, \sigma^b)$ of (non-empty) messages from the message space \mathcal{M} that it obtained during the execution from processing onions. Let v_r be the number of messages that P_r received during the run, i.e., $v_r \stackrel{\text{def}}{=} |\mathcal{O}_r^{\Pi, \mathcal{A}}(1^\lambda, \sigma^b)|$. (These statistics are part of the adversary's view in the anonymity game.)

We define the statistics for the corrupt parties differently since \mathcal{C} does not get to observe how many messages the corrupt parties output; indeed it is not even clear what it means for a corrupt party to produce an output. For each recipient $P_r \in \text{Bad}$, let v_r correspond to the number of onions that \mathcal{C} has routed to an adversarial participant P' such that (1) they had been formed by an honest participant with P_r as the recipient; and (2) all the participants after P' on the remainder of this onion's route are controlled by the adversary. In other

words, v_r is the number of onions from honest participants that P_r would receive if, internal to the adversary, all the onions are processed and delivered to their next destinations. We define this formally below.

Let $\text{msPairs}(P_r)$ denote the set of message-sender pairs for P_r . That is, for every $(m, P_s) \in \text{msPairs}(P_r)$, the input σ_s for P_s includes the message-recipient pair (m, P_r) , i.e., $(m, P_r) \in \sigma_s$. Let $\text{receivableOnions}(P_r)$ be the following set of onions: An onion O is in $\text{receivableOnions}(P_r)$ if there exists a message-sender pair $(m, P_s) \in \text{msPairs}(P_r)$ such that

- i. O was formed by \mathcal{C} (on behalf of P_s) by running FormOnion on input the message m , a routing path $P^\rightarrow = (Q_1, \dots, Q_{d-1}, P_r)$ ending in P_r , the public keys $\text{pk}(P^\rightarrow)$ of the parties on the path, and a sequence s^\rightarrow of nonces, i.e., $O \in \{O_1, \dots, O_d\}$ where $(O_1, \dots, O_d) \leftarrow \text{FormOnion}(m, (P^\rightarrow), \text{pk}(P^\rightarrow), s^\rightarrow)$;
- ii. letting i denote the position of O in the output of the FormOnion call, either $i = 1$, or the $(i - 1)^{\text{st}}$ intermediary Q_{i-1} on the routing path is honest; and
- iii. O is “peelable all the way” by \mathcal{A} ; i.e., Q_i, \dots, Q_{d-1}, P_r are all adversarial.

For each adversarial recipient $P_r \in \text{Bad}$, we define the statistic v_r to be the number of onions in $\text{receivableOnions}(P_r)$ that the challenger sent out during the execution.

Let $\mathbf{v} = (v_1, v_2, \dots, v_N)$. \mathcal{C} provides these statistics \mathbf{v} alone (and not the rest of the view) to the distinguisher \mathcal{D} , who outputs a guess b' for the challenge bit and wins the game if $b' = b$, i.e. if it correctly determines whether the challenger ran the protocol on input σ^0 or σ^1 . The definition for equalizing is as follows:

► **Definition 5** (Equalizing). *A messaging protocol $\Pi(1^\lambda, \text{pp}, \text{states}, \$, \sigma)$ equalizes for the adversary class \mathbb{A} w.r.t. the input set Σ if for every adversary $\mathcal{A} \in \mathbb{A}$ and distinguisher \mathcal{D} , \mathcal{D} wins $\text{EqualizingGame}(1^\lambda, \Pi, \mathcal{A}, \mathcal{D}, \Sigma)$ with negligible advantage, i.e., $|\Pr[\mathcal{D} \text{ wins } \text{EqualizingGame}(1^\lambda, \Pi, \mathcal{A}, \mathcal{D}, \Sigma)] - \frac{1}{2}| = \text{negl}(\lambda)$.*

The protocol computationally (resp. statistically) equalizes if the adversaries and the distinguishers are computationally bounded (resp. unbounded).

Clearly, a protocol that satisfies anonymity must equalize:

► **Theorem 6.** *For any adversary class \mathbb{A} , a protocol that is anonymous for \mathbb{A} w.r.t. the input set Σ equalizes for \mathbb{A} w.r.t. Σ .*

Proof. If \mathcal{D} can guess b based on the statistics \mathbf{v} alone, then the adversary \mathcal{A} who has access to the entire view of its interaction with \mathcal{C} can guess b also. (It is also easy to see that a protocol need not satisfy anonymity in order to satisfy equalizing. Thus, equalizing is necessary but not sufficient to achieve anonymity.) ◀

3.3 Mixing in the simple I/O setting

Mixing is a property of *onion routing* protocols. Informally, an onion routing protocol mixes if the protocol sufficiently shuffles the honest parties’ “message-bearing” onions. That is, once an honestly generated onion has traveled far enough, getting peeled at every intermediary, the adversary cannot trace it to the original sender. If the adversary is the recipient of the message contained in the onion, it should not be able to trace it to the sender provided the message itself does not reveal the sender.

Formally, mixing is defined with respect to the mixing game. To keep things simple, we present the definition in the simple I/O setting, but this can be extended to any arbitrary input set.

The mixing game. Let $\mathcal{OE} = (\text{Gen}, \text{FormOnion}, \text{ProcOnion})$ be a secure onion encryption scheme. $\text{MixingGame}(1^\lambda, \Pi, \mathcal{A})$ is parametrized by the security parameter 1^λ , an onion routing protocol Π , and an adversary \mathcal{A} .

First, the adversary \mathcal{A} and the challenger \mathcal{C} set up the parties' keys (exactly as we described above for the anonymity game): \mathcal{A} chooses a subset $\text{Bad} \subseteq \mathcal{P}$ of the parties to corrupt and sends Bad to \mathcal{C} . For each honest party in $\mathcal{P} \setminus \text{Bad}$, \mathcal{C} generates a key pair for the party by running the onion encryption scheme's key generation algorithm Gen and sends the public keys $\text{pk}(\mathcal{P} \setminus \text{Bad})$ of the honest parties to the adversary \mathcal{A} . \mathcal{A} picks the keys for the corrupted parties and sends the public-key portions $\text{pk}(\text{Bad})$ to \mathcal{C} .

Next, the input is selected: \mathcal{A} identifies a set $\text{S} \subseteq \mathcal{P} \setminus \text{Bad}$ of honest *target senders* and a set $\text{R} \subseteq \mathcal{P}$, $|\text{R}| = |\text{S}|$ of *target receivers*. In addition to S and R , \mathcal{A} also decides *part* of the input; for every non-target sender $P_s \in \mathcal{P} \setminus \text{S}$, \mathcal{A} chooses a message m and a unique non-target recipient $P_r \in \mathcal{P} \setminus \text{R}$ such that P_s 's input becomes $\sigma_s = \{(m, P_r)\}$; and for every target recipient $P_r \in \text{R}$, \mathcal{A} chooses a message m_r to be sent to P_r . We call the portion of the input that \mathcal{A} decides “*the partial input vector*,” and denote it $\tilde{\sigma}$. \mathcal{A} sends $(\text{S}, \text{R}, \tilde{\sigma})$ to the challenger \mathcal{C} . \mathcal{C} supplies the rest of the input vector $\sigma = (\sigma_1, \dots, \sigma_N)$ by choosing a random bijection g from S to R ; each $P_s \in \text{S}$ is instructed to send the message $m_{g(P_s)}$ to $g(P_s) \in \text{R}$, i.e., $\sigma_s = \{(m_{g(P_s)}, g(P_s))\}$ where the message $m_{g(P_s)}$ was supplied by \mathcal{A} as part of the partial input vector.

Next, \mathcal{C} interacts with \mathcal{A} in an execution of protocol Π on input σ with \mathcal{C} acting as the honest parties adhering to the protocol and \mathcal{A} controlling the corrupted parties. Whenever the protocol Π specifies for an onion to be formed or processed, \mathcal{C} runs the onion encryption scheme's onion-forming algorithm FormOnion or onion-processing algorithm ProcOnion .

Let O_R be the set of onions received by the parties in R .

At the end of the execution, \mathcal{A} chooses two onions $O_s, O_{\bar{s}} \in \text{O}_\text{R}$ and a target sender $P_s \in \text{S}$ and outputs $(O_s, O_{\bar{s}}, P_s)$.

Let an onion O be a “*valid challenge onion*” if (i) there exists a message $m_r \in \mathcal{M}$ and a target recipient $P_r \in \text{R}$ such that m_r is \mathcal{A} 's choice for the message to be sent to P_r , and (ii) O is the last onion to be received by the recipient over the network in the onion evolution generated by \mathcal{C} on behalf of one of the target senders running FormOnion on the message m_r and a routing path ending in P_r .

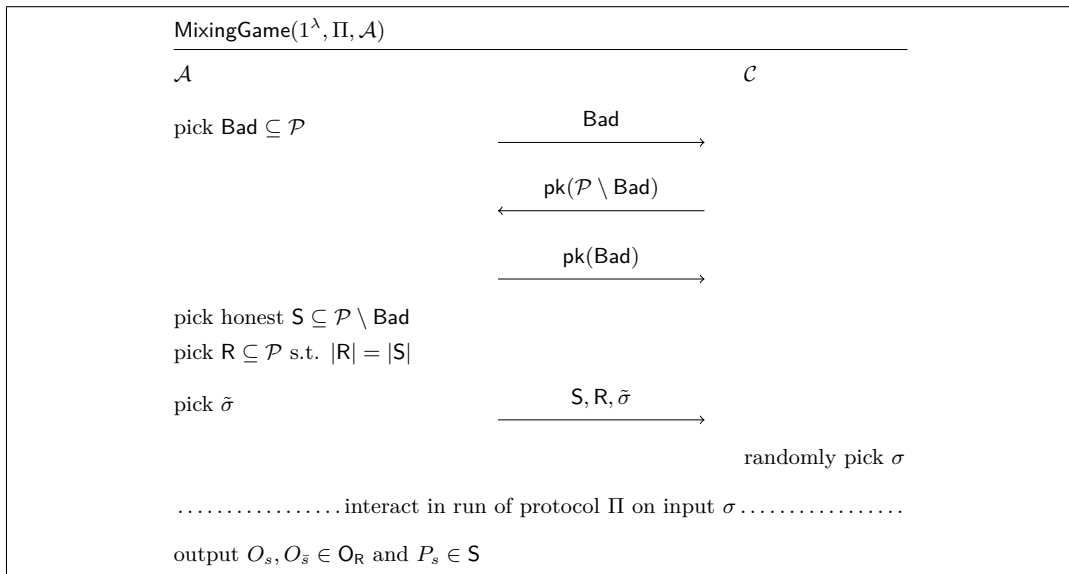
Let $\text{sender}(O_s)$ be the sender of O_s , and let $\text{sender}(O_{\bar{s}})$ be the sender of $O_{\bar{s}}$. To maximize his chances of winning the game, the adversary wants both O_s and $O_{\bar{s}}$ to be valid challenge onions such that O_s was sent by P_s , while $O_{\bar{s}}$ was not. Formally, if \mathcal{A} chose two valid challenge onions, and $\{P_s\} \subset \{\text{sender}(O_s), \text{sender}(O_{\bar{s}})\} \subseteq \text{S}$, then \mathcal{A} wins iff $P_s = \text{sender}(O_s)$. Otherwise, if \mathcal{A} did not choose two valid challenge onions, or if $\{P_s\} \not\subseteq \{\text{sender}(O_s), \text{sender}(O_{\bar{s}})\}$ or $\{\text{sender}(O_s), \text{sender}(O_{\bar{s}})\} \not\subseteq \text{S}$, then \mathcal{A} wins with probability one-half. See Figure 1 for a quick reference to the mixing game.

We now define mixing as follows.

► **Definition 7 (Mixing).** *An onion routing protocol $\Pi(1^\lambda, \text{pp}, \text{states}, \$, \sigma)$ mixes conditioned on the event E for the adversary class \mathbb{A} if given E , every adversary $\mathcal{A} \in \mathbb{A}$ wins $\text{MixingGame}(1^\lambda, \Pi, \mathcal{A})$ with negligible advantage, i.e., $|\Pr[\mathcal{A} \text{ wins } \text{MixingGame}(1^\lambda, \Pi, \mathcal{A}) \mid E] - \frac{1}{2}| = \text{negl}(\lambda)$.*

The protocol computationally (resp. statistically) mixes if the adversaries in \mathbb{A} are computationally bounded (resp. unbounded).

Now that we have defined mixing formally, let us walk the reader through our definitional choices. The starting intuition is that this definition needs to capture that it should be hard for the adversary to pinpoint the origin of an onion received by one of the target recipients.



■ **Figure 1** Schematic of the mixing game.

This goal comes with a caveat that of course an adversary can determine the sender of an onion that one of the target senders has just created, or, more generally, that hasn't traveled very far and hasn't had a chance to mix with any onions from other target senders. Hence, we need to restrict the set of onions on which the adversary can win to a set of onions that have traveled far and have already had a chance to mix with other onions. This is why we have the requirement that the onion be a valid challenge onion. Intuitively, a valid challenge onion is one that was formed by a target sender and has already arrived at its destination, a target recipient, and now the adversary's job is to figure out where it came from.

Next, let us explain why, to win the game, the adversary must produce two valid challenge onions, and correctly attribute one of them to a sender P_s , while the other must have originated with another target sender. What does it mean that the adversary cannot trace an onion? One intuitive approach would be to say: the adversary's chances of winning the game where he picks just one onion and guesses its origin are close to a simulator's chances of winning a game where he just guesses a sender, and the challenger picks the onion uniformly at random and independently of the simulator's guess. The problem with this approach is that we don't know the best strategy for such a simulator and with what probability it would succeed. So our approach is to have the adversary pick a sender and two onions. "Mixing" means that, if it so happens that exactly one of them comes from P_s and the other comes from another target sender, then try as he may, the adversary cannot tell which is which any better than by guessing randomly; and if it doesn't happen that way, then the adversary wins with probability one-half.

4 Main tools: checkpoint onions and merging onions

We describe the main ingredients for our constructions: checkpoint onions (a tool that was introduced in prior work [3]) and a new tool: merging onions.

4.1 Checkpoint onions

Our goal is to achieve anonymity by ensuring that our protocol mixes and equalizes in the presence of an active adversary that drops onions. The challenge is: if the adversary drops too many onions, then the remaining ones don't have enough onions to mix with, and so the resulting protocol will not mix. Checkpoint onions give the honest participants a way of checking that there are still enough onions in the system for mixing to be possible.

A *checkpoint onion* O is a dummy onion (containing the empty message \perp) formed by a party P that travels through the network until, at a pre-determined checkpoint round r , it arrives at the intermediary I , who is expecting it. If it fails to arrive, then I is alerted to the activity of an active adversary. More precisely, let $F(\cdot, \cdot)$ be a pseudo-random function over two inputs, keyed by $\text{sk}(P, I)$ which is a secret key shared between P and I . Let b be a binary predicate. Let \mathcal{D} be the *diagnostic rounds*; the honest parties test whether enough onions remain in the system after these rounds. For each intermediary I and each round $r \in \mathcal{D}$, P determines whether or not to create a checkpoint onion that will arrive at I at round r by computing $f = F(\text{sk}(P, I), (r, 0))$ and then checking if $b(f) = 1$; if so, P creates this checkpoint onion. Similarly, the intermediary I will know to expect a checkpoint onion from P at round r by computing $f = F(\text{sk}(P, I), (r, 0))$ and then checking if $b(f) = 1$.

P forms O by running `FormOnion` on input the empty message \perp , a randomly chosen routing path $P^\rightarrow = (I_1, \dots, I_d)$, the public keys associated with parties on P^\rightarrow , and a sequence (s_1, \dots, s_{d-1}) of nonces. The nonce s_r which will be received by I , is the value that I will know to expect: $s_r = F(\text{sk}(P, I), (r, 1))$; the rest are random nonces. The reason that I will know to expect s_r is that I can compute it too, since $\text{sk}(P, I)$ is shared between P and I . Of course, the shared key $\text{sk}(P, I)$ need not be set up in advance: it can be generated from an existing PKI, e.g., using Diffie-Hellman.

If the adversary drops an onion belonging to the same evolution as O before it reaches I , I will detect it: it will detect that no onion with nonce s_r was received in round r . (Since F is pseudorandom, it is highly unlikely that another onion peels to the same nonce value.)

4.2 Merging onions

Checkpoint onions help with mixing, but not with equalizing. If our routing protocol just has every sender form one “message-bearing” onion to its recipient and send it along in addition to a set of checkpoint onions (as in the protocol Π_a of Ando, Lysyanskaya, and Upfal [3]), then an adversary who targets the sender Alice can cause Alice's recipient Bob to receive the message with a smaller probability than her alternative recipient, Bill; so this protocol will not equalize and, from Theorem 6, has no hope of achieving anonymity.

So how can we design a protocol that equalizes? One approach is to detect when the adversary drops any onions at all (e.g., using verifiable shuffling) [25, 30] and abort when that happens. While this approach equalizes, it is not at all fault-tolerant. To achieve fault tolerance and equalizing, the protocol must be able to react to the adversary dropping onions in a way that is less dramatic than total abort. This can be accomplished by using a new tool: merging onions. The idea here is that a sender P can create two onions, O_1 and O_2 that bear the same message to the same recipient R . Further, they will be routed through the same intermediary I , arriving at I at the same round r . Let O'_1 (resp. O'_2) denote the r^{th} layer of O_1 (resp. O_2) that arrives at I at round r . When I peels both O'_1 and O'_2 , I discovers that they are (essentially) the same onion, and only forwards one of them to the next destination. If I receives just one of them (because the other one had been dropped by the adversary), then it forwards it to the next destination too.

Why does this approach help with equalizing? Suppose we have a protocol in which every participant creates two message-bearing onions that merge at round r . Suppose that the adversary targets the sender Alice and succeeds in dropping one of her two outgoing merging onions. Since these onions were supposed to merge at round r , after round r , there are just as many onions for which Alice was the sender (namely, just one onion) as for any other participant. In general, of course, the adversary may drop more than one onion belonging to Alice. In fact, in order to guarantee that any of Alice’s onions survive with overwhelming probability when the adversary controls a constant fraction of the network’s nodes, Alice needs to send out a superlogarithmic (in the security parameter λ) number of onions. In order to equalize the number of onions that make it to each destination, our protocol will have to create not a pair, but $2^h = \Omega(\text{polylog } \lambda)$ merging onions, organized in a binary tree of height h .

We now illustrate how to form 2^h merging onions through a toy example for $h = 3$. We first construct a binary tree graph of height $3 = \log 8$. We label the root vertex of the tree v , and the left-child and right-child of v , v_0 , and v_1 . More generally, the left-child of a vertex v_w is v_{w0} , and the right-child of v_w is v_{w1} , so that the leaf vertices are: v_{000} , v_{001} , v_{010} , v_{011} , v_{100} , v_{101} , v_{110} , and v_{111} . Each of these leaf vertices corresponds to a separate onion.

Let y denote a fixed number of rounds; this will later correspond to the length of an “epoch.” Next, for each vertex v_i of the graph, we choose a random sequence $I_i^{\rightarrow} = (I_i^1, \dots, I_i^y)$ of y parties and a random sequence $s_i^{\rightarrow} = (s_i^1, \dots, s_i^y)$ of y nonces, i.e., $\forall j \in [y]$, $I_i^j \leftarrow \mathcal{P}$ and $s_i^j \leftarrow \mathcal{S}$. Let the “direct path from a leaf vertex v_ℓ to the root” be the path that begins with v_ℓ and recursively moves to its parent vertex until the root vertex v is reached. For example, the direct path from v_{101} to the root is $(v_{101}, v_{10}, v_1, v)$. Let the “sequence of intermediaries corresponding to leaf vertex v_ℓ ” be the sequence of parties corresponding to the parties on the direct path from v_ℓ to the root, e.g., for v_{101} , it is $(I_{101}^{\rightarrow}, I_{10}^{\rightarrow}, I_1^{\rightarrow}, I^{\rightarrow})$, where I^{\rightarrow} is the sequence of parties assigned to the root. Let the “sequence of nonces corresponding to leaf vertex v_ℓ ” be the sequence of nonces corresponding to the parties on the direct path from v_ℓ to the root, e.g., for v_{101} , it is $(s_{101}^{\rightarrow}, s_{10}^{\rightarrow}, s_1^{\rightarrow}, s^{\rightarrow})$, where s^{\rightarrow} is the sequence of nonces assigned to the root.

For each leaf vertex v_ℓ , we form an onion O_ℓ^1 using the message m from the input, the routing path $(I_{101}^{\rightarrow}, I_{10}^{\rightarrow}, I_1^{\rightarrow}, I^{\rightarrow}, R)$ where R is the recipient from the input, the public key associated with the routing path, and the sequence $(s_{101}^{\rightarrow}, s_{10}^{\rightarrow}, s_1^{\rightarrow}, s^{\rightarrow})$ of nonces. We can generalize this idea to generate an arbitrarily large set of merging onions by using an appropriately large binary tree.

5 A stepping stone construction, Π_Δ

Let us extend the toy example construction we just saw to a protocol, $\Pi_\Delta^{x,y,t}$, which is a stepping stone for our main construction. $\Pi_\Delta^{x,y,t}$ is pronounced “Pi-tree” from the fact that the onions’ routing paths are structured like a binary tree graph and is parametrized by the number x of merging onions per sender (this is also the expected number of checkpoint onions per sender), the number y of rounds per epoch, and the threshold t for missing checkpoint nonces per diagnostic round. (We will generally omit the superscript for better readability.)

We use a secure onion encryption scheme $\mathcal{OE} = (\text{Gen}, \text{FormOnion}, \text{ProcOnion})$ as a building block. During the setup phase, the participants set up their keys. Every honest party P sets up his/her keys $(\text{pk}(P), \text{sk}(P))$ by running the onion encryption scheme’s key generation algorithm Gen .

The onion-forming phase. During the onion-forming phase, each honest party P creates two types of onions: merging onions and checkpoint onions.

- On input $\{(m, R)\}$, P forms a set of x merging onions using the number y of rounds in an epoch, the message m , and the recipient R .
- In addition to merging onions, P generates (on average) x checkpoint onions using the set $\mathcal{D} = \{y, 2y, \dots, (\log x + 1)y\}$ as the diagnostic rounds. (Appropriate functions are chosen for $F(\cdot, \cdot)$ and $b(\cdot)$ such that P generates x checkpoint onions in expectation. See *Checkpoint onions* in Section 4 to recall how these functions are used for generating checkpoint onions.)

For both merging onions and checkpoint onions, the length of the routing path is fixed; it is $(\log x + 1)y + 1$.

The execution phase. All onions are created during the onion-forming phase and released simultaneously in the first round of the execution phase.

- After each round r of the execution phase, P peels all onions it received at the r^{th} round and merges mergeable onions (i.e., if two onions peel to the same nonce value, drop one of them at random).
- If r is a diagnostic round (i.e., $r \in \mathcal{D}$), P runs the following diagnostic test: Let $\text{Ckpts}(P, r)$ denote the set of checkpoints that P expects to see from peeling the onions between rounds r and $r + 1$. P counts how many checkpoints from $\text{Ckpts}(P, r)$ are missing. If the number exceeds a fixed threshold value t , then P aborts. Otherwise, P continues for another round by sending the processed onions to their respective next destinations in random order.
- At the end of the execution phase, P peels the onions it received at the last round and outputs the set of (non-empty) messages it received.

► **Remark 8.** $\Pi_{\Delta}^{x,y,t}$ is anonymous from the adversary who corrupts up to κ fraction of the parties when (i) the onion encryption scheme is secure, (ii) the number x of onions formed by each (honest) party is $\Omega(2^{\lceil \log(\chi(\log \chi + 1)) \rceil})$ where $\chi = \max(\sqrt{N \log^{2+\epsilon} \lambda}, \log^{2(1+\epsilon)} \lambda)$, (iii) the number y of rounds per epoch is $\Omega(\log^{1+\epsilon} \lambda)$, and (iv) the threshold t is $2(1 - \delta)(1 - \kappa)^3 \kappa \log^{1+\epsilon} \lambda$. (See the full version of the paper for the proof.) The reason that $\Pi_{\Delta}^{x,y,t}$ needs so many onions is that the adversary can target Alice and drop a lot of her onions before the honest participants realize (via checkpoint onions) the presence of an attack and abort. The protocol Π_{\bowtie} presented in the next section improves on this by giving the routing paths enough structure that missing onions can be detected sooner.

6 Our main construction, Π_{\bowtie}

In this section, we present our main construction Π_{\bowtie} (pronounced “Pi-butterfly”). Π_{\bowtie} uses a variant of a butterfly graph described below.

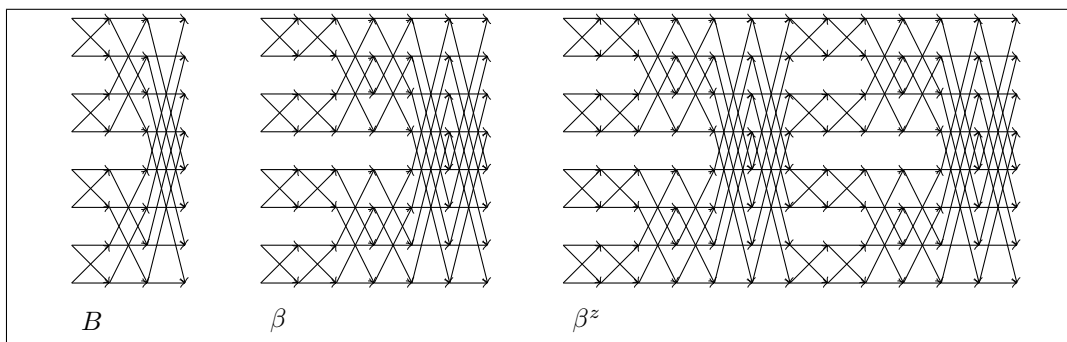
6.1 The butterfly network and variants

Recall [26, Chapter 4.5.2] that the *butterfly network* $B = (V(B), E(B))$ is a directed graph on $(n + 1)2^n$ vertices. The vertices are organized into $N = 2^n$ rows and $n + 1$ columns, so each vertex has an address (r, c) where $1 \leq r \leq N$ and $0 \leq c \leq n$. Vertices in column i represent potential locations of a data packet (here, an onion) at epoch i ; each participant P has a dedicated row. An edge from (P, i) to $(Q, i + 1)$ means that an onion can travel from participant P to participant Q in epoch i . The edges of the specific butterfly network that will be useful for us are $E(B) = \{((P, i), (Q, i + 1)) \mid P = Q \text{ or binary representations of } P \text{ and } Q \text{ differ in position } i + 1 \text{ only}\}$.

Let J and J' be two participants whose binary representation differs in bit $i + 1$ only. In Π_{\boxtimes} , epoch i is dedicated to having an onion bounce y times between J and J' . This way, by the end of the epoch, the onions that J and J' held at the beginning of the epoch will be mixed together if one of them is honest. More formally, the onions travel along the edges of a *stretched* butterfly network, defined as follows: its $N(ny + 1)$ vertices are organized into N rows and $ny + 1$ columns; and its edges are: $E(\beta) = \{((P, j), (Q, j + 1)) \mid \text{for } i = \lfloor j/y \rfloor, ((P, i), (Q, i + 1)) \in E(B)\}$.

However, what if both J and J' are adversarial? Then sending the onions through the stretched butterfly network just once will result in the adversary knowing the i^{th} bit of an onion's destination! So to prevent this, we will send the onions through the *iterated* stretched butterfly network. For an integer z , let β^z denote the stretched butterfly network iterated z times. More precisely, β^z is a directed graph in which the vertices are organized into N rows and $nyz + 1$ columns, i.e., a vertex has an address (r, c) where $1 \leq r \leq N$ and $0 \leq c \leq nyz$. The edges are as follows: $E(\beta^z) = \{((P, j), (Q, j + 1)) \mid \text{for } i = j \bmod ny, ((P, i), (Q, i + 1)) \in E(\beta)\}$.

To summarize, we begin with a butterfly network B , then we stretch it by y to get β , then we iterate it z times to get β^z ; see Figure 2. By a “walk through β^z ” we mean a sequence (J_0, \dots, J_{nyz}) such that, for each $i < nyz$, $((J_i, i), (J_{i+1}, i + 1)) \in E(\beta^z)$. A random walk from a node J_0 is a sequence that begins with J_0 such that for $i > 0$, each J_i is a walk selected uniformly at random conditioned on the first i elements being (J_0, \dots, J_{i-1}) . A random walk starting at any address can be sampled efficiently.



■ **Figure 2** Diagrams of the butterfly network B , the stretched butterfly network β , and the iterated stretched butterfly network β^z for $n = \log(8) = 3$, and $y = z = 2$.

6.2 Description of the construction

$\Pi_{\boxtimes}^{x,y,z,t}$ consists of setup, the onion-forming phase, and the execution phase. It is parameterized by the number x of merging onions per sender, the number y of rounds per epoch, the number z of iterations of a variant of a butterfly graph, and the threshold t for missing checkpoint nonces. The execution phase is further divided into the *mixing sub-phase* and the *equalizing sub-phase*. The iterated stretched butterfly graph determines routing options for the mixing sub-phase.

Let $\mathcal{OE} = (\text{Gen}, \text{FormOnion}, \text{ProcOnion})$ be a secure onion encryption scheme. During setup, each honest participant P generates its public key pair $(\text{pk}(P), \text{sk}(P))$ using Gen .

6.2.1 The onion-forming phase

On input $\{(m, R)\}$, each honest party P generates exactly x merging onions and (on average) x checkpoint onions. To form an onion, P first needs to pick a path for it. Each onion will (potentially) travel to $d \stackrel{\text{def}}{=} (nyz + 1) + y \log x + 1$ parties to reach its destination: the first $nyz + 1$ steps involve a random walk through the iterated stretched butterfly network (the mixing sub-phase), and the next $y \log x + 1$ steps will take the onion through the equalizing sub-phase and to the recipient.

To begin with, P generates the x merging onions as follows: Let T be the binary tree of height $\log x$. Let k be an address of a node in T (i.e., k is a binary string of length at most $\log x$); let v_k denote this node. I.e., $V(T) = \{v_k \mid k \text{ is a binary string, } |k| \leq \log x\}$. To each non-leaf vertex v_k in T , P assigns a sequence of y random parties and y random nonces; let $I_{v_k}^{\rightarrow} = (I_{v_k,1}, \dots, I_{v_k,y})$ denote the sequence of vertices and $s_{v_k}^{\rightarrow} = (s_{v_k,1}, \dots, s_{v_k,y})$ denote the sequence of nonces corresponding to vertex v_k . (Up until this step, this is exactly how merging onions are formed in Π_{Δ} .) For each leaf vertex v_{ℓ} , P picks a random walk through the iterated stretched butterfly β^z and $nyz + 1$ random nonces; let $J_{v_{\ell}}^{\rightarrow} = (J_{v_{\ell},0}, \dots, J_{v_{\ell},nyz})$, denote the random walk, and let $t_{v_{\ell}}^{\rightarrow} = (t_{v_{\ell},0}, \dots, t_{v_{\ell},nyz})$ be the sequence of nonces.

Let v_{ℓ} be a leaf of T . Let $v_{\ell,i} = v_{k_i}$ where k_i is the i -bit prefix of ℓ . I.e. $v_{\ell,\ell} = v_{\ell}$ and $v_{\ell,0} = v_{\varepsilon}$, and $(v_{\ell,h}, v_{\ell,h-1}, \dots, v_{\ell,0})$ is the path from v_{ℓ} to the root of the tree, where $h = \log x$. P will create an onion O_{ℓ} for each leaf v_{ℓ} . Its routing path is $I_{\ell}^{\rightarrow} = (J_{v_{\ell}}^{\rightarrow}, I_{\ell,2}^{\rightarrow}, \dots, I_{\ell,h}^{\rightarrow}, R)$ where $J_{v_{\ell}}^{\rightarrow}$ is as defined above, $I_{\ell,i}^{\rightarrow} = I_{k_i}^{\rightarrow}$ where k_i is the i -bit prefix of ℓ , and R is the recipient, and such that $|I_{\ell}^{\rightarrow}| = d$. Similarly, let $s_{\ell}^{\rightarrow} = (t_{v_{\ell}}^{\rightarrow}, s_{\ell,2}^{\rightarrow}, \dots, s_{\ell,h}^{\rightarrow})$ denote the sequence of nonces corresponding to this path. To form the onion O_{ℓ} corresponding to v_{ℓ} , P runs the algorithm `FormOnion` on the message m , the routing path I_{ℓ}^{\rightarrow} , the public keys associated with the routing path, and the nonce sequence s_{ℓ}^{\rightarrow} .

After forming the merging onions, P generates the checkpoint onions. Just as in Π_{Δ} , the execution phase consists of epochs, and the last round of every epoch is a diagnostic round. Here, each epoch lasts y rounds, thus round $r > 0$ is a diagnostic round if r is a multiple of y . For each diagnostic round r and for each intermediary I , P uses the pseudorandom function $F_{\text{sk}(P,I)}(r, 0)$ to determine whether to form a checkpoint onion to send to I at round r , and if so, calculates the nonce $s = F_{\text{sk}(P,I)}(r, 1)$.

When $F_{\text{sk}(P,I)}(r, 0) = 1$, P generates a checkpoint onion to be verified by party I in round r . Recall that $d \stackrel{\text{def}}{=} (nyz + 1) + y \log x + 1$; so round d is the last round of the execution phase. Since the checkpoint onion should not be distinguishable from a merging one during the mixing sub-phase, it needs to travel over the edges of the iterated stretched butterfly network for the first $nyz + 1$ rounds, and follow a random path through the network during the equalizing sub-phase, all the way until the last round d .

As a result, for $r \geq nyz + 1$, P generates the routing path by first picking a random walk $J^{0 \rightarrow nyz} = (J_0, \dots, J_{nyz})$ through the iterated stretched butterfly network starting at a random node J_0 , and then choosing each participant on the next part of the path $J^{nyz+1 \rightarrow r-1} = (J_{nyz+1}, \dots, J_{r-1})$ uniformly at random from \mathcal{P} . Next, $J_r = I$, and each router on the remaining stretch of the path $J^{r+1 \rightarrow d}$ is, again, chosen uniformly at random from \mathcal{P} . So the resulting routing path is $J_{I,r}^{\rightarrow} = (J^{0 \rightarrow nyz}, J^{nyz+1 \rightarrow r-1}, J_r, J^{r+1 \rightarrow d})$. P chooses the corresponding nonces $\{s_{I,r,j}\}_{j \in \{0, \dots, d-1\} \setminus \{r\}}$ uniformly at random, sets $s_{I,r,r} = s$, and gives the resulting routing path, sequence $(s_{I,r,0}, \dots, s_{I,r,d-1})$ of nonces and the empty message to `FormOnion` to obtain checkpoint onion $O_{I,r}$.

If $r \leq nyz$, then round r occurs during the mixing sub-phase, as the onion is making its way through the butterfly network. So its path has to be formed in such a way that it arrives at I at round r ; but it needs to be a randomly chosen path conditioned on this event (so that

a checkpoint onion's path is distributed the same way as one of a merging onion). Let $J^{0 \rightarrow nyz}$ be a random walk through β^z that is at address I at round r . Let each intermediary in the sequence $J^{nyz+1 \rightarrow d}$ be chosen uniformly at random from \mathcal{P} . Again, for $j \neq r$, $0 \leq j \leq d-1$, the nonce $s_{I,r,j}$ is chosen at random, while $s_{I,r,r} = s$. Let $J_{I,r}^{\rightarrow} = (J^{0 \rightarrow nyz}, J^{nyz+1 \rightarrow d})$. Run `FormOnion` on input the routing path $J_{I,r}^{\rightarrow}$, sequence of nonces $s_{I,r}^{\rightarrow}$ and the empty message to obtain checkpoint onion $O_{I,r}$.

► **Remark 9.** In both Π_{Δ} and Π_{\boxtimes} , the onion layers are tagged with their respective round number to prevent replay attacks. If by peeling an onion received at round r , an honest relaying party observes a round number $r' \neq r$, the party drops the onion. (We can, therefore, assume that replay attacks do not happen. We can safely do so since the security of the onion encryption scheme prevents the adversary from modifying the onions formed by honest participants in any meaningful way. See, for example Ando and Lysyanskaya's work on onion encryption [2], for a sufficiently strong construction.)

6.2.2 The execution phase

At the beginning of the execution phase, each party P is *live*. P 's status will change from live to *aborted* if it ever receives a special abort message from another party. An aborted party sends the special abort message to a random sample of x parties. (A slight technicality is that, since all messages must be onions, the abort message is a specially formed onion.)

- For each $r \in \{0, \dots, d-1\}$, each live honest party P first peels all the onions it received at the r^{th} round. It merges onions that are mergeable: if it received two onions that have the same nonce, then it drops one of them, selected at random, and sends the other one to its next destination.
- If r is a diagnostic round (i.e., a multiple of y), then P runs the diagnostic test: P compares the number of checkpoint onions it expects to receive with the number it received. For every participant $Q \in \mathcal{P}$, if $F_{\text{sk}(Q,P)}(r, 0) = 1$, then P expects to receive a checkpoint onion with nonce $s = F_{\text{sk}(Q,P)}(r, 1)$ in this round. In the mixing sub-phase, if fewer than t checkpoint onions are missing so far in the protocol run (not just in this round, but cumulatively), then P continues the run by processing all the other onions. Otherwise, P 's status changes: it is no longer live but becomes an *aborted* party. In the equalizing sub-phase, change status to aborted if there are t or more missing checkpoint onions in this round, else continue.
- At the last round (round d) of the execution phase, P peels the onions it received and outputs the set of (non-empty) messages it received.

6.3 Proof that Π_{\boxtimes} is anonymous, robust, and efficient

In this section, we will prove that there exists a parameter setting (for x , y , z , and t) such that Π_{\boxtimes} is simultaneously anonymous, fault-tolerant, and efficient.

Our measure of efficiency is *onion cost per user*, which measures how many onions are transmitted by each user in the protocol. This is an appropriate measure when the parties pass primarily onions to each other. It is also an attractive measure of complexity because it is algorithm-independent: If we measured complexity in bits, it would change depending on which underlying encryption scheme was used. Since an onion contains as many layers as there are intermediaries, its bit complexity scales linearly with the number of intermediaries. (We assume that every message m can be contained in a single onion.) To translate our lower bound from onion complexity to bits, we will consider onions to be at least as long (in bits) as the message m being transmitted and the routing information. More formally,

► **Definition 10** (Onion cost). Let $\text{out}_i^{\Pi, \mathcal{A}}(1^\lambda, \sigma)$ denote the number of onions formed by an honest party that party P_i transmits directly to another party in a protocol run of Π with adversary \mathcal{A} , security parameter λ and σ . The onion cost of Π is $\text{OC}^{\Pi, \mathcal{A}}(1^\lambda, \Sigma) \stackrel{\text{def}}{=} \mathbb{E}_{\sigma, i, \$} \left[\text{out}_i^{\Pi, \mathcal{A}}(1^\lambda, \sigma) \right]$. The expectation is taken over the input $\sigma \leftarrow \$\Sigma$, the party $P_i \leftarrow \$\mathcal{P}$, and the randomness $\$$ of the protocol.

For an adversary class \mathbb{A} , the onion cost of Π interacting with \mathbb{A} w.r.t. Σ is the maximum onion cost over the adversaries in \mathbb{A} , i.e., $\text{OC}^{\Pi, \mathbb{A}}(1^\lambda, \Sigma) \stackrel{\text{def}}{=} \max_{\mathcal{A} \in \mathbb{A}} \text{OC}^{\Pi, \mathcal{A}}(1^\lambda, \Sigma)$.

Our formal notion of fault tolerance is *robustness*, defined below:

► **Definition 11** (Robustness). A messaging protocol Π is robust if in every interaction in which the adversary drops at most a logarithmic (in the security parameter) number of message packets, Π delivers all messages sent out by honest participants w.o.p.

Let \mathbb{A}_κ denote the class of active adversaries who can corrupt up to a constant κ fraction of the participants. In this section, we will prove the following upper bound on onion cost:

► **Theorem 12.** For any constants $\kappa < \frac{1}{2}$ and $\gamma_1, \gamma_2 > 0$, there is a setting of x, y, z , and t such that $\Pi_{\boxtimes}^{x, y, z, t}$ is robust and anonymous from the adversary class \mathbb{A}_κ with onion cost at most $\gamma_1 \log N \log^{3+\gamma_2} \lambda$ (in the presence of \mathbb{A}_κ), where λ is the security parameter and $N = \omega(\log \lambda)$ is the number of participants.

Proof. Recall that the number of corruptions is $\kappa < \frac{1}{2}$. Set ϵ_1 such that $\gamma_1 = 6\epsilon_1^3$ and ϵ_2 such that $\gamma_2 = 3\epsilon_2$. Let $x = y = z = \epsilon_1 \log^{1+\epsilon_2} \lambda$. Let an onion (layer) be *commutable* if (i) an honest party formed it, and (ii) it is not a checkpoint onion for verification by an adversarial party (more precisely, it does not belong to the same evolution as a checkpoint onion for verification by an adversarial party); and let $t = \frac{W}{3}$, where $W = \frac{(1-\kappa)x}{z \log N + \log x}$ is the expected number of commutable checkpoint nonces at a party at a diagnostic round.

Having set the parameters, we wish to show that the protocol $\Pi_{\boxtimes}^{x, y, z, t}$ (a) is robust; (b) has onion cost $\text{OC} \leq \gamma_1 \log N \log^{3+\gamma_2} \lambda$; and (c) is anonymous, provided that the underlying onion encryption scheme is secure.

Part (a) is true by inspection.

To see why (b) follows, recall that each participant forms x merging onions and, on average, x checkpoint onions; let X be the maximum number of onions formed by an honest party. Each of these onions will need to be processed in each round, so $\text{OC} \leq Xd$, where d is the number of rounds. Using Chernoff bounds, $X < 3x$ with overwhelming probability. The number of rounds is $d = (nyz + 1) + y \log x + 1$; for our setting of parameters, therefore, $\text{OC} \leq 6\epsilon_1^3 \log N \log^{3(1+\epsilon_2)} \lambda$.

We show part (c) via a series of lemmas that follow. First, we invoke the UC composition theorem of Canetti [8] in order to replace cryptographic algorithms for onion encryption with ideal encryption; this allows our further analysis to assume that onions reveal nothing to an intermediary I other than the information that is intended for I (Lemma 13). Let $\Pi_{\boxtimes}^{\text{ideal}}$ be the resulting protocol. Next, we argue that $\Pi_{\boxtimes}^{\text{ideal}}$ is an indifferent onion routing protocol (Lemma 15). This is helpful because then we will be able to invoke Theorem 3. Third, we discard, for the purposes of analysis, all the checkpoint onions that are checked by the adversary; we show that if a protocol mixes (resp. equalizes) in this setting, then it mixes (resp. equalizes). Finally, we show that in this setting, Π_{\boxtimes} mixes (Lemma 16) and equalizes (Lemma 17). Then, putting it all together, we get our desired result. ◀

► **Lemma 13.** *Let Π be onion routing protocol that makes use of an onion encryption scheme that is UC-secure [8] under a computational assumption A . Let Π^{ideal} be the same protocol, but the onion encryption scheme is replaced by the ideal onion encryption functionality of Camenisch and Lysyanskaya [7]. If Π^{ideal} is anonymous, then Π is anonymous under assumption A .*

Proof. The Lemma follows by the UC composition theorem of Canetti [8]. ◀

► **Remark 14.** Since CCA2-secure public-key encryption UC-realizes the ideal public-key encryption functionality of Canetti, and in Π_{\bowtie} , the adversary already knows how many layers of a given onion have already been peeled, forming onions by using CCA2-secure encryption to encrypt each layer will also result in an anonymous Π_{\bowtie} .

► **Lemma 15.** Π_{\bowtie}^{ideal} is indifferent.

Proof. In Π_{\bowtie}^{ideal} , the length of each routing path is fixed, and the intermediaries and nonces of honestly formed onion layers do not depend on the input σ to the protocol. The procedure for generating intermediaries and nonces takes as input only the values x , y , and z . Thus, by definition, Π_{\bowtie}^{ideal} is indifferent. ◀

For the subsequent lemmas (Lemmas 16-18), we analyze only *commutable* onions.

► **Lemma 16.** *With parameters x , y , z , and t defined as above, Π_{\bowtie}^{ideal} mixes for the adversary who corrupts up to half of the parties.*

Proof sketch. If Π_{\bowtie}^{ideal} delivers messages in the final round d , then w.o.p., the adversary dropped (at most) a constant fraction of the commutable checkpoint onions before the last epoch: The adversary cannot drop more than a constant fraction of all commutable onions without also dropping a proportional number of checkpoint onions. This is because if the adversary were to drop more than a constant fraction of all commutable onions, then, from known probability concentration bounds [22], w.o.p., the adversary would drop close to a proportional number of checkpoint onions, which, in turn, would cause all honest parties to abort the run. Combining this with Chernoff bounds we get: during each round of the penultimate epoch e , each honest party processed a polylogarithmic (in the security parameter) number of commutable onions. From Chernoff bounds, we also get: during epoch e , each commutable onion went to an honest party a polylogarithmic number of times. Thus, either the Π_{\bowtie}^{ideal} aborts, or it sufficiently shuffles the commutable onions during the penultimate epoch since shuffling for a polylogarithmic number of rounds with a polylogarithmic number of other onions is sufficient for mixing. Either way, Π_{\bowtie}^{ideal} mixes. ◀

► **Lemma 17.** *With parameters x , y , z , and t defined as above, Π_{\bowtie}^{ideal} equalizes for the adversary who corrupts up to half of the parties, who also receives everything about non-commutable onions as an auxiliary input.*

Before proving Lemma 17, let us prove the following:

► **Lemma 18.** *Let Π_{\bowtie}^{ideal} run with parameters x , y , z , and t are as defined above on input σ , with A corrupting up to half of the participants, and receiving an auxiliary input about non-commutable onions as an auxiliary input. If there is an unaborted honest party at the beginning of the equalizing phase, then with overwhelming probability for each honest party P , at least $\frac{1-\kappa}{9}$ of P 's merging onions remained undropped by the adversary at the end of the mixing phase. (Recall that κ is the corruption rate.)*

Proof sketch. In the first round, the adversary \mathcal{A} knows the sender of each commutable onion. As the protocol progresses, \mathcal{A} loses track of this information. Thus, \mathcal{A} 's optimal tactic is to target Alice upfront by dropping every onion that might have come from Alice that is routed to an adversarial party during the first three rounds of the first epoch (as well as the last round of the epoch).

In the first round, some of Alice's onions route to a corrupt party; \mathcal{A} drops all of these. However, from Chernoff bounds, w.o.p., at least a constant fraction of Alice's onion go to an honest party first. Let O be such an onion, and let P be the honest party that receives O in the first round. Recall that during each epoch of the mixing phase, P shuffles onions back and forth with another party P' . \mathcal{A} can attempt to drop O if P' is corrupt. However, even if P' is corrupt, \mathcal{A} cannot drop O if it arrives at P first and remains at P during rounds 2 and 3 (and return to P at round y) – so, using probability concentration bounds, $\frac{1-\kappa}{9}$ of the time. Thus, even if \mathcal{A} employs the optimal tactic for dropping Alice's onions, (at least) $\frac{1-\kappa}{9}$ of Alice's onions will make it to the equalizing phase. Since \mathcal{A} cannot do better than this, this proves Lemma 18. ◀

Proof sketch of Lemma 17. From Lemma 18, if Π_{\boxtimes}^{ideal} continues into the equalizing phase, then a constant fraction of each honest party's merging onions are still in play at the start of the equalizing phase. However, Lemma 18 does not guarantee that there will be an epoch $i > nyz$ such that the number of Alice's merging onions at epoch i , $\text{numMO}_{\text{Alice},i}$, will be close to that of Allison's, $\text{numMO}_{\text{Allison},i}$. To prove that Π_{\boxtimes}^{ideal} equalizes, we need to show that there exists an epoch $i \leq d$ such that (for any two parties Alice and Allison), $\text{numMO}_{\text{Alice},i} \approx \text{numMO}_{\text{Allison},i}$. If \mathcal{A} doesn't drop any commutable onions during the equalizing phase, then this condition is satisfied by the merging of onions.

So what can \mathcal{A} do? The only information that \mathcal{A} has for guessing where any commutable onion came from is which onions are part of a mergeable pair and which are not; this is because the onions are shuffled during the mixing phase and each epoch of the equalizing phase. Let a *singleton* be a commutable onion that is not part of a mergeable pair; note that it can be either a checkpoint onion or a merging onion. W.l.o.g., suppose that \mathcal{A} dropped more of Alice's onions upfront (during the mixing phase) than Allison's. Then, at the start of the equalizing phase, it is likely that more singletons are Alice's merging onions than Allison's merging onions. So, \mathcal{A} can attempt to prevent the numbers of merging onions from evening out by dropping singletons. We can show that the best that \mathcal{A} can do is to drop as many singletons as possible (without causing the protocol to be aborted) at the beginning of the equalizing phase. (Of course, \mathcal{A} could also drop onions that belong in a mergeable pair, but this would only help to even out the numbers of merging pairs.) Even if \mathcal{A} does this, there exists an epoch $i \leq d$ such that $\text{numMO}_{\text{Alice},i} \approx \text{numMO}_{\text{Allison},i}$.

Armed with Lemma 18 and the above analysis, we can prove that Π_{\boxtimes}^{ideal} equalizes. If the adversary drops too many onions during the mixing phase, then Π_{\boxtimes}^{ideal} equalizes since every honest party stops participating (Lemma 18), and so no one receives their message. Otherwise, Π_{\boxtimes}^{ideal} equalizes since enough of each sender's merging onions make it to the equalizing phase (Lemma 18), and the numbers of merging onions are eventually evened out by the merging of onions (above). ◀

7 Our lower bound: polylog onion cost is required

In this section, we present our lower bound: an onion routing protocol can be anonymous from the active adversary only if the onion cost is superlogarithmic in the security parameter. Our lower bound holds for protocols that are minimally functional for the active adversary.

We call this notion weakly robustness, defined below. The reason this definition is weaker than robustness (Definition 11) is that here we only insist that the protocol guarantee delivery for senders whose onions are never dropped.

► **Definition 19** (Weakly robust). *Let $\Pi(1^\lambda, \text{pp}, \text{states}, \$, \sigma)$ be an onion routing protocol and let \mathcal{A} be an adversary attacking Π that drops at most $\mathcal{O}(\log(\lambda))$ onions. Π is weakly robust if whenever \mathcal{A} doesn't drop any onions sent by honest party P , P 's message will be delivered to its recipient with overwhelming probability.*

► **Theorem 20.** *If the onion routing protocol $\Pi(1^\lambda, \text{pp}, \text{states}, \$, \sigma)$ is weakly robust and (computationally) anonymous from the adversary \mathcal{A} who corrupts up to a constant fraction of the parties and drops at most $f(\lambda) = \mathcal{O}(\log(\lambda))$ onions, then the onion cost of Π interacting with \mathcal{A} is $\omega(f(\lambda))$.*

Proof sketch. Let us give the intuition for the proof of this theorem. If an honest P_i sends out only $\mathcal{O}(\log(\lambda))$ onions, then an adversary that chooses which participants to corrupt uniformly at random has a $1/\lambda^{\mathcal{O}(1)}$ chance of controlling each and every participant that ever receives an onion directly from P_i . (This is because $\mathcal{O}(\log(\lambda)) = \mathcal{O}(\log(N))$, since λ and N are polynomially related.) Thus with non-negligible probability it can cut off P_i entirely by dropping all of the onions it sends out, guaranteeing that the intended recipient of P_i 's message never receives the message; yet, by weak robustness (Definition 19), we can show that there will be some recipient whose probability of receiving his message is high. Therefore, Π will not equalize (Definition 5): based on who failed to receive the message, it is possible to determine whether P_i 's intended recipient was Bob or Bill. Since it does not equalize, by Theorem 6, it is not anonymous. See the full version of this paper for the proof. ◀

8 Conclusion and future work

Here, we mention a few extensions of our results: We proved that the required onion cost for an onion routing protocol to provide robustness and (computational) anonymity from the active adversary is polylogarithmic in the security parameter. Our proof for the lower bound can be used to prove the stronger result that polylogarithmic onion cost is required even when (i) the adversary observes the traffic on only $\Theta(1)$ fraction of the links and or when (ii) the security definition is weakened to (computational) differential privacy. (iii) Also, while we explicitly showed this to be the case for the simple I/O setting, the result holds more generally whenever any party can send a message to any other party.

We also proved the existence of a robust and anonymous onion routing protocol with polylogarithmic (in the security parameter) onion cost. (iv) This result also extends beyond the simple I/O setting; our onion routing protocol is anonymous w.r.t. any input set where the size of each party's input is fixed.

There is a small gap between our lower and upper bounds. A natural direction for future work is to close this gap.

References

- 1 Mário S. Alvim, Miguel E. Andrés, Konstantinos Chatzikoikolakis, Pierpaolo Degano, and Catuscia Palamidessi. Differential privacy: on the trade-off between utility and information leakage. In *FAST 2011*, pages 39–54. Springer, 2011.

- 2 Megumi Ando and Anna Lysyanskaya. Cryptographic shallots: A formal treatment of reliable onion encryption. *IACR Cryptol. ePrint Arch.*, 2020:215, 2020. URL: <https://eprint.iacr.org/2020/215>.
- 3 Megumi Ando, Anna Lysyanskaya, and Eli Upfal. Practical and provably secure onion routing. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPICs*, pages 144:1–144:14. Schloss Dagstuhl, July 2018. doi:10.4230/LIPICs.ICALP.2018.144.
- 4 Michael Backes, Ian Goldberg, Aniket Kate, and Esfandiar Mohammadi. Provably secure and practical onion routing. In *Computer Security Foundations Symposium (CSF), 2012 IEEE 25th*, pages 369–385. IEEE, 2012.
- 5 Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. Anoa: A framework for analyzing anonymous communication protocols. In *Computer Security Foundations Symposium (CSF), 2013 IEEE 26th*, pages 163–178. IEEE, 2013.
- 6 Ron Berman, Amos Fiat, and Amnon Ta-Shma. Provable unlinkability against traffic analysis. In Ari Juels, editor, *FC 2004*, volume 3110 of *LNCS*, pages 266–280. Springer, Heidelberg, February 2004.
- 7 Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 169–187. Springer, Heidelberg, August 2005. doi:10.1007/11535218_11.
- 8 Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. doi:10.1109/SFCS.2001.959888.
- 9 Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 42–51. ACM, 1993. doi:10.1145/167088.167105.
- 10 Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Information and Computation*, 206(2–4):378–401, 2008.
- 11 David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981. doi:10.1145/358549.358563.
- 12 David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, January 1988. doi:10.1007/BF00206326.
- 13 Miranda Christ. New lower bounds on the complexity of provably anonymous onion routing. Undergraduate honors thesis, Brown University, Providence, RI 02912 USA, 2020.
- 14 David A. Cooper and Kenneth P. Birman. Preserving privacy in a network of mobile computers. In *1995 IEEE Symposium on Security and Privacy*, pages 26–38. IEEE Computer Society Press, 1995. doi:10.1109/SECPRI.1995.398920.
- 15 Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy*, pages 321–338. IEEE Computer Society Press, May 2015. doi:10.1109/SP.2015.27.
- 16 Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure multiparty computation*. Cambridge University Press, 2015.
- 17 Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two. In *2018 IEEE Symposium on Security and Privacy*, pages 108–126. IEEE Computer Society Press, May 2018. doi:10.1109/SP.2018.00011.
- 18 Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320, 2004.
- 19 Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540. Springer, Heidelberg, May 2004. doi:10.1007/978-3-540-24676-3_31.

- 20 Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.
- 21 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. doi:10.1145/28395.28420.
- 22 Don Hush and Clint Scovel. Concentration of the hypergeometric distribution. *Statistics & Probability Letters*, 75(2):127–132, 2005.
- 23 Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul F. Syverson. Users get routed: traffic correlation on tor by realistic adversaries. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 337–348. ACM Press, November 2013. doi:10.1145/2508859.2516651.
- 24 Christiane Kuhn, Martin Beck, and Thorsten Strufe. Breaking and (partially) fixing provably secure onion routing. *CoRR*, abs/1910.13772, 2019. arXiv:1910.13772.
- 25 Albert Kwon, Henry Corrigan-Gibbs, Srinivas Devadas, and Bryan Ford. Atom: Horizontally scaling strong anonymity. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 406–422. ACM, 2017. doi:10.1145/3132747.3132755.
- 26 Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press, 2005.
- 27 Lasse Øverlier and Paul Syverson. Locating hidden servers. In *2006 IEEE Symposium on Security and Privacy*, pages 100–114. IEEE Computer Society Press, May 2006. doi:10.1109/SP.2006.24.
- 28 Charles Rackoff and Daniel R. Simon. Cryptographic defense against traffic analysis. In *25th ACM STOC*, pages 672–681. ACM Press, May 1993. doi:10.1145/167088.167260.
- 29 Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. RAPTOR: Routing Attacks on Privacy in Tor. In *USENIX Security Symposium*, pages 271–286, 2015.
- 30 Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nikolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 423–440. ACM, 2017. doi:10.1145/3132747.3132783.
- 31 Jelle van den Hooff, David Lazar, Matei Zaharia, and Nikolai Zeldovich. Vuvuzela: scalable private messaging resistant to traffic analysis. In Ethan L. Miller and Steven Hand, editors, *Proceedings of the 25th Symposium on Operating Systems Principles, SOSP 2015, Monterey, CA, USA, October 4-7, 2015*, pages 137–152. ACM, 2015. doi:10.1145/2815400.2815417.
- 32 Ryan Wails, Yixin Sun, Aaron Johnson, Mung Chiang, and Prateek Mittal. Tempest: Temporal dynamics in anonymity systems. *PoPETs*, 2018(3):22–42, 2018.

Broadcast Secret-Sharing, Bounds and Applications

Ivan Bjerre Damgård ✉

Department of Computer Science, Aarhus University, Denmark

Kasper Green Larsen ✉

Department of Computer Science, Aarhus University, Denmark

Sophia Yakoubov ✉

Department of Computer Science, Aarhus University, Denmark

Abstract

Consider a sender \mathcal{S} and a group of n recipients. \mathcal{S} holds a secret message \mathbf{m} of length l bits and the goal is to allow \mathcal{S} to create a secret sharing of \mathbf{m} with privacy threshold t among the recipients, by broadcasting a single message \mathbf{c} to the recipients. Our goal is to do this with information theoretic security in a model with a simple form of correlated randomness. Namely, for each subset \mathcal{A} of recipients of size q , \mathcal{S} may share a random key with all recipients in \mathcal{A} . (The keys shared with different subsets \mathcal{A} must be independent.) We call this *Broadcast Secret-Sharing (BSS)* with parameters l, n, t and q .

Our main question is: how large must \mathbf{c} be, as a function of the parameters? We show that $\frac{n-t}{q}l$ is a lower bound, and we show an upper bound of $(\frac{n(t+1)}{q+t} - t)l$, matching the lower bound whenever $t = 0$, or when $q = 1$ or $n - t$.

When $q = n - t$, the size of \mathbf{c} is exactly l which is clearly minimal. The protocol demonstrating the upper bound in this case requires \mathcal{S} to share a key with *every* subset of size $n - t$. We show that this overhead cannot be avoided when \mathbf{c} has minimal size.

We also show that if access is additionally given to an idealized PRG, the lower bound on ciphertext size becomes $\frac{n-t}{q}\lambda + l - \text{negl}(\lambda)$ (where λ is the length of the input to the PRG). The upper bound becomes $(\frac{n(t+1)}{q+t} - t)\lambda + l$.

BSS can be applied directly to secret-key threshold encryption. We can also consider a setting where the correlated randomness is generated using computationally secure and non-interactive key exchange, where we assume that each recipient has an (independently generated) public key for this purpose. In this model, any protocol for non-interactive secret sharing becomes an *ad hoc threshold encryption (ATE) scheme*, which is a threshold encryption scheme with no trusted setup beyond a PKI. Our upper bounds imply new ATE schemes, and our lower bound becomes a lower bound on the ciphertext size in any ATE scheme that uses a key exchange functionality and no other cryptographic primitives.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques

Keywords and phrases Secret-Sharing, Ad-hoc Threshold Encryption

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.10

Funding *Ivan Bjerre Damgård*: The European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO).

Kasper Green Larsen: Independent Research Fund Denmark (DFR) Sapere Aude Research Leader grant No 9064-00068B.

Sophia Yakoubov: The European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO).



© Ivan Bjerre Damgård, Kasper Green Larsen, and Sophia Yakoubov;
licensed under Creative Commons License CC-BY 4.0

2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 10; pp. 10:1–10:20



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

In this paper, we consider the following scenario: We have a sender \mathcal{S} and a group of n recipients. \mathcal{S} holds a secret message m of length l bits, and the goal is to allow \mathcal{S} to create a secret sharing of m with privacy threshold t among the recipients. This should be done by broadcasting a single message c to the recipients, followed by local computation by the recipients.

Our goal is to do this with information theoretic security, and since this is clearly impossible in the plain model, we consider a model with correlated randomness.

Note that, if the correlated randomness is “strong enough”, the problem becomes trivial: we could ask that \mathcal{S} has a random secret r of the same length as m and the recipients have shares of r in, e.g., Shamir’s secret sharing scheme. Now, \mathcal{S} can broadcast $m - r$ which is clearly of minimal size, and the recipients adjust their shares accordingly. The problem, however, is that each instance of the correlation can only be used once. And if we want to use the Shamir-based solution several times, the only known approach is to create every correlation instance from scratch, implying a communication cost for every instance. In other words, there is no known way to create new instances from old ones using only local computation, not even if we settle for computational security.

We therefore choose an arguably simpler and easier to implement form of correlated randomness where \mathcal{S} shares random strings with one or more of the recipients. More precisely, for each subset \mathcal{A} of recipients of size q , \mathcal{S} may share a secret random bit string $s_{\mathcal{A}}$ with all recipients in \mathcal{A} . Note that this form of correlated randomness can be set up using only communication between the sender and the receivers; receivers do not need to interact. Furthermore, from one instance of such correlated randomness, the parties can generate as many new (pseudorandom) instances as they like using a PRF and *only local computation*. These properties are very useful for applications. See, for instance, Section 1.1.

For any q , we also allow \mathcal{S} to share a secret with any subset smaller than q ¹. This means that, for larger q , we have stronger forms of correlated randomness.

We consider protocols where \mathcal{S} computes c from m and all the shared secrets ($s_{\mathcal{A}}$ ’s). Then c is broadcast, and each recipient computes his share of m from c and the shared secrets he holds. Security means that c and the information held by up to t recipients contain no information on m , but c and the information held by any $t + 1$ recipients determine m .

We call the notion we just sketched *Broadcast Secret-Sharing (BSS)*, with parameters l , n , t and q . In the following, we will sometimes refer to c as the *ciphertext* and the correlated randomness as *shared keys*, which is motivated by the fact that any broadcast secret sharing scheme can be used as is for a secret key threshold encryption scheme. More on this interpretation below.

Our main question is: how large must c be, as a function of the parameters? And, as a secondary question, how much secret correlated data do we need? To the best of our knowledge, these questions, as well the notion of broadcast secret-sharing, have not been considered before.

Let l_c be the length of c . It is easy to see that

$$l \leq l_c \leq n \cdot l.$$

Namely, c must always carry enough information to transmit m to the receivers – and on the other hand, \mathcal{S} can always solve the problem by sharing a one-time pad key with each

¹ The motivation is that, for virtually any way to implement the shared randomness, \mathcal{S} could always share with $q' < q$ parties by imagining $q - q'$ virtual parties and emulate these herself.

receiver, then making a standard secret sharing of m and letting c consist of the one-time pad encryptions of each of the shares.

In this paper, we show the much stronger conditions

$$\frac{n-t}{q}l \leq l_c \leq \left(\frac{n(t+1)}{q+t} - t\right)l.$$

Note that our upper bound matches the lower bound whenever $t = 0$ or when $q = 1$ or $n - t$. Note also that when $q = n - t$, the size of c is exactly l which is minimal, so $q = n - t$ is the largest value it makes sense to consider. The protocol demonstrating the upper bound in this case requires \mathcal{S} to share a key with *every* subset of size $n - t$. We show that this (possibly exponential) overhead cannot be avoided when c has minimal size.

The *BSS* schemes we mentioned so far produce Shamir secret-sharings as output. In the final part of the paper, we show that if access is additionally given to an idealized PRG², other solutions become possible. Namely, the sender chooses a PRG-input, shares it among the receivers using the best available *BSS*, and one-time pad encrypts the message using the output from the PRG. Note that this produces a non-standard, non-linear secret sharing. The lower bound on ciphertext size becomes $\frac{n-t}{q}\lambda + l - \text{negl}(\lambda)$ (where λ is the length of the input to the PRG). The upper bound becomes $\left(\frac{n(t+1)}{q+t} - t\right)\lambda + l$.

1.1 Applications

We believe broadcast secret-sharing is interesting in its own right, and we describe below a couple of applications that make use of a *BSS*-scheme “out of the box”. As further motivation, we also consider in the following subsection two different ways to provide the correlated randomness, leading to other applications.

1.1.1 (Secret-Key) Threshold Encryption

The first application is to secret-key threshold encryption, where a sender sends a ciphertext to set of receivers such that only large enough subsets can decrypt. The main difference between broadcast secret sharing and secret-key threshold encryption is that, in secret-key threshold encryption, it is important that the shared keys be *reusable*. We can easily achieve this by interpreting each key shared between \mathcal{S} and a (subset of) receiver(s) as a key for a pseudorandom function (PRF) ϕ . To encrypt, \mathcal{S} chooses a random nonce r , and for each shared key K , computes $\phi_K(r)$. Note that these PRF values form a (pseudorandom) set of values that can be used as fresh correlated randomness for the broadcast secret-sharing scheme we use. \mathcal{S} now uses this scheme to share her message m among the receivers, resulting in a ciphertext c , and sends the pair (r, c) . Decryption can clearly be done by any subset consisting of at least $t + 1$ receivers, and no smaller subset learns anything, which follows easily from security of the PRF and the underlying *BSS*-scheme. Note that decryption requires minimal interaction: each receiver just has to send his share to the others.

Note also that this application works exactly for the simple form of correlated randomness we use, where \mathcal{S} knows some keys, and each receiver knows a subset of them. Had we allowed a more complicated correlation, the receivers could not have generated new (pseudorandom) correlations of the same form simply by applying the PRF locally.

² We warn the reader that in an actual implementation, a real PRG would have to be used, and the scheme would only be computationally secure.

1.1.2 Secure Multiparty Computation

A second application of BSS is to use it to non-interactively supply input to a secret-sharing based multiparty computation protocol, where the keys held by the sender and receivers can be generated in an earlier setup phase. Given an ideal functionality for distributing keys, we get information theoretic security if the keys are used once. But if we are happy with computational security, we can use a PRF as explained in the previous subsection to extend the key material and support any number of inputs. Note that this will not work when using the well-known method of “pre-cooking” a Shamir secret sharing of a random value known to the sender. Note also that our construction generates Shamir secret-sharings and so is compatible with standard MPC protocols.

1.2 Implementing Shared Keys

Broadcast secret sharing assumes keys shared between the sender and (subsets of) the receiver(s). To discuss the use of BSS in practice, we must also consider the distribution of these keys. We suggest two approaches: non-interactive key exchange (NIKE), and quantum key agreement.

1.2.1 Using NIKE to get (Public-Key) Ad-Hoc Threshold Encryption

In this subsection, we discuss a way to generate the shared keys on the fly, via computationally secure and non-interactive key exchange. Here, we assume that each recipient has an (independently generated) public key and secret key for this purpose.

In this model, any protocol for BSS (including our upper bounds) implies a (public-key) *ad hoc threshold encryption (ATE) scheme*, which is a threshold encryption scheme with no trusted setup beyond a PKI. Namely, the sender creates a ciphertext that includes the information required for the key exchange as well as the c created for broadcast secret-sharing of the message m . To decrypt, at least $t+1$ recipients will first compute the shared randomness using the key exchange, then use this to compute their shares, and finally exchange the shares to reconstruct m . In the related work section below, we give more background on ATE and its relation to standard threshold encryption.

Note that for $q = 1$ the non-interactive key exchange can be done very efficiently based on the DDH assumption: if each receiver i has a public key of form g^{x_i} in some appropriate group, then \mathcal{S} just needs to include a single element g^r for random r in the ciphertext, then the shared key will be of form $g^{x_i r}$ for receiver i . A similar solution for $q = 2$ can be designed using pairing friendly groups. Thus, for these cases, our upper bounds become (essentially) upper bounds on the ciphertext size of the corresponding ATE-scheme. In particular, the ATE-scheme that follows from this and our construction for $q = 2$ has smaller ciphertext size than the best previous scheme of Daza *et al.* [4]. For instance, when $t = 1$, that scheme has ciphertext size $(n - 1)l$ while we can obtain $(\frac{2n}{3} - 1)l$.

Less efficient non-interactive key exchange solutions also exist for larger values of q . They can be constructed from multilinear maps, indistinguishability obfuscation [2], universal samplers [8, 7] (which can be built from indistinguishability obfuscation or functional encryption), or encryption combiners satisfying perfect independence [9] (which can be built from universal samplers).

On the other hand, in this setting, our lower bound becomes a lower bound on the ciphertext size in any ATE scheme that uses an ideal functionality for key exchange (and perhaps for PRG), and no other cryptographic primitives. We formalize the demand that no other cryptographic primitives are used by requiring that the scheme is information theoretically secure when using the ideal functionalities.

We stress that these lower bounds hold for ATE-schemes that have access to the cryptographic primitives only via the ideal functionalities they implement. This is more restrictive than if black-box access were given to the corresponding algorithms; one might say that we allow the protocol to use them “only as intended”. However, to the best of our knowledge, no general lower bound was known for ATE before.

1.2.2 Using Quantum Agreement

The correlated randomness needed for BSS can also be provided in a setting where the sender shares entangled quantum states with each of the receivers. As is well known, if sender and receiver share a pair of particles that are in the so-called EPR state, then measuring each particle results in the same random bit being obtained by both parties. Moreover, as long as the state really is the pure EPR state, no third party has any information on the randomness obtained. Thus this setting gives us exactly what we want for $q = 1$, with perfect security assuming perfect ability to prepare states and measure them. The same is true if one assumes that sender and receiver has executed a secure quantum key exchange protocol at some earlier time.

The case of $q > 1$ also has a quantum implementation, namely if we assume that the sender shares multipartite entangled states with subsets of receivers. In a multipartite entangled state, each involved party holds a particle, and the global state of the particles can be designed to be fully entangled so that local measurements return the same random result for all parties.

1.3 Related Work

1.3.1 Threshold Secret-Key Cryptosystems

There is not much work on secret-key (symmetric) cryptosystems where the decryption and/or the encryption process can be distributed among a number of parties. A formal study of this was done by Agrawal et al. [1], in which formal security definitions and constructions were given for the case where both encryption and decryption is distributed. Our construction is in a different model where only the decryption is distributed. This allows us to offer new tradeoffs for constructions using only secret-key primitives and no public-key techniques, which is usually the more efficient case. The one construction from [1] using only secret-key primitives (a PRF) is very similar to our solution where $q = n - t$. It has minimal ciphertext size l but requires $\binom{n}{t}$ keys, potentially leading to exponential in n overhead. At the other extreme, we have the trivial solution where $q = 1$ and the sender secret-shares the message and sends a share to each receiver, leading to ciphertext size nl and a total of n keys. However, the construction leading to our upper bound implies a spectrum of options “in between”, namely we can get ciphertext size $(\frac{n(t+1)}{q+t} - t)l$ using $\frac{n}{q+t} \binom{q+t}{t}$ keys.

1.3.2 Threshold Public-Key Cryptosystems

The concept of public-key threshold encryption is very well known. It goes back at least to Desmedt *et al.* [5], and has since then been studied in a very long line of research. For this type of scheme, the key generation outputs a public key \mathbf{pk} and a set of secret keys $\mathbf{sk}_1, \dots, \mathbf{sk}_n$ which are generated with respect to a threshold value t , where $0 \leq t < n$. Informally, the important security properties are that given any set of at least $t + 1$ secret keys, one can decrypt a ciphertext encrypted under \mathbf{pk} , while the encryption remains secure even given any set of t secret keys. For efficiency, ciphertexts should have size independent of n .

Requiring a single trusted execution of key generation can be very limiting, particularly in a system where parties may join at any point, or where senders want to dynamically choose subsets of the parties to be the recipients of a particular message. *Dynamic* threshold public-key encryption, introduced by Delerablée and Pointcheval [6], has a reduced setup requirement where the sender can pick the set of n recipients at encryption time; however, each recipient's secret key must be derived from a common master secret key, so a trusted authority is still necessary. *Ad hoc threshold encryption* (ATE), first introduced by Daza *et al.* [4] as *threshold broadcast encryption*³ (motivated by its applicability to mobile ad hoc networks), requires no trusted setup beyond the absolute minimum – a PKI.

ATE considers a universe of users, where each user i has a public key pk_i and corresponding secret key sk_i , and where all key pairs are independently generated. A sender can select a set \mathcal{R} of n users and a threshold value t at the time at which he decides to send a message m . He can then construct a ciphertext $c = E_{\text{pk}_{\mathcal{R}}, t}(m)$, where $\text{pk}_{\mathcal{R}}$ is the set of public keys belonging to parties in \mathcal{R} . ATE requires properties similar to those of standard threshold encryption: namely, that any $t + 1$ parties in \mathcal{R} can decrypt, while the encryption remains semantically secure even given the secret keys of any t parties in \mathcal{R} .

Clearly, ATE has a number of attractive properties that standard threshold encryption lacks: no trusted authority, and the ability to decide on the set of receivers and the threshold on the fly. On the other hand, it is not clear that an ATE ciphertext can be as small as a standard one. The best known solution is from Daza *et al.* [4]. They show how to get ciphertext size linear in $n - t$. This solution is in our model discussed earlier (though it was not presented this way). Namely, it combines a BSS-scheme with non-interactive key exchange, where $q = 1$. In fact, their BSS scheme is a special case of our upper bound.

In this context, our lower bound shows that the ATE scheme of Daza *et al.* has optimal ciphertext size in the class of ATE schemes that use non-interactive key exchange with $q = 1$ and no other cryptographic tools (but as mentioned above, it can be improved using $q = 2$). To the best of our knowledge, our bound is the first lower bound obtained for ATE schemes.

Reyzin *et al.* [10] show that using indistinguishability obfuscation, as well as a few standard primitives, it is possible to get ciphertext size independent of n . There are several reasons, however, why this is not a very satisfactory answer. For one thing, the construction requires that senders (as well as receivers) have public and secret keys, which is not usually assumed for ATE. Moreover, obfuscation requires strong assumptions; and with current state of the art techniques, it comes at the price of a huge loss of efficiency in practice.

1.3.3 Pseudorandom Secret-Sharing

In [3], Cramer *et al.* show that, in a model where sufficiently many independent random values are generated and each player is given an appropriate subset of these, the players can locally convert this information to a random Shamir secret-sharing (with a fixed threshold that depends on the set-up). This model is a somewhat similar to ours. The crucial difference, however, is that we have a distinguished player - the sender - who knows all the values and can send a single message to the others. This allows us to create secret-sharings with any threshold, and while we do make use of their technique in our construction, we need additional new ideas to do so.

³ One should note that ATE for $t = 0$ is very similar to broadcast encryption: each party can decrypt on his own. However, in broadcast encryption, centralized key generation is usually allowed (or at least key generation is coordinated between receivers). This is exactly what is not allowed in ATE.

1.4 Open Problems

There is a very rich space of problems to explore. The most obvious open question is of course to close the gap between the upper and the lower bound on ciphertext size. Another problem is to understand how large the correlated randomness must be. Can the lower bound for minimal ciphertext size be generalized, or is there a way to get polynomial size randomness when the ciphertext is (close to) minimal size?

2 Definitions

In this section, we give the syntax and security definitions for broadcast secret sharing (BSS).

We consider the following random variables:

- $S_{\mathcal{A}}$, the random variable shared by the sender with the q parties in the set \mathcal{A} ,
- the message M , and
- the ciphertext C .

For ease of notation, we also let U be the random variable giving all the secrets $S_{\mathcal{A}}$ shared by the sender with any subset of receivers, U_i be the random variable giving all the secrets held by party \mathcal{P}_i (that is, $U_i = \{S_{\mathcal{A}}\}_{i \in \mathcal{A}}$), and $U_{\mathcal{A}}$ be the random variable giving the union of all the secrets held by parties in \mathcal{A} .

We use uppercase variables – S, U, M, C – to refer to distributions, and lowercase variables – s, u, m, c – to refer to concrete values.

2.1 BSS Syntax

We assume that any BSS scheme comes with a specification of finite sets from where the random variables are to be chosen. Hence, when we say in the following “any distribution of M ”, for instance, this means any distribution over the specified set of outcomes.

A BSS scheme with parameters (l, n, t, q) consists of two algorithms, described below.

$E_{u_{\mathcal{R}}}(\mathbf{m}) \rightarrow \mathbf{c}$ is a secret sharing algorithm (which we also sometimes dub *encryption*) that uses a set of keys $u_{\mathcal{R}} = \{u_i\}_{i \in \mathcal{R}}$ belonging to the parties in the size- n set \mathcal{R} of intended recipients (where each u_i consists of all secrets known to sets \mathcal{A} where $i \in \mathcal{A}$) to transform a length- l message \mathbf{m} into a secret sharing (or *ciphertext*) \mathbf{c} .

$D_{u_{\mathcal{A}}}(\mathbf{c}) \rightarrow \mathbf{m}$ is a reconstruction (or *decryption*) algorithm that uses keys $u_{\mathcal{A}} = \{u_i\}_{i \in \mathcal{A}}$ belonging to a subset \mathcal{A} of the intended recipient set \mathcal{R} (where $|\mathcal{A}| > t$) to recover the message \mathbf{m} from the sharing / ciphertext \mathbf{c} .

2.2 BSS Security

Informally, a BSS scheme is secure if any t parties in the designated set of receivers \mathcal{R} can learn nothing about a message from a ciphertext, but any $t + 1$ parties in \mathcal{R} can recover the message. More precisely:

► **Definition 1** (BSS Perfect Security). *A BSS scheme (E, D) is perfectly secure with threshold t if for any set of receivers \mathcal{R} of size n , for $C = E_{U_{\mathcal{R}}}(M)$, the following two properties hold for any distribution of M :*

Security *For any $\mathcal{A} \subset \mathcal{R}$ of size at most t , we have $H(M|C, U_{\mathcal{A}}) = H(M)$.*

Correctness *For any $\mathcal{A} \subset \mathcal{R}$ of size greater than t , we have $H(M|C, U_{\mathcal{A}}) = 0$. Furthermore, $M = D_{U_{\mathcal{R}}}(C)$.*

10:8 Broadcast Secret-Sharing, Bounds and Applications

We can define statistical security similarly, where we assume that the distribution of the variables may also depend on a security parameter λ , but we always assume that the parameters l, n, t are polynomial in λ .

► **Definition 2** (BSS Statistical Security). A BSS scheme (E, D) is statistically secure with threshold t if for any set of receivers \mathcal{R} of size n , for $C = E_{U_{\mathcal{R}}}(\mathbf{M})$, the following two properties hold for any distribution of msg :

Security For any $\mathcal{A} \subset \mathcal{R}$ of size at most t , we have $H(\mathbf{M}|C, U_{\mathcal{A}}) \geq H(\mathbf{M}) - \text{negl}(\lambda)$.

Correctness For any $\mathcal{A} \subset \mathcal{R}$ of size greater than t , we have $H(\mathbf{M}|C, U_{\mathcal{A}}) \leq \text{negl}(\lambda)$. Furthermore, $\mathbf{M} = D_{U_{\mathcal{R}}}(C)$ with overwhelming probability.

Finally we define a different type of security that we will need later for technical reasons. It is designed for a situation where $t = 0$, so C alone reveals nothing about the message. Moreover, each player on her own can learn l' bits of the message, but not necessarily the entire message.

► **Definition 3** (BSS l' -Security). A BSS scheme (E, D) is l' -secure if for any set of receivers \mathcal{R} of size n , for $C = E_{U_{\mathcal{R}}}(\mathbf{M})$, the following two properties hold for any distribution of \mathbf{M} and some $l' \leq H(\mathbf{M})$:

Security $H(\mathbf{M}|C) \geq H(\mathbf{M}) - \text{negl}(\lambda)$.

Correctness For any receiver \mathcal{P}_i we have $H(\mathbf{M}|C, U_i) \leq H(\mathbf{M}) - l' + \text{negl}(\lambda)$.

Clearly, if a BSS-scheme is l' -secure for $l' = H(\mathbf{M})$, it is statistically secure in the case where $t = 0$.

3 Lower Bounds for Broadcast Secret Sharing

In this section, we prove a lower bound for BSS schemes with statistical security. Throughout the proofs, we consider sending a uniform random message \mathbf{M} of l bits. We then prove that the corresponding ciphertext of a BSS scheme must (roughly) satisfy $H(C) \geq nH(\mathbf{M})/q = nl/q$. Since the entropy of a random variable giving a bit string is a lower bound on its expected length (Shannon's source coding theorem), this also lower bounds the length of the ciphertext. We prove the lower bound in steps, starting with the warm-up case $t = 0, q = 1$ and then extending it to arbitrary q and finally also to arbitrary t .

3.1 Warm-Up: BSS with $t = 0$ and $q = 1$

We start with a lower bound proof in the simple setup with threshold $t = 0$ and shared keys among $q = 1$ recipients. We let the message \mathbf{M} be a uniform random bit string of length l (hence $H(\mathbf{M}) = l$). We prove the following lower bound, where $\text{negl}(\lambda)$ may be replaced by 0 for perfect security:

► **Theorem 4.** For any BSS scheme with statistical security, n recipients, threshold $t = 0$ and sharing of keys with $q = 1$ recipients, we must have:

$$H(C) \geq n(l - \text{negl}(\lambda)).$$

To prove the lower bound, let S_i for $i = 1, \dots, n$ denote the shared key received by the i 'th recipient (for $q = 1$, only i receives that random key). The high level idea in our proof is to argue that C must contain a lot of information about the randomness S_i for every index i . Since the shared keys are independent, this implies a lower bound on the entropy of C . More formally, consider the mutual information $I(S_i; C | \mathbf{M}, S_1, \dots, S_{i-1})$. We will show:

► **Lemma 5.** *For all recipients i , it holds that $I(C; S_i | M, S_1, \dots, S_{i-1}) \geq l - \text{negl}(\lambda)$.*

Before proving Lemma 5, let us see how we use it to prove Theorem 4. Using non-negativity of entropy and the chain rule of mutual information, we have

$$\begin{aligned} H(C) &\geq H(C | M) \\ &\geq H(C | M) - H(C | M, S_1, \dots, S_n) \\ &= I(C; S_1, \dots, S_n | M) \\ &= \sum_{i=1}^n I(C; S_i | M, S_1, \dots, S_{i-1}) \\ &\geq n(l - \text{negl}(\lambda)). \end{aligned}$$

This completes the proof of Theorem 4. Thus what remains is to prove Lemma 5.

Proof of Lemma 5. The basic idea in the proof of Lemma 5 is that C and S_i together reveal M , thus collectively they must have $l - \text{negl}(\lambda)$ bits of information about M . Since S_1, \dots, S_i alone have no information about M , those $l - \text{negl}(\lambda)$ bits must be accounted for in $I(C; S_i | M, S_1, \dots, S_{i-1})$. We prove that formally in the following. By definition, the mutual information in Lemma 5 equals:

$$\begin{aligned} I(C; S_i | M, S_1, \dots, S_{i-1}) &= \\ H(S_i | M, S_1, \dots, S_{i-1}) - H(S_i | C, M, S_1, \dots, S_{i-1}). \end{aligned}$$

The message M and all the shared keys are independent, hence $H(S_i | M, S_1, \dots, S_{i-1}) = H(S_i)$. Since entropy may only increase by dropping variables we condition on, we also conclude $H(S_i | C, M, S_1, \dots, S_{i-1}) \leq H(S_i | C, M)$. Using the definition of mutual information, we thus have:

$$\begin{aligned} I(S_i; C | M, S_1, \dots, S_{i-1}) &\geq H(S_i) - H(S_i | C, M) \\ &= I(S_i; C, M) \\ &= H(C, M) - H(C, M | S_i). \end{aligned}$$

Since the ciphertext C contains no information about M alone (up to $\text{negl}(\lambda)$), we have $H(C, M) = H(C) + H(M | C) \geq H(C) + H(M) - \text{negl}(\lambda)$. By the chain rule of entropy, we have $H(C, M | S_i) = H(C | S_i) + H(M | C, S_i) \leq H(C) + H(M | C, S_i)$. But $H(M | C, S_i) \leq \text{negl}(\lambda)$ since recipient i can recover M from C and S_i . We therefore have:

$$\begin{aligned} I(S_i; C | M, S_1, \dots, S_{i-1}) &\geq H(C) + H(M) - \text{negl}(\lambda) - (H(C) + \text{negl}(\lambda)) \\ &= H(M) - \text{negl}(\lambda) \\ &= l - \text{negl}(\lambda). \end{aligned} \quad \blacktriangleleft$$

3.2 BSS with $t = 0$

In this section, we generalize the lower bound from Section 3.1 to $q \geq 1$ (still assuming $t = 0$ and that the message M is a uniform random l bit string):

► **Theorem 6.** *For any BSS scheme with statistical security, n recipients, security threshold $t = 0$ and sharing of keys with q recipients, we must have:*

$$H(C) \geq n(l - \text{negl}(\lambda))/q.$$

10:10 Broadcast Secret-Sharing, Bounds and Applications

To show this, we will show a stronger statement that will be useful for other purposes in the following:

► **Theorem 7.** *For any l' -secure BSS scheme with n recipients, and sharing of keys with q recipients, we must have:*

$$H(C) \geq n(l' - \text{negl}(\lambda))/q.$$

Clearly, this result implies Theorem 6: when M is uniform and $H(M) = l$, the assumption in Theorem 6 is equivalent to requiring l -security.

The basic idea in the proof for $q = 1$ was to argue that the ciphertext C contained a lot of information about each S_i . Formally, Lemma 5 showed that $I(C; S_i | M, S_1, \dots, S_{i-1}) \geq l - \text{negl}(\lambda)$. In the following, we discuss the obstacles we face when generalizing the proof to $q \geq 1$ and show how we overcome them.

First, in order to prove Lemma 5, we used the fact that S_i together with C revealed M to conclude that $I(C; S_i | M, S_1, \dots, S_{i-1}) \geq l - \text{negl}(\lambda)$. Considering instead l' -security this statement would be $I(C; S_i | M, S_1, \dots, S_{i-1}) \geq l' - \text{negl}(\lambda)$ and it could be proved in exactly the same way for $q = 1$.

However, since a recipient may now use all his shared keys to recover M , we define a random variable U_i for each recipient i : We let U_i denote all shared keys held by recipient i ($U_i = \{S_{\mathcal{A}}\}_{i \in \mathcal{A}}$). Intuitively, the analog of Lemma 5 would state that $I(C; U_i | M, U_1, \dots, U_{i-1}) \geq l' - \text{negl}(\lambda)$.

With this definition of U_i we again have that U_i and C together reveal l' bits of M . Unfortunately, the sets of shared keys held by different recipients are not disjoint. This means that U_i may depend on U_1, \dots, U_{i-1} and thus the lower bound on the mutual information is not necessarily true.

Our key idea for addressing the above issue is to further partition U_i into subset $U_{i,1}, \dots, U_{i,q}$ where $U_{i,k}$ contains all shared keys $S_{\mathcal{A}}$ for which i is the k 'th smallest index in \mathcal{A} . Note that with this definition $U_{i,k}$ and $U_{j,k}$ with $i \neq j$ are disjoint sets of shared keys (only one index can be the k 'th smallest in a set \mathcal{A}) and thus are independent. The same holds for $U_{i,j}$ and $U_{i,k}$ with $j \neq k$ (i cannot both be the j 'th and k 'th smallest index in \mathcal{A}). Finally, we also define $F_{i,k}$ to denote the set of all shared keys $S_{\mathcal{A}}$ in which i is the largest index in \mathcal{A} and $|\mathcal{A}| < k$. Our generalization of Lemma 5 then becomes:

► **Lemma 8.** *There is an index $k \in \{1, \dots, q\}$ such that*

$$\sum_{i=1}^n I(U_{i,k} F_{i,k}; C | M, U_{i+1,k}, F_{i+1,k}, \dots, U_{n,k}, F_{n,k}) \geq n(l' - \text{negl}(\lambda))/q.$$

Before proving Lemma 8, let us see that it implies Theorem 6. We have:

$$\begin{aligned} H(C) &\geq H(C | M) \\ &\geq H(C | M) - H(C | M, U_{1,k}, F_{1,k}, \dots, U_{n,k}, F_{n,k}) \\ &= I(C; U_{1,k}, F_{1,k}, \dots, U_{n,k}, F_{n,k} | M) \\ &= \sum_{i=1}^n I(C; U_{i,k}, F_{i,k} | M, U_{i+1,k}, F_{i+1,k}, \dots, U_{n,k}, F_{n,k}) \\ &\geq n(l' - \text{negl}(\lambda))/q. \end{aligned}$$

What remains is thus to prove Lemma 8. The key step in doing so is to replace each mutual information in the sum by a term that only depends on the sets $U_{i,1}, \dots, U_{i,q}$ seen by the i 'th

recipient. The rewriting is quite non-trivial and crucially relies on the fact that we applied the chain rule in reverse order of indices such that we condition on $U_{j,k}, F_{j,k}$ for indices $j > i$. The rewriting we make uses the following:

► **Lemma 9.** *For every recipient i and every index $k \in \{1, \dots, q\}$ we have*

$$I(U_{i,k} F_{i,k}; C \mid M, U_{i+1,k}, F_{i+1,k}, \dots, U_{n,k}, F_{n,k}) \geq I(U_{i,k}; C \mid M, U_{i,1}, \dots, U_{i,k-1}).$$

Let us first use Lemma 9 to prove Lemma 8.

Proof of Lemma 8. Consider summing over all recipients and all choices of k , applying Lemma 9 on each term:

$$\begin{aligned} \sum_{k=1}^q \sum_{i=1}^n I(U_{i,k} F_{i,k}; C \mid M, U_{i+1,k}, F_{i+1,k}, \dots, U_{n,k}, F_{n,k}) &\geq \\ \sum_{k=1}^q \sum_{i=1}^n I(U_{i,k}; C \mid M, U_{i,1}, \dots, U_{i,k-1}) &= \\ \sum_{i=1}^n \sum_{k=1}^q I(U_{i,k}; C \mid M, U_{i,1}, \dots, U_{i,k-1}) &= \\ \sum_{i=1}^n I(U_{i,1}, \dots, U_{i,q}; C \mid M) &= \\ \sum_{i=1}^n I(U_i; C \mid M). \end{aligned}$$

Since U_i and M are independent, we have $I(U_i; C \mid M) = H(U_i \mid M) - H(U_i \mid C, M) = H(U_i) - H(U_i \mid C, M) = I(U_i; C, M) = H(C, M) - H(C, M \mid U_i)$. Since M cannot be recovered from C , we have

$$H(C, M) = H(C) + H(M \mid C) \geq H(C) + H(M) - \text{negl}(\lambda).$$

By the chain rule, $H(C, M \mid U_i) = H(C \mid U_i) + H(M \mid C, U_i) \leq H(C) + H(M \mid C, U_i)$. But, by l' -security, l' bits of M are determined from C and U_i , more precisely

$$H(M \mid C, U_i) \leq H(M) - l' + \text{negl}(\lambda).$$

We have thus shown $I(U_i; C \mid M) \geq H(C) + H(M) - \text{negl}(\lambda) - (H(C) + H(M) - l' + \text{negl}(\lambda)) = l' - \text{negl}(\lambda)$. We therefore have:

$$\begin{aligned} \sum_{k=1}^q \sum_{i=1}^n I(U_{i,k} F_{i,k}; C \mid M, U_{i+1,k}, F_{i+1,k}, \dots, U_{n,k}, F_{n,k}) &\geq \\ \sum_{i=1}^n l' - \text{negl}(\lambda) &= \\ n(l' - \text{negl}(\lambda)). \end{aligned}$$

Averaging over all choices of k completes the proof of Lemma 8. ◀

To finish, we thus need to prove Lemma 9:

10:12 Broadcast Secret-Sharing, Bounds and Applications

Proof of Lemma 9. We need to show that for all recipients i and every index k , it holds that

$$I(\mathbf{U}_{i,k} F_{i,k}; \mathbf{C} \mid \mathbf{M}, \mathbf{U}_{i+1,k}, F_{i+1,k}, \dots, \mathbf{U}_{n,k}, F_{n,k}) \geq I(\mathbf{U}_{i,k}; \mathbf{C} \mid \mathbf{M}, \mathbf{U}_{i,1}, \dots, \mathbf{U}_{i,k-1}).$$

The main observation needed in the proof is the fact every shared key in $\mathbf{U}_{i,1}, \dots, \mathbf{U}_{i,k}$ also appears in $\mathbf{U}_{i,k}, F_{i,k}, \dots, \mathbf{U}_{n,k}, F_{n,k}$. More formally, we start by observing that:

$$\begin{aligned} I(\mathbf{U}_{i,k} F_{i,k}; \mathbf{C} \mid \mathbf{M}, \mathbf{U}_{i+1,k}, F_{i+1,k}, \dots, \mathbf{U}_{n,k}, F_{n,k}) &\geq \\ I(\mathbf{U}_{i,k}; \mathbf{C} \mid \mathbf{M}, F_{i,k}, \mathbf{U}_{i+1,k}, F_{i+1,k}, \dots, \mathbf{U}_{n,k}, F_{n,k}) &= \\ H(\mathbf{U}_{i,k} \mid \mathbf{M}, F_{i,k}, \mathbf{U}_{i+1,k}, \dots, F_{n,k}) - H(\mathbf{U}_{i,k} \mid \mathbf{C}, \mathbf{M}, F_{i,k}, \mathbf{U}_{i+1,k}, \dots, F_{n,k}). \end{aligned}$$

Notice that the set of shared keys $\mathbf{U}_{i,k}$ is disjoint from the sets $\mathbf{U}_{j,k}$ with $j \neq i$. This holds since for any set of receivers \mathcal{A} , only one receiver can be the k 'th smallest. Moreover, $\mathbf{U}_{i,k}$ is also disjoint from $F_{j,k}$ for all j . This is true since $F_{j,k}$ contains only shared keys for sets of receivers with cardinality less than k . This means that $\mathbf{U}_{i,k}$ is independent of $\mathbf{M}, F_{i,k}, \mathbf{U}_{i+1,k}, F_{i+1,k}, \dots, \mathbf{U}_{n,k}, F_{n,k}$ and thus we have

$$H(\mathbf{U}_{i,k} \mid \mathbf{M}, F_{i,k}, \mathbf{U}_{i+1,k}, \dots, F_{n,k}) = H(\mathbf{U}_{i,k}).$$

We therefore have:

$$\begin{aligned} I(\mathbf{U}_{i,k} F_{i,k}; \mathbf{C} \mid \mathbf{M}, \mathbf{U}_{i+1,k}, F_{i+1,k}, \dots, \mathbf{U}_{n,k}, F_{n,k}) &\geq \\ H(\mathbf{U}_{i,k}) - H(\mathbf{U}_{i,k} \mid \mathbf{C}, \mathbf{M}, F_{i,k}, \mathbf{U}_{i+1,k}, F_{i+1,k}, \dots, \mathbf{U}_{n,k}, F_{n,k}). \end{aligned}$$

Since entropy may only increase by removing variables that we condition on, we remove all shared keys from $F_{i,k}, \mathbf{U}_{i+1,k}, F_{i+1,k}, \dots, \mathbf{U}_{n,k}, F_{n,k}$ which do not appear in $\mathbf{U}_{i,1}, \dots, \mathbf{U}_{i,k-1}$. We claim that we are left with precisely the full set of shared keys appearing in $\mathbf{U}_{i,1}, \dots, \mathbf{U}_{i,k-1}$. To see this, consider a shared key $S_{\mathcal{A}}$ appearing in $\mathbf{U}_{i,j}$ for some $j < k$. Assume first that i is the largest index in the set \mathcal{A} . Then the cardinality of \mathcal{A} is $j < k$ and we have $S_{\mathcal{A}} \in F_{i,k}$ by definition of $F_{i,k}$. Next, assume that the cardinality of \mathcal{A} is less than k , but i is not the largest index in \mathcal{A} . Let $i' > i$ be the largest index. Then by definition, we have $S_{\mathcal{A}} \in F_{i',k}$. Finally, assume that the cardinality of \mathcal{A} is at least k . Let $i' > i$ be the k 'th smallest index in \mathcal{A} , then $S_{\mathcal{A}} \in \mathbf{U}_{i',k}$. In all cases, we have that $S_{\mathcal{A}}$ is in one of $F_{i,k}, \mathbf{U}_{i+1,k}, F_{i+1,k}, \dots, \mathbf{U}_{n,k}, F_{n,k}$ and we conclude that we are left with $\mathbf{U}_{i,1}, \dots, \mathbf{U}_{i,k-1}$. We therefore have:

$$\begin{aligned} I(\mathbf{U}_{i,k} F_{i,k}; \mathbf{C} \mid \mathbf{M}, \mathbf{U}_{i+1,k}, F_{i+1,k}, \dots, \mathbf{U}_{n,k}, F_{n,k}) &\geq \\ H(\mathbf{U}_{i,k}) - H(\mathbf{U}_{i,k} \mid \mathbf{C}, \mathbf{M}, \mathbf{U}_{i,1}, \dots, \mathbf{U}_{i,k-1}). \end{aligned}$$

Conditioning on a random variable may only decrease entropy, we can therefore bound the above by:

$$\begin{aligned} I(\mathbf{U}_{i,k} F_{i,k}; \mathbf{C} \mid \mathbf{M}, \mathbf{U}_{i+1,k}, F_{i+1,k}, \dots, \mathbf{U}_{n,k}, F_{n,k}) &\geq \\ H(\mathbf{U}_{i,k} \mid \mathbf{M}, \mathbf{U}_{i,1}, \dots, \mathbf{U}_{i,k-1}) - H(\mathbf{U}_{i,k} \mid \mathbf{C}, \mathbf{M}, \mathbf{U}_{i,1}, \dots, \mathbf{U}_{i,k-1}) &= \\ I(\mathbf{U}_{i,k}; \mathbf{C} \mid \mathbf{M}, \mathbf{U}_{i,1}, \dots, \mathbf{U}_{i,k-1}). \end{aligned}$$

This concludes the proof of Lemma 9 and thus also of Theorem 7. ◀

3.3 Final BSS Lower Bound

In this section, we finally extend the lower bound in Theorem 6 to the general case of $t \geq 0$ and $q \geq 1$. Our final result is the following:

► **Theorem 10.** *For any BSS scheme with statistical security, n recipients, security threshold t and sharing of keys with q recipients, we must have:*

$$H(C) \geq (n - t)(1 - \text{negl}(\lambda))/q.$$

The proof follows via a reduction from the case with $t = 0$ (Theorem 6). The basic idea is to show that any BSS scheme for arbitrary threshold $t \geq 0$ can be converted into a scheme for $t = 0$ and $n - t$ receivers. This is done by treating the first t receivers as dummy receivers for which all shared keys are public information. This way, we get a BSS scheme with $t = 0$ for the remaining receivers $t + 1, \dots, n$.

In detail, consider all shared keys U_1, \dots, U_t held by the first t parties in a BSS scheme with threshold t . Consider any concrete instantiation u_1, \dots, u_t of the random variables and let E_{u_1, \dots, u_t} denote the event that $U_i = u_i$ for $i = 1, \dots, t$. We will prove that for most instantiations of $U_1 = u_1, \dots, U_t = u_t$, conditioned on E_{u_1, \dots, u_t} , the BSS statistical security definitions hold for the remaining $n - t$ receivers with threshold $t = 0$. Formally, we require that:

Security We have $H(M | C, E_{u_1, \dots, u_t}) \geq H(M) - \text{negl}(\lambda)$.

Correctness For any receiver i with $i \in \{t + 1, \dots, n\}$, we have

$$H(M | C, U_i, E_{u_1, \dots, u_t}) \leq \text{negl}(\lambda).$$

Call u_1, \dots, u_t *typical* if they satisfies the above Security and Correctness. If u_1, \dots, u_t are typical, then we have a BSS scheme with threshold $t = 0$ for the remaining $n - t$ receivers $t + 1, \dots, n$ if we hard code $U_1 = u_1, \dots, U_t = u_t$ and let those be shared knowledge. Therefore, by Theorem 6, it must be the case for typical u_1, \dots, u_t , that

$$H(C | E_{u_1, \dots, u_t}) \geq \frac{n - t}{q}(1 - \text{negl}(\lambda)).$$

We will show:

► **Lemma 11.** U_1, \dots, U_t are typical with probability at least $1 - \text{negl}(\lambda)$.

Before we prove Lemma 11, we use the lemma to finish the proof of Theorem 10. We see that

$$\begin{aligned} H(C) &\geq H(C | U_1, \dots, U_t) \\ &= \sum_{u_1, \dots, u_t} H(C | E_{u_1, \dots, u_t}) \Pr[E_{u_1, \dots, u_t}] \\ &\geq \sum_{u_1, \dots, u_t: u_1, \dots, u_t \text{ are typical}} H(C | E_{u_1, \dots, u_t}) \Pr[E_{u_1, \dots, u_t}] \\ &\geq \frac{n - t}{q}(1 - \text{negl}(\lambda)) \Pr[U_1, \dots, U_t \text{ are typical}] \\ &= \frac{n - t}{q}(1 - \text{negl}(\lambda)). \end{aligned}$$

What remains is thus to prove Lemma 11.

10:14 Broadcast Secret-Sharing, Bounds and Applications

Proof of Lemma 11. Let $X(u_1, \dots, u_t)$ take the value $H(M) - H(M | C, E_{u_1, \dots, u_t})$. Observe that since M is independent of U_1, \dots, U_t , we have $H(M) = H(M | E_{u_1, \dots, u_t})$ and thus $X(u_1, \dots, u_t) = H(M | E_{u_1, \dots, u_t}) - H(M | C, E_{u_1, \dots, u_t})$. Conditioning on C may only decrease entropy, hence X is non-negative for all u_1, \dots, u_t . It follows by Markov's inequality that

$$\Pr \left[X(U_1, \dots, U_t) > \sqrt{\mathbb{E}[X(U_1, \dots, U_t)]} \right] < \sqrt{\mathbb{E}[X(U_1, \dots, U_t)]}.$$

Now recall from the security requirements of a BSS scheme with threshold t that:

$$\begin{aligned} H(M) - \text{negl}(\lambda) &\leq H(M | C, U_1, \dots, U_t) \\ &= \sum_{u_1, \dots, u_t} H(M | C, E_{u_1, \dots, u_t}) \Pr[E_{u_1, \dots, u_t}], \end{aligned}$$

which implies

$$\begin{aligned} \mathbb{E}[X(U_1, \dots, U_t)] &= H(M) - \sum_{u_1, \dots, u_t} H(M | C, E_{u_1, \dots, u_t}) \Pr[E_{u_1, \dots, u_t}] \\ &\leq \text{negl}(\lambda). \end{aligned}$$

Thus by Markov's, we have $\Pr \left[X(U_1, \dots, U_t) > \text{negl}(\lambda) \right] < \text{negl}(\lambda)$.

Next, for any receiver $i > t$, define $Y_i(u_1, \dots, u_t)$ to take the value $H(M | C, U_i, E_{u_1, \dots, u_t})$. Since entropy is always non-negative, so is Y_i . By definition of conditional entropy, we have $\mathbb{E}[Y_i(U_1, \dots, U_t)] = H(M | C, U_i, U_1, \dots, U_t)$. Thus from Markov's we again have $\Pr[Y_i(U_1, \dots, U_t) > \text{negl}(\lambda)] < \text{negl}(\lambda)$. It finally follows by a union bound that with probability at least $1 - (n-t+1)\text{negl}(\lambda) = 1 - \text{negl}(\lambda)$, we simultaneously have $X(U_1, \dots, U_t) < \text{negl}(\lambda)$ and $Y_i(U_1, \dots, U_t) < \text{negl}(\lambda)$ for all $i = t+1, \dots, n$. That is, U_1, \dots, U_t are typical with probability at least $1 - \text{negl}(\lambda)$. ◀

4 Upper Bound on Ciphertext Size

In this section, we explore constructions of broadcast secret-sharing.

4.1 Building Block: Pseudorandom Secret Sharing

Our results in this section leverage *pseudorandom secret sharing*, which is a technique for the local (that is, non-interactive) conversion of a replicated secret sharing to a Shamir secret sharing.

A *replicated secret sharing* for the $(t+1)$ -out-of- n threshold access structure proceeds as follows. First, the dealer splits the secret M into $\binom{n}{t}$ additive secret shares, where each share $r_{\mathcal{A}}$ corresponds to a different maximally unqualified set \mathcal{A} of size t . Then, the complement of each set \mathcal{A} (that is, the $n-t$ parties that are *not* in \mathcal{A}) are all given $r_{\mathcal{A}}$. It is then clear that any maximally unqualified set \mathcal{A} is only missing knowledge of one share $r_{\mathcal{A}}$, which any additional party holds.

Pseudorandom secret sharing [3] locally converts such a replicated secret sharing into a Shamir secret sharing (a degree- t polynomial f with $f(0) = M$ as the secret, and $f(i) = s_i$ as party i 's share for $i \in [1, \dots, n]$). Pseudorandom secret sharing proceeds as follows: let $f_{\mathcal{A}}$ be the degree- t polynomial such that $f_{\mathcal{A}}(0) = 1$, and $f_{\mathcal{A}}(i) = 0$ for all $i \in \mathcal{A}$. Each player \mathcal{P}_i can then compute their Shamir share as

$$s_i = \sum_{\mathcal{A} \subseteq [n]: |\mathcal{A}|=t, i \notin \mathcal{A}} r_{\mathcal{A}} f_{\mathcal{A}}(i).$$

We stress that, despite the name, pseudorandom secret-sharing as presented here provides perfect information theoretic security. The name comes from an application of the technique that uses pseudorandom functions.

Cramer, Damgård and Ishai [3] also prove a lower bound, stated in Theorem 12.

► **Theorem 12** (From [3]). *Fewer than $\binom{n}{t}$ independent random values shared among various subsets of parties cannot be locally converted into a $(t + 1)$ -out-of- n threshold secret sharing.*

4.2 Lower Bounding the Correlated Randomness When $H(\mathbf{C}) = H(\mathbf{M})$

► **Theorem 13.** *For any perfectly secure BSS scheme with threshold $t = \theta(n)$, if $H(\mathbf{C}) = H(\mathbf{M})$, then correlated randomness of exponential size is necessary.*

Proof. If $H(\mathbf{C}) = H(\mathbf{M})$, then for any distribution of keys, there is exactly one ciphertext that corresponds to any given message. Therefore, choosing a ciphertext at random (without considering the correlated randomness) will always give a valid ciphertext that corresponds to some message, no matter which value the randomness takes. Choosing the randomness and ciphertext simultaneously independently at random thus produces a random $(t + 1)$ -out-of- n secret sharing (where the ciphertext is simply an additional random value given to all parties). So, the exponential lower bound by Cramer *et al.* [3] (Theorem 12) on amount of independent randomness that can be converted into a $(t + 1)$ -out-of- n secret sharing applies. ◀

4.3 The Upper Bound

Construction 1 below achieves optimal ciphertext size whenever $t = 0$, or when $q = 0$ or when q is the maximal relevant value $n - t$. This construction leverages the techniques of replicated or pseudorandom secret sharing. The price we pay is that the overhead in terms of size of correlated randomness is sometimes exponential (that is, the sender and each of the receivers must use an exponential number of shared random values). Whether this happens depends on the parameter values.

► **Construction 1.** *Let $n' = q + t$. We partition the recipients into $\frac{n}{n'}$ subsets of size $n' = q + t$. (We assume for simplicity that $n' = q + t$ divides n .) An arbitrary but fixed one of these subsets is chosen and named \mathcal{B} . This is done publicly once and for all. We also assign once and for all a unique point in a suitable finite field to each recipient.*

Consider now any of the above subsets \mathcal{A} . We set up the correlated randomness such that the sender \mathcal{S} shares a random value with any subset of \mathcal{A} , of size $n' - t = q$. These values form a random replicated secret-sharing among the players in \mathcal{A} and hence, using the technique from [3], \mathcal{S} can share a random polynomial $f_{\mathcal{A}}$ of degree at most t with the participants in \mathcal{A} , using only the correlated randomness. Concretely, \mathcal{S} knows $f_{\mathcal{A}}$ and each player in \mathcal{A} knows a point on $f_{\mathcal{A}}$.

The ciphertext consists of $\mathbf{m} + f_{\mathcal{B}}(0)$ and $f_{\mathcal{B}} - f_{\mathcal{A}}$ for every subset $\mathcal{A} \neq \mathcal{B}$.

Each recipient locally computes from the correlated randomness $f_{\mathcal{A}}(i)$ where \mathcal{A} is the subset she is in and i is her assigned point in the field. Then she computes $f_{\mathcal{B}}(i) = f_{\mathcal{A}}(i) + (f_{\mathcal{B}} - f_{\mathcal{A}})(i)$. To reconstruct, any subset of size at least $t + 1$ can interpolate $f_{\mathcal{B}}$ and compute $\mathbf{m} = (\mathbf{m} + f_{\mathcal{B}}(0)) - f_{\mathcal{B}}(0)$.

The security of this construction follows trivially from the security of replicated secret sharing: each $f_{\mathcal{A}}$ is uniformly random of degree at most t and so $f_{\mathcal{B}} - f_{\mathcal{A}}$ contains no information on \mathbf{m} , even given $\mathbf{m} + f_{\mathcal{B}}(0)$. Since each polynomial $f_{\mathcal{B}} - f_{\mathcal{A}}$ can be specified using $t + 1$ coefficients, the ciphertext size is

$$((t + 1)(n/(q + t) - 1) + 1)l = (n(t + 1)/(q + t) - t)l.$$

10:16 Broadcast Secret-Sharing, Bounds and Applications

The size of the shared keys (correlated randomness) is $n/(q+t) \cdot \binom{q+t}{t}$ field elements. This can be as much as $\binom{n}{t}$ and so may be exponential in n . But as we showed above, at least when $q = n - t$, this overhead cannot be avoided.

5 Bounds Additionally Assuming an Idealized PRG

In this section, we add to our BSS model an idealized pseudorandom generator (PRG); an idealized functionality that takes in a random length- λ seed, and outputs a longer random value. (As long as the output is at least one bit longer than the input, we can bootstrap the PRG to give arbitrarily long outputs. In our case, the output length that most often makes sense is l , the length of the message.) Our BSS algorithms are augmented with oracle access to the idealized PRG.

We make some assumptions on how the BSS protocol may use the idealized PRG:

- **Definition 14.** *An admissible BSS-protocol satisfies the following:*
- *For any subset of receivers, any PRG-seed chosen by the sender can either be computed using what that subset of receivers knows, or has full entropy (possibly up to a negligible loss).*
 - *During the sharing phase, the sender chooses all seeds that are input to PRG uniformly, independently of anything else.*
 - *The idealized PRG is not called with any shared keys as input.*

In the following we will only consider admissible BSS constructions. The motivation for this is as follows:

- We want to make sure that an admissible protocol can be turned into a construction in the real world by replacing the idealized PRG by a real PRG construction. Now, if a seed has (essentially) full entropy in the view of the adversary, then (and only then) can we use the standard security of a real PRG to conclude that the output is pseudorandom. Seeds for which the adversary has partial information are not useful in this sense, and we may as well give the adversary full information on that seed for free. This is why we assume that in the view of a subset of receivers, any seed that the sender chose can either be computed or has (essentially) full entropy. However, for a seed to be potentially useful it must have full entropy in the first place, which is why we assume that the sender chooses all seeds uniformly, independently of anything else.
- We assume that the idealized PRG is not called using shared keys as input for simplicity, because this does not cost us any generality: calls to the PRG using shared keys as input is equivalent to asking for longer shared keys. In both cases, the result is a greater amount of correlated randomness.

Finally, we will assume that privacy only needs to hold given ability to call the PRG a polynomial number of times. The reason for this is that otherwise protocols that actually make use of the PRG could not ensure that the message is hidden from a non-qualified subset of receivers. As an example, suppose the sender secret-shares a seed s and includes in the ciphertext a one-time pad encryption $m \oplus PRG(s)$. A completely unbounded adversary can call the PRG on all inputs and, once all the outputs are given, the only uncertainty she has is which seed the sender used. Then, if m is longer than s , it cannot have full entropy.

To be able to talk about the information a set of receivers can get from the oracle, we abuse notation and let $PRG(\mathcal{C}, \mathcal{U}_A)$ denote the random variable that is obtained by calling the PRG on inputs that are selected by an unbounded randomized algorithm that gets $\mathcal{C}, \mathcal{U}_A$ as input. The algorithm only returns a polynomial number of outputs. For simplicity of notation, we suppress the algorithm and the random coins it uses.

► **Definition 15** (BSS Statistical Security with PRG). *A BSS scheme (E, D) is statistically secure with threshold t with respect to a random oracle PRG if for any set of receivers \mathcal{R} of size n , for $C = E_{\mathcal{U}_{\mathcal{R}}}^{PRG}(M)$, the following two properties hold for any distribution of M :*

Security *For any $\mathcal{A} \subset \mathcal{R}$ of size at most t , we have $H(M|C, \mathcal{U}_{\mathcal{A}}, PRG(C, \mathcal{U}_{\mathcal{A}})) \geq H(M) - \text{negl}(\lambda)$.*

Correctness *For any $\mathcal{A} \subset \mathcal{R}$ of size greater than t , $H(M|C, \mathcal{U}_{\mathcal{A}}, PRG(C, \mathcal{U}_{\mathcal{A}})) \leq \text{negl}(\lambda)$. Furthermore, $M = D_{\mathcal{U}_{\mathcal{R}}}^{PRG}(C)$ with overwhelming probability.*

5.1 Lower Bound on Ciphertext Size

► **Theorem 16.** *Consider any BSS scheme that is statistically secure with threshold t with respect to PRG (which takes inputs of size λ) and shares messages of length $l \geq \lambda$. If the scheme is admissible it holds that*

$$H(C) \geq \frac{n-t}{q} \lambda + l - \delta(\lambda)$$

for a negligible function $\delta(\lambda)$.

To show the above theorem, consider first a scheme that satisfies the assumption with threshold $t = 0$, so then the only unqualified set of receivers is the empty set. Since the scheme is admissible, there is a (possibly empty) set of seeds \mathcal{S} that were chosen by the sender, but where each seed in \mathcal{S} has full entropy given the ciphertext C , and all other seeds are determined by C .

We claim that we can transform this scheme into a new one (for a different distribution of messages) that is l' -secure (Definition 3) with $l' = \lambda$. In particular, this will be a scheme where the PRG is not available. Recall that in such a scheme a qualified subset of receivers can determine at least l' bits of the message.

To this end, we define the message M' in the new scheme to be the original M concatenated with the seeds in \mathcal{S} . Reconstruction in the new scheme by a qualified set \mathcal{A} works as follows: If at least one seed $s \in \mathcal{S}$ is determined by $C, \mathcal{U}_{\mathcal{A}}$, then return s . Otherwise, by admissibility, all seeds in \mathcal{S} have full entropy given $C, \mathcal{U}_{\mathcal{A}}$. Consider the random variable $PRG(C, \mathcal{U}_{\mathcal{A}})$ that would have been used for reconstruction in the original scheme. Notice that since this variable is formed by calling the PRG a polynomial number of times, the inputs used will overlap with \mathcal{S} with only negligible probability. Therefore unless this overlap event happens, access to the PRG can be perfectly simulated without calling the PRG, simply by choosing fresh randomness to play the role of the PRG's output. Hence, we can return M with overwhelming probability without calling the PRG, so $H(M|C, \mathcal{U}_{\mathcal{A}})$ is negligible, even without access to the PRG.

Since $l \geq \lambda$, we have shown that given $C, \mathcal{U}_{\mathcal{A}}$ for a qualified set \mathcal{A} , the entropy of M' drops by at least l' bits (up to a negligible amount), and this is the correctness property of Definition 3.

The security property of Definition 3 follows immediately from admissibility and from the security property of Definition 15: given only C , all seeds in \mathcal{S} have full entropy and $H(M|C, \mathcal{U}_{\mathcal{A}}, PRG(C, \mathcal{U}_{\mathcal{A}}))$ can only increase if we take away the PRG and therefore do not condition on $PRG(C, \mathcal{U}_{\mathcal{A}})$.

We can now apply Theorem 7 and since we did not change the distribution of C , we conclude:

► **Lemma 17.** *For any BSS-scheme satisfying Definition 15 with $t = 0$, we have:*

$$H(C) \geq n(\lambda - \delta(\lambda))/q.$$

10:18 Broadcast Secret-Sharing, Bounds and Applications

Proof of Theorem 16. Given any BSS-scheme satisfying Definition 15, we can construct from this a new scheme for $n' = n - t$ receivers and threshold 0 (but the same ciphertext distribution). This is done by fixing the shared keys of the first t players and making them public, exactly as in the proof of Theorem 10, so we will not repeat the details here. We then apply the above lemma, and conclude that $H(C) \geq (n - t)(\lambda - \delta(\lambda))/q$. We finally obtain Theorem 16 by also noting that C must carry enough information to determine the message, so we can add l to the lower bound. ◀

5.2 Upper Bound

Construction 2 describes how, using an idealized PRG in addition to shared keys, we can achieve

$$H(C) = (n(t + 1)/(q + t) - t)\lambda + l.$$

► **Construction 2.** *The sender chooses a random PRG seed, uses the scheme from Construction 1 to share this seed among the receivers, and uses the PRG output on this seed to one-time-pad-encrypt the message.*

Ciphertext size and correctness follow trivially from Construction 1. As for security, it follows from security of Construction 1 that an unqualified set \mathcal{A} of receivers has no information on the seed chosen by the sender. Hence the event that the (polynomial number of) inputs to the PRG chosen by \mathcal{A} include the sender's seed has negligible probability. Unless this event happens, the message has full entropy, so the security property follows.

It is important to remark that, unlike Construction 1, Construction 2 does not give the receivers a Shamir secret sharing of the message, but rather of a PRG seed. The receivers reconstruct by first recovering the PRG seed, and then expanding that seed and using the resulting longer string to recover the message. The downside is that this not a linear secret sharing of the message. However, the upside is that, since a PRG seed can be used to generate an arbitrarily long pseudorandom string, the shared PRG seed can be re-used and the sender can share additional messages of length l to the same set of receivers by sending only l additional bits.

6 Application: Ad hoc Threshold Encryption

We can use any (l, n, t, q) BSS scheme together with any non-interactive key exchange (NIKE) scheme for $q + 1$ parties to get (l, n, t) ad hoc threshold encryption (ATE). Informally, the message sender uses the NIKE scheme to set up the correlated randomness for BSS non-interactively. She simply generates a fresh NIKE key pair, uses the secret key to derive shared secrets with every size- q subset of receivers, uses those shared secrets to run BSS, and sends the NIKE public key along with the resulting ciphertext to enable the recipients to derive the same shared secrets.

We sketch the definitions of NIKE and ATE below, and formalize how ATE can be instantiated from NIKE and BSS.

6.1 NIKE Definitions

A non-interactive key exchange (NIKE) scheme consists of two algorithms:

$KG(1^\lambda) \rightarrow (\mathbf{pk}, \mathbf{sk})$ is a randomized key generation algorithm that takes in the security parameter λ and returns a public-private key pair.

$KA(\mathbf{sk}_i, \mathbf{pk}_{\mathcal{A}}) \rightarrow \mathbf{s}$ is a key agreement algorithm that takes in one secret key and q public keys $\mathbf{pk}_{\mathcal{A}} = \{\mathbf{pk}_j\}_{j \in \mathcal{A}}$ and returns a shared secret.

Informally, a NIKE scheme for q parties is *correct* as long as, for any $i \in \mathcal{A}$ (where $|\mathcal{A}| = q + 1$), $\mathbf{s}_{\mathcal{A}} \leftarrow KA(\mathbf{sk}_i, \{\mathbf{pk}_j\}_{j \in \mathcal{A}, j \neq i})$ gives the same value. It is *secure* as long as, given $\{\mathbf{pk}_i\}_{i \in \mathcal{A}}$ (but none of the associated secret keys \mathbf{sk}_i), $\mathbf{s}_{\mathcal{A}}$ is computationally indistinguishable from random.

6.2 ATE Definitions

An ad hoc threshold encryption (ATE) scheme consists of three algorithms:

$KG(1^\lambda) \rightarrow (\mathbf{pk}, \mathbf{sk})$ is a randomized key generation algorithm that takes in the security parameter λ and returns a public-private key pair.

$E_{\mathbf{pk}_{\mathcal{R}}}(\mathbf{m}) \rightarrow \mathbf{c}$ is an encryption algorithm that encrypts a message \mathbf{m} to a set of public keys $\mathbf{pk}_{\mathcal{R}} = \{\mathbf{pk}_i\}_{i \in \mathcal{R}}$ belonging to the parties in the intended recipient set \mathcal{R} in such a way that any size- $(t + 1)$ subset of the recipient set should jointly be able to decrypt.

$D_{\mathbf{pk}_{\mathcal{R}}, \mathbf{sk}_{\mathcal{A}}}(\mathbf{c}) \rightarrow \mathbf{m}$ is a decryption algorithm that uses secret keys $\mathbf{sk}_{\mathcal{A}} = \{\mathbf{sk}_i\}_{i \in \mathcal{A}}$ belonging to a subset \mathcal{A} of the intended recipient set \mathcal{R} (where $|\mathcal{A}| > t$) to decrypt the ciphertext \mathbf{c} and recover the message \mathbf{m} .

Informally, an (l, n, t) ATE scheme is *correct* if $D(E(\mathbf{M})) = \mathbf{M}$ (where D and E are run with the appropriate keys). It is *secure* if, for any two messages \mathbf{m}_0 and \mathbf{m}_1 of the same length l , $\mathbf{c}_0 = E_{\mathbf{pk}_{\mathcal{R}}}(\mathbf{m}_0)$ and $\mathbf{c}_1 = E_{\mathbf{pk}_{\mathcal{R}}}(\mathbf{m}_1)$ are computationally indistinguishable even given t or fewer of the secret keys \mathbf{sk}_i , $i \in \mathcal{A}$.

6.3 ATE from NIKE and BSS

We can build an ATE scheme from a NIKE scheme and a BSS scheme as follows:

$KG(1^\lambda) \rightarrow (\mathbf{pk}, \mathbf{sk})$:

1. Return $(\mathbf{pk}, \mathbf{sk}) \leftarrow NIKE.KG(1^\lambda)$.

$E_{\mathbf{pk}_{\mathcal{R}}}(\mathbf{m})$:

1. Run $(\mathbf{pk}, \mathbf{sk}) \leftarrow NIKE.KG(1^\lambda)$.
2. For every size- q subset $\mathcal{A} \subseteq \mathcal{R}$, run $\mathbf{s}_{\mathcal{A}} \leftarrow NIKE.KA(\mathbf{sk}, \mathbf{pk}_{\mathcal{A}})$.
3. Run $BSS.c \leftarrow BSS.E_{\mathbf{u}_{\mathcal{R}}}(\mathbf{m})$.
4. Return $(\mathbf{pk}, BSS.c)$.

$D_{\mathbf{pk}_{\mathcal{R}}, \mathbf{sk}_{\mathcal{A}}}(\mathbf{c} = (\mathbf{pk}, BSS.c))$:

1. For every party $i \in \mathcal{A}$, for every size- q subset \mathcal{A}' such that $i \in \mathcal{A}'$, run

$$\mathbf{s}_{\mathcal{A}'} \leftarrow NIKE.KA(\mathbf{sk}_i, \{\mathbf{pk}\} \cup \{\mathbf{pk}_j\}_{j \in \mathcal{A}', j \neq i}).$$

2. Recall that $\mathbf{u}_{\mathcal{A}}$ denotes $\{\mathbf{s}_{\mathcal{A}'}\}_{\mathcal{A}' \cup \mathcal{A} \neq \emptyset}$. Return $\mathbf{m} \leftarrow BSS.D_{\mathbf{u}_{\mathcal{A}}}(BSS.c)$.

The size of a ciphertext in this scheme will be equal to the size of the corresponding BSS ciphertext plus the size of a NIKE public key.

6.4 From ATE and NIKE to BSS

Assume we have an ATE-scheme whose algorithms use an ideal NIKE functionality. We also assume that the ATE scheme is statistically secure when using the ideal NIKE functionality, that is, ciphertexts of different messages are statistically indistinguishable, and the message has full entropy in the view of a non-qualified set of receivers (up to a negligible amount).

From this, we can obtain a BSS scheme: the keys returned from the NIKE functionality become the correlated randomness, the encryption algorithm becomes the sharing algorithm, and the view of each receiver (including the ciphertext) is a share. Reconstruction is done by emulating the decryption protocol.

It therefore follows that our lower bound for BSS ciphertext size is also a lower bound for ciphertext size in any ATE scheme of the type we assumed.

References

- 1 Shashank Agrawal, Payman Mohassel, Pratyay Mukherjee, and Peter Rindal. DiSE: distributed symmetric-key encryption. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1993–2010, 2018.
- 2 Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014. doi:10.1007/978-3-662-44371-2_27.
- 3 Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 342–362. Springer, Heidelberg, February 2005.
- 4 Vanesa Daza, Javier Herranz, Paz Morillo, and Carla Ràfols. Ad-hoc threshold broadcast encryption with shorter ciphertexts. *Electron. Notes Theor. Comput. Sci.*, 192(2):3–15, May 2008. doi:10.1016/j.entcs.2008.05.002.
- 5 Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *26th ACM STOC*, pages 522–533. ACM Press, May 1994.
- 6 Cécile Delerablée and David Pointcheval. Dynamic threshold public-key encryption. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 317–334. Springer, Heidelberg, August 2008.
- 7 Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfuscation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 156–181. Springer, Heidelberg, April / May 2017.
- 8 Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal samplers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 715–744. Springer, Heidelberg, December 2016. doi:10.1007/978-3-662-53890-6_24.
- 9 Fermi Ma and Mark Zhandry. Encryptor combiners: A unified approach to multiparty NIKE, (H)IBE, and broadcast encryption. Cryptology ePrint Archive, Report 2017/152, 2017. URL: <http://eprint.iacr.org/2017/152>.
- 10 Leonid Reyzin, Adam Smith, and Sophia Yakoubov. Turning hate into love: Homomorphic ad hoc threshold encryption for scalable mpc. Cryptology ePrint Archive, Report 2018/997, 2018. URL: <https://eprint.iacr.org/2018/997>.

Locally Reconstructable Non-Malleable Secret Sharing

Bhavana Kanukurthi ✉

Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India

Sai Lakshmi Bhavana Obbattu ✉

Microsoft Research, Bangalore, India

Sruthi Sekar ✉

Department of Mathematics, Indian Institute of Science, Bangalore, India

Jenit Tomy ✉

Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India

Abstract

Non-malleable secret sharing (NMSS) schemes, introduced by Goyal and Kumar (STOC 2018), ensure that a secret m can be distributed into shares m_1, \dots, m_n (for some n), such that any t (a parameter $\leq n$) shares can be reconstructed to recover the secret m , any $t - 1$ shares doesn't leak information about m and even if the shares that are used for reconstruction are tampered, it is guaranteed that the reconstruction of these tampered shares will either result in the original m or something independent of m . Since their introduction, non-malleable secret sharing schemes sparked a very impressive line of research.

In this work, we introduce a feature of local reconstructability in NMSS, which allows reconstruction of any portion of a secret by reading just a few locations of the shares. This is a useful feature, especially when the secret is long or when the shares are stored in a distributed manner on a communication network. In this work, we give a compiler that takes in any non-malleable secret sharing scheme and compiles it into a *locally reconstructable non-malleable secret sharing scheme*. To secret share a message consisting of k blocks of length ρ each, our scheme would only require reading $\rho + \log k$ bits (in addition to a few more bits, whose quantity is independent of ρ and k) from each party's share (of a reconstruction set) to locally reconstruct a single block of the message.

We show an application of our locally reconstructable non-malleable secret sharing scheme to a computational *non-malleable secure message transmission scheme in the pre-processing model*, with an improved communication complexity, when transmitting multiple messages.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases Information Theoretic Cryptography, Secret Sharing, Non-malleability, Local Reconstructability

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.11

Funding *Bhavana Kanukurthi*: Microsoft Research Grant.

Sruthi Sekar: Research supported in part by TCS Research Grant.

Acknowledgements We thank the reviewers for their useful comments and suggestions.

1 Introduction

Secret Sharing Schemes

Secret sharing schemes [29, 6] allow a dealer holding a secret m , to distribute the secret across a set of parties P_1, P_2, \dots, P_n as shares m_1, m_2, \dots, m_n such that subsets of parties authorised by the dealer can reconstruct the secret m and all the other subsets of parties have no information about the secret. Secret sharing schemes are fundamental building blocks in secure computation.



© Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, Sruthi Sekar, and Jenit Tomy; licensed under Creative Commons License CC-BY 4.0

2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 11; pp. 11:1–11:19



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Non-malleability

Non-malleable secret sharing schemes (NMSS) were introduced by Goyal and Kumar [18]. They ensure that if the shares of an authorised set are tampered, then reconstruction of these tampered shares is either same as the original secret or it is some independent of the original secret. Since their introduction, NMSS received wide attention with a long line of work [18, 19, 31, 3, 17, 2, 8, 25, 28, 26], [12, 9]. NMSS are built specific to the class of tampering that the shares undergo. This is because without any restriction of the tampering it is impossible to build NMSS as the tampering function can take all the shares of an authorised set and reconstruct m , compute the shares of $m + 1$ with respect to this authorised set and sets them as the tampered shares. In this case, the reconstruction of the tampered shares will give $m + 1$, which is not same as m but is very much related to m . Tampering families that were studied so far in the context of NMSS are a) *independent tampering*: tampering of a share depends solely on itself and is independent of the other shares b) *joint tampering*: tampering of a share can depend on few other shares c) *affine tampering*: All shares can be tampered together, but the tampering function is restricted to be an affine function.

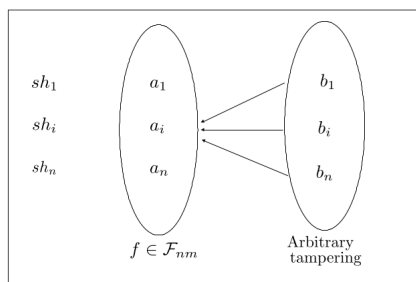
Locality Reconstructability/Recoverability

Inspired from the (well studied) notion of locality in the context of codes (error correcting codes [21, 10] and Non-malleable codes [15, 11, 14]), we study the notion of local reconstructability in the context of secret sharing schemes. A secret sharing scheme is locally reconstructable, if it facilitates retrieval of a portion of the underlying secret such that one does not need to read through the entire share of each party in an authorised set but instead can just read a few locations from shares of parties in the authorised set.

1.1 Our Result

- We define the notion of *locally reconstructable non-malleable secret sharing schemes* (inspired from [15]), which are non-malleable secret sharing schemes infused with the feature of local reconstructability. Suppose the secret to be shared is parsed as a sequence of blocks $m = (m_1, \dots, m_k)$. Assume m is shared, the shares are (possibly) tampered and let $\tilde{m} = (\tilde{m}_1, \dots, \tilde{m}_k)$ denote the reconstruction of the tampered shares. The non-malleability guarantee is that, either \tilde{m} is independent of m or there exists an efficiently samplable set description $\mathcal{I} \subset \{1, \dots, k\}$ (independent of m) such that for $i \in \mathcal{I}$, $\tilde{m}_i = \perp$ and for $i \notin \mathcal{I}$, $\tilde{m}_i = m_i$.
- We show how to compile any non-malleable secret sharing scheme secure against some tampering model \mathcal{F}_{nm} into a locally reconstructable Non-malleable secret sharing scheme.
- **Our tampering model:** The above compiled scheme is non-malleable against the following tampering family. Parse each share sh_i as consisting two parts a_i and b_i , i.e $sh_i = (a_i, b_i)$. b_i 's (for $i \in \{1, \dots, n\}$) can be tampered jointly and arbitrarily but independent of any a_j . All a_i 's can be tampered together as per the tampering allowed by the underlying non-malleable secret sharing i.e by any $f \in \mathcal{F}_{nm}$. In addition, we can allow the description of this tampering function to depend on the values (b_1, \dots, b_n) . We give a pictorial representation of our tampering model in Figure 1. We will call this family as the *Lookahead family* as tampering of a_i 's can depend on b_i 's but not vice-versa¹.

¹ While this may seem like an artificial model of tampering, we indeed show an application of this model to a non-malleable secure message transmission protocol.



■ **Figure 1** Lookahead tampering. Solid arrows signify that the tampering function of a_i 's (which is f) can depend on b_i 's.

- **Parameters:** Let each block of the message be ρ bits long, where ρ is some polynomial in the computational security parameter λ . Upon appropriate instantiation, we have an LRNMSS with share length of each party being $k(\rho + \log k + 2\lambda) + r(2\lambda)$, where $r(\alpha)$ denotes the length of a share of the underlying NMSS upon sharing an α -bit secret. Asymptotically, for long messages the rate $(\frac{\text{message length}}{\text{share length per party}})$ of the compiled locally reconstructable NMSS is $\frac{1}{1+o(1)}$ when $(\log k \ll \rho)$. To locally reconstruct any particular block, each party of an authorised set needs to read only $\rho + \log k + r(2\lambda) + 2\lambda$ bits. Note that this quantity only depends logarithmically on k .
- **Non-malleable secure message transmission in the pre-processing model:** We show an application of our locally reconstructable non-malleable secret sharing scheme to a computational non-malleable message transmission protocol, in the pre-processing model (where a sender and a receiver communicate, first in a message-independent offline phase and then in a message dependent online phase). We show that a combination of our locality feature and the pre-processing, helps us improve communication, specially when the sender wants to transmit multiple messages.

1.2 Technical Overview

We parse the secret to be shared as a sequence of blocks (typically of same length) $m = (m_1, \dots, m_k)$. Let NMSHare denote the non-malleable secret sharing scheme to be compiled into a locally reconstructable NMSS. Let Encrypt be any symmetric key authenticated encryption scheme. Then our compiler proceeds as follows.

- Choose an authentication encryption key K
- Secret share K using NMSHare. Let a_1, \dots, a_n be the shares.
- Encrypt each block $m_i (i \in \{1, \dots, k\})$ along with its location stamp, $c_i \leftarrow \text{Encrypt}_K(m_i || i)$
- For all $i \in \{1, \dots, n\}$, set $b_i = (c_1, c_2, \dots, c_k)$
- Output $sh_i = (a_i, b_i)$

The scheme is locally reconstructable as to recover (say) j th block, the authorised set of parties need to put together only (their respective) a_i 's and c_j (which is given to them as part of b_i). Then they can check the consistency of these c_j 's, reconstruct K and decrypt c_j using K to obtain m_j . If any of the above checks fail, the parties abort. This reconstruction procedure can be naturally extended to recover all the blocks. The works of [24, 1, 13, 17] use similar techniques to improve the rate, while the focus of our work is to achieve locality.

Now we provide a very brief idea of why the above scheme is non-malleable. Suppose even after tampering, if the tampered authenticated encryption key remained the same, then any tampering of the ciphertexts would be detected by the integrity of authenticated

11:4 Locally Reconstructable Non-Malleable Secret Sharing

encryption. If the tampered authenticated key turns out to be independent of K , then all the information about K is lost in the shares, and the ciphertexts do not reveal anything about the messages they encrypt by the indistinguishability of encryption. We provide more details in the technical sections.

Our tampering model does not allow the tampering of ciphertexts to depend on shares of the encryption key. Allowing this kind of tampering will result in the tampered ciphertexts depending indirectly on the encryption key, which would break the encryption security. Although our scheme can be made secure against individual tampering by using secret sharing schemes with stronger security guarantees (e.g. leakage resilient schemes) as the underlying scheme, this trail would worsen the rate and deviate from our focus on building a rate-1 scheme.

While the above model for tampering our scheme seems artificial, it is indeed natural when we apply it in the context of secure message transmission. Particularly, we will send the shares a_1, \dots, a_n of the key K in the offline phase (independent of the message to be transmitted) and then send the ciphertext c_i corresponding to message m_i in the online phase. Here, the online tampering of the ciphertext, indeed will be independent of the offline transmissions.

1.3 Organization of the paper

We provide preliminaries in Section 2. Then, we present our LRNMSS definition in Section 3.1. We define the tampering model in Section 3.2. Our construction and security proof of the locally reconstructable non-malleable secret sharing scheme appears in Section 3.3 and Section 3.4, respectively. We also explain how to instantiate the construction in Section 3.5. In Section 4, we provide an application for our LRNMSS scheme to a non-malleable secure message transmission protocol in the pre-processing model.

2 Preliminaries

2.1 Notations

The set of all natural numbers is denoted by \mathbb{N} . $x \leftarrow X$ denotes sampling from a probability distribution X . All logarithms are base 2. For any two sets S and S' , $S \setminus S' := \{x : x \in S, x \notin S'\}$, is the set of elements in S that are not in S' . Let $[n]$ denote the set $\{1, 2, \dots, n\}$. Let $[n]$ represents the set of all elements. Then, the complement of the set I denoted by $\bar{I} := \{x : x \in [n], x \notin I\}$ is the set of all the elements that are not in I . For any set $T \subseteq [n]$ and a function f outputting n -tuples, $f(\cdot)_T$ represents the output of f restricted to the set T . $\text{negl}(x)$ represents negligible function in x . For any two distributions A and B , $A \approx_c B$ means that the distributions A and B are computationally indistinguishable.

2.2 Authenticated Encryption

An encryption scheme consists of a tuple of polynomial-time algorithms $\mathcal{E} = (\text{Gen}, \text{Encrypt}, \text{Decrypt})$ with key space \mathcal{K} , message space \mathcal{M} and ciphertext space \mathcal{C} such that:

- The randomized algorithm **Gen** takes as input the security parameter $\lambda \in \mathbb{N}$ and outputs a uniform key $sk \in \mathcal{K}$.
- The randomized algorithm **Encrypt** takes as input a key $sk \in \mathcal{K}$ and a message $m \in \mathcal{M}$ and outputs a ciphertext $c \in \mathcal{C}$.
- The deterministic algorithm **Decrypt** takes as input a key $sk \in \mathcal{K}$ and ciphertext $c \in \{0, 1\}^*$ and outputs a value $m \in \mathcal{M} \cup \{\perp\}$, where \perp denotes an invalid ciphertext.

► **Definition 1** ([22, 4, 5]). An encryption scheme $\mathcal{E} = (\text{Gen}, \text{Encrypt}, \text{Decrypt})$ is called a symmetric-key authenticated encryption scheme if it satisfies the following properties:

1. **Correctness.** For all $m \in \mathcal{M}$,

$$\Pr[sk \leftarrow \text{Gen}(1^\lambda); \text{Decrypt}_{sk}(\text{Encrypt}_{sk}(m)) = m] = 1$$

(where probability is taken over randomness of Gen and Encrypt)

2. **Semantic Security.** For any non-uniform PPT adversary \mathcal{A} , it holds that $|2 \cdot \text{Adv}_{\mathcal{E}}^{\text{priv}}(\mathcal{A}) - 1| = \text{negl}(\lambda)$, where

$$\text{Adv}_{\mathcal{E}}^{\text{priv}} = \Pr[sk \leftarrow \text{Gen}(1^\lambda); b \leftarrow \{0, 1\} : \mathcal{A}^{LR_{sk,b}(\cdot, \cdot)}(1^\lambda) = b].$$

Here, the left-or-right encryption oracle $LR_{sk,b}(\cdot, \cdot)$ with $b \in \{0, 1\}$ and inputs $m_0, m_1 \in \mathcal{M}$ for $|m_0| = |m_1|$, is defined as:

$$LR_{sk,b}(m_0, m_1) := \text{Encrypt}_{sk}(m_b).$$

3. **Authenticity.** For any non-uniform PPT adversary \mathcal{A} , it holds that $\text{Adv}_{\mathcal{E}}^{\text{auth}}(\mathcal{A}) = \text{negl}(\lambda)$ where

$$\text{Adv}_{\mathcal{E}}^{\text{auth}}(\mathcal{A}) = \Pr[sk \leftarrow \text{Gen}(1^\lambda), c \leftarrow \mathcal{A}^{\text{Encrypt}_{sk}(\cdot)} : c \notin Q \wedge \text{Decrypt}_{sk}(c) \neq \perp]$$

where Q is list of ciphertexts received by \mathcal{A} through the encryption oracle.

2.3 Secret Sharing Schemes

We will be considering computational secret sharing scheme throughout this paper.

► **Definition 2.** Let \mathcal{M} be finite set of secrets, where $|\mathcal{M}| \geq 2$. A scheme $\Sigma = (\text{Share}, \text{Rec})$ consists of a randomized sharing function $\text{Share} : \mathcal{M} \rightarrow \mathcal{S}_1 \times \dots \times \mathcal{S}_n$ which takes as input a secret $M \in \mathcal{M}$ and outputs n shares (s_1, \dots, s_n) where each $s_i \in \mathcal{S}_i$. The scheme Σ is called a (t, n) -threshold secret sharing scheme with message space \mathcal{M} if the following properties hold:

1. **Correctness.** For any set $T \subseteq [n]$ such that $|T| \geq t$, there exists a deterministic reconstruction function $\text{Rec} : \otimes_{i \in T} \mathcal{S}_i \rightarrow \mathcal{M}$ such that for every $M \in \mathcal{M}$,

$$\Pr[\text{Rec}(\text{Share}(M)_T) = M] = 1$$

(over the randomness of the sharing function)

2. **Privacy (Computational).** For any set $U \subseteq [n]$ such that $|U| < t$, and for every pair of secrets $M_0, M_1 \in \mathcal{M}$,

$$\{\text{Share}(M_0)_U\} \approx_c \{\text{Share}(M_1)_U\}$$

2.4 Non-malleable Secret Sharing Schemes

Non-malleable secret sharing schemes were first studied in [18]. We will be considering the computational variant of their definition.

► **Definition 3.** Let $\Sigma = (\text{Share}, \text{Rec})$ be a (t, n) -secret sharing scheme for message space \mathcal{M} . Let $\mathcal{F} \subseteq \{f : \mathcal{S}_1 \times \dots \times \mathcal{S}_n \rightarrow \mathcal{S}_1 \times \dots \times \mathcal{S}_n\}$ be some family of tampering functions. The scheme is said to be non-malleable w.r.t \mathcal{F} if for each $f \in \mathcal{F}$ and set $T \subseteq [n]$ such that $|T| = t$, there exists a distribution $\text{NMSim}^{f,T}$ such that $\forall m \in \mathcal{M}$,

$$\text{NMTamper}_m^{f,T} \approx_c \text{NMIdeal}_m^{\text{NMSim}^{f,T}}$$

where $\text{NMTamper}_m^{f,T}$ and $\text{NMIdeal}_m^{\text{NMSim}^{f,T}}$ are distributions defined as below:

$$\text{NMTamper}_m^{f,T} = \left\{ \begin{array}{l} \text{shares} \leftarrow \text{Share}(m) \\ \widetilde{\text{shares}} \leftarrow f(\text{shares}) \\ \widetilde{m} \leftarrow \text{Rec}(\widetilde{\text{shares}}_T) \\ \text{Output } \widetilde{m} \end{array} \right\}$$

$$NMIdeal_m^{NMSim^{f,T}} = \left\{ \begin{array}{l} \tilde{m} \leftarrow NMSim^{f,T} \\ \text{If } \tilde{m} = \text{same}^*, \text{ Output } m \\ \text{Else, Output } \tilde{m} \end{array} \right\}$$

3 Locally Reconstructable Non-malleable Secret Sharing Scheme(LRNMSSS)

In this section, we define and construct non-malleable secret sharing scheme with local reconstructability. Intuitively, this gives a way to secret share blocks of messages such that in order to recover a single block of message, a small number of bits from each share in a reconstruction set is needed.

3.1 LRNMSS - Definition

► **Definition 4.** (*LRNMSSS*) Let $(\text{Share}, \text{Rec})$ be a (t, n) -secret sharing scheme for message space \mathcal{M} . The scheme $\Sigma = (\text{Share}, \text{Local}, \text{Rec})$ is called a (t, n, p) -locally reconstructable non-malleable secret sharing scheme for with message space $\mathcal{M} = \mathcal{M}_1 \times \dots \times \mathcal{M}_k$ and $\mathcal{M}_i \subseteq \{0, 1\}^\rho \forall i \in [k]$ if:

1. **Local Reconstruction.** For any $M = (m_1, \dots, m_k) \in \mathcal{M}$ where $m_i \in \mathcal{M}_i \forall i \in [k]$, for any $i \in [k]$ and for any set $T \subseteq [n]$ such that $|T| \geq t$, there exists a deterministic function Local such that,

$$Pr[\text{Local}^{\text{Share}(M)^T}(i) = m_i] = 1$$

where Local reads at most p bits from each share in T .

2. **Non-malleability.** Let \mathcal{F} be some family of tampering functions. The scheme is said to be non-malleable w.r.t \mathcal{F} if for each $f \in \mathcal{F}$ and set $T \subseteq [n]$ such that $|T| = t$, there exists a distribution $\text{Sim}^{f,T}$ such that $\forall M \in \mathcal{M}$,

$$\text{Tamper}_M^{f,T} \approx_c \text{Ideal}_M^{\text{Sim}^{f,T}}$$

where $\text{Tamper}_M^{f,T}$ and $\text{Ideal}_M^{\text{Sim}^{f,T}}$ are distributions defined as below:

$$\text{Tamper}_M^{f,T} = \left\{ \begin{array}{l} \text{shares} \leftarrow \text{Share}(M) \\ \widetilde{\text{shares}} \leftarrow f(\text{shares}) \\ \widetilde{M} \leftarrow \text{Rec}(\widetilde{\text{shares}}_T) \\ \text{Output } \widetilde{M} \end{array} \right\}$$

$$\text{Ideal}_M^{\text{Sim}^{f,T}} = \left\{ \begin{array}{l} (\mathcal{I}^*, M^*) \leftarrow \text{Sim}^{f,T} \\ \text{If } \mathcal{I}^* = [k], \text{ set } \widetilde{M} = M^* \\ \text{Else, set } \widetilde{M}|_{\mathcal{I}^*} = \perp \text{ and } \widetilde{M}|_{\overline{\mathcal{I}^*}} = M|_{\overline{\mathcal{I}^*}} \\ \text{Output } \widetilde{M} \end{array} \right\}$$

Now we describe the tampering model we consider in this paper.

3.2 Our Model - Lookahead Tampering

The message is partitioned into k blocks of length ρ . Let Share be a sharing function which takes as input a message $M \in \{0, 1\}^{k\rho}$ and outputs n shares, namely $\text{share}_1, \dots, \text{share}_n$ where each $\text{share}_i \in \{0, 1\}^{\hat{\gamma}} \times \{0, 1\}^{k\hat{\rho}}$. Each share can be viewed as $k + 1$ blocks². The first block

² To have correspondence with the explanation in the introduction, one can consider the first block of each share to be a_i and the remaining k blocks to be b_i .

is of length $\hat{\gamma}$ and next k blocks are of length $\hat{\rho}$. Let $\mathcal{F}_{nm}(\subseteq \{ f \mid f : \{0,1\}^{n\hat{\gamma}} \rightarrow \{0,1\}^{n\hat{\gamma}} \})$ be some set of tampering functions. We define a lookahead tampering family \mathcal{F} specific to \mathcal{F}_{nm} . The tampering function family consists of functions of the form (f_1, f_2) where f_1 takes as input the first blocks of the shares under the constraint that $f_1 \in \mathcal{F}_{nm}$. The rest of the blocks in the shares are hardwired in function f_1 . The function f_2 takes as input the last k blocks of all the shares.

Our tampering family is defined as :

$$\mathcal{F} = \left\{ f : (f_1, f_2) \mid \begin{array}{l} \forall (x_1, \dots, x_{nk}) \in \{0,1\}^{nk\hat{\rho}}, f_1(\cdot, x_1, \dots, x_{nk}) \in \mathcal{F}_{nm}, \\ f_2 : \{0,1\}^{nk\hat{\rho}} \rightarrow \{0,1\}^{nk\hat{\rho}} \end{array} \right\}$$

3.3 Our Construction

We use the following building blocks for the LRNMSS compiler.

1. A symmetric key authenticated encryption scheme $\mathcal{E} = (\text{Gen}, \text{Encrypt}, \text{Decrypt})$ as in Def. 1 with key space $\mathcal{K} \subseteq \{0,1\}^\gamma$, message space $\mathcal{M} \subseteq \{0,1\}^{\rho+\log k}$, where k is the number of message blocks defined in Def. 4 and ciphertext space $\mathcal{C} \subseteq \{0,1\}^{\hat{\rho}}$.
2. A non-malleable secret sharing scheme $\Sigma' = (\text{NMShare}_n^t, \text{NMRec}_n^t)$, which is non-malleable w.r.t \mathcal{F}_{nm} as in Def. 3 with message space $\mathcal{M} \subseteq \{0,1\}^\gamma$ and share-space $\mathcal{S}_i \subseteq \{0,1\}^{\hat{\gamma}}$ for all $i \in [n]$.

Our construction combines a symmetric key authenticated encryption scheme(Def. 1) and a non-malleable secret sharing scheme (Def. 3) to obtain a locally reconstructable non-malleable secret sharing scheme(Def. 4).

Upon receiving a message having k blocks, **Share** function generates a key using the **Gen** function of the authenticated encryption and then encrypts each of the message block along with its index using the key. The key is shared using a non-malleable secret sharing scheme. A share of the key along with the ciphertexts constitutes a share of LRNMSSS.

On input an index i , **Local** function reads first and $(i+1)^{th}$ block of every share in a reconstruction set. Using the first blocks, **Local** recovers key using NMRec_n^t function of the non-malleable secret sharing. A consistency check is also made to make sure that all the $(i+1)^{th}$ blocks are the same. The $(i+1)^{th}$ block is decrypted using the recovered key. It performs a check to make sure that the index decrypted is the same as input index. **Local** outputs decrypted message block.

Rec function reads the shares in a reconstruction set and parses the shares as $k+1$ blocks. First block correspond to the shares of the key. Using NMRec_n^t , it recovers the key. It performs consistency checks to make sure that all the ciphertext corresponding to an index are same. **Rec** outputs the concatenation of the decrypted messages.

The LRNMSS compiler is defined as:

Share(M): On input $M = (m_1, m_2, \dots, m_k)$,

1. $sk \leftarrow \text{Gen}(1^\lambda)$.
2. $e_j \leftarrow \text{Encrypt}_{sk}(m_j, j) \forall j \in [k]$.
3. $(sk_1, \dots, sk_n) \leftarrow \text{NMShare}_n^t(sk)$.
4. Output $share_i = (sk_i, e_1, \dots, e_k)$.

Local^{share $_T$} (j) : For any reconstruction set $T = \{i_1, \dots, i_t\} \subseteq [n]$ such that $|T| \geq t$, sk_i and e_j^i represents the first and $(j + 1)^{th}$ block of $share_i$ respectively. Local reads $(sk_{i_1}, e_j^{i_1}), \dots, (sk_{i_t}, e_j^{i_t})$ from $share_T$ on input index $j \in [k]$ and evaluates,

1. If $\exists i_a, i_b$ s.t. $e_j^{i_a} \neq e_j^{i_b}$, output \perp and terminate.
2. Else, recover $sk \leftarrow \text{NMRec}_n^t(sk_{i_1}, \dots, sk_{i_t})$.
3. Recover $(\tilde{m}_j, \tilde{j}) \leftarrow \text{Decrypt}_{sk}(e_j^{i_1})$.
 - a. If $\tilde{j} \neq j$, output \perp and terminate.
4. Output \tilde{m}_j .

Rec($share_T$) : For any reconstruction set $T = \{i_1, \dots, i_t\} \subseteq [n]$ such that $|T| \geq t$ and input $share_T = share_{i_1}, \dots, share_{i_t}$,

1. Parse $share_{i_j}$ as $(sk_{i_j}, e_1^{i_j}, \dots, e_k^{i_j})$.
2. $sk \leftarrow \text{NMRec}_n^t(sk_{i_1}, \dots, sk_{i_t})$.
3. For each $j \in [k]$,
 - a. If $\exists i_a, i_b$ s.t. $e_j^{i_a} \neq e_j^{i_b}$, set $\tilde{m}_j = \perp$.
 - b. Else, $(\tilde{m}_j, \tilde{j}) \leftarrow \text{Decrypt}_{sk}(e_j^{i_1})$.
 - i. If $\tilde{j} \neq j$, set $\tilde{m}_j = \perp$.
4. Output $\tilde{m}_1, \dots, \tilde{m}_k$.

3.4 Security Analysis

► Theorem 5. Let $\mathcal{E} = (\text{Gen}, \text{Encrypt}, \text{Decrypt})$ be a symmetric key authenticated encryption scheme with ciphertext space $\mathcal{C} \subseteq \{0, 1\}^{\hat{\rho}}$, $\Sigma' = (\text{NMShare}_n^t, \text{NMRec}_n^t)$ be a (t, n) non-malleable secret sharing scheme which is non-malleable w.r.t \mathcal{F}_{nm} with share space $\mathcal{S}_i \subseteq \{0, 1\}^{\hat{\gamma}}$ for all $i \in [n]$. Then, the scheme $\Sigma = (\text{Share}, \text{Local}, \text{Rec})$ defined above is a $(t, n, \hat{\gamma} + \hat{\rho})$ -locally reconstructable non-malleable secret sharing scheme which is non-malleable with respect to \mathcal{F} , the lookahead family specific to \mathcal{F}_{nm} .

Proof. Correctness. The correctness of the scheme follows from the correctness of the underlying secret sharing scheme and encryption scheme.

Local Reconstruction. On input an index i , Local function reads first block and $i + 1^{th}$ block of the shares in a reconstruction set T . It recovers key from the first blocks of the shares using NMRec_n^t . That would require Local to read $\hat{\gamma}$ bits from each share in T .

Local then checks if the $i + 1^{th}$ block of each share in T are same or not. This would require Local to read $\hat{\rho}$ bits from each share in T . If yes, the $i + 1^{th}$ ciphertext block is decrypted using the recovered key. It performs a check to make sure that $i + 1^{th}$ ciphertext block corresponds to the message block m_i . It checks if the decrypted index is the same as input index.

Thus, Local needs to read $p = (\hat{\gamma} + \hat{\rho})$ bits from each share in T . Total number of bits to be retrieved to reconstruct a single block m_i is $t(\hat{\gamma} + \hat{\rho})$.

Privacy. Let $T \subset [n]$ with $|T| < t$, be an arbitrary set. We wish to show that for any two messages $M^0, M^1 \in \mathcal{M}$, $\text{Share}(M^0)_T \approx_c \text{Share}(M^1)_T$.

We show this through a sequence of hybrids:

- **Hybrid₀**: This corresponds to the shares of $M^0 = (m_1^0, \dots, m_k^0)$ in the set T . Generate $sk \leftarrow \text{Gen}(1^\lambda)$. Further, $e_j \leftarrow \text{Encrypt}_{sk}(m_j^0, j)$ for $j \in [k]$ and $(sk_1, \dots, sk_n) \leftarrow \text{NMShare}_n^t(sk)$. Set $share_i = (sk_i, e_1, \dots, e_k)$ for each $i \in T$. Output $\{share_i\}_{i \in T}$.
- **Hybrid₁**: Generate a new key $sk' \leftarrow \text{Gen}(1^\lambda)$ and replace the shares of sk in the set T with the shares of sk' .
Generate $sk \leftarrow \text{Gen}(1^\lambda)$ and $sk' \leftarrow \text{Gen}(1^\lambda)$. Further, $e_j \leftarrow \text{Encrypt}_{sk}(m_j^0, j)$ for $j \in [k]$ and $(sk'_1, \dots, sk'_n) \leftarrow \text{NMShare}_n^t(sk')$. Set $share_i = (sk'_i, e_1, \dots, e_k)$ for each $i \in T$. Output $\{share_i\}_{i \in T}$.
- **Hybrid₂**: Replace the encryptions of message M^0 with encryptions of message $M^1 = (m_1^1, \dots, m_k^1)$.
Generate $sk \leftarrow \text{Gen}(1^\lambda)$ and $sk' \leftarrow \text{Gen}(1^\lambda)$. Further, $e'_j \leftarrow \text{Encrypt}_{sk}(m_j^1, j)$ for $j \in [k]$ and $(sk'_1, \dots, sk'_n) \leftarrow \text{NMShare}_n^t(sk')$. Set $share_i = (sk'_i, e'_1, \dots, e'_k)$ for each $i \in T$. Output $\{share_i\}_{i \in T}$.
- **Hybrid₃**: This corresponds to the shares of $M^1 = (m_1^1, \dots, m_k^1)$ in the set T . Generate $sk \leftarrow \text{Gen}(1^\lambda)$. Further, $e'_j \leftarrow \text{Encrypt}_{sk}(m_j^1, j)$ for $j \in [k]$ and $(sk_1, \dots, sk_n) \leftarrow \text{NMShare}_n^t(sk)$. Set $share_i = (sk_i, e'_1, \dots, e'_k)$ for each $i \in T$. Output $\{share_i\}_{i \in T}$.

Here, $\text{Hybrid}_0 \equiv \text{Share}(M^0)_T$ and $\text{Hybrid}_3 \equiv \text{Share}(M^1)_T$.

By the computational privacy of $\Sigma' = (\text{NMShare}_n^t, \text{NMRec}_n^t)$, we get $\text{Hybrid}_0 \approx_c \text{Hybrid}_1$ and by the semantic security of $\mathcal{E} = (\text{Gen}, \text{Encrypt}, \text{Decrypt})$, it follows $\text{Hybrid}_1 \approx_c \text{Hybrid}_2$. Finally, again by the computational privacy of $\Sigma' = (\text{NMShare}_n^t, \text{NMRec}_n^t)$, it follows $\text{Hybrid}_2 \approx_c \text{Hybrid}_3$.

Thus, $\text{Share}(M^0)_T \equiv \text{Hybrid}_0 \approx_c \text{Hybrid}_1 \approx_c \text{Hybrid}_2 \approx_c \text{Hybrid}_3 \equiv \text{Share}(M^1)_T$.

Non-malleability. To show the non-malleability of our scheme, we need to show that $\forall f \in \mathcal{F}, \forall T \subseteq [n]$ such that $|T| = t$, $\exists \text{Sim}^{f,T}$ such that $\forall M \in \mathcal{M}$

$$\text{Tamper}_M^{f,T} \approx_c \text{Ideal}_M^{\text{Sim}^{f,T}}$$

For any $f = (f_1, f_2) \in \mathcal{F}$ and any reconstruction set $T = \{i_1, \dots, i_t\}$, we begin by describing the simulator $\text{Sim}^{f,T}$.

For each $(e_1, \dots, e_k) \in \{0, 1\}^{k\hat{\rho}}$, we define a function $g : \{0, 1\}^{n\hat{\gamma}} \rightarrow \{0, 1\}^{n\hat{\gamma}}$, hardwired with n copies of (e_1, \dots, e_k) , as $g(x) = f_1(x, (e_1^i, \dots, e_k^i)_{i \in [n]})$, $\forall x \in \{0, 1\}^{n\hat{\gamma}}$ where $(e_1^i, \dots, e_k^i) = (e_1, \dots, e_k), \forall i \in [n]$. Hence, by definition of \mathcal{F} , $g \in \mathcal{F}_{nm}$. Let $\text{NMSim}^{g,T}$ be the simulator for the underlying NMSS (which is non-malleable w.r.t \mathcal{F}_{nm}) and ϕ denote the empty string.

11:10 Locally Reconstructable Non-Malleable Secret Sharing

$\text{Sim}^{f,T}$:

1. $sk \leftarrow \text{Gen}(1^\lambda)$.
2. $e_j \leftarrow \text{Encrypt}_{sk}(0^{\rho+\log k}) \forall j \in [k]$.
3. Set $e_j^i = e_j \forall j \in [k], \forall i \in [n]$ and hardwire them in g .
4. $\widetilde{sk} \leftarrow \text{NMSim}^{g,T}$.
5. If $\widetilde{sk} = \text{same}^*$,
 - a. $(\widetilde{e}_1^i, \dots, \widetilde{e}_k^i)_{i \in [n]} \leftarrow f_2((e_1^i, \dots, e_k^i)_{i \in [n]})$.
 - b. $\mathcal{I} = \{j : \widetilde{e}_j^{i_a} \neq \widetilde{e}_j^{i_b} \text{ for some } i_a, i_b \in T\}$.
 - c. $\mathcal{I}_1 = \{j : \exists i_c \in T \text{ s.t. } \widetilde{e}_j^{i_c} \neq e_j \text{ and } \forall i_a, i_b \in T, \widetilde{e}_j^{i_a} = \widetilde{e}_j^{i_b}\}$.
 - d. Set $(\mathcal{I}^*, M^*) = (\mathcal{I} \cup \mathcal{I}_1, \phi)$.
6. Else,
 - a. $(\widetilde{e}_1^i, \dots, \widetilde{e}_k^i)_{i \in [n]} \leftarrow f_2((e_1^i, \dots, e_k^i)_{i \in [n]})$.
 - b. $\mathcal{I} = \{j : \widetilde{e}_j^{i_a} \neq \widetilde{e}_j^{i_b} \text{ for some } i_a, i_b \in T\}$.
 - c. $\forall j \in \mathcal{I}$, set $\widetilde{m}_j = \perp$.
 - d. $\forall j \notin \mathcal{I}$, $(\widetilde{m}_j, \widetilde{j}) \leftarrow \text{Decrypt}_{\widetilde{sk}}(\widetilde{e}_j^{i_1})$.
 - i. If $\widetilde{j} \neq j$, set $\widetilde{m}_j = \perp$.
 - e. Set $(\mathcal{I}^*, M^*) = ([k], \widetilde{m}_1, \dots, \widetilde{m}_k)$.
7. Output (\mathcal{I}^*, M^*) .

For any $f = (f_1, f_2) \in \mathcal{F}$, any reconstruction set $T = \{i_1, \dots, i_t\}$ and any message $M = (m_1, \dots, m_k) \in \mathcal{M}$, we define the tamper distribution, $\text{Tamper}_M^{f,T}$ as below.

$\text{Tamper}_M^{f,T}$:

1. $sk \leftarrow \text{Gen}(1^\lambda)$.
2. $e_j \leftarrow \text{Encrypt}_{sk}(m_j, j) \forall j \in [k]$.
3. Set $e_j^i = e_j \forall j \in [k], \forall i \in [n]$.
4. $(sk_1, \dots, sk_n) \leftarrow \text{NMShare}_n^t(sk)$.
5. $\widetilde{\text{share}}_i = (sk_i, \widetilde{e}_1^i, \dots, \widetilde{e}_k^i)$.
6. $(\widetilde{\text{share}}_1, \dots, \widetilde{\text{share}}_n) \leftarrow f(\text{share}_1, \dots, \text{share}_n)$.
7. Parse $\widetilde{\text{share}}_i$ as $(\widetilde{sk}_i, \widetilde{e}_1^i, \dots, \widetilde{e}_k^i) \forall i \in [n]$.
8. $\widetilde{sk} \leftarrow \text{NMRec}_n^t(\widetilde{sk}_{i_1}, \dots, \widetilde{sk}_{i_t})$.
9. For each $j \in [k]$
 - a. If $\exists i_a, i_b \in T$ s.t. $\widetilde{e}_j^{i_a} \neq \widetilde{e}_j^{i_b}$, set $\widetilde{m}_j = \perp$.
 - b. Else, $(\widetilde{m}_j, \widetilde{j}) \leftarrow \text{Decrypt}_{\widetilde{sk}}(\widetilde{e}_j^{i_1})$.
 - i. If $\widetilde{j} \neq j$, set $\widetilde{m}_j = \perp$.
10. Output $\widetilde{m}_1, \dots, \widetilde{m}_k$.

Now, through a sequence of hybrids, we show that the $\text{Tamper}_M^{f,T}$ and $\text{Ideal}_M^{\text{Sim}^{f,T}}$ are computationally indistinguishable.

We define the first hybrid, which only has some notational changes with respect to $\text{Tamper}_M^{f,T}$ and is equivalent to it.

Rewriting $\text{Tamper}_M^{f,T}$ as $\text{Hybrid1}_M^{f,T}$: $\text{Hybrid1}_M^{f,T}$ is the same as the tampering experiment with few differences. We expand the function f giving f_1 and f_2 . Then, f_2 is placed after the non-malleable secret reconstruction, because NMRec_n^t doesn't depend on the output of f_2 . A new variable \mathcal{I} is also defined to maintain the indices having inconsistent ciphertexts.

Steps (5) – (10) of $\text{Tamper}_M^{f,T}$ is replaced with the following steps (5) – (10) in $\text{Hybrid1}_M^{f,T}$.

5. $\widetilde{sk}_1, \dots, \widetilde{sk}_n \leftarrow f_1(sk_1, \dots, sk_n, (e_1^i, \dots, e_k^i)_{i \in [n]})$.
6. $\widetilde{sk} \leftarrow \text{NMRec}_n^t(\widetilde{sk}_{i_1}, \dots, \widetilde{sk}_{i_t})$.

7. $(\tilde{e}_1^i, \dots, \tilde{e}_k^i)_{i \in [n]} \leftarrow f_2((e_1^i, \dots, e_k^i)_{i \in [n]})$.
8. $\mathcal{I} = \{j : \tilde{e}_j^{i_a} \neq \tilde{e}_j^{i_b} \text{ for some } i_a, i_b \in T\}$.
9. a. $\forall j \in \mathcal{I}$, set $\tilde{m}_j = \perp$.
b. $\forall j \notin \mathcal{I}$, $(\tilde{m}_j, \tilde{j}) \leftarrow \text{Decrypt}_{sk}^{\sim}(\tilde{e}_j^{i_1})$.
i. If $\tilde{j} \neq j$, set $\tilde{m}_j = \perp$.
10. Output $\tilde{m}_1, \dots, \tilde{m}_k$.

▷ Claim 6. For any $M \in \mathcal{M}$, $f \in \mathcal{F}$, and any set $T \subseteq [n]$ such that $|T| = t$,

$$\text{Tamper}_M^{f,T} \equiv \text{Hybrid1}_M^{f,T}.$$

Proof. Clearly, only the notations were modified in $\text{Hybrid1}_M^{f,T}$ and the distribution remains the same. Hence $\text{Tamper}_M^{f,T}$ is identical to $\text{Hybrid1}_M^{f,T}$. ◁

In our next hybrid, we use the non-malleability of our underlying NMSS. Hence, we replace the tamper distribution of the underlying NMSS with its simulator.

Going from $\text{Hybrid1}_M^{f,T}$ to $\text{Hybrid2}_M^{f,T}$: $\text{Hybrid2}_M^{f,T}$ is the same as $\text{Hybrid1}_M^{f,T}$, except that the simulator, $\text{NMSim}^{g,T}$, for the underlying NMSS, $\Sigma' = (\text{NMShare}_n^t, \text{NMRec}_n^t)$, is used to generate the tampered key \tilde{sk} .

Steps (4) – (6) of $\text{Hybrid1}_M^{f,T}$ is replaced with the following steps (4) – (5) in $\text{Hybrid2}_M^{f,T}$.

4. $\tilde{sk} \leftarrow \text{NMSim}^{g,T}$.
5. If $\tilde{sk} = \text{same}^*$, set $\tilde{sk} = sk$.

▷ Claim 7. If $\Sigma' = (\text{NMShare}_n^t, \text{NMRec}_n^t)$ is a non-malleable secret sharing scheme w.r.t. \mathcal{F}_{nm} , then for any $M \in \mathcal{M}$, $f \in \mathcal{F}$, and any set $T \subseteq [n]$ such that $|T| = t$,

$$\text{Hybrid1}_M^{f,T} \approx_c \text{Hybrid2}_M^{f,T}.$$

Proof. If the two hybrids are computationally distinguishable, we can build an adversary \mathcal{A} breaking the non-malleable property of the Σ' . Let D be the distinguisher that can distinguish between $\text{Hybrid1}_M^{f,T}$ and $\text{Hybrid2}_M^{f,T}$.

The adversary \mathcal{A} is defined as follows:

1. \mathcal{A} generates $sk \leftarrow \text{Gen}(1^\lambda)$.
2. \mathcal{A} computes $e_j \leftarrow \text{Encrypt}_{sk}(m_j, j)$ for $j \in [k]$.
3. Set $e_j^i = e_j \forall j \in [k], \forall i \in [n]$.
4. \mathcal{A} sends $sk, g, (e_1^i, \dots, e_k^i)_{i \in [n]}$ to the challenger.
5. \mathcal{A} after receiving challenge \tilde{sk} from the challenger, does the following:
 - a. $(\tilde{e}_1^i, \dots, \tilde{e}_k^i)_{i \in [n]} \leftarrow f_2((e_1^i, \dots, e_k^i)_{i \in [n]})$
 - b. $\mathcal{I} = \{j : \tilde{e}_j^{i_a} \neq \tilde{e}_j^{i_b} \text{ for some } i_a, i_b \in T\}$
 - c. $\forall j \in \mathcal{I}$, set $\tilde{m}_j = \perp$
 - d. $\forall j \notin \mathcal{I}$, $(\tilde{m}_j, \tilde{j}) \leftarrow \text{Decrypt}_{\tilde{sk}}^{\sim}(\tilde{e}_j^{i_1})$
i. If $\tilde{j} \neq j$, set $\tilde{m}_j = \perp$.
 - e. Sends $D(\tilde{m}_1, \dots, \tilde{m}_k)$ to the challenger.

If the challenge corresponds to the output of the tampered experiment $\text{NMTamper}_{sk}^{g,T}$, then D will be invoked with distribution corresponding to $\text{Hybrid1}_M^{f,T}$. Otherwise, D will be invoked with distribution corresponds to the simulated experiment $\text{NMIdeal}_{sk}^{\text{NMSim}^{g,T}}$. This contradicts the non-malleability property of Σ' . ◁

11:12 Locally Reconstructable Non-Malleable Secret Sharing

In the next hybrid, we only make a few notational changes, leading to an identical distribution.

Rewriting $\text{Hybrid2}_M^{f,T}$ as $\text{Hybrid3}_M^{f,T}$: In $\text{Hybrid3}_M^{f,T}$, $\widetilde{sk} = \text{same}^*$ and $\widetilde{sk} \neq \text{same}^*$ are considered as two different cases. When $\widetilde{sk} = \text{same}^*$, a new variable \mathcal{I}_1 is defined to maintain the indices having consistent, but tampered ciphertexts. For all the indices other than those in $\mathcal{I} \cup \mathcal{I}_1$, the key and the ciphertexts were not tampered. Decrypt outputs the original message block, m_i , at those indices.

Steps (5) – (10) of $\text{Hybrid2}_M^{f,T}$ is replaced with the following steps (5) – (7) in $\text{Hybrid3}_M^{f,T}$.

5. If $\widetilde{sk} = \text{same}^*$,
 - a. $(\tilde{e}_1^i, \dots, \tilde{e}_k^i)_{i \in [n]} \leftarrow f_2((e_1^i, \dots, e_k^i)_{i \in [n]})$.
 - b. $\mathcal{I} = \{j : \tilde{e}_j^{i_a} \neq \tilde{e}_j^{i_b} \text{ for some } i_a, i_b \in T\}$.
 - c. $\mathcal{I}_1 = \{j : \exists i_c \in T \text{ s.t. } \tilde{e}_j^{i_c} \neq e_j \text{ and } \forall i_a, i_b \in T, \tilde{e}_j^{i_a} = \tilde{e}_j^{i_b}\}$.
 - d. $\forall j \in \mathcal{I}$, set $\widetilde{m}_j = \perp$.
 - e. $\forall j \in \mathcal{I}_1$, $(\widetilde{m}_j, j) \leftarrow \text{Decrypt}_{sk}(\tilde{e}_j^{i_1})$.
 - i. If $\tilde{j} \neq j$, set $\widetilde{m}_j = \perp$.
 - f. $\forall j \notin \mathcal{I} \cup \mathcal{I}_1$, set $\widetilde{m}_j = m_j$.
6. Else,
 - a. $(\tilde{e}_1^i, \dots, \tilde{e}_k^i)_{i \in [n]} \leftarrow f_2((e_1^i, \dots, e_k^i)_{i \in [n]})$.
 - b. $\mathcal{I} = \{j : \tilde{e}_j^{i_a} \neq \tilde{e}_j^{i_b} \text{ for some } i_a, i_b \in T\}$.
 - c. $\forall j \in \mathcal{I}$, set $\widetilde{m}_j = \perp$.
 - d. $\forall j \notin \mathcal{I}$, $(\widetilde{m}_j, j) \leftarrow \text{Decrypt}_{\widetilde{sk}}(\tilde{e}_j^{i_1})$.
 - i. If $\tilde{j} \neq j$, set $\widetilde{m}_j = \perp$.
7. Output $\widetilde{m}_1, \dots, \widetilde{m}_k$.

▷ **Claim 8.** For any $M \in \mathcal{M}$, $f \in \mathcal{F}$, and any set $T \subseteq [n]$ such that $|T| = t$ indices,

$$\text{Hybrid2}_M^{f,T} \equiv \text{Hybrid3}_M^{f,T}.$$

Proof. The only difference between these hybrids are that instead of a single case in $\text{Hybrid2}_M^{f,T}$, two cases were introduced in $\text{Hybrid3}_M^{f,T}$ with both the cases executing the same steps. Hence, they are identical distributions. ◁

In our next hybrid, we use the authenticity property of the underlying authenticated encryption scheme in order to completely remove the use of the original secret key sk .

Going from $\text{Hybrid3}_M^{f,T}$ to $\text{Hybrid4}_M^{f,T}$: In $\text{Hybrid4}_M^{f,T}$, the decrypted messages corresponding to the indices in the set \mathcal{I}_1 are set to \perp .

Step 5(e) in $\text{Hybrid3}_M^{f,T}$ is replaced with the following step in $\text{Hybrid4}_M^{f,T}$.

5. e. $\forall j \in \mathcal{I}_1$, set $\widetilde{m}_j = \perp$.

▷ **Claim 9.** If $\mathcal{E} = (\text{Gen}, \text{Encrypt}, \text{Decrypt})$ is an authenticated symmetric key encryption scheme, then for any $M \in \mathcal{M}$, $f \in \mathcal{F}$, and any set $T \subseteq [n]$ such that $|T| = t$,

$$\text{Hybrid3}_M^{f,T} \approx_c \text{Hybrid4}_M^{f,T}.$$

Proof. If the hybrids are computationally distinguishable, we can build an adversary which can break the authenticity property of the encryption scheme. Note that the two hybrids differ only in the case where $\widetilde{sk} = \text{same}^*$ and the set \mathcal{I}_1 is non-empty and are identical otherwise. Pick a message $M = (m_1, \dots, m_k) \in \mathcal{M}$ for which the two hybrids are distinguishable.

Adversary \mathcal{A} , which can compute a valid new ciphertext, is defined as:

1. \mathcal{A} sends $(m_j, j)_{j \in [k]}$ as queries to the challenger.

2. \mathcal{A} on receiving e_1, \dots, e_k from the challenger, does the following
 - a. Set $e_j^i = e_j \forall j \in [k], \forall i \in [n]$.
 - b. $\widetilde{sk} \leftarrow \text{NMSim}^{g,T}$.
 - c. $(\widetilde{e}_1^i, \dots, \widetilde{e}_k^i)_{i \in [n]} \leftarrow f_2((e_1^i, \dots, e_k^i)_{i \in [n]})$.
 - d. $\mathcal{I}_1 = \{j : \exists i_c \in T \text{ s.t. } \widetilde{e}_j^{i_c} \neq e_j \text{ and } \forall i_a, i_b \in T, \widetilde{e}_j^{i_a} = \widetilde{e}_j^{i_b}\}$.
 - e. \mathcal{A} sends $(\widetilde{e}_j^{i_1})_{j \in \mathcal{I}_1}$ to the challenger.

From our assumption that the two hybrids are distinguishable, we know that there exists some $j \in \mathcal{I}_1$ such that the corresponding ciphertext $\widetilde{e}_j^{i_1}$ is valid, i.e., $\text{Decrypt}_{sk}(\widetilde{e}_j^{i_1}) \neq \perp$. Since $j \in \mathcal{I}_1$, we know that $\widetilde{e}_j^{i_a} = \widetilde{e}_j^{i_b}$ for all $i_a, i_b \in T$ and $\widetilde{e}_j^{i_1} \neq e_j$. Moreover, because of the index j being appended to the message, $\widetilde{e}_j^{i_1} \neq e_q$, for each $q \in [k]$. This implies that the adversary \mathcal{A} outputs a valid ciphertext, which it did not receive as a challenge, hence breaking the authenticity of the encryption. \triangleleft

Next, we use the semantic security of the authentication scheme to move to a hybrid where, the actual message is no longer used in the encryption.

Going from $\text{Hybrid4}_M^{f,T}$ to $\text{Hybrid5}_M^{f,T}$: In $\text{Hybrid5}_M^{f,T}$, the ciphertexts corresponding to M are replaced with the ciphertexts corresponding to $0^{\rho+\log k}$.

Step 2 in $\text{Hybrid4}_M^{f,T}$ is replaced with the following step in $\text{Hybrid5}_M^{f,T}$.

2. $e_j \leftarrow \text{Encrypt}_{sk}(0^{\rho+\log k}) \forall j \in [k]$

\triangleright Claim 10. If $\mathcal{E} = (\text{Gen}, \text{Encrypt}, \text{Decrypt})$ is an authenticated encryption scheme, then for any $M \in \mathcal{M}, f \in \mathcal{F}$, and any set $T \subseteq [n]$ such that $|T| = t$,

$$\text{Hybrid4}_M^{f,T} \approx_c \text{Hybrid5}_M^{f,T}.$$

Proof. Assume to the contrary that, there exists $M \in \mathcal{M}$ and a distinguisher D that can distinguish between the hybrids $\text{Hybrid4}_M^{f,T}$ and $\text{Hybrid5}_M^{f,T}$. The distinguisher D can be used to construct another distinguisher D_1 which violates the semantic security of the underlying encryption scheme \mathcal{E} .

The distinguisher D_1 is defined as follows:

1. D_1 sets $M_0 = (m_j, j)_{j \in [k]}, M_1 = 0^{k(\rho+\log k)}$.
2. D_1 sends (M_0, M_1) to the challenger.
3. D_1 on receiving (e_1, \dots, e_k) from the challenger, does the following,
 - a. Set $e_j^i = e_j \forall j \in [k], \forall i \in [n]$.
 - b. $\widetilde{sk} \leftarrow \text{NMSim}^{g,T}$.
 - c. If $\widetilde{sk} = \text{same}^*$,
 - i. $(\widetilde{e}_1^i, \dots, \widetilde{e}_k^i)_{i \in [n]} \leftarrow f_2((e_1^i, \dots, e_k^i)_{i \in [n]})$.
 - ii. $\mathcal{I} = \{j : \widetilde{e}_j^{i_a} \neq \widetilde{e}_j^{i_b} \text{ for some } i_a, i_b \in T\}$.
 - iii. $\mathcal{I}_1 = \{j : \exists i_c \in T \text{ s.t. } \widetilde{e}_j^{i_c} \neq e_j \text{ and } \forall i_a, i_b \in T, \widetilde{e}_j^{i_a} = \widetilde{e}_j^{i_b}\}$.
 - iv. $\forall j \in \mathcal{I}$, set $\widetilde{m}_j = \perp$.
 - v. $\forall j \in \mathcal{I}_1$, set $\widetilde{m}_j = \perp$.
 - vi. $\forall j \notin \mathcal{I} \cup \mathcal{I}_1$, set $\widetilde{m}_j = m_j$.
 - d. Else,
 - i. $(\widetilde{e}_1^i, \dots, \widetilde{e}_k^i)_{i \in [n]} \leftarrow f_2((e_1^i, \dots, e_k^i)_{i \in [n]})$.
 - ii. $\mathcal{I} = \{j : \widetilde{e}_j^{i_a} \neq \widetilde{e}_j^{i_b} \text{ for some } i_a, i_b \in T\}$.
 - iii. $\forall j \in \mathcal{I}$, set $\widetilde{m}_j = \perp$.
 - iv. $\forall j \notin \mathcal{I}$, $(\widetilde{m}_j, \widetilde{j}) \leftarrow \text{Decrypt}_{\widetilde{sk}}(\widetilde{e}_j^{i_1})$.
 - A. If $\widetilde{j} \neq j$, set $\widetilde{m}_j = \perp$.
- e. D_1 sends $D(\widetilde{m}_1, \dots, \widetilde{m}_k)$ to the challenger.

11:14 Locally Reconstructable Non-Malleable Secret Sharing

If the challenge corresponds to the ciphertext of M_0 , then D will be invoked with the distribution corresponding to $\text{Hybrid4}_M^{f,T}$. Otherwise, the distribution corresponds to $\text{Hybrid5}_M^{f,T}$. This contradicts the semantic security of \mathcal{E} . \triangleleft

We finally make some notational changes to the above hybrid, to get an identical distribution, which would be $\text{IdealSim}_M^{f,T}$.

Rewriting $\text{Hybrid5}_M^{f,T}$ as $\text{Hybrid6}_M^{f,T}$: The new variable \mathcal{I}^* represents the set of tampered indices. If shares are entirely tampered, the output will be independent of the original message. If the tampering function doesn't change first blocks of shares, then any modification will output \perp . \mathcal{I}^* keeps track of these indices. The simulator for the LRNMSSS outputs tampered indices along with message vector. If the shares are entirely tampered, $\text{Hybrid6}_M^{f,T}$ will output the message vector which is independent of original message. If the first block is not tampered, the $\text{Hybrid6}_M^{f,T}$ outputs \perp at indices in \mathcal{I}^* and original messages for other indices.

Steps (5) – (7) of $\text{Hybrid5}_M^{f,T}$ are replaced with the following steps in $\text{Hybrid6}_M^{f,T}$

5. If $sk = \text{same}^*$,
 - a. $(\tilde{e}_1^i, \dots, \tilde{e}_k^i)_{i \in [n]} \leftarrow f_2((e_1^i, \dots, e_k^i)_{i \in [n]})$.
 - b. $\mathcal{I} = \{j : \tilde{e}_j^{i_a} \neq \tilde{e}_j^{i_b} \text{ for some } i_a, i_b \in T\}$.
 - c. $\mathcal{I}_1 = \{j : \exists i_c \in T \text{ s.t. } \tilde{e}_j^{i_c} \neq e_j \text{ and } \forall i_a, i_b \in T, \tilde{e}_j^{i_a} = \tilde{e}_j^{i_b}\}$.
 - d. Set $(\mathcal{I}^*, M^*) = (\mathcal{I} \cup \mathcal{I}_1, \phi)$.
6. Else,
 - a. $(\tilde{e}_1^i, \dots, \tilde{e}_k^i)_{i \in [n]} \leftarrow f_2((e_1^i, \dots, e_k^i)_{i \in [n]})$.
 - b. $\mathcal{I} = \{j : \tilde{e}_j^{i_a} \neq \tilde{e}_j^{i_b} \text{ for some } i_a, i_b \in T\}$.
 - c. $\forall j \in \mathcal{I}$, set $\tilde{m}_j = \perp$.
 - d. $\forall j \notin \mathcal{I}$, $(\tilde{m}_j, \tilde{j}) \leftarrow \text{Decrypt}_{sk}(\tilde{e}_j^{i_1})$.
 - i. If $\tilde{j} \neq j$, set $\tilde{m}_j = \perp$.
 - e. Set $(\mathcal{I}^*, M^*) = ([k], \tilde{m}_1, \dots, \tilde{m}_k)$.
7. If $\mathcal{I}^* = [k]$, set $\tilde{M} = M^*$.
8. Else, set $\tilde{M}|_{\mathcal{I}^*} = \perp$ and $\tilde{M}|_{\overline{\mathcal{I}^*}} = M|_{\overline{\mathcal{I}^*}}$.
9. Output \tilde{M} .

\triangleright Claim 11. For any $M \in \mathcal{M}$, $f \in \mathcal{F}$, and any set $T \subseteq [n]$ such that $|T| = t$ indices,

$$\text{Hybrid5}_M^{f,T} \equiv \text{Hybrid6}_M^{f,T}.$$

Proof. When $\mathcal{I}^* = [k]$, key can be tampered or not. If the key was tampered, M^* stores the messages decrypted using the tampered key at all the indices. $\text{Hybrid5}_M^{f,T}$ also outputs the decrypted messages at all indices when the key is tampered. If the key was not tampered, M^* stores empty vector. $\text{Hybrid6}_M^{f,T}$ outputs M^* in that case. If $\mathcal{I}^* = [k]$, then $\mathcal{I} \cup \mathcal{I}_1 = [k]$. $\text{Hybrid5}_M^{f,T}$ outputs \perp at all the indices.

When $\mathcal{I}^* \neq [k]$, key was not tampered. Both the hybrids output \perp at those indices in $\mathcal{I}^* = \mathcal{I} \cup \mathcal{I}_1$. On all the other indices, $\text{Hybrid5}_M^{f,T}$ and $\text{Hybrid6}_M^{f,T}$ outputs original message corresponding to their indices. Both the hybrids behave the same for all the cases. Thus, $\text{Hybrid5}_M^{f,T}$ and $\text{Hybrid6}_M^{f,T}$ are identical. \triangleleft

From our description of the simulator, $\text{Sim}^{f,T}$, clearly $\text{Hybrid6}_M^{f,T}$ is the same as $\text{IdealSim}_M^{f,T}$.

By the previous claims, we get

$$\begin{aligned} \text{Tamper}_M^{f,T} &\equiv \text{Hybrid1}_M^{f,T} \approx_c \text{Hybrid2}_M^{f,T} \equiv \text{Hybrid3}_M^{f,T} \approx_c \text{Hybrid4}_M^{f,T} \\ \text{Hybrid4}_M^{f,T} &\approx_c \text{Hybrid5}_M^{f,T} \equiv \text{Hybrid6}_M^{f,T} \equiv \text{IdealSim}_M^{f,T} \end{aligned} \quad \blacktriangleleft$$

3.5 Instantiation

Let the secret to be shared in LRNMSS consists of k blocks, with size of each block ρ , where ρ is some polynomial in the computational security parameter λ . Let $(\text{NMShare}_n^t, \text{NMRec}_n^t)$ be a NMSS with length of the each share being $r(\alpha)$ when a α -bit secret is shared. Authenticated encryption $(\text{Gen}, \text{Encrypt}, \text{Dec})$ scheme is instantiated with *Encrypt-and-authenticate* scheme mentioned in Section 4.5 of [20]. We let the encryption key, randomness and tag to be of length $2\lambda, \lambda$ and λ respectively. The encryption scheme takes messages of length $\rho + \log k$, outputs a ciphertext of length $\rho + \log k + 2\lambda$.

- For messages of length $k\rho$, a single share of LRNMSS will be of length $k(\rho + \log k + 2\lambda) + r(2\lambda)$.
- Thus, the rate of LRNMSS is $\frac{k\rho}{k(\rho + \log k + 2\lambda) + r(2\lambda)}$.
- For long messages, rate = $\frac{1}{1+o(1)}$ assuming $\log k \ll \rho$.
- For local reconstruction, **Local** is required to read $\rho + \log k + 2\lambda + r(2\lambda)$ bits from each share in a reconstruction set.

4 Computational Non-malleable Multi-message Transmission in the Pre-processing Model

Perfectly secure message transmission (SMT) was introduced in [16], where a sender S wants to transmit a message m to a receiver R , through n wires between them, ensuring that perfect secrecy is guaranteed, even in the presence of an eavesdropping adversary looking at a bounded number of wires, and perfect resiliency is guaranteed, even in the presence of an adversary controlling a bounded number of wires completely. Post their introduction, SMTs have been studied in several works [16, 30, 32, 27, 23]. Non-malleable secure message transmission (NMSMT) was introduced in [18], where the goal is to guarantee non-malleability in the presence of an adversary who can tamper all n wires according to some tampering model (i.e., the tampered message m' is guaranteed to be either same as the original message m , or is completely independent of it). Further, they build NMSMTs using non-malleable secret sharing schemes.

In this work, we show an application of our LRNMSS scheme to build a computational SMT protocol in the pre-processing model, that allows the sender and receiver to communicate in two phases: a message-independent *offline phase* and a message-dependent *online phase*, to non-malleably send multiple messages to the receiver, while saving on the online communication. Formally, we allow S and R to first communicate in an offline phase, where S sends messages x_1, \dots, x_n to R (which are all independent of the messages to be transmitted in the online phase). In the online phase, S can securely send message m by sending a single message c , through one wire, to R . In both the online and offline phase, the adversary can tamper the messages being sent (with the restriction that each wire for the offline phase communication can be arbitrarily tampered independent of each other, and the single wire for the online phase communication, can be arbitrarily tampered independent of the offline communication). The guarantee is that the tampered messages are either the same or are independent of the original messages. To transmit k messages, each of size $\rho = \text{poly}(\lambda)$ (for security parameter λ), our protocol requires an offline communication of 2λ bits per wire (with n wires in total) and an online communication of $\rho + \log k + 2\lambda$ bits per message. In comparison, even if we instantiate the NMSMT protocol of [18] with a rate-1 computational non-malleable secret sharing scheme [7, 17] (as in our construction), to send k messages of length ρ each, the protocol would need to communicate $n\rho$ bits (in total) per message in a

11:16 Locally Reconstructable Non-Malleable Secret Sharing

single online phase. Hence, by introducing an offline phase and by leveraging the locality of our LRNMSS construction, we save a factor of n in the online communication required.

We now define our model for computational non-malleable multi-message transmission formally.

► **Definition 12** (Computational Non-malleable Multi-message Transmission). *Let S and R denote the sender and receiver of the message transmission protocol, respectively and let \mathcal{M} denote the message space from which S wants to transmit messages to R . S and R communicate in two phases: in the offline phase, they communicate through n wires connecting them, and in the online phase, they communicate through a single wire. In the offline phase, S sends messages x_1, \dots, x_n to R (each x_i is sent through the i -th wire). In the online phase, to transmit a message m to R , S sends the message c . Let $\pi(m_1, \dots, m_k, S, R)$ denote an execution of the protocol to transmit k messages m_1, \dots, m_k (involving a single offline phase message and k online phase messages). We say that $\pi(\cdot, S, R)$ is a k -non-malleable multi-message transmission protocol with respect to a tampering family $\mathcal{F}_{\text{split}}$, if it satisfies the following properties:*

1. **Correctness:** *For all messages $m_1, \dots, m_k \in \mathcal{M}$, at the end of an honest execution of the protocol $\pi(m_1, \dots, m_k, S, R)$, the receiver receives the messages m_1, \dots, m_k , with probability 1.*
2. **Computational Privacy:** *For every adversary \mathcal{A} that can see at most $n - 1$ wires in the offline phase and the single wire of the online phase, and for each pair of multi-messages $(m_1, \dots, m_k), (m'_1, \dots, m'_k)$,*

$$\pi_{\mathcal{A}}^{\text{view}}(m_1, \dots, m_k, S, R) \approx_c \pi_{\mathcal{A}}^{\text{view}}(m'_1, \dots, m'_k, S, R),$$

where $\pi_{\mathcal{A}}^{\text{view}}(m_1, \dots, m_k, S, R)$ denotes the distribution corresponding to the view of \mathcal{A} in the protocol execution $\pi(m_1, \dots, m_k, S, R)$, which includes the messages sent through $n - 1$ wires in the offline phase and the messages sent in the online phase.

3. **Non-malleability:**

Tampering Family $\mathcal{F}_{\text{split}}$: *We allow each wire of the offline phase to be tampered independent of each other, and the online phase messages are tampered independent of all offline messages (but may depend on each other). Hence, each $f \in \mathcal{F}_{\text{split}}$ consists of functions f_1, \dots, f_n, g , where each f_i acts on wire i of the offline phase and g acts on the online phase messages.*

For each $f \in \mathcal{F}_{\text{split}}$, there exists a distribution Sim^f over \mathcal{M} , such that, for all sets of messages (m_1, \dots, m_k) ,

$$\text{Tamper}_{m_1, \dots, m_k}^f \approx_c \text{Copy}(m_1, \dots, m_k, \text{Sim}^f),$$

where $\text{Tamper}_{m_1, \dots, m_k}^f$ and $\text{Copy}(m_1, \dots, m_k, \text{Sim}^f)$ are defined as follows:

$$\text{Tamper}_{m_1, \dots, m_k}^f = \left\{ \begin{array}{l} (x_1, \dots, x_n, c_1, \dots, c_k) \leftarrow \pi(m_1, \dots, m_k, S, R) \\ (x'_1, \dots, x'_n, c'_1, \dots, c'_k) = f((x_1, \dots, x_n, c_1, \dots, c_k)) \\ (m'_1, \dots, m'_k) \leftarrow R(x'_1, \dots, x'_n, c'_1, \dots, c'_k) \end{array} \right\}$$

$$\text{Copy}(m_1, \dots, m_k, \text{Sim}^f) = \left\{ \begin{array}{l} (I, m_1^*, \dots, m_k^*) \leftarrow \text{Sim}^f \\ \text{If } I = [k], \text{ set } m' = m_1^*, \dots, m_k^* \\ \text{Else, set } m'|_I = \perp, \text{ and } m'|_{\bar{I}} = (m_i)_{i \in \bar{I}} \\ \text{Output : } m' \end{array} \right\}$$

Construction. Consider our LRNMSS construction from 3.3, specifically for n -threshold setting. S generates $sk \leftarrow \text{Gen}(1^\lambda)$ and sends the shares $(sk_1, \dots, sk_n) \leftarrow \text{NMShare}_n^n(sk)$ through the n -wires in the offline phase (sk_i is sent through wire i , for each $i \in [n]$). For each message m_j ($j \in [k]$) in the online phase, S sends the ciphertext $c_j = \text{Encrypt}_{sk}(m_j, j)$ to R . Now, clearly, R can reconstruct to recover sk from the offline communication and decrypt each ciphertext from the online phase³

► **Theorem 13.** *Let the messages being transmitted be of ρ bits each. If $(\text{NMShare}_n^n, \text{NMRec}_n^n)$ is a n -threshold computational non-malleable secret sharing scheme against independent tampering (each share tampered independently and arbitrarily) with rate 1 and $(\text{Gen}, \text{Encrypt}, \text{Decrypt})$ is a symmetric key authenticated encryption scheme, then the above construction describes a k -non-malleable multi-message transmission protocol with respect to the tampering family $\mathcal{F}_{\text{split}}$ with an offline communication complexity of $2n\lambda$ bits (through all wires combined) and an online communication complexity of $(\rho + \log k + 2\lambda)$ bits, per message sent.*

Proof. The correctness and computational privacy of the protocol directly follow from the correctness and privacy of our LRNMSS scheme.

For non-malleability, note that the tampering model $\mathcal{F}_{\text{split}}$, is in fact weaker than the tampering model \mathcal{F} of our LRNMSS. Note that, tampering of the shares sent in the offline phase indeed belong to \mathcal{F}_{nm} (here, the tampering doesn't depend on the ciphertexts sent in the online phase) and the ciphertexts are all tampered independent of the offline shares. Hence, by the non-malleability of our LRNMSS scheme, the non-malleability of our SMT protocol follows.

Communication Cost. Let each message being transmitted be of ρ bits, k be the number of messages transmitted, λ be the security parameter, n be the number of wires in the offline phase. If we instantiate our protocol with the rate 1 computational NMSS scheme of [17, 7], we get a total offline communication complexity of $2n\lambda$ bits and an online communication complexity of $(\rho + \log k + 2\lambda)$ bits, per message. ◀

References

- 1 Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 393–417, 2016. doi:10.1007/978-3-662-49099-0_15.
- 2 Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, João L. Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret sharing schemes for general access structures. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 510–539. Springer, 2019. doi:10.1007/978-3-030-26951-7_18.
- 3 Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EURO-CRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part*

³ Note that, we can use a two split state non-malleable code (which is implicitly a 2-out-of-2 NMSS) in the offline phase, if non-malleability is the only concern. However, with the use of an n -out-of- n NMSS, we get stronger security guarantees, where the adversary gets more eavesdropping power ($n - 1$ wires).

- I*, volume 11476 of *Lecture Notes in Computer Science*, pages 593–622. Springer, 2019. doi:10.1007/978-3-030-17653-2_20.
- 4 Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 531–545. Springer, 2000.
 - 5 Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 317–330. Springer, 2000.
 - 6 G.R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the 1979 AFIPS National Computer Conference*, pages 313–317, Monval, NJ, USA, 1979. AFIPS Press.
 - 7 Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, and Daniele Venturi. Non-malleable secret sharing against bounded joint-tampering attacks in the plain model. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 127–155. Springer, 2020. doi:10.1007/978-3-030-56877-1_5.
 - 8 Gianluca Brian, Antonio Faonio, and Daniele Venturi. Continuously non-malleable secret sharing for general access structures. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 211–232. Springer, 2019. doi:10.1007/978-3-030-36033-7_8.
 - 9 Nishanth Chandran, Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Constant rate (non-malleable) secret sharing schemes tolerating joint adaptive leakage. Cryptology ePrint Archive, Report 2020/1252, 2020. URL: <https://eprint.iacr.org/2020/1252>.
 - 10 Nishanth Chandran, Bhavana Kanukurthi, and Rafail Ostrovsky. Locally updatable and locally decodable codes. In Yehuda Lindell, editor, *Theory of Cryptography*, pages 489–514, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
 - 11 Nishanth Chandran, Bhavana Kanukurthi, and Srinivasan Raghuraman. Information-theoretic local non-malleable codes and their applications. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 367–392. Springer, 2016. doi:10.1007/978-3-662-49099-0_14.
 - 12 Eshan Chattopadhyay, Jesse Goodman, Vipul Goyal, and Xin Li. Leakage-resilient extractors and secret-sharing against bounded collusion protocols. *Electron. Colloquium Comput. Complex.*, 27:60, 2020. URL: <https://eccc.weizmann.ac.il/report/2020/060>.
 - 13 Sandro Coretti, Antonio Faonio, and Daniele Venturi. Rate-optimizing compilers for continuously non-malleable codes. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *Applied Cryptography and Network Security - 17th International Conference, ACNS 2019, Bogota, Colombia, June 5-7, 2019, Proceedings*, volume 11464 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2019. doi:10.1007/978-3-030-21568-2_1.
 - 14 Dana Dachman-Soled, Mukul Kulkarni, and Aria Shahverdi. Local non-malleable codes in the bounded retrieval model. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography - PKC 2018*, pages 281–311, Cham, 2018. Springer International Publishing.
 - 15 Dana Dachman-Soled, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Locally decodable and updatable non-malleable codes and their applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography*, pages 427–450, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
 - 16 Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *J. ACM*, 40(1):17–47, 1993. doi:10.1145/138027.138036.
 - 17 Antonio Faonio and Daniele Venturi. Non-malleable secret sharing in the computational setting: Adaptive tampering, noisy-leakage resilience, and improved rate. In Alexandra

- Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 448–479. Springer, 2019. doi:10.1007/978-3-030-26951-7_16.
- 18 Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 685–698. ACM, 2018. doi:10.1145/3188745.3188872.
 - 19 Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing for general access structures. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 501–530, Cham, 2018. Springer International Publishing.
 - 20 Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2014.
 - 21 Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, STOC '00*, page 80–86, New York, NY, USA, 2000. Association for Computing Machinery. doi:10.1145/335305.335315.
 - 22 Jonathan Katz and Moti Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In *International Workshop on Fast Software Encryption*, pages 284–299. Springer, 2000.
 - 23 Ravi Kishore, Ashutosh Kumar, Chiranjeevi Vanarasa, and Kannan Srinathan. On the price of proactivizing round-optimal perfectly secret message transmission. *IEEE Trans. Inf. Theory*, 64(2):1404–1422, 2018. doi:10.1109/TIT.2017.2776099.
 - 24 Hugo Krawczyk. Secret sharing made short. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 136–146. Springer, 1993. doi:10.1007/3-540-48329-2_12.
 - 25 Ashutosh Kumar, Raghu Meka, and Amit Sahai. Leakage-resilient secret sharing against colluding parties. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 636–660. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00045.
 - 26 Ashutosh Kumar, Raghu Meka, and David Zuckerman. Bounded collusion protocols, cylinder-intersection extractors and leakage-resilient secret sharing. *Electron. Colloquium Comput. Complex.*, 27:55, 2020. URL: <https://eccc.weizmann.ac.il/report/2020/055>.
 - 27 Kaoru Kurosawa and Kazuhiro Suzuki. Truly efficient 2-round perfectly secure message transmission scheme. *IEEE Trans. Inf. Theory*, 55(11):5223–5232, 2009. doi:10.1109/TIT.2009.2030434.
 - 28 Fuchun Lin, Mahdi Cheraghchi, Venkatesan Guruswami, Reihaneh Safavi-Naini, and Huaxiong Wang. Non-malleable secret sharing against affine tampering. *CoRR*, abs/1902.06195, 2019. arXiv:1902.06195.
 - 29 Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
 - 30 K. Srinathan, Arvind Narayanan, and C. Pandu Rangan. Optimal perfectly secure message transmission. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 545–561. Springer, 2004. doi:10.1007/978-3-540-28628-8_33.
 - 31 Akshayaram Srinivasan and Prashant Nalini Vasudevan. Leakage resilient secret sharing and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 480–509, Cham, 2019. Springer International Publishing.
 - 32 Yongge Wang and Yvo Desmedt. Perfectly secure message transmission revisited. *IEEE Trans. Inf. Theory*, 54(6):2582–2595, 2008. doi:10.1109/TIT.2008.921676.

Linear Threshold Secret-Sharing with Binary Reconstruction

Marshall Ball ✉

University of Washington, Seattle, WA, USA

Alper Çakan ✉

Bogazici University, Istanbul, Turkey

Tal Malkin ✉

Columbia University, New York, NY, USA

Abstract

Motivated in part by applications in lattice-based cryptography, we initiate the study of the size of linear threshold (t -out-of- n) secret-sharing where the linear reconstruction function is restricted to coefficients in $\{0, 1\}$. We also study the complexity of such schemes with the additional requirement that the joint distribution of the shares of any unauthorized set of parties is not only independent of the secret, but also uniformly distributed. We prove upper and lower bounds on the share size of such schemes, where the size is measured by the total number of field elements distributed to the parties. We prove our results by defining and investigating an equivalent variant of Karchmer and Wigderson’s Monotone Span Programs [CCC, 1993].

One ramification of our results is that a natural variant of Shamir’s classic scheme [Comm. of ACM, 1979], where bit-decomposition is applied to each share, is optimal for when the underlying field has characteristic 2. Another ramification is that schemes obtained from monotone formulae are optimal for certain threshold values when the field’s characteristic is any constant.

For schemes with the uniform distribution requirement, we show that they must use $\Omega(n \log n)$ field elements, for all thresholds $2 < t < n$ and regardless of the field. Moreover, this is tight up to constant factors for the special cases where any $t = n - 1$ parties can reconstruct, as well as for any threshold when the field characteristic is 2.

2012 ACM Subject Classification Theory of computation \rightarrow Computational complexity and cryptography

Keywords and phrases Secret sharing, Span programs, Lattice-based cryptography

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.12

Related Version *Full Version:* <https://eccc.weizmann.ac.il/report/2021/050/>

Funding *Marshall Ball:* This material is based upon work partly supported by the National Science Foundation under Grant #2030859 to the Computing Research Association for the CIFellows Project, a grant from the Columbia-IBM center for Blockchain and Data Transparency, and by JP Morgan Chase & Co. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of JPMorgan Chase & Co. or its affiliates, the National Science Foundation, or the Computing Research Association.

Alper Çakan: Part of this work was performed while this author was visiting Columbia University in NY, USA.

Tal Malkin: This research was supported in part by a grant from the Columbia-IBM center for Blockchain and Data Transparency, and by JPMorgan Chase & Co. Any views or opinions expressed herein are solely those of the authors, and may differ from the views and opinions expressed by JPMorgan Chase & Co. or its affiliates.

Acknowledgements We are grateful to Hoeteck Wee who proposed the problem to us, and provided many useful comments and suggestions towards solving it. We also thank anonymous referees for helpful comments.



© Marshall Ball, Alper Çakan, and Tal Malkin;
licensed under Creative Commons License CC-BY 4.0
2nd Conference on Information-Theoretic Cryptography (ITC 2021).
Editor: Stefano Tessaro; Article No. 12; pp. 12:1–12:22



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Threshold secret sharing, introduced by Blakley [7] and Shamir [23], allows a dealer to distribute n shares of a secret value to n distinct parties, such that any t parties can reconstruct the secret from their shares, but any cohort of fewer than t parties can glean nothing about the secret value. While originally introduced in the context of secure data storage, secret sharing has since found a myriad of applications in cryptography and beyond (e.g., see references in [5]).

A particularly useful and well-understood variant of secret-sharing is *linear* secret-sharing schemes: schemes where the secret is represented as a field element, the shares are comprised of collections of field elements, and any t parties can reconstruct the secret by applying a linear function to the field elements in their shares. A canonical example of a linear secret-sharing scheme is Shamir's scheme [23].¹

Shamir's scheme enjoys some desirable properties (in addition to linearity): each share is comprised of just a single field element (an optimal share size), and additionally the residual distribution of shares corresponding to any unauthorized set (any $\leq t - 1$ shares) is uniform. A major drawback of Shamir's scheme is that it is *not* black-box in the underlying field: it requires a field of size at least $n + 1$ (even when sharing a 1-bit secret). In particular, the reconstruction coefficients may be *arbitrary elements* in this large field.

Another classical linear secret sharing scheme that *is* black-box in the underlying field is that of Benaloh and Leichter [6]. Their scheme is recursively defined with respect to any monotone formula computing the access structure (in our case, t -out-of- n threshold function):

- **Initialization.** Assign the secret, s , to the output wire of the formula
- **Recursion.** Given a (sub)-formula with output labeled s' :
 - If the top gate is OR, assign both input wires to that gate s' and recurse on both subformulas,
 - If the top gate is AND, assign left input wire to that gate uniformly random r and the right input wire $s' - r$, and recurse on both subformulas,
 - If the (sub)formula is an input variable, x_i , concatenate s' to the i th share.

This scheme works for any underlying field, and allows for reconstruction with *binary*, or $\{0, 1\}$, *coefficients*. Unfortunately, it does not enjoy the advantages of Shamir's scheme: unauthorized shares are clearly not uniform in general, and moreover the size of shares is comparable to the size of the formula, which can be quite large. For the particular case of $n/2$ -out-of- n thresholds, or majority, the smallest known formula is of size $n^{5.3}$ [24] (and in fact the smallest bound on *explicit* monotone formulas computing majority gives size approximately n^{5000} [3, 22, 19]).

In this work, we ask whether it is possible to get the best of both worlds:

(Q1) Are there linear threshold secret sharing schemes that admit small shares *and* reconstruction via binary coefficients?

(Q2) Moreover, are there such schemes where, additionally, unauthorized shares are jointly uniformly distributed?

With the general question (Q1) in mind, we initiate the study of *linear threshold secret-sharing with binary reconstruction*, where the coefficients of all linear reconstruction functions

¹ Recall that to share a secret s in Shamir's scheme, one chooses a random polynomial p of degree $t - 1$, such that $p(0) = s$. The i th share is then simply $p(\alpha_i)$ where $\{\alpha_1, \dots, \alpha_n\}$ is a set of distinct non-zero field elements.

are simply 0 and 1 (See Definition 5). That is, the secret can be reconstructed by a sum of some subset of the field elements making up (sufficiently many) shares. Specifically, we are interested in the minimum size of the shares for such schemes, quantified in terms of the total number of field elements. We observe how known and folklore results yield upper bounds for this question, then prove lower bounds. We also investigate the minimum share size of such schemes under the additional requirement from (Q2), that unauthorized sets of shares are uniformly distributed. We again prove upper and lower bounds. Almost all our upper bounds are black-box schemes that do not place any restrictions on the field (in particular, when the secret is from a small field, each field element can be represented by a small number of bits). Our lower bounds are tight in some cases (depending on the field characteristic and the threshold). Our technical starting point is the tight connection between monotone span programs and linear secret sharing, shown by Karchmer and Wigderson [20]. Following in their footsteps and much subsequent work on linear secret sharing, we begin by defining restricted models of monotone span programs that are equivalent to the notions of secret sharing we are interested in, and prove our main results within these models.

While we believe this topic to be natural and interesting in its own right, in Section 1.3 we highlight some surprising applications of such schemes from recent results in lattice-based cryptography.

1.1 Our Results

We summarize our results below. We focus on threshold $1 < t < n$, since for $t = 1$ and $t = n$ there is an immediate upper bound of n (one field element per share),² and this is clearly optimal (since for linear schemes shares have to consist of field elements).

On threshold secret sharing with binary reconstruction

A simple folklore construction of secret sharing with binary reconstruction involves a simple bit-decomposition of Shamir's scheme. Suppose we are working over the field $\mathbb{F} = GF(p^c)$ with $p^c \geq n + 1$. Let $\mathbb{L} = GF(p)$, $m = \lceil \log(p) \rceil$ and let $g \in \mathbb{F}$ be such that $\mathbb{F} = \mathbb{L}(g)$. If Shamir's scheme would deal a share $s \in \mathbb{F}_q$ then the corresponding shares in the modified scheme is $s' = (s \cdot g^i \cdot 2^j)_{i=0, j=0}^{i=c-1, j=m-1}$. To see that such a scheme admits binary reconstruction, suppose Shamir's scheme would require multiplication by reconstruction coefficient α . Then, we know that, for $i \in \{0, \dots, c-1\}, j \in \{0, \dots, m-1\}$ there is $\beta_{ij} \in \{0, 1\}$ such that $\alpha = \sum_{i=0}^{c-1} \sum_{j=0}^{m-1} \beta_{ij} \cdot g^i \cdot 2^j$. So, in the modified scheme, the party can obtain the product $\alpha \cdot s$ as $\sum_{i=0}^{c-1} \sum_{j=0}^{m-1} \beta_{ij} \cdot (g^i \cdot 2^j \cdot s)$, which uses only $\{0, 1\}$ as coefficients.

This yields an upper bound of total share size $O(n \log |\mathbb{F}|)$ for threshold secret sharing with binary reconstruction, where \mathbb{F} is required to be of size at least $n + 1$. To obtain a black-box upper bound, we start instead with Benaloh and Leichter's scheme [6]. As mentioned above, for the special case of majority, instantiating with Valiant's probabilistic construction of a monotone formula for majority [24] gives a black-box linear threshold scheme with binary reconstruction, where the total share size is $O(n^{5.3})$. For the general case, Boppana [10] gave a probabilistic construction of monotone formulas computing t -out-of- n Threshold functions that yields total share size $O((\min\{t, n-t\})^{4.3} n \log n)$. To our knowledge, no fully-explicit scheme of comparable size is known.

² For $t = 1$ every share is the secret, and for $t = n$ we can use additive secret sharing.

We observe that it may be possible to improve on these upper-bounds if one starts from small *series-parallel undirected contact networks* (See Definition 9) for the threshold function, which are potentially smaller than corresponding monotone-formula. Explicit undirected contact networks without the series-parallel restriction are known to beat Boppana’s bound. Additionally, as is the case with [6], this connection applies for arbitrary access structures, beyond threshold.

Finally, for the special case that the field has characteristic two, a $O(n \log n)$ upper bound is given by Karchmer and Wigderson [20]. We note that this may also yield non-trivial results for access structures other than threshold, as it makes use of a general connection to monotone-span programs.

Moving to lower bounds, we first show that any linear threshold scheme with binary reconstruction for $1 < t < n$ requires total share size at least $2n - 1$. Next, we prove a lower bound of $n \lceil \log_{\text{char}(\mathbb{F})} n \rceil$ total share size for fields of characteristic $\text{char}(\mathbb{F})$. This indicates that the only hope of achieving linear total share complexity must follow the folklore scheme and utilize large characteristic where $\text{char}(\mathbb{F}) = \Omega(n)$.

Resolving the gap between the general upper and lower bounds remains an open problem. However, for the specific case of secret sharing with binary reconstruction over finite fields with *characteristic 2*, the second lower bound $n \lceil \log_2 n \rceil$ above is tight, matching the [20] upper bound.

Note that Bogdanov, Guo, and Komargodski [8] gave a lower bound of $\Omega\left(\frac{n \log(n)}{\log|\mathbb{F}|}\right)$ for general threshold schemes. Our bound (for linear threshold schemes with binary reconstruction) is higher for any non-prime field, by a factor of $\frac{\log|\mathbb{F}|}{\log \text{char}(\mathbb{F})}$.

On uniformly-distributed unauthorized shares in threshold secret sharing with binary reconstruction

Note that neither the folklore construction specified above nor that of Benaloh and Leichter yield schemes with uniformly distributed unauthorized shares. Do such schemes exist?

Yes, in fact, we prove that *subfield decomposition* applied to Karchmer and Wigderson’s scheme for characteristic 2 [20] indeed yields uniformly distributed unauthorized shares (with total share size $O(n \log n)$). This is tight for the case of characteristic 2, as follows from the above mentioned lower bounds.

More generally, for all fields and access structures, we show connections between share size of secret sharing schemes with uniform unauthorized shares and the complexity of the access structure in a new, *restricted* span program model (see Section 1.2). Furthermore, we introduce various other connections to known models like contact networks and and we show that constructions in these new models yield an upper bound on total share size for threshold secret sharing of $\min\left\{\binom{n}{t}t, \binom{n}{t-2}t(n-t) \log(n-t)\right\}$. For the special cases of $t = 2$ and $t = n - 1$, we show an upper bound of $O(n \log(n))$.

Using extremal set theory (and, alternately, graph theory) we show a general lower bound of $\Omega(n \log n)$ on total share size of threshold schemes with binary reconstruction and uniform unauthorized shares for *any underlying field*. Recall that if unauthorized shares may be arbitrarily distributed, we only know comparable bounds for fields with *constant* characteristic. Also, observe that for the special case of $t = n - 1$, this lower bound is tight, as follows from the upper bound mentioned above. However, there is a gap between these bounds for various values of t , and we show that significantly improving *either* the upper bound *or* the lower bound will require different techniques than the ones used here.

Our results are summarized in the following table.

0,1 Recons	Unauth-uniform	Lower	Upper	Remarks ($1 < t < n$ in all cases)
	✓	n [20]	n [20]	Upper bound requires $ \mathbb{F} \geq n + 1$
✓		$\max\{n \log_{\text{char}(\mathbb{F})}(n), 2n - 1\}$ If $\text{char}(\mathbb{F}) = O(1)$, this is $\Omega(n \log(n))$	$O((\min\{t, n - t\})^{4.3} n \log(n))$ [17] or $O(n \log(\mathbb{F}))$	Second upper bound requires $ \mathbb{F} \geq n + 1$ and bit decomposition
✓		$\Omega(n \log(n))$	$O(n \log(n))$ [20]	$\text{char}(\mathbb{F}) = 2$
✓	✓	$\Omega(n \log(n))$	$O(n \log(n))$	$\text{char}(\mathbb{F}) = 2$
✓	✓	$\Omega(n \log(n))$	$\min\{O\left(\binom{n}{t-2} t(n-t) \log(n-t)\right), \binom{n}{t} t\}$	$2 < t < n$
✓	✓	$\Omega(n \log(n))$	$O(n \log(n))$	$t = n - 1$

1.2 Technical Overview

As discussed above, we introduce two new models of linear secret sharing schemes with perfect privacy. In the first, we restrict the linear reconstruction functions to use coefficients only from a fixed, small set. While both for generality and for ease in some of the proofs, we define the model in a general way to allow for any set here, we will mostly be interested in the case where this set is $\{0, 1\}$. In the second model, we impose the additional requirement that the joint distribution of the shares of any unauthorized set of parties be uniform. While we also prove some general results about these models, our main focus will be computing threshold functions in these models. For both models, we are concerned with the total number of shares (field elements) distributed to the parties. To show upper and lower bounds for this quantity, we use the following equivalence.

1.2.1 Equivalence to New Span Program Models

A *monotone span program* consists of a matrix, M , over some vector space where the rows are labeled by input variables, x_1, \dots, x_n . A monotone span program accepts an input if and only if the rows corresponding to inputs $x_i = 1$ span the all ones vector (using arbitrary coefficients).

The first model we define requires that any *authorized* submatrix be able to span the fixed target only using a fixed set of span coefficients. However, note that the requirement that the *unauthorized* submatrices cannot span the target vector stays the same, that is, it has to hold for any span, without restrictions to the coefficients. In the second model, we further add the *uniformity* requirement that any *unauthorized* submatrix have full row rank. We extend the well-known equivalence between linear secret sharing schemes with perfect privacy and monotone span programs to show that both the coefficients and the uniformity are preserved between these new models.

1.2.2 Upper Bounds

While our focus will be on lower bounds, we explore some upper bounds for the case of coefficient set $\{0, 1\}$, in order to show that some of our lower bounds are tight. On top of the folklore version Shamir's scheme which requires $|\mathbb{F}| \geq n + 1$ and bit decomposition of field elements, we explore some other methods that yield upper bounds for the non-uniform model for all fields. We define two new contact network variations that lead to upper bounds for our monotone span programs, and hence for our secret sharing models. The first model requires that the graph underlying the contact network be a series-parallel graph. We show

that the contact network to span program construction of [20] leads to coefficients $\{0, 1\}$. For the uniform scheme case, the second contact network model we define further requires that the subgraph be acyclic when the input is unauthorized. We show that the same construction from this model yields uniform restricted span programs. We further define a new *non-local* monotone formula model that forbids computing disjunction of small conjunctions. We show that, for the case of threshold function, converting this type of formula to a contact network using the known conversion gives us a network with the acyclicity property defined above. We show upper bounds for this formula model that utilizes an existing explicit formula construction for the special case of $t = 2$. We further show some lower bounds by using extremal combinatorics regarding intersections of fixed size subsets of a set and prove that such a model has to have distinct subtrees/subformulae computing almost all subsets of $[n]$. Our lower bounds show that the upper bounds we give are close to optimal.

We finally show that *decomposing* a program is the optimal method when we want to restrict the coefficients to a subset that is a subfield, even when we working with the stronger uniform model. This implies a tight lower bound for the case where the field characteristic does not grow with the number of parties and the threshold value is constant.

1.2.3 Lower Bounds

Our lower bounds are in two cases. For the general case, we first show a new canonical span program definition for our new span program models, and then show that the size preserving conversion into the canonical model also preserves the coefficient set. Then, we prove that there is a size-preserving conversion that lets us switch the coefficient set with the matrix entry set, at the cost of taking the *dual* of the computed function. Using these results, we show that the subfield decomposition method is optimal, as mentioned above. For the uniform case, we show a field independent $n \log_2(n)$ lower bound for computing any threshold function ($2 < t < n$) in the *uniform* span program model. We do this by showing that if we can find a large family of authorized subsets of parties that have a fixed core subset and have large pairwise intersections, then the total share size must also be large or else we can find cancellations in span equations, which leads to a violation of the uniformity. We start with a primitive version of the argument that gives the lower bound for some cases and then make it more flexible in the next step. Then, we go on to show lower bounds for various threshold values. Finally, we show that a single, condensed and graph-theoretic argument can show the same lower bound for (almost) all threshold values. Finally, using Ahlswede-Khachatrian Complete Intersection Theorem [2] we also show that the proof technique we present cannot give a lower bound that is asymptotically better than the one shown here. More specifically for the case where the coefficient set is $\{0, 1\}$, the lower bound we give matches the upper bound we give above for any threshold value and a field of characteristic 2 or any field and threshold value $t = n - 1$. This shows that the bound we give is optimal for both threshold-independent or field-agnostic lower bounds.

1.3 Secret Sharing with Binary Reconstruction in Lattice-Based Cryptography

We describe two recent applications of linear threshold secret sharing in lattice-based cryptography that require such restrictions on reconstruction coefficients. The first highlights the utility of binary reconstruction coefficients, and the second highlights the additional utility of requiring unauthorized shares to be uniformly distributed. Understanding the share size of such schemes has immediate ramifications to the efficiency of these constructions. We anticipate that schemes admitting such simple reconstruction will find applications beyond those presented here.

Threshold Cryptosystems

Threshold cryptography refers to settings where a cryptographic secret is shared amongst n parties in such a manner that if any t of them come together they can accomplish a task, but security is preserved so long as fewer than t parties are corrupted. Boneh et al. [9] construct Threshold Fully Homomorphic Encryption (TFHE), a primitive that was effectively complete for threshold cryptography in general. In TFHE, an encryption key is made public and n parties are given shares for an associated decryption scheme. Given data encrypted under the public key, each party can *independently* perform a computation on the encrypted data, homomorphically, before using their share of the decryption key to perform a “partial decryption.” Any t partial decryptions can be combined to recover the result of computation in the clear and semantic security holds even if an adversary corrupts $t - 1$ parties. An important property is compactness: the size of the ciphertext is independent of the number of decryptors and does not grow with complexity of homomorphic computation. (Without compactness there are trivial solutions.)

Boneh et al. [9] showed TFHE schemes could be constructed from the Learning with Errors (LWE) assumption, and since publication numerous further applications have been found in situations requiring secure computation with limited interaction. The authors, in fact, gave two constructions of TFHE from LWE, both relying on linear secret sharing. At a high level, both schemes take advantage of the fact that decryption in LWE-based FHE schemes is effectively an inner product between the secret key and the ciphertext. As such, the natural thing to do is secret-share the secret key using a linear secret sharing scheme and perform the inner products locally with each share of the key and simply perform linear reconstruction on the resulting partial decryptions. The problem is that taking an inner product does not immediately decrypt, but instead yields the plaintext plus some small noise. Thus, if the linear reconstruction function has large coefficients, this noise will not remain small and the “reconstructed noise” may occlude the reconstructed plaintext.

Boneh et al. propose to get around this by using schemes that only use binary reconstruction coefficients. The authors conclude by observing that such a scheme exists for any access structure computable by monotone Boolean formulas (including threshold functions) – the Benaloh and Leichter [6] scheme we described above. Unfortunately, as also noted above, this results in a scheme where the share size scales polynomially with the circuit size. Consequently, this also leads to large keys in the TFHE scheme, and comparatively high noise growth. Hence, any improvement to linear threshold secret-sharing with binary reconstruction will immediately result in an improved TFHE scheme.

Boneh et al. additionally proposed a solution that uses Shamir’s scheme as is and instead modifies the noise distribution of a specific FHE scheme. Unfortunately, the resulting ciphertext is not immediately compact and requires further compilation with a non-threshold FHE scheme. As a result, new ideas are needed to yield a scheme with practical parameters.

Fuzzy Identity-Based Encryption and Attribute-Based Encryption

Attribute-Based Encryption (ABE) is a public-key encryption scheme with fine-grained access control. Unlike in a traditional public-key encryption, in ABE an authority can issue secret keys bound to predicates, sk_P , associated with some single public key, pk . Given an encryption of m (encrypted under pk), a party holding sk_P can recover m if and only if $P(m) = 1$. Fuzzy Identity-Based Encryption (Fuzzy IBE) refers to the specific case that the family of allowable predicates are restricted to threshold functions.

Prior to 2013 [11, 16], ABE schemes for NC^1 were known from pairing-based assumptions, but not lattice-based assumptions. As outlined in [1, Appendix B], a tempting paradigm for achieving such an object involves viewing the predicate P as specifying an access structure for a linear secret sharing scheme and sampling LWE trapdoors with the shares baked in, which in turn allow decryption when the receiver is holding authorized shares for the message. We refer the reader to [1, Appendix B] for details, but this goes awry for similar reasons to the above. Correctness is not achieved due to noise growth when the reconstruction coefficients are large. Thus, small reconstruction coefficients are needed. However, in this case the classic scheme of [6] does not yield a secure ABE via the recipe of [1], because unauthorized sets of shares may contain correlations that damage the LWE security. If the secret sharing scheme has the additional property that unauthorized shares are uniformly distributed, the scheme is secure.

Agrawal et al. [1] invoke this recipe with Shamir's scheme to construct Fuzzy IBE. However, to deal with the large reconstruction coefficient in Shamir's scheme, they are required to modify the noise distribution (in a similar manner to the second TFHE construction of [9]). The resulting scheme requires a larger base field $((\ell)!)^2$ times larger, where ℓ is the length of an "identity". Consequently, linear secret sharing with binary coefficients and uniformly distributed unauthorized shares immediately yields practical improvements to Fuzzy IBE from LWE.

2 Preliminaries

► **Notation.** Unless otherwise specified, any column or row representation of a vector is according to the standard basis of \mathbb{F}^d for the appropriate value of d . Similarly, any matrix $M_{k \times \ell}$ is a representation over the standard bases of \mathbb{F}^k and \mathbb{F}^ℓ . For a matrix $M_{k \times \ell}$ over a field \mathbb{F} , and a subset $A \subset \mathbb{F}$, $\text{Rowspan}_A(M)$ denotes the set $\{vM \mid v \in A^{1 \times k}\}$. When \mathbb{F} is clear from the context and $A = \mathbb{F}$, we will drop the subscript. By $\mathbf{1}$ ($\mathbf{0}$), we denote the unique row vector whose entries are all ones (zeroes) in the implicit basis of appropriate dimension, and its dimension will be clear from the context. We will consider elements of $\{0, 1\}^n$ and subsets of $[n]$ interchangeably in the natural way. Th_n^t denotes the t -out-of- n threshold function, i.e., the function $Th_n^t : \{0, 1\}^n \rightarrow \{0, 1\}$ where $Th_n^t(x) = 1$ if and only if $|x| \geq t$. For any set A , $x \in A^n$ and $i \in [n]$, x_i denotes the i^{th} component of x . We show the degree of a field extension \mathbb{F} over \mathbb{L} as $|\mathbb{F} : \mathbb{L}|$.

The following generalizes the span program model of [20].

► **Definition 1.** Fix a field \mathbb{F} and two sets $A, B \subseteq \mathbb{F}$. A restricted span program over (\mathbb{F}, A, B) is a labeled matrix $\hat{M}(M, \rho)$ where $M_{k \times \ell}$ is a matrix over \mathbb{F} with entries only in A and $\rho : \text{rows}(M) \rightarrow \{x_i^\epsilon \mid i \in [n], \epsilon \in \{0, 1\}\}$. For any $v \in \{0, 1\}^n$, M_v denotes the submatrix consisting of rows $r \in \text{rows}(M)$ such that $\rho(r) = x_i^\epsilon$ with $\epsilon = v_i$ for some $i \in [n]$.

We say that \hat{M} computes $f : \{0, 1\}^n \rightarrow \{0, 1\}$ if for all $x \in \{0, 1\}^n$,

$$\begin{cases} \mathbf{1} \in \text{Rowspan}_B(M_x), & \text{if } f(x) = 1 \\ \mathbf{1} \notin \text{Rowspan}_B(M_x), & \text{if } f(x) = 0 \end{cases}$$

We define $\text{size}(\hat{M})$ to be the number of rows in M , $\text{rows}(\hat{M}, i)$ to be the rows of $i \in [n]$, that is, $\{r \in \text{rows}(M) \mid \rho(r) = x_i^\epsilon \text{ for some } \epsilon \in \{0, 1\}\}$, and $\text{rowcount}(\hat{M}, i)$ to be $|\text{rows}(\hat{M}, i)|$. More generally, for any $P \subset [n]$, we take $\text{rows}(\hat{M}, P)$ and $\text{rowcount}(\hat{M}, P)$ to denote $\bigcup_{i \in P} \text{rows}(\hat{M}, i)$ and $\sum_{i \in P} \text{rowcount}(\hat{M}, i)$, respectively.

For any $f : \{0, 1\}^n \rightarrow \{0, 1\}$, we denote the set of all restricted span programs over (\mathbb{F}, A, B) computing f as $\text{SP}_{A,B,\mathbb{F}}(f)$ and the smallest program size in this set as $\text{size}(\text{SP}_{A,B,\mathbb{F}}(f))$.

We will usually refer to the span program \hat{M} and its underlying matrix M interchangeably, denoting both as M .

► **Definition 2.** Let $\hat{M}(M, \rho)$ be a restricted span program computing $f : \{0, 1\}^n \rightarrow \{0, 1\}$ over (\mathbb{F}, A, B) . If M_x has full row rank as an \mathbb{F} -matrix for all $x \in \{0, 1\}^n$ such that $f(x) = 0$, then we call \hat{M} a uniform program.

Similar to the above, $\text{SP}_{A,B,\mathbb{F}} - \text{Uniform}(f)$ and $\text{size}(\text{SP}_{A,B,\mathbb{F}} - \text{Uniform}(f))$ denote the set of uniform restricted span programs computing f and the size of the smallest program in this set, respectively.

For both models defined above and similar models that will be defined below, the qualifier *monotone* will mean that all labels are of the form x_i^1 . The corresponding sets will be denoted as $\text{MSP}_{A,B,\mathbb{F}}(f)$ and $\text{MSP}_{A,B,\mathbb{F}} - \text{Uniform}(f)$.

► **Remark 3.** In the context of span programs, we will refer to $\mathbf{1}$ as the target vector. For usual span programs, it is well known that any two definitions with different non-zero target vectors are equivalent, since a program can be converted to be a program for another target vector through a simple change of basis. However, we have to be more careful with the restricted span programs.

It's easy to see that the set of coefficients, B , is preserved when we change the basis. The entry set, however, requires a more detailed investigation, and we avoid it since we won't need it here. The uniformity is similar to the set of coefficients and is preserved.

2.1 Restricted, Information-Theoretically Secure Linear Secret Sharing Schemes

In this section, we define the new secret sharing models that motivate the definitions of the restricted span program models of the previous section. We will also extend the known equivalence between the linear secret sharing schemes with perfect privacy and monotone span programs to between their new counterparts.

► **Definition 4.** [5] Fix number of parties $n \in \mathbb{Z}^+$, and sets R, S, S_1, \dots, S_n . A secret sharing with perfect privacy scheme realizing the access function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ over the domain of secrets S and domains of shares S_1, \dots, S_n with random input domain R is a family of functions $(\text{share}, (\text{reconstruct}_P)_{P \subseteq [n]})$ where $\text{share} : S \times R \rightarrow S_1 \times \dots \times S_n$ and $\text{reconstruct}_P : (\times_{i \in P} S_i) \rightarrow S$ satisfy the following for all $P \subseteq [n]$.

Correctness If $f(P) = 1$, then $\Pr_{r \sim R}[\text{reconstruct}_P(\text{share}(s, r)_P) = s] = 1$

Perfect Privacy If $f(P) = 0$, then, for all $a, b \in S, v \in \times_{i \in P} S_i$,

$$\Pr_{r \sim R}[\text{share}(a, r)_P = v] = \Pr_{r \sim R}[\text{share}(b, r)_P = v]$$

Here, share_P refers to the joint share vector of subset P , that is, components indexed $i \in [n]$ of share_P with $i \in P$.

In this work, unless otherwise stated, secrets and shares will be from a single field, that is, $S = \mathbb{F}$ and $S_i = \mathbb{F}^{c_i}$, where $c_i \in \mathbb{N}$, for all $i \in [n]$. The size of a scheme is the total number of field elements distributed, that is, $\sum_{i=1}^n c_i$.

We call a scheme linear when the domain of secret and domain of shares are all a field \mathbb{F} and all the reconstruction functions are linear on the shares.

12:10 Linear Threshold Secret-Sharing with Binary Reconstruction

► **Definition 5.** Fix a field \mathbb{F} and a set $B \subseteq \mathbb{F}$. A restricted secret sharing scheme S (share, reconstruct) over (B, \mathbb{F}) is a linear secret sharing scheme over \mathbb{F} with perfect privacy such that the reconstruction coefficients are only from B . If for a restricted secret sharing scheme, the joint distribution of the shares of any unauthorized set is uniform, then the scheme is called a uniform scheme.

To provide intuition, throughout the text, we sometimes use the secret sharing nomenclature for span programs, such as referring to the rows labeled x_i^ϵ as the rows of party i or referring to x with $f(x) = 0$ as an *unauthorized* input.

We extend the equivalence proof of [4] to show that the set of coefficients and uniformity are preserved.

► **Lemma 6.** For any field \mathbb{F} , sets $A, B \subseteq \mathbb{F}$, function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and $M \in \text{MSP}_{A, B, \mathbb{F}}(f)$, there is a restricted secret sharing scheme S realizing f over (B, \mathbb{F}) with $\text{size}(S) = \text{size}(M)$. Furthermore, if M is uniform, S is also uniform.

Proof. Let $k \times \ell$ be the size of M . For each $s \in \mathbb{F}$, define $N_s = \{v \in \mathbb{F}^{\ell \times 1} \mid \mathbf{1}v = s\}$ and fix an arbitrary indexing $\gamma_s : [|N_s|] \rightarrow N_s$. Let $R = [|N_1|]$, also noting that $|N_s| = |N_1|$ for all $s \in \mathbb{F}$. We construct the scheme (share, reconstruct, R) over (B, \mathbb{F}) as follows.

Define $\text{share}(s, r) = M\gamma_s(r)$ where in the resulting vector, an entry will be a share piece for party j if the corresponding row in M is labeled x_j^1 .

Consider any P such that $f(P) = 1$. Then, there is u_P with entries in B such that $u_P M_P = \mathbf{1}$. Hence, $u_P(M_P\gamma_s(r)) = u_P M_P\gamma_s(r) = \mathbf{1}\gamma_s(r) = s$. Therefore, we define $\text{reconstruct}_P(q) = u_P q$ and we have correctness.

Now consider any P such that $f(P) = 0$. Pick $u \in \mathbb{F}^{\ell \times 1}$ such that $M_P u = \mathbf{0}$ and $\mathbf{1}u = 1$. Such u exists since $\mathbf{1} \notin \text{Rowspan}_{\mathbb{F}}(M_P)$. For any $s_1, s_2 \in \mathbb{F}$ and any $c \in \mathbb{F}$, we will show that $\phi(r) = \gamma_{s_2}^{-1}((s_2 - s_1)u + \gamma_{s_1}(r))$ is a bijection from $\{r \in R \mid M_P\gamma_{s_1}(r) = c\}$ to $\{r \in R \mid M_P\gamma_{s_2}(r) = c\}$. First of all, it's well defined: $\beta(x) = ((s_2 - s_1)u + x)$ is a bijection from N_{s_1} to N_{s_2} since $\mathbf{1}\beta(x) = s_2 - s_1 + \mathbf{1}x = s_2$. A similar argument shows that $\gamma_{s_1}^{-1}((s_1 - s_2)u + \gamma_{s_2}(r))$ is also well-defined and acts as the inverse of $\phi(r)$, hence proving our claim.

Lastly, we prove that uniformity is preserved. Assume that M is uniform, and we claim the scheme constructed above is uniform. Again consider any P such that $f(P) = 0$. Since we want to show that all share vectors of the appropriate dimension have non-zero and equal probability, observe that it's enough to show that for each $s \in \mathbb{F}$ and $c_1, c_2 \in \mathbb{F}^{\text{rowcount}(M, P) \times 1}$, there is a bijection between $\{r \in R \mid M_P\gamma_{s_1}(r) = c_1\}$ and $\{r \in R \mid M_P\gamma_{s_2}(r) = c_2\}$ and that both are non-empty sets. But there is indeed a bijection since $\{v \in \mathbb{F}^{\ell \times 1} \mid M_P v = c_1, \mathbf{1}v = s\}$ and $\{v \in \mathbb{F}^{\ell \times 1} \mid M_P v = c_2, \mathbf{1}v = s\}$ are both translations of $\{v \in \mathbb{F}^{\ell \times 1} \mid M_P v = 0, \mathbf{1}v = 0\}$ and since γ_s is also a bijection. Finally, observe that when we concatenate the row vector $\mathbf{1}$ to M_P it still has full row rank since M_P has full row rank and $\mathbf{1} \notin \text{Rowspan}(M_P)$. Hence, $\{v \in \mathbb{F}^{\ell \times 1} \mid M_P v = c_1, \mathbf{1}v = s\}$ is always non-empty. ◀

► **Lemma 7.** For any field \mathbb{F} , set $B \subseteq \mathbb{F}$, function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and a restricted secret sharing scheme S realizing S over (B, \mathbb{F}) , there is $M \in \text{MSP}_{\mathbb{F}, B, \mathbb{F}}(f)$ with $\text{size}(M) = \text{size}(S)$. Furthermore, if S is uniform, M is also uniform.

Proof. Let R be the domain of the random input of share and fix any ordering of R and S . We define the matrix $M_{\text{size}(S) \times (|R| |\mathbb{F}|)}$ as follows. Index the columns of M by $(r, s) \in R \times \mathbb{F}$, ordering first by the index of s and then by the index of r . Set the column labeled (r, s) to

share(s, r). Index the rows by the party indices. That is, if the i^{th} entry of the joint share vector belongs to party j , label the i^{th} row of M with x_j^1 . Finally, let the target vector w be the concatenation of $[s_i \ s_i \ \dots \ s_i]_{1 \times |R|}$ for $i = 1$ to $|\mathbb{F}|$ in that order.

Consider any fixed $r \in R$ and $s \in \mathbb{F}$. Take $P \subseteq [n]$ such that $f(P) = 1$. Let v be the joint share vector of P and let k be its dimension. Then, by the correctness of the secret sharing scheme, there is $c = [c_1 \ c_2 \ \dots \ c_k]_{1 \times |R|} \in B^k$ such that, $\text{reconstruct}_P(v) = \sum_{i=1}^k c_i v_i = s$. Then, we see that $cM_P = w$. The case when $f(P) = 0$ is proven similarly by contradiction.

Lastly, we show that uniformity is preserved. Take any P such that $f(P) = 0$. Consider M_P and let ℓ be its number of rows. By the uniformity of the secret sharing scheme, for any $i \in [\ell]$ and for all $s \in \mathbb{F}$, there is $r \in R$ such that the column labeled (r, s) is e_i , the vector with 1 in the i^{th} coordinate and 0 in all the others. Hence, $\text{rank}(M_P) = \ell$. ◀

► **Remark 8.** Observe that in the proof of Lemma 7, instead of requiring that the joint distribution of the shares of the unauthorized sets be uniform, we could show the same results with the weaker assumption that the support of those distributions are equal to their respective spaces or even just that those supports span their respective spaces. In fact, based on this observation, we can see that any such *weaker* scheme can be converted to a uniform scheme by first applying Lemma 7 and then Lemma 6 while preserving the total share size.

3 Upper Bounds

In this paper, our focus will be lower bounds. However, we do present upper bounds for reference.

3.1 Upper Bounds for $\text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}}(\text{Th}_n^t)$

Note that, while the upper bounds here work for any field; for suitable fields, the folklore construction discussed in the introduction, Shamir's scheme with bit decomposition, yields better upper bounds.

► **Definition 9** ([17]). *An undirected contact network (UCN) (G, s, t, μ) is an undirected graph $G = (V, E)$ with edges labeled by variables or their negations, that is $\mu : E \rightarrow \{x_i^\epsilon \mid i \in [n], \epsilon \in \{0, 1\}\}$, and two designated vertices, source $s \in V$ and terminal $t \in V$. For any $u \in \{0, 1\}^n$, E_u is defined to be $\{e \in E : \mu(e) = x_i^\epsilon \text{ with } u_i = \epsilon\}$ and G_u is (V, E_u) .*

A UCN is said to compute a function $f(x_1, \dots, x_n) : \{0, 1\}^n \rightarrow \{0, 1\}$ if for all $x \in \{0, 1\}^n$, $f(x) = 1$ if and only if there is a path from s to t in G_x . The size of a UCN is defined to be the number of edges its graph has, $|E|$.

An undirected monotone contact network (UMCN) is a UCN where all edges are labeled by (non-negated) variables, namely $\epsilon = 1$. A UCN is series-parallel if the underlying network graph is series-parallel.

Note that the same construction is named *symmetric branching programs* in [20] and we will use the terms interchangeably. Also, as in the case of span programs, we will refer to the contact network and its underlying graph interchangeably.

Now we present a lemma from [20] which allows us to obtain upper bounds for span programs using known contact network and formula sizes. Additionally, we observe that when the underlying graph of a contact network is series-parallel, the proof actually gives a program in $\text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}}(f)$. The proof, with this observation, can be found in the full version of this paper.

► **Lemma 10.** [20] Fix a field \mathbb{F} . A UCN $G = (V, E)$ computing a function f can be converted into a span program of the same size computing f . Also, if the network is monotone, so is the resulting program. Finally, if the network is series-parallel, the resulting program is in $\text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}}(f)$.

Using the monotone formula upper bounds stated in [10], we get the following upper bounds.

► **Theorem 11.** [10] $\text{size}(\text{UMCN}(\text{Th}_n^t)) \leq O((\min\{t, n-t\})^{4.3} n \log(n))$

► **Corollary 12.** $\text{size}(\text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}}(\text{Th}_n^t)) \leq O((\min\{t, n-t\})^{4.3} n \log(n))$

3.2 Upper Bounds for $\text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}} - \text{Uniform}(\text{Th}_n^t)$

While the monotone UCN model does not give $\text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}} - \text{Uniform}$ directly, requiring that the G_x be acyclic when $f(x) = 0$ is enough to get this property. Note that, in case of Th_n^t , this is equivalent to each cycle of the contact network having at least t distinct variables. We use the following restricted models to get $\text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}} - \text{Uniform}$ upper bounds.

► **Definition 13.** Let $\hat{G}(G, s, t, \mu)$ be a UCN computing $f : \{0, 1\}^n \rightarrow \{0, 1\}$. If G_x is acyclic for all $x \in \{0, 1\}^n$ such that $f(x) = 0$, then we call \hat{G} a uniform network.

► **Lemma 14.** In Lemma 10, if the network is uniform, then so is the resulting program.

Proof. Let G be a uniform UCN computing f and let M be the corresponding span program obtained using Lemma 10. For a contradiction, suppose there is x with $f(x) = 0$ such that M_x does not have full row rank. Then, there is a linear dependency $\sum_i c_i u_i = 0$ where $\{u_i\}_i$ is rows(M_x) and c_i are not all 0. Consider only those i such that $c_i \neq 0$. Consider any u_i and its corresponding edge in the network, (v, w) . u_i is a difference of two basis vectors by construction. Therefore, for those basis vectors to be eliminated, there should be distinct j, k such that the edge of u_j touches v , the edge of u_k touches w and $c_j, c_k \neq 0$. Continuing like this, we get a connected subset of vertices of G_x such that each vertex of it has degree at least 2 in G_x , which implies G_x is cyclic. ◀

► **Definition 15.** Restricted monotone formulae for threshold functions.

A restricted monotone formula for a threshold function is a monotone formula³ F computing Th_n^t for some t, n such that OR gates cannot have as their input a literal; their inputs can only be outputs of other gates, and if an input of an OR gate is the output of a pure AND subtree⁴, that subtree must be effectively computing $\bigwedge_{i \in S} x_i$ for some $S \subseteq [n]$ with $|S| \geq t-1$. We will interchangeably consider formulae as functions and as trees. We denote the set of restricted monotone formulae computing Th_n^t as $\text{RestrictedFormula}(t, n)$.

► **Lemma 16.** For any restricted monotone formula $F \in \text{RestrictedFormula}(t, n)$ and for any field \mathbb{F} , there is an $\hat{M} \in \text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}} - \text{Uniform}(\text{Th}_n^t)$ with $\text{size}(\hat{M}) \leq \text{size}(F)$.

Proof. The proof is presented in the full version of the paper. ◀

Note that we are not claiming that this is the optimal way of constructing an $\text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}} - \text{Uniform}$ from a contact network or a formula, but these uniform network and restricted formula definitions are natural and readily give such programs.

³ We require fan-in = 2 and allow only AND, OR gates. Formula size is the number of gates.

⁴ An AND gate which doesn't have any OR gates below

Now, we show some upper bounds for $\text{MSP}_{\mathbb{F},\{0,1\},\mathbb{F}} - \text{Uniform}(\text{Th}_n^t)$ that we obtain from contact networks and restricted formulae.

► **Theorem 17.** $\text{size}(\text{MSP}_{\mathbb{F},\{0,1\},\mathbb{F}} - \text{Uniform}(\text{Th}_n^t)) \leq \binom{n}{t}t$

Sketch. Use the sum of minterms form of f . ◀

Some optimal monotone formula upper bounds such as the one in [24] are shown probabilistically. However, [13, Section 1] gives a code-based explicit construction, which is still optimal for $t = \Theta(1)$ and $t = n - \Theta(1)$. In fact, below we will invoke his construction only for $t = 2$.

The following is an elementary construction using [13], which nevertheless improves upon the naive upper bound by a factor of $\frac{n}{\log(n)}$ in some cases.

► **Theorem 18.** $\text{size}(\text{RestrictedFormula}(t, n)) \leq O\left(\binom{n}{t-2}t(n-t)\log(n-t)\right)$.

Proof. Observe that the threshold function Th_n^t for $t > 2$ can be written in terms of Th_{n-t+2}^2 as follows: $\text{Th}_n^t(x_1, x_2, \dots, x_n) = \bigvee_{S=\{i_1, i_2, \dots, i_{t-2}\} \subseteq [n]} x_{i_1} x_{i_2} \dots x_{i_{t-2}} \text{Th}_{n-t+2}^2([n]-S)$. Based on this, do the following for each $S = \{i_1, i_2, \dots, i_{t-2}\} \subseteq [n]$ and OR the resulting formulae. Apply the construction of [13] to get a formula for Th_{n-t+2}^2 , and then replace each literal x_j (where $j \in [n] - S$) with $x_j x_{i_1} x_{i_2} \dots x_{i_{t-2}}$. Note that this replacement only increases the size of each formula for Th_{n-t+2}^2 by a factor of $t - 1$.

Since the formula for Th_{n-t+2}^2 is of size $O((n-t)\log(n-t))$, the formula we get for Th_n^t is of size $O\left(\binom{n}{t-2}t(n-t)\log(n-t)\right)$. ◀

We show below that the upper bounds obtained above are close to optimal for this model.

► **Theorem 19.** $\text{size}(\text{RestrictedFormula}(t, n)) \geq \Omega\left(\frac{\binom{n}{t-1}}{n-t}\right)$

Proof. Consider any $S \subseteq [n]$ with $|S| = t$. We will show that there must be $S' \subseteq S$ with $|S'| = t - 1$ such that there is a pure AND subtree of the formula computing $\bigwedge_{i \in S'} x_i$.

Assume this is true for now. Since we cannot re-use a computation result in a formula, we conclude that the minimum size of the formula is $t|\mathcal{F}^*|$ where \mathcal{F}^* is the smallest collection of size $t - 1$ subsets of $[n]$ that includes a size $t - 1$ subset of each size t subset of $[n]$. Observe that, for each $K \subseteq [n]$, $|K| = t - 1$, \mathcal{F}^* has to include a subset $K' \subseteq [n]$, $|K'| = t - 1$ with $|K \cap K'| \geq t - 2$. Suppose otherwise. Then, let i be such that $i \notin K$ and consider $K \cup \{i\}$. This size t subset won't have any size $t - 1$ subsets that's in \mathcal{F}^* (i.e., $K \cup \{i\}$ won't be covered). Based on this, we conclude that $|\mathcal{F}^*| \geq \gamma(J(n, t - 1))$ where $\gamma(J(n, t - 1))$ is the domination number of the Johnson graph $J(n, t - 1)$. It's an elementary result that $\gamma(G) \geq \frac{|V(G)|}{\Delta(G)+1}$ for any graph G , where $\Delta(G)$ is the maximum degree of G . Therefore, we get $|\mathcal{F}^*| \geq \Omega\left(\frac{\binom{n}{t-1}}{\binom{n-t}{t}}\right)$ since $J(n, t - 1)$ is $(t - 1)(n - t + 1)$ regular.

Now, we need to prove our initial claim. Consider any $S \subseteq [n]$ with $|S| = t$ and its evaluation by this formula. First of all, it's easy to see that the formula must contain at least 1 OR gate. Start at the root vertex of the formula. Since an AND gate means both subtrees evaluate to 1, we can descend down to an OR gate that must evaluate to 1. After this point, if we get to an OR gate, we recursively call the descend procedure for the child that evaluates to 1. We stop when we have reached an AND gate.

By the definition of RestrictedFormula , it's easy to see that this descending procedure will terminate at an AND gate that has output 1 in this case and that's computing $\bigwedge_{i \in S'} x_i$ for some $S' \subseteq [n]$ with $|S'| \geq t - 1$ in general. If $|S'| > t - 1$, we must have $S = S'$, which means (by descending one more level) that a subset of S is computed by a pure AND subtree. If $|S'| = t - 1$, this implies $S' \subset S$, which again proves our claim. ◀

12:14 Linear Threshold Secret-Sharing with Binary Reconstruction

As discussed above, this model is not the only way we can obtain $\text{MSP}_{\mathbb{F},\{0,1\},\mathbb{F}} - \text{Uniform}$ upper bounds through contact networks or formulae. Below we use a more direct analysis to obtain a better upper bound for a specific case.

► **Definition 20.** For a function $f : \{0,1\}^n \rightarrow \{0,1\}$, define its dual $f' : \{0,1\}^n \rightarrow \{0,1\}$ as $f'(x_1, x_2, \dots, x_n) = f(\overline{x_1}, \overline{x_2}, \dots, \overline{x_n})$.

Observe that dual of a monotone formula is again a monotone formula of the same size. It's easy to see that dual of Th_n^t is Th_n^{n-t+1} .

► **Theorem 21.** $\text{size}(\text{MSP}_{\mathbb{F},\{0,1\},\mathbb{F}} - \text{Uniform}(\text{Th}_n^t)) \leq O(n \log(n))$ for $t = 2$ and $t = n - 1$.

Proof. For $t = 2$, the requirement that each cycle contain at least t distinct variables is trivially satisfied. This shows that any formula upper bound for $t = 2$ transfers to our case. So we just use [13] formula directly to get $O(n \log(n))$. We cannot hope for a better upper bound through formulae since it is known that there is a $\Omega(n \log(n))$ lower bound for monotone formulae for $t = 2$ [17].

For $t = n - 1$, take the dual of [13] formula constructed for $t = 2$. Any parallel part in this construction corresponds to A_0^j and A_1^j of [13, Section 1], and their union contains all the variables by the definition given there. Hence, any cycle contains all n variables. ◀

3.3 Subfield Decomposition

The method of converting a program over \mathbb{F} to a program over the subfield \mathbb{L} is useful for us in the case when $\text{char}(\mathbb{F}) = 2$, since then $\{0,1\}$ is a subfield.

[20, Theorem 12] uses subfield decomposition method to show upper bounds for $\text{MSP}_{\mathbb{F}_2, \mathbb{F}_2, \mathbb{F}_2}(\text{Th}_n^t)$ through Shamir's secret sharing scheme over larger fields of characteristic 2. [12, Lemma 3] uses the same method for *integer span programs*. Here, we show that this method also preserves uniformity. We modify the decomposition slightly to be able to show the uniformity, so we first show in detail the correctness of the method in our context.

► **Theorem 22.** Let \mathbb{L} be a subfield of \mathbb{F} . Then, $\text{size}(\text{MSP}_{\mathbb{L}, \mathbb{L}, \mathbb{L}}(f)) \leq \text{size}(\text{MSP}_{\mathbb{F}, \mathbb{F}, \mathbb{F}}(f)) \cdot |\mathbb{F} : \mathbb{L}|$ and $\text{size}(\text{MSP}_{\mathbb{L}, \mathbb{L}, \mathbb{L}} - \text{Uniform}(f)) \leq \text{size}(\text{MSP}_{\mathbb{F}, \mathbb{F}, \mathbb{F}} - \text{Uniform}(f)) \cdot |\mathbb{F} : \mathbb{L}|$

Proof. Let $\{a_0, a_1, \dots, a_{\ell-1}\}$ be an \mathbb{L} -basis of \mathbb{F} where $\ell = |\mathbb{F} : \mathbb{L}|$ and $a_0 = 1$. For any $x \in \mathbb{F}$, let N_x denote the $\ell \times \ell$ matrix whose k^{th} row is the \mathbb{L} -coordinates of $a_k x$ as a row vector. We omit the proof here, but it's easy to show that $N_{xy} = N_x N_y$ and $N_{x+y} = N_x + N_y$ for all $x, y \in \mathbb{F}$ using the fact that multiplication is linear. Finally, for any matrix A with entries in \mathbb{F} , let \hat{A} denote the matrix created by replacing each entry x of A with N_x .

Let $M_{s \times k} \in \text{MSP}_{\mathbb{F}, \mathbb{F}, \mathbb{F}} - \text{Uniform}(\text{Th}_n^t)$ with target vector $w_{1 \times k} = [1, 0, \dots, 0]$. We claim $\hat{M} \in \text{MSP}_{\mathbb{L}, \mathbb{L}, \mathbb{L}}(\text{Th}_n^t)$ with target vector $w_{1 \times k\ell} = [1, 0, \dots, 0]$. Note that it is fine to use these target vectors since we can change target vectors at the end to return to the original model.

First, the correctness. Let $A \subseteq [n]$ be such that $f(A) = 1$. Then, there is v such that $vM_A = w_{1 \times k}$. Hence, $\hat{v}\hat{M}_A = (w_{1 \times k})$. Considering the \mathbb{L} -coordinates of 0 and 1, it is easy to see that only keeping the first row gives us $(\hat{v})_1 \hat{M}_A = w_{1 \times k\ell}$.

Then, the security. Let A be such that $f(A) = 0$. Then, $w_{1 \times k} \notin \text{Rowspan}(M_A)$. Then, $\tilde{M}_A v = ([0, \dots, 0, 1]^T)_{(s_A+1) \times 1}$ has a solution, where \tilde{L} is the matrix obtained by appending $[1, 0, \dots, 0]^T$ at the bottom of L for any matrix L . Then, consider the multiplication $(\hat{M}_A)\hat{v}$, and drop the last $(\ell - 1)$ rows of (\hat{M}_A) and all except the first column of \hat{v} . Denote these as P and u respectively. Observe that we have $Pu = ([0, \dots, 0, 1]^T)_{(s_A\ell+1) \times 1}$ and $P = (\hat{M}_A)$. Therefore, $w_{k\ell \times 1} \notin \text{Rowspan}(\hat{M}_A)$.

Finally, the uniformity. Let A be such that $f(A) = 0$. Assume for a contradiction that \hat{M}_A does not have full row rank. Then, there is a row vector $v \neq 0$ with entries in \mathbb{L} such that $v\hat{M}_A = [0, \dots, 0]_{1 \times k\ell}$. Now, observe that there is (unique) $(v_c)_{1 \times s_A}$ such that the first row of \hat{v}_c is equal to v . Then, we can see that the first row of $\hat{v}_c\hat{M}_A$ is all zeroes. We claim this is a contradiction. Consider $v_c M_A$. Since v is not 0, v_c is not 0 either. By definition, we have that M_A has full row rank. Hence, $v_c M_A$ is not all zero. Therefore, the first row of $\hat{v}_c\hat{M}_A$ cannot be all zeroes. \blacktriangleleft

► **Corollary 23.** $\text{size}(\text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}} - \text{Uniform}(\text{Th}_n^t)) \leq O(n \log(n))$ for any \mathbb{F} with $\text{char}(\mathbb{F}) = 2$.

4 Lower Bounds

4.1 $\text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}}(\text{Th}_n^t)$

► **Theorem 24.** For any field \mathbb{F} of finite characteristic $\text{char}(\mathbb{F})$ and any t with $2 \leq t \leq n-1$, we have $\text{size}(\text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}}(\text{Th}_n^t)) \geq n \log_{\text{char}(\mathbb{F})}(n)$

Since we have the upper bound $O(n \log_2(n))$ for any field \mathbb{F} with $\text{char}(\mathbb{F}) = 2$, that is obtained through bit decomposition, we conclude that the lower bound is tight for such \mathbb{F} . Furthermore, using monotone contact networks, we get the same upper bound for any field \mathbb{F} (of any characteristic including 0) and for threshold $t = \Theta(1)$ or $n - \Theta(1)$, we again conclude that the lower bound is tight for the case of $\Theta(1)$ characteristic and such threshold values.

We begin by outlining the proof of the theorem. First, we will show that conversion into a canonical form that preserves the program size and the coefficient set B . Then, we will prove that there is again a size preserving conversion between $\text{MSP}_{A,B,\mathbb{F}}(f)$ and $\text{MSP}_{B,A,\mathbb{F}}(f')$ where f' is the dual of f , inspired by [14]. Lastly, we show $\text{size}(\text{MSP}_{\{0,1\}, \mathbb{F}, \mathbb{F}}(\text{Th}_n^t)) \geq n \log_{\text{char}(\mathbb{F})}(n)$ using an adaptation of a theorem of [20].

4.1.1 Canonical Forms

We start with canonical forms. The following definition is from [20].

► **Definition 25.** Let M be a span program computing f . We say that M is canonical if the columns of M are in one-to-one correspondence with $U = f^{-1}(0) \subset \{0,1\}^n$ and for every $u \in U$, the column corresponding to u in M_u is $\mathbf{0}$. We denote the class of canonical monotone span programs as $\text{MSPCanon}_{A,B,\mathbb{F}}$.

Observe that this condition automatically implies the security condition: since the column u of M_u will be $\mathbf{0}$, M_u cannot span $\mathbf{1}$. Therefore, we can think of this condition as replacing the security condition.

With a small modification, construction of [20, Theorem 6] preserves the set of coefficients. We observe this below and also the fact that in some cases the set of entries is also preserved. Proof given in the full version.

► **Lemma 26.** For any $M \in \text{MSP}_{A,B,\mathbb{F}}(f)$, there is $N \in \text{MSP}_{\mathbb{F},B,\mathbb{F}} - \text{Canonical}(f)$ with $\text{size}(M) = \text{size}(N)$. Furthermore, if A is a subfield of \mathbb{F} , then $N \in \text{MSP}_{A,B,\mathbb{F}} - \text{Canonical}(f)$

4.1.2 Switching the Sets A and B

The following lemma is inspired by [14, Theorem 3.4]. The complete proof is presented in the full version.

► **Lemma 27.** *For any $M \in \text{MSP}_{A,B,\mathbb{F}} - \text{Canonical}(f)$, there is $N \in \text{MSP}_{B,A,\mathbb{F}} - \text{Canonical}(f')$ with $\text{size}(M) = \text{size}(N)$.*

► **Corollary 28.** $\text{size}(\text{MSP}_{A,B,\mathbb{F}} - \text{Canonical}(f)) = \text{size}(\text{MSP}_{B,A,\mathbb{F}} - \text{Canonical}(f'))$

4.1.3 Proof of the Main Theorem

► **Definition 29.** [20] *An function $g : \{0,1\}^\ell \rightarrow \{0,1\}$ is called a restriction of a function $f : \{0,1\}^n \rightarrow \{0,1\}$ if g can be obtained by hardwiring (each to 0 or 1 independently) some of the inputs of f .*

► **Lemma 30.** *Let g be a restriction of $f : \{0,1\}^n \rightarrow \{0,1\}$. Then, for any $M \in \text{MSP}_{A,B,\mathbb{F}} - \text{Canonical}(f)$, there is $N \in \text{MSP}_{A,B,\mathbb{F}} - \text{Canonical}(g)$ with $\text{size}(N) \leq \text{size}(M)$.*

Proof. See the proof of [20, Theorem 7]. It's easy to see that the construction there preserves A and B . ◀

► **Lemma 31.** *If A, B, \mathbb{F} are all fields such that $A \subseteq B \subseteq \mathbb{F}$, then $\text{MSP}_{A,B,\mathbb{F}}(f) = \text{MSP}_{A,A,A}(f)$*

Proof. Consider any $M \in \text{MSP}_{A,B,\mathbb{F}}(f)$, we will show $M \in \text{MSP}_{A,A,A}(f)$. Let d_i be $\text{rowcount}(M, i)$ for $i \in [n]$. Consider any authorized input $v \in f^{-1}(1)$. Then, there is a row vector $r \in B^{\sum_{i \in v} d_i}$ such that $rM_v = \mathbf{1}$. Since $\mathbf{1}$ and M_v both have their entries in A , then there is $r' \in A^{\sum_{i \in v} d_i}$ such that $r'M_v = \mathbf{1}$, since a solution in an extension field implies a solution in the subfield (see [18], for example).

The security condition is trivial: for an unauthorized input $u \in f^{-1}(0)$, since M_u cannot \mathbb{F} -span $\mathbf{1}$, then it cannot A -span it either.

Now, take any $N \in \text{MSP}_{A,A,A}(f)$, we will show $N \in \text{MSP}_{A,B,\mathbb{F}}(f)$. Since $A \subset B$, the coefficient set condition is trivially satisfied. Finally, consider any unauthorized input $u \in f^{-1}(0)$. Assume for a contradiction there is $r \in \mathbb{F}^{\sum_{i \in u} d_i}$ such that $rN_u = \mathbf{1}$. As above, this would imply existence of $r' \in A^{\sum_{i \in u} d_i}$ such that $r'N_u = \mathbf{1}$, which is a contradiction. ◀

Finally, the proof of the main theorem. [20, Theorem 11] gives an algebraic variation of a lower bound proof for Th_n^2 formula size to show that $\text{size}(\text{MSP}_{\mathbb{F}_2, \mathbb{F}_2, \mathbb{F}_2}(Th_n^2)) \geq n \log_2(n)$. Here, we use the same counting argument in a more general setting along with the lemmas above to show results for the restricted model.

Proof. Let \mathbb{L} be the prime subfield of \mathbb{F} . Observe that $\text{size}(\text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}}(Th_n^t)) \geq \text{size}(\text{MSP}_{\mathbb{F}, \mathbb{L}, \mathbb{F}}(Th_n^t))$ since $\{0,1\} \subseteq \mathbb{L}$. We will mainly work with \mathbb{L} in the proof.

We will prove $\text{size}(\text{MSP}_{\mathbb{L}, \mathbb{F}, \mathbb{F}}(Th_n^2)) \geq n \log_{|\mathbb{L}|}(n)$. Assume this is true for now. By Lemma 31, $\text{size}(\text{MSP}_{\mathbb{L}, \mathbb{F}, \mathbb{F}}(Th_n^2)) = \text{size}(\text{MSP}_{\mathbb{L}, \mathbb{L}, \mathbb{L}}(Th_n^2))$. Then, by Corollary 28, $\text{size}(\text{MSP}_{\mathbb{L}, \mathbb{L}, \mathbb{L}}(Th_n^2)) = \text{size}(\text{MSP}_{\mathbb{L}, \mathbb{L}, \mathbb{L}}(Th_n^{n-1}))$. Again by Lemma 31, $\text{size}(\text{MSP}_{\mathbb{L}, \mathbb{L}, \mathbb{L}}(Th_n^{n-1})) = \text{size}(\text{MSP}_{\mathbb{L}, \mathbb{F}, \mathbb{F}}(Th_n^{n-1}))$. Therefore, $\text{size}(\text{MSP}_{\mathbb{L}, \mathbb{F}, \mathbb{F}}(Th_n^{n-1})) \geq n \log_{|\mathbb{L}|}(n)$. Finally, again by using Corollary 28, we get $\text{size}(\text{MSP}_{\mathbb{F}, \mathbb{L}, \mathbb{F}}(Th_n^2)) \geq n \log_{|\mathbb{L}|}(n)$ and $\text{size}(\text{MSP}_{\mathbb{F}, \mathbb{L}, \mathbb{F}}(Th_n^{n-1})) \geq n \log_{|\mathbb{L}|}(n)$

For any $t \leq n-1$, when we hardwire $n-t-1$ inputs of Th_n^t to 0, we get Th_{t+1}^t . Hence, by Lemma 30, $\text{size}(\text{MSP}_{\mathbb{F}, \mathbb{L}, \mathbb{F}}(Th_n^t)) \geq (t+1) \log_{\text{char}(\mathbb{F})}(t+1)$. Therefore, for any t with $\frac{n}{2} \leq t \leq n-1$, $\text{size}(\text{MSP}_{\mathbb{F}, \mathbb{L}, \mathbb{F}}(Th_n^t)) \geq \Omega(n \log_{\text{char}(\mathbb{F})}(n))$. Lastly, note that by Lemma 26, $\text{size}(\text{MSP}_{\mathbb{F}, \mathbb{L}, \mathbb{F}}(Th_n^t)) = \text{size}(\text{MSP}_{\mathbb{F}, \mathbb{L}, \mathbb{F}}(Th_n^t))$. This proves the main inequality for $\frac{n}{2} \leq t \leq n-1$.

Similar to above, for any $t \geq 2$, we can hardwire $t-2$ inputs of Th_n^t to 1 and get Th_{n-t+2}^2 . Therefore, for t with $\frac{n}{2} \geq t \geq 2$, we get $size(MSP_{\mathbb{F},\mathbb{L},\mathbb{F}}(Th_n^t)) \geq \Omega(n \log_{char(\mathbb{F})}(n))$, hence proving the main inequality for all $2 \leq t \leq n-1$.

Now we prove $size(MSP_{\mathbb{L},\mathbb{F},\mathbb{F}}(Th_n^2)) \geq n \log_{|\mathbb{L}|}(n)$. Take any $M \in MSP_{\mathbb{L},\mathbb{F},\mathbb{F}}(Th_n^2)$. Let ℓ be the number of columns of M and d_i be the number of rows of i . For any $a \in \mathbb{L} \setminus \{0\}$, define the set of column vectors $R_a := \{r \in \mathbb{L}^\ell : \mathbf{1}r = a\}$. Also, for all $i \in [n]$, $R_{i,a} := \{r \in \mathbb{L}^\ell : M'_i r = w_{d_i,a}\}$ where M'_i is a matrix with $d_i + 1$ rows with first d_i rows set to $M_{\{i\}}$ and the last row set to $\mathbf{1}$. $w_{d_i,a}$ is the column vector of size $d_i + 1$ with first d_i rows equal to 0 and the last row equal to a . It's easy to see that $\bigcup_{i \in [n]} R_{i,a} \subseteq R_a$. Also, for any $i, j \in [n]$ with $i \neq j$, we have $R_{i,a} \cap R_{j,a} = \emptyset$. We prove the disjointness by contradiction as follows. Suppose there is $r \in R_{i,a} \cap R_{j,a}$. Let $\{b_m\}_{m \in [d_i]}$ and $\{c_k\}_{k \in [d_j]}$ be the rows of parties i and j respectively. Since $t = 2$, there is $\{\beta_m\}_{m \in [d_i]}, \{\gamma_k\}_{k \in [d_j]} \subseteq \mathbb{L}$ such that $\sum_{m=1}^{d_i} \beta_m b_m + \sum_{k=1}^{d_j} \gamma_k c_k = \mathbf{1}$. Multiplying by r on both sides and considering the definitions of R_i, R_j , we get $0 = \mathbf{1}r$. This is a contradiction since $\mathbf{1}r = a \neq 0$ by definition.

Using disjointness, we get $\sum_{i=1}^n |R_{i,a}| \leq |R_a|$. Now, observe that R_a is defined by a single linear equation in \mathbb{L} . Hence, $|R_a| = |\mathbb{L}|^{\ell-1}$. Similarly, $|R_{i,a}| = |\mathbb{L}|^{\ell - rank_{\mathbb{L}}(M'_i)}$. Note that here we used the fact that $\mathbf{1}$ is not in \mathbb{L} -span of M'_i (since $t > 1$), which shows the non-homogeneous equation system defining $R_{i,a}$ is not *inconsistent*. Using the fact $rank_{\mathbb{L}}(M'_i) \leq d_i + 1$, we now have $\sum_{i=1}^n |\mathbb{L}|^{\ell-1-d_i} \leq |\mathbb{L}|^{\ell-1}$. Applying the arithmetic-geometric mean inequality (or Jensen's inequality directly), we get $\sum_{i=1}^n d_i \geq n \log_{|\mathbb{L}|}(n)$. ◀

► **Remark 32.** To get a lower bound when the field characteristic grows with n , one approach that looks promising is to consider r with entries in $\{0, 1\}$ and consider programs with entries in $\{0, 1\}$, instead of in \mathbb{L} . In fact, one can use a linear recursion⁵ or use combinatorial approaches directly to see that there are $\sum_{k=0}^{\lfloor \frac{\ell-1}{k} \rfloor} \binom{\ell}{char(\mathbb{F})k+1}$ solutions to $\mathbf{1}r = 1$ with r having entries in $\{0, 1\}$. However, the other side is problematic: sets $R_{i,1}$ can have small sizes that are independent of d_i . For example, in the case $\ell = 2n$, $char(\mathbb{F}) = n^2 + 1$, it's possible that $|R_{i,a}| = 1$, no matter how large or small d_i is,⁶ which renders this approach useless.

We finish this section with a lower bound that works for all fields, albeit it's an asymptotically insignificant result. Nevertheless, the approach will be useful in the next section for proving lower bounds for the uniform model.

► **Theorem 33.** $size(MSP_{\mathbb{F},\{0,1\},\mathbb{F}}(Th_n^t)) \geq 2n - 1$ for all t such that $1 < t < n$.

Proof. Consider any $M \in MSP_{\mathbb{F},\{0,1\},\mathbb{F}}(Th_n^t)$. We will show that there can be at most one $i \in [n]$ such that $rowcount(M, i) = 1$. For a contradiction, without loss of generality, assume that $rowcount(M, i) = rowcount(M, t+1) = 1$.

Consider the following authorized sets A_1, A_2, A_3 and the unauthorized set U_1 . $A_1 = \{1, 2, \dots, t-1, t\}$, $A_2 = \{1, 2, \dots, t-1, t+1\}$, $A_3 = \{2, 3, \dots, t-1, t, t+1\}$, $U_1 = \{2, 3, \dots, t-1, t\}$. Observe that, for any $i \in [n]$, when A is a minterm, $rowcount(M, i) = 1$ and $i \in A$, the coefficient of the single row of i must be nonzero.

⁵ This leads to a block diagonal matrix with block size $char(\mathbb{F})$ and each block being circulant, which can be solved with standard techniques.

⁶ Consider the case when there is a row of full of 1s except the last column. Since $\ell < char(\mathbb{F})$, this forces all of the first $\ell - 1$ coordinates of r to be 0. Then, the last entry is forced to be 1 by the last row of the linear system. Hence, there is only 1 solution.

Since rows of A_1 and A_2 can span $\mathbf{1}$, there is a $0 - 1$ combination of rows of these sets that are equal to each other. Canceling out the row of party 1, we see that rows of parties $\{2, \dots, t - 1, t\}$ can \mathbb{F} -span the only row of party $t + 1$.

A_3 is also authorized, so it can span $\mathbf{1}$. But we can get rid of the row of party $t + 1$ in this span equation by replacing it with what we obtained above. Thus, U_1 can \mathbb{F} -span the target, which violates the security condition. ◀

4.2 Lower Bounds for Uniform Schemes via Extremal Sets

In this section, we prove a $\Omega(n \log(n))$ lower bound for computing thresholds functions with *uniform* restricted span programs with $\{0, 1\}$ coefficients. Recall such a restricted span program, $\hat{M}(M, \rho)$, computing f is said to be uniform, if for all x such that $\text{Th}_n^t(x) = 0$ M_x has full row rank. Roughly, we show that if we can find a large family of authorized subsets that have a fixed core subset and have large pairwise intersections, then the total share size must also be large.

We start with a primitive version of the argument and then make it more flexible in the next step. Then, we go on to show lower bounds for various threshold values.

Finally, we show that a single, condensed version can show the same lower bound for (almost) all threshold values and then show that this is the optimal lower bound that can be shown with the technique we give here.

► **Theorem 34.** *Suppose $t + (2^c - 1)^{(t-1)} < n$ for some $2 < t < n$ and $c \in \mathbb{N}^+$. Then, there cannot be $M \in \text{MSP}_{\mathbb{F}, \{0,1\}, \mathbb{F}} - \text{Uniform}(\text{Th}_n^t)$ where $\text{rowcount}(M, i) = c$ for all $i \in [n]$.*

Proof. Suppose otherwise. Let $v_{i,j}$ denote the j^{th} row of party i for $i = 1, \dots, n$ and $j = 1, \dots, c$.

Consider the subset of parties $A = \{1, 2, \dots, (t - 1)\}$. If we add any one more party to this set, it will be able to $0,1$ span the target vector $w = \mathbf{1}$. Note that no matter which party we add, we will have that, for each $i = 1, 2, \dots, t - 1$, the coefficient of $v_{i,j}$ is non-zero for at least one value of $j = 1, \dots, c$. (Assume otherwise for some party i . Then its rows are contributing 0 to the span, so we can just drop party i and get a party set of $(t - 1)$ parties that can span w , which is a contradiction).

Therefore, there are $(2^c - 1)^{(t-1)}$ possible coefficient combinations for the rows of parties $1, 2, \dots, t - 1$ in any case where we add another party to them to span $\mathbf{1}$.

So, consider the parties $t, t + 1, \dots, t + (2^c - 1)^{(t-1)}$ (this is where we use the inequality assumption with c, t, n). If we add party t to the set $A = \{1, 2, \dots, (t - 1)\}$, they will be able to span $\mathbf{1}$. If we instead add party $t + 1$, again they will be able to span $\mathbf{1}$ (since that makes t many parties). It continues like this for all values $t, t + 1, \dots, (2^c - 1)^{(t-1)}$

Now, we have $(2^c - 1)^{(t-1)} + 1$ span equations giving $\mathbf{1}$, where, in each of them we have t parties (first $t-1$ parties and one another party). Furthermore, in each of them, not all coefficients of the rows of a given party is 0 (due to reasoning above: we can go down to $t-1$ parties otherwise). By the pigeonhole principle, there must be two equations (without loss of generality, say they are the ones with party t and party $t+1$ respectively) where all the row coefficients of the parties $1, 2, \dots, t - 1$ are the same. Remembering that both equations are equal to $\mathbf{1}$, we can equate them and cancel everything related to rows of parties $1, 2, \dots, t - 1$.

Now, we have an equation of the following form: $b_1 v_{t,1} + b_2 v_{t,2} + \dots + b_c v_{t,c} = d_1 v_{t+1,1} + d_2 v_{t+1,2} + \dots + d_c v_{t+1,c}$. That is, some linear combination of rows of party t is equal to some (not necessarily the same coefficients) linear combination of rows of party $t + 1$.

Finally, consider the unauthorized set of two parties: party t and party $t + 1$ (since $t > 2$). By above, the submatrix of these two parties does not have full row rank, which is a contradiction. ◀

We generalize the proof method shown above by making the number of parties that we try to cancel a parameter, along with the number of span equations we use. We will call these parameters x and ℓ respectively, and the proof method *x-fixed- ℓ -minterms* proof.

Proof. *x-fixed- ℓ -minterms* proof. Suppose in the proof above, instead of considering $(2^c - 1)^{(t-1)} + 1$ equations, we consider ℓ different equations for some parameter ℓ , corresponding to ℓ many distinct minimal (that is, of size t) authorized sets. We also require that all the minimal sets contain the first x parties, for some parameter x . Finally, we require that the union P of parties involved in pair of minimal sets, satisfy $|P - [x]| < t$. If there is a way of choosing a family of minimal sets satisfying these, we will call it a minimal set choosing strategy $Y_{x,\ell,t}$. It's easy to see that we also need $1 < x < t$.

Fix some x, ℓ, c such that there is a strategy $Y_{x,\ell,t}$ and $\ell > (2^c - 1)^x$. Then, there cannot be an MSP01-Uniform program where all n of the parties get c rows each. We prove by contradiction as follows.

Suppose otherwise. Then, we can invoke strategy $Y_{x,\ell,t}$ to get ℓ different span equations. Since $\ell > (2^c - 1)^x$; by the pigeonhole principle, there has to be two equations where the first x parties have exactly the same coefficients for each of their rows. Call the parties involved in those two equations P_1 and P_2 . By cancellation, we get a linear dependence between rows of $(P_1 \cup P_2) - [x]$. By the definition of a strategy, we have $|(P_1 \cup P_2) - [x]| < t$. Hence, the fact that the submatrix of $(P_1 \cup P_2) - [x]$ is not of full row rank is a contradiction.

We can remove the requirement that all parties get the same number of rows as follows. Observe that the pigeonhole principle would still work if we assume that c is the largest number of rows that a party among the first x parties has. However, we are not required to invoke this proof with the *actual* first x parties. Instead, re-label parties so that parties $2, 3, \dots, x$ are the parties with smallest number of rows. Then, invoke the proof by re-labeling the first party to be any party except one of those $x - 1$ parties with smallest number of rows. Now, if we have the lower bound c^* under the assumption that all parties get the same number of rows, then in the general case, we get $\text{rowcount}(M, i) \geq c^*$ for all $i \in [n]$ except $x - 1$ many of them. Hence, the total number of rows is lower bounded by $(n - x + 1)c^* + (x - 1)$.

Finally, it's easy to see that the impossibility result for $\ell > (2^c - 1)^x$ corresponds to the lower bound $c > \frac{\log_2(\ell)}{x}$. Hence, we get the following theorem. ◀

► **Theorem 35.** *If there is a strategy $Y_{x,\ell,t}$, then we have $\text{size}(\text{MSP}_{\mathbb{F},\{0,1\},\mathbb{F}} - \text{Uniform}(\text{Th}_n^t)) > (n - x + 1)\frac{\log_2(\ell)}{x} + (x - 1)$.*

We now show some strategies for various cases and the corresponding lower bounds.

► **Lemma 36.** *If $t + \ell - 1 \leq n$ and $x \geq 2$, then there is a strategy $Y_{x,\ell,t}$.*

Proof. On top of the first x parties, for each minimal set, add parties $\{x + 1, x + 2, \dots, t - 1, t + i - 1\}$ for $i = 1, \dots, \ell$. This gives us ℓ minimal sets, and we never run out of parties since $t + \ell - 1 \leq n$. Finally, the union of any two minimal sets contains $t - 1 - (x + 1) + 1 + 2$ parties, which is $\leq t - 1$ since $x \geq 2$. ◀

► **Corollary 37.** $\text{size}(\text{MSP}_{\mathbb{F},\{0,1\},\mathbb{F}} - \text{Uniform}(\text{Th}_n^t)) \geq \Omega(n \log(n - t))$ for $t \geq 3$.

Proof. Invoke the *x-fixed- ℓ -minterms* proof using the strategy $Y_{x,\ell,t}$ for $x = 2$ and $\ell = n - t + 1$. ◀

► **Corollary 38.** $\text{size}(\text{MSP}_{\mathbb{F},\{0,1\},\mathbb{F}} - \text{Uniform}(\text{Th}_n^t)) \geq \Omega(n \log(n))$ for the majority function ($t = \lceil \frac{n}{2} \rceil$).

12:20 Linear Threshold Secret-Sharing with Binary Reconstruction

► **Lemma 39.** *If $\ell \leq \binom{n-x}{t-x}$ and $x \geq \min\{n-t+1, \frac{t+1}{2}\}$, then there is a strategy $Y_{x,\ell,t}$.*

Proof. Simply pick all possible subsets of size $(t-x)$ of the set $\{x+1, x+2, \dots, n\}$. $\ell \leq \binom{n-x}{t-x}$ guarantees that we can produce ℓ minimal sets without running out of possible subsets, and $x \geq \min\{n-t+1, \frac{t+1}{2}\}$ guarantees the pairwise union size requirement (We don't prove it here, but it can be obtained using the elementary inequalities $|A \cup B| \leq |A| + |B|$ and $|U - A| \cup |U - B| \leq |U - A| + |U - B|$ where U contains both A, B .) ◀

► **Corollary 40.** *$\text{size}(\text{MSP}_{\mathbb{F},\{0,1\},\mathbb{F}} - \text{Uniform}(\text{Th}_n^t)) \geq \Omega((n-x) \frac{\log(\binom{n-x}{t-x})}{x})$ for $t \geq 3$ where $x = \min\{n-t+1, \frac{t+1}{2}\}$*

Proof. Use the strategy shown above with $\ell = \binom{n-x}{t-x}$ and $x = \min\{n-t+1, \frac{t+1}{2}\}$. Again, this is the best lower bound we can get from this family of strategies. ◀

► **Corollary 41.** *For any $t = n - \Theta(1)$ and $t = \Theta(1)$, except for $t = 0, 1, 2, n$, we have $\text{size}(\text{MSP}_{\mathbb{F},\{0,1\},\mathbb{F}} - \text{Uniform}(\text{Th}_n^t)) \geq \Omega(n \log(n))$.*

Proof. Just use the elementary inequality $\binom{n}{k} \geq (\frac{n}{k})^k$ with Corollary 40. The other side $\binom{n}{k} \leq (\frac{en}{k})^k$ shows that this is the best lower bound we can get for these thresholds using this family of strategies. ◀

It turns out that we can show all of these bounds, or in fact more, by a single graph theoretic argument: one that uses the properties of Johnson graphs. This reduction is only applicable when $x = 2$, but later we show that the lower bound (which applies to almost all threshold values) we get from this is the best lower bound we can get for any value of x .

► **Theorem 42.** *For any $3 \leq t \leq n-1$, we have $\text{size}(\text{MSP}_{\mathbb{F},\{0,1\},\mathbb{F}} - \text{Uniform}(\text{Th}_n^t)) \geq \Omega(n \log(n))$.*

Proof. Let $x = 2$. Then, let P_1, P_2 be any pair of subsets of size t provided by a fixed strategy. It's easy to show that $|(P_1 \cup P_2) - [x]| \leq t-1$ implies $|(P_1 - [x]) \cap (P_2 - [x])| \geq t-3$. Since $P_1 \neq P_2$ and $|P_1 - [x]| = |P_2 - [x]| = t-2$, we get $|(P_1 \cup P_2) - [x]| = t-3$. This shows that $P_1 - [x], P_2 - [x]$ must be adjacent in the Johnson graph $J := J_{n-2,t-2}$. This was for any pair P_1, P_2 , which means that we are looking for the largest clique in J . Its size is the clique number of the graph and is denoted $\omega(J)$.

[15, Section 16.6] states that $\chi(J_{n,k}) \leq n$, where $\chi(G)$ denotes the chromatic number of graph G . Since $\chi(G) \geq \omega(G)$ for any G , we conclude that $\omega(J) \leq n$.

In fact, for $t \leq \frac{n}{2}$, the largest clique that gives us this lower bound is the elementary sliding window family we used in Corollary 37. Furthermore, the same family/clique is one of the two simple cliques demonstrated in [15, Section 6.1]. Taking into account the other clique they show, we get $\omega(J) \geq \max\{n-t+1, t-1\} \geq \frac{n}{2}$. Hence, we get a $n \log(n)$ lower bound for all $3 \leq t \leq n-1$, thus proving Theorem 42. ◀

Lastly, we give the following result. It might indicate that x -fixed- ℓ -minterms method might not be using the full power of the 0,1 restriction, and results specific for binary matrices (and their ranks) might lead to better lower bounds for $\text{size}(\text{MSP}_{\mathbb{F},\{0,1\},\mathbb{F}} - \text{Uniform}(\text{Th}_n^t))$.

► **Corollary 43.** *Let $B \subseteq \mathbb{F}$ and $0 \in B$. Any x -fixed- ℓ -minterms based lower bound we get for $\text{size}(\text{MSP}_{\mathbb{F},\{0,1\},\mathbb{F}} - \text{Uniform}(\text{Th}_n^t))$ also works for $\text{size}(\text{MSP}_{\mathbb{F},B,\mathbb{F}} - \text{Uniform}(\text{Th}_n^t))$ when we change the base of the logarithm from 2 to $|B|$. In particular, for constant $|B|$, the lower bound stays the same asymptotically.*

Proof. Just change the base 2 to $|B|$ in the pigeonhole principle argument of x -fixed- ℓ -minterms proof. ◀

4.3 Limitations

While the fact that various values of x provided $\Omega(n \log(n))$ lower bound for various threshold values was promising that better lower bounds could be obtained by setting $x > 2$, it turns out that just using $x = 2$ is sufficient.

► **Lemma 44.** *The best lower bound we can obtain using the x -fixed- ℓ -minterms method is $\Omega(n \log(n))$.*

Proof. Here, we give a sketch of the proof and the complete proof is presented in the full version. By Ahlswede-Khachatrian Complete Intersection Theorem [2]⁷, which provides bounds for strategies (or families of subsets in their terminology) for all possible values, we conclude the following.

If there is an integer r such that $0 \leq r \leq x-1$ and $x(2 + \frac{t-2x}{r+1}) < n-x < x(2 + \frac{t-2x}{r})$, then the largest family a strategy $Y_{x,\ell,t}$ can provide is $F_r = \{A \subset \{x+1, x+2, \dots, n\} : |A| = (t-x), |A \cap \{x+1, x+2, \dots, t-x+1+2r\}| \geq t-2x+1+r\}$. Then, under the assumption that such r exists, it's easy to see that $\ell = |F_r| \leq \sum_{j=t-2x+1+r}^{t-2x+1+2r} \binom{t-2x+1+2r}{j} \sum_{j=t-2x+1+r}^{t-2x+1+2r} \binom{n-t+2x-1-2r}{t-x-j}$. Then, $\log(\ell) \leq \log((r+1) \binom{t+1+2r}{j}) + \log((r+1) \binom{n-t+2x-1-2r}{j})$. Here, we used the fact that $r \leq \frac{t-2x+1+2r}{2} \leq \frac{t-2x+1+2r}{2}$ and $x-1-r \leq \frac{n-t+2x-1-2r}{2}$ and that the binomial coefficients are larger towards the middle.

Continuing by using $r+1 \leq x$, $t+1+2r \leq 4n$, $x-1-r \leq x$, $n-t+2x-1-2r \leq n+2x \leq 4n$ and $x \leq \frac{4n}{2}$, after multiple steps and by using the inequality $\binom{n}{k} \leq (\frac{en}{k})^k$ we get $\log(\ell) \leq 4x \log(4en)$. Hence, $\frac{\log(\ell)}{x} \leq O(\log(n))$.

Finally, the case where there is no such integer r . First of all, observe that if $t \leq 2x$, we get $\frac{\log(\ell)}{x} \leq \frac{\log(\binom{n-x}{t-x})}{x} \leq \frac{t-x}{x} \log(n) \leq \log(n)$. Similarly, $n \leq 3x$ implies $\frac{\log(\ell)}{x} \leq \frac{\log(2^n)}{x} = \frac{n}{x} \leq 3$.

Therefore, we can assume $t > 2x$ and $n > 3x$. Under this, the inequality condition provided for r above becomes $x \frac{t-2x}{n-3x} - 1 < r < x \frac{t-2x}{n-3x}$.

It's easy to see that if $\frac{t-2x}{n-3x} \leq 1$, we can pick an integer r that both satisfies this and is in the range $0 \leq r \leq x-1$. Hence, we only need to focus on the case $t-2x > n-3x$, or $x > n-t$ equivalently. In that case, $\frac{\log(\ell)}{x} \leq \frac{\log(\binom{n-x}{t-x})}{x} = \frac{\log(\binom{n-x}{n-t})}{x} \leq \frac{n-t}{x} \log(\frac{e(n-x)}{n-t}) \leq \frac{n-t}{x} \log(en) \leq O(\log(n))$ ◀

The fact that we have $O(n \log(n))$ upper bound for fields of characteristic 2 shows that field-agnostic approaches like the one here cannot yield lower bounds better than $\Omega(n \log(n))$. With this lemma, we also showed that subset-counting approaches like the one presented is not likely to yield better lower bounds even if they were specifically for fields with characteristic different than 2.


References

- 1 Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Functional encryption for threshold functions (or fuzzy IBE) from lattices. In *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 280–297. Springer, 2012.
- 2 Rudolf Ahlswede and Levon H. Khachatrian. The complete intersection theorem for systems of finite sets. *European Journal of Combinatorics*, 18(2):125–136, 1997. URL: <http://www.sciencedirect.com/science/article/pii/S0195669885700923>.

⁷ See [21] if you are only interested in the theorem statement.

- 3 Miklós Ajtai, János Komlós, and Endre Szemerédi. Sorting in $c \log n$ parallel sets. *Comb.*, 3(1):1–19, 1983.
- 4 Amos Beimel. Secure schemes for secret sharing and key distribution, 1996.
- 5 Amos Beimel. Secret-sharing schemes: A survey. In *IWCC*, volume 6639 of *Lecture Notes in Computer Science*, pages 11–46. Springer, 2011.
- 6 Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer, 1988.
- 7 George Robert Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318. IEEE, 1979.
- 8 Andrej Bogdanov, Siyao Guo, and Ilan Komargodski. Threshold secret sharing requires a linear size alphabet. In *Theory of Cryptography Conference*, pages 471–484. Springer, 2016.
- 9 Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO (1)*, volume 10991 of *Lecture Notes in Computer Science*, pages 565–596. Springer, 2018.
- 10 Ravi B. Boppana. Amplification of probabilistic boolean formulas. *Adv. Comput. Res.*, 5:27–45, 1989.
- 11 Xavier Boyen. Attribute-based functional encryption on lattices. In *TCC*, volume 7785 of *Lecture Notes in Computer Science*, pages 122–142. Springer, 2013.
- 12 Ronald Cramer and Serge Fehr. Optimal black-box secret sharing over arbitrary abelian groups. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '02*, page 272–287, Berlin, Heidelberg, 2002. Springer-Verlag.
- 13 J. Friedman. Constructing $o(n \log n)$ size monotone formulae for the k -th elementary symmetric polynomial of n boolean variables. In *25th Annual Symposium on Foundations of Computer Science, 1984.*, pages 506–515, 1984.
- 14 Anna Gal. A characterization of span program size and improved lower bounds for monotone span programs. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, STOC '98*, page 429–437, New York, NY, USA, 1998. Association for Computing Machinery. doi:10.1145/276698.276855.
- 15 Christopher Godsil and Karen Meagher. *Erdős–Ko–Rado Theorems: Algebraic Approaches*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2015.
- 16 Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, pages 545–554. ACM, 2013.
- 17 M. M. Halldorsson, J. Radhakrishnan, and K. V. Subrahmanyam. Directed vs. undirected monotone contact networks for threshold functions. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 604–613, 1993.
- 18 Kenneth Hoffman and Ray A. Kunze. *Linear Algebra*. PHI Learning, second edition, 2004. URL: <http://www.worldcat.org/isbn/8120302702>.
- 19 Shlomo Hoory, Avner Magen, and Toniann Pitassi. Monotone circuits for the majority function. In *APPROX-RANDOM*, volume 4110 of *Lecture Notes in Computer Science*, pages 410–425. Springer, 2006.
- 20 M. Karchmer and A. Wigderson. On span programs. In *[1993] Proceedings of the Eighth Annual Structure in Complexity Theory Conference*, pages 102–111, 1993.
- 21 Gyula O.H. Katona. Around the complete intersection theorem. *Discrete Applied Mathematics*, 216:618–621, 2017. Levon Khachatryan’s Legacy in Extremal Combinatorics. URL: <http://www.sciencedirect.com/science/article/pii/S0166218X1600010X>.
- 22 Mike Paterson. Improved sorting networks with $o(\log N)$ depth. *Algorithmica*, 5(1):65–92, 1990.
- 23 Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
- 24 Leslie G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5(3):363–366, 1984.

Doubly-Affine Extractors, and Their Applications

Yevgeniy Dodis 

New York University, NY, USA

Kevin Yeo 

Google, New York, NY, USA

Columbia University, New York, NY, USA

Abstract

In this work we challenge the common misconception that information-theoretic (IT) privacy is too impractical to be used in the real-world: we propose to build simple and *reusable* IT-encryption solutions whose only efficiency penalty (compared to computationally-secure schemes) comes from a large secret key size, which is often a rather minor inconvenience, as storage is cheap. In particular, our solutions are *stateless* and *locally computable at the optimal rate*, meaning that honest parties do not maintain state and read only (optimally) small portions of their large keys with every use.

Moreover, we also propose a novel architecture for outsourcing the storage of these long keys to a network of semi-trusted servers, trading the need to store large secrets with the assumption that it is hard to simultaneously compromise too many publicly accessible ad-hoc servers. Our architecture supports *everlasting privacy* and *post-application security* of the derived one-time keys, resolving two major limitations of a related model for outsourcing key storage, called bounded storage model.

Both of these results come from nearly optimal constructions of so called *doubly-affine extractors*: locally-computable, seeded extractors $\mathbf{Ext}(X, S)$ which are linear functions of X (for any fixed seed S), and protect against bounded affine leakage on X . This holds unconditionally, even if (a) affine leakage may *adaptively depend* on the extracted key $R = \mathbf{Ext}(X, S)$; and (b) the seed S is only *computationally secure*. Neither of these properties are possible with general-leakage extractors.

2012 ACM Subject Classification Security and privacy \rightarrow Information-theoretic techniques

Keywords and phrases extractors, information-theoretic privacy, everlasting privacy

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.13

Related Version *Full Version*: <https://eprint.iacr.org/2021/637> [13]

Funding *Yevgeniy Dodis*: Yevgeniy Dodis was partially supported by gifts from VMware Labs, Facebook and Google, and NSF grants 1815546, 2055578.

1 Introduction

Information-theoretic (IT) security is very attractive as it enables provably secure schemes that resist advances in computational power, novel cryptanalysis, or the possibility of quantum computers. This is especially important for privacy applications, where huge amounts of encrypted communication are being stored and recorded, with the danger that all these communications could be decrypted years later. Unfortunately, the famous impossibility result of Shannon [28, 10] states that IT-secure schemes come at a price: the secret should be at least as large as the message. The traditional interpretation of this negative result is that one must settle for much weaker computational security, so as to make the problem of key distribution feasible.

1.1 Reusable IT-Encryption

As we observe, just because the secret key must be large does not make the system automatically impractical. In fact, since local storage is often cheap, a (necessarily) large secret key X might be a very reasonable price to pay for *unconditional security*, provided this is the



© Yevgeniy Dodis and Kevin Yeo;
licensed under Creative Commons License CC-BY 4.0
2nd Conference on Information-Theoretic Cryptography (ITC 2021).
Editor: Stefano Tessaro; Article No. 13; pp. 13:1–13:23



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

only efficiency penalty when compared to computationally-secure schemes. For the purposes of this work, we will interpret this latter requirement by demanding that our solutions are simultaneously *stateless* and *locally-computable*.

Stateless. Our first requirement demands that parties keep no state beyond storing the original key X . This is crucial when there are many parties that cannot easily remain synchronized together. For example, this could be the case when the secret key is shared by $k \gg 2$ parties and each party cannot view all transmissions (possibly intentionally). It also removes out-of-sync errors as a common potential source of insecurity that cause multiples parties to accidentally reuse the same portion of the key. In particular, statelessness rules out trivial solutions where parties slowly utilize consecutive (fresh) portions shared key X (e.g., as one-time pads) until X “runs out” that is infeasible with many parties.

Locally Computable. Our second requirement of local compatibility ensures that each concrete application of the scheme (both as sender and receiver) only accesses the long secret key X in very few locations. We call this number of locations p the *probe complexity*, and the ratio $\alpha = \frac{m}{p} \in [0; 1]$ the *locality* of a given solution. Requiring $p \ll |X|$ rules out elegant (stateless) solutions using so called ℓ -wise independent hash functions, because all conventional ℓ -wise independent hash functions read the entire long key X for every evaluation.

Rate of IT-encryption. We can now define our first motivating question more formally. As the “price” for being stateless and locally computable, we introduce a “waste” parameter $\beta > 0$, indicating that the total length of all the messages we wish to (statelessly) encrypt is bounded by $(1 - \beta)|X|$. Given this “waste” $\beta > 0$ and message length m , our goal is to minimize the probe complexity p (and maximize locality α).

It is not hard to see (this follows from the more general observation of [30]) that $p \geq m/\beta$ (i.e., $\alpha \leq \beta$), which is indeed independent of the key length $|X|$. Intuitively, since in any stateless scheme up to $(1 - \beta)$ -fraction of X might have been already used to encrypt prior messages, one needs to sample on average $1/\beta$ bits of X to get an “unused” bit. We ask if this exact bound is tight, and optimal locality $\alpha \approx \beta$ can be achieved, perhaps up to an *sub-linear additive* loss (that we denote by $o(m)$ while omitting the security parameter λ from notation)?

► **Open Problem 1.** *Design practical, stateless and reusable IT-secure encryption schemes of m -bit messages with n -bit key and probe complexity $p = m/\beta + o(m)$, where $(1 - \beta)n$ is the upper bound on the total length of the encrypted messages.*

As one of our main results, we present the first affirmative solution to this open question, achieving $p = m/\beta + O(\sqrt{\lambda(m + \lambda)}) = m/\beta + o(m)$, whenever message length $m = \omega(\lambda)$. Our scheme is quite practical. We present the exact expression with no hidden constants in Theorems 13 and 15, and demonstrate concrete improvements over prior state-of-the-art solutions [5, 4] in Section 6.

1.2 Delegating Storage

While solving Open Problem 1 means that long secrets is the *only* “price” for unconditional security (when compared to computationally-secure schemes), we would like to do even better, and delegate the storage of these large keys (to a cloud provider as an example). Since this clearly overcomes the Shannon’s impossibility result, we require strong trust assumptions with the storage server.

To achieve this ambitious goal, our encryption scheme for message M will compute a ciphertext $C = (S, R \oplus M)$, where $R = \mathbf{Ext}(X, S)$, \mathbf{Ext} is a carefully chosen locally computable extractor [30] and S is a fresh random seed chosen by the sender. At a high level, we would like the server to store X instead of the users, and only the honest sender/receiver(s) be able to retrieve the correct one-time pads $R = \mathbf{Ext}(X, S)$ from the server.

Basic Architecture. As the first attempt, imagine some *virtual server* \mathbf{T} will choose the long random string X , for the altruistic purpose of helping users utilize “reusable IT-security”. In our final architecture, the virtual server will be emulated by a network of semi-trusted servers under (still strong, but) more plausible trust and communication assumptions. But, for now, we will think of \mathbf{T} as not only stateless and locally computable, but also *fully trusted, incorruptible, and having private channels to any user who contacts it*.¹

Since \mathbf{T} might not even know its user base, we assume that \mathbf{T} is truly public, and does not perform any explicit authentication. Hence, anybody, *including the attacker* \mathbf{A} , can send a seed S to \mathbf{T} , and get back the value $R = \mathbf{Ext}(X, S)$. However, we assume that the total length of one-time pads obtained by the attacker is bounded by $(1 - \beta)|X|$ that may be achieved by ensuring that servers stop responding after a certain number of requests.

Sharing the seed. In the setting of Section 1.1, the seed S used to derive $R = \mathbf{Ext}(X, S)$ was sent in the clear, as part of the ciphertext $C = (S, R \oplus M)$. This was fine since access to X was given only to the honest users, and not the attacker \mathbf{A} . Now, however, S cannot be sent in the clear, as then \mathbf{A} can directly query \mathbf{T} on S , just as the honest recipient would.

Now, we need some mechanism how only the authorized parties learn S . Moreover, they need to do this repeatedly for every new message M . This looks like a chicken-and-egg problem, as transmitting fresh seeds while protecting their privacy *unconditionally* is as hard as solving the reusable IT-encryption. To resolve this dilemma, we would like for this idea to work even if the seed S is shared using some *computationally-secure* mechanism. As natural examples, parties can (a) run fresh Diffie-Hellman key agreement to generate S ; or (b) share a short symmetric key k once, and have the sender computationally encrypt seed S using k ; or (c) in the public-key setting, computationally encrypt S using the receiver’s public key. In all of these scenarios, we want to claim that M remains private *forever*, as long as the privacy of S is not broken *during the lifetime of server* \mathbf{T} . We call this notion of privacy *everlasting privacy*, following the terminology of [2].

Allowing adversarial seeds. In the setting of Section 1.1, the attacker could only observe prior extractor outputs $\mathbf{Ext}(X, S_i)$ on *honestly chosen*, random seeds S_i . In contrast, the current setting enables the adversary \mathbf{A} to learn outputs $\mathbf{Ext}(X, S_i)$ on *adversarial* seeds S_i . Moreover, the seed S_i could be chosen effectively depending on the “challenge one-time pad” $R = \mathbf{Ext}(X, S)$. With respect to the security game, if \mathbf{A} observes “challenge encryption” $P = R \oplus M$ and knows the message $M \in \{V_0, V_1\}$. Therefore, \mathbf{A} can deduce $R \in \{P \oplus V_0, P \oplus V_1\}$ without knowing the “challenge seed” S . Hence we want our architecture to be *post-application secure* [12]. That is, it should be safe to let the attacker interact with the servers, even after the honest parties use the challenge encryption P (either large portions of P or all of P may be leaked to \mathbf{A}).

¹ These strong assumptions will be substantially weakened, once we replace the virtual server by several “real” servers. This is similar in spirit to IT secret sharing [27] and MPC literature [8], which assume private channels to uncorrupted servers. However, we will do even better, as there are no “consistency requirements” for generating randomness. For example, our servers will be ad hoc, and do not communicate (or possibly even know!) about each other. See Section 5.2.

To summarize the preceding discussion, to soundly realize our basic architecture for key delegation, the chosen extractor $\mathbf{Ext}(X, S)$ must be (a) everlastingly private with computationally-secure seeds; and (b) post-application secure.

► **Open Problem 2.** *Design IT-secure, locally computable extractor \mathbf{Ext} for the sound implementation of the basic delegation architecture. In particular, \mathbf{Ext} should be post-application secure and support computationally-secure seeds.*

In this work we design the first architecture which supports these guarantees. Moreover, we show how to distribute the virtual server among several servers, only some of which can be trusted. Our solution is (a) *ad-hoc*, meaning servers do not need to know about or coordinate with each other, and (b) almost fully *stateless*, meaning the servers need to maintain minimal-to-no state except to ensure that adversaries view a bounded amount of leakage.

1.3 Locally Computable Extractors to the Rescue?

Our first hope is that standard locally computable extractors (LCEs), as originally formalized in the context of Bounded Storage Model (BSM) encryption [30], would be precisely what we need to solve Open Problems 1 and 2. Such an extractor $\mathbf{Ext}(X, S)$ is guaranteed to work on a uniform, n -bit key X , even despite the attacker obtaining up to $(1 - \beta)n$ leakage bits $L = f(X)$, for any function f of attacker's choice. In particular, by modeling previous extractor outputs $\mathbf{Ext}(X, S_i)$ as leakage on X , the resulting scheme appears to suffice for our purposes of building reusable IT-encryption. Unfortunately, general LCEs do not work for either of our questions, both quantitatively and qualitatively.

Suboptimal Rate. While the best upper bound [30] on the locality α of LCEs is the same $\alpha \leq \beta$ as we have in our simpler setting, we currently do not have schemes matching this bound. Thus, this approach will not help us resolve Open Problem 1. The best known scheme of [5] achieves $\alpha_0(\beta) \approx -\log_2(1 - h_2^{-1}(\beta))$, where $h_2(z) = -z \log_2(z) - (1 - z) \log_2(1 - z)$ is the binary entropy function, and $h_2^{-1}(\beta)$ takes the smaller of two possible inverses. It is easy to see that $\alpha_0(\beta) \ll \beta$ for all β even slightly bounded away from 0 and 1. For example, $\alpha_0(0.5) = 0.168 \ll 0.5$ and $\alpha_0(0.1) = 0.019 \ll 0.1$. In fact, [5] proved the optimality of their particular proof technique based on the so called “subkey prediction lemma” [1, 5, 4] (although it is conceivable a better non-asymptotic LCE bound will be found with a different technique; e.g., those from [24, 30]).

Computationally Secure Seeds. Just like in our setting from Section 1.2, supporting computationally secure seeds would be a huge win for the BSM setting. Surprisingly, several works [19, 15] showed that the BSM (and LCE) is too general to handle computationally-secure seeds. First, everlasting privacy in the BSM may not be reduced in a black-box manner to any computational assumption [19]. Second, there are explicit examples of computationally secure mechanisms to generate S which would break BSM security for *any* LCE. For example, if the attacker knows encryption Z of S under some fully homomorphic encryption (FHE), this still leaves S computationally secure. Yet, the attacker can *efficiently* evaluate $\mathbf{Ext}(X, \cdot)$ inside the ciphertext as its compact leakage function $L = f(X, Z)$, learning FHE of the one-time pad $R = \mathbf{Ext}(X, S)$. When the attacker later becomes unbounded, it can break the FHE, and learn R .

Fortunately, this attack on BSM does not translate to our setting in the delegated storage setting. The servers will refuse to homomorphically evaluate the given ciphertext, as this does not correspond to evaluating $\mathbf{Ext}(X, S_i)$ on some seed S_i . However, it shows that we need a more refined approach to *provably* achieve everlasting privacy.

Post-Application Security. In the traditional BSM setting, the leakage $L = f(X)$ may only depend on the random source X . For post-application security, however, we allow the leakage function to also depend on R ; that is, $L = f(X, R)$. Unfortunately, general LCEs cannot be post-application secure, at least for the interesting setting when $|S| < |R|$.² To see this, consider a boolean leakage function $f(X, R)$ which is 1 if and only there exists some seed S which yields $R = \mathbf{Ext}(X, S)$. When $|S| < |R|$, such $f(X, R)$ will always be true with “real” R , but almost never true with random R (see our full version for more details).

Once again, this attack does not translate to our setting, as such leakage does not correspond to evaluating $\mathbf{Ext}(X, S_i)$ on some seed S_i . However, it shows that we need a more refined approach to *provably* achieve post-application-security.

1.4 Doubly-Affine Extractors

To overcome the limitations of existing LCEs, we notice that, for both of our application scenarios, the leakage of X consists of values $\mathbf{Ext}(X, S_i)$ for various seeds S_i . Hence, we can try to design efficient extractors which are *only secure against leakage of their own outputs*. Moving forward, we will denote LCEs with this property as simply extractors. With this approach, we will resolve both Open Problems 1 and 2.

Linearity. We observe that most existing LCEs [2, 15, 22, 30] are *linear (affine) functions* of X (for any fixed S). For our settings, an affine extractor only needs to be secure against what is called *affine* leakage functions resulting from previous extractor outputs. Such extractors are called *affine extractors* [16], and certainly appear easier than “general leakage” extractors. However, until now affine extractors have only been considered in the seedless setting. As a result, these seedless affine extractors are neither locally computable, nor linear.

In this work, we initiate the study of *seeded* affine extractors which are both locally computable and linear functions of the source X . For conciseness, we will call such (seeded, locally computable) extractors *doubly-affine*, where “affine” now refers to both the leakage and the extractor itself.

Our Model and its Advantages. The formal security game for doubly-affine extractors is given in Figure 1. The attack is split in two stage. In this first state, the attacker \mathbf{A}_1 is given challenge output R (either $\mathbf{Ext}(X, S)$ or uniform), and can make up to $\ell = (1 - \beta)n$ adaptive affine leakage queries. Since these queries are adversarial and our extractor is linear, they can model extractor outputs $\mathbf{Ext}(X, S^*)$ on adversarial seeds S^* . Thus, *post-application security is built into the definition*.

In the second stage, the attacker \mathbf{A}_2 is given the seed S , but cannot make any more leakage queries. This models everlasting privacy, although not necessarily with respect to computationally secure seeds (yet). To model the latter concern, we augment the basic definition in Figure 1 to a seemingly more advanced setting in Figure 2, where some abstract seed-generating procedure $\Sigma(S')$ outputs the extractor seed S and the side-information Z . This side information Z is given to \mathbf{A}_1 to help making its affine queries, and the entire seed S' used by the computationally secure seed generator is given to the second-stage attacker \mathbf{A}_2 . We require that the game in Figure 2 is secure against any computationally bounded \mathbf{A}_1 and *unbounded* \mathbf{A}_2 , as long as S remains pseudorandom to \mathbf{A}_1 given Z .

² Setting $|S| \geq |R|$ is uninteresting for BSM, as parties can then use S instead of R .

For example, to model Diffie-Hellman key exchange, we set $S' = (a, b)$, $Z = (g^a, g^b)$ and $S = \mathbf{Prg}(g^{ab})$, for some pseudorandom generator \mathbf{Prg} . The fact that we give the values a and b to \mathbf{A}_2 now accurately models the fact that an unbounded attacker can break discrete log of g^a and g^b eventually, and thus learn more information than simply recovering the extractor seed $S = \mathbf{Prg}(g^{ab})$.

Fortunately, unlike the setting of general LCEs, we show that *any* doubly-affine extractor satisfying a the simpler definition in Figure 1 will also automatically satisfy the definition in Figure 2. Thus, our basic definition in Figure 1 also covers *everlasting security against computationally secure seeds*.

Parameters and Efficiency. Last, but not the least, restricting to linear leakage allows to solve our nearly optimal probe complexity $\mathbf{p} \approx \mathbf{m}/\beta$, settling Open Problem 1 in the affirmative. As we show in Section 6, our construction is also concretely efficient, making it attractive for real-world applications where IT-security matters.

As an additional advantage, it gracefully extends to a more refined model of local computability, where in addition to the probe complexity \mathbf{p} , we also wish to minimize the number of non-contiguous memory c blocks one needs to read the required \mathbf{p} bits. We call this parameter c *cache complexity*, as it roughly corresponds to the number of cache misses to read c non-contiguous regions of memory. Unlike probe complexity, cache complexity does not have to grow with the number of extracted bits \mathbf{m} , and can be as small as $c = O(\lambda)$, where λ is the security parameter. Indeed, our main construction generally gives $\mathbf{p} = \frac{\mathbf{m}}{\beta} \cdot (1 + \sqrt{\frac{O(\lambda)}{c}})$.

To summarize, doubly-affine extractors provide all the properties we need to simultaneously resolve Open Problems 1 and 2.

1.5 Our Constructions and Techniques

Our doubly-affine extractor follow the sample-then-extract approach introduced by Vadhan [30] for LCE. The first sampling step selects a subset I of \mathbf{p} bits of X , denoted by $Y = X|_I$, and the second step applies a non-local extractor to Y to produce the final output R . In our work, we improve the parameters for both steps, when the leakage is restricted to be affine.

Sampler Improvement. The two samplers we analyze were already considered by prior work on LCEs [23, 24, 30, 5], but our work presents new, improved analyses for the case of affine leakage. As our key insight, we prove (see Theorem 7) that the optimal affine leakage strategy against *any* sampler is to select some *physical* $(1 - \beta)\mathbf{n}$ bits of X . In other words, the best adversarial strategy is to simply try and guess as many locations of the sampled bits as possible.

This result is surprising for two reasons. First, the same equivalence is false for general-leakage samplers: [5] shows that even simple (but highly non-linear) leakage functions may greatly outperform physical-bit leakage. Second, the equivalence is false for the overall setting of doubly-affine extractors. Ignoring locality, for example, parity of all \mathbf{n} bits of X is trivially secure against bounded leakage of up to $(\mathbf{n} - 1)$ bits, but is trivially insecure against a single-bit of affine leakage.

Once we reduce to physical-bit leakage, a simple Chernoff bound easily implies that the number of “non-leaked” physical bits in a “random-enough” \mathbf{p} -bit sample of X is highly concentrated around its expected value $\beta\mathbf{p}$ – a conclusion which would seem highly non-obvious without our equivalence.

Extractor Improvement. Once we know that the sample Y has entropy of approximately βp in the adversary's view, we can apply any non-local linear \mathbf{Ext}' to extract $m \approx \beta p$ bits from Y . For *concrete security*, especially for small values of m , we still want to optimize the *entropy loss* ($\beta p - m$). Using extractors for general leakage, this entropy loss is known to be at least 2λ [26], and this bound is easily achieved by many linear extractors (e.g., [20]).

Once again, we observe that our non-local extractor only has to withstand affine leakage. In particular, we show that the optimal entropy loss for (non-local) doubly-affine extractors is only λ , *saving a factor of two over general-leakage extractors*. We present a general construction of such non-local, doubly-affine extractors from rank-preserving matrices (see [11]), that may be instantiated from a variety of concrete matrices such as Toeplitz matrices.

Seed Length. In our analysis, we did not optimize the length s of S . Existentially, we show that all our improvements are possible (unconditionally) with $s = O(\lambda)$, while our concrete constructions use larger seed length $s = O(m + \lambda \log(n))$. Such a seed-length is quite acceptable for most applications, as this only increases the ciphertext length (or communication with the server \mathbf{T}) by a constant factor. Moreover, to match computationally-secure encryption schemes with optimal ciphertext length $m + O(\lambda)$, we use the fact that doubly-affine extractors are *everlastingly private* with computationally-secure seeds. Hence, we can use any stream cipher (e.g., SALS20 or CHACHA) to expand a λ -bit seed S' into the required longer seed $S = \mathbf{Prg}(S')$, *while maintaining IT-security*.

Our reusable IT-encryption in Section 1.1 has ciphertext $(S', \mathbf{Ext}(X, \mathbf{Prg}(S')) \oplus M)$ of optimal length $(m + \lambda)$, matching that of computationally secure schemes! Similarly, the communication complexity when interacting with our virtual server \mathbf{T} from Section 1.2 can be made optimal: λ “upstream” bits S' from the users, and m “downstream” bits $R = \mathbf{Ext}(X, \mathbf{Prg}(S'))$ from the server. We stress that we need an extremely weak kind of computational-security security for the \mathbf{Prg} : just ability to fool a concrete, and easily computable statistical test (which we know a random expanding function satisfies w.h.p.). Thus, it seems *extremely plausible* that SALS20 or CHACHA satisfy this combinatorial property *unconditionally*. Nevertheless, it is a good theoretical question to improve the seed length s to the optimal value $O(\lambda)$.

1.6 Applications

Replicated Setting. We generalize the setting of Section 1.1, where the entire long secret key X is replicated among several trusted parties. We already saw that doubly-affine extractors immediately give locally computable, CPA-secure encryption $C = (S, \mathbf{Ext}(X, S) \oplus M)$ *with optimal locality* in this setting. In fact, the same scheme is trivially CCA1-secure, since doubly-affine extractors support leakage of extractor outputs on adversarial seeds S . To get CCA2 security, and even achieve the strongest notion of authenticated encryption (AE) [6], the parties can additionally share a short key for *computationally-secure* MAC, and use this fixed key to authenticate the ciphertext $C = (S, P)$ above. In this variant, the authenticity is computational, but the privacy is everlasting, as long as the MAC is not broken while the post-challenge decryption oracle is used. Moreover, all these schemes are still everlastingly private with computationally-secure seeds S , allowing to achieve optimal ciphertext length $(m + O(\lambda))$.

Distributed Setting. We generalize the setting of Section 1.2 where parties delegate the storage of X to several servers. We already saw that doubly-affine extractors are enough in the single server case, as they provide the required post-application security and support

computationally secure seeds. For example, the server \mathbf{T} can help the parties to achieve all the efficiency benefits of (symmetric-key) authenticated encryption with associated data (AEAD) plus *everlasting privacy*. All the parties need to do is use a computationally-secure AEAD (using a short shared key) to encrypt the seed S ,³ instead of the message M . Similar techniques also work in the public-key setting, where S can be appropriately encrypted using the receiver’s public-key.

While relying on a single trusted server \mathbf{T} may be challenging due to privacy of the channel, we can consider distributed setting with multiple servers $t \geq 2$ where we use the standard assumption in the information-theoretic literature that channels between the user and at least g servers are secure and the remaining $t - g$ channels may be compromised. This is an assumption that has been used in many prior seminal works including information-theoretic secret sharing [27], multi-party computation [8] and secure message transmission [14].

Specifically, we extend our architecture to the setting of $t \geq 2$ servers, who jointly emulate the virtual server \mathbf{T} . In more detail, each of the t servers will independently generate and store a subset of the random source X . For the distributed setting, we only assume that $g \leq t$ servers are honest with a private channel to users. Moreover, the servers do not need to coordinate, or even know each other’s existence: each simply picks a random string, and provides access to its random string to users.

We also consider two cases where the $(t - g)$ corrupted servers are either honest-but-curious or malicious. Our honest-but-curious solution works for any $g \geq 1$, and achieves multiplicative overhead roughly t/g for the user, as compared to the single-server case. In particular, each server accesses and returns a *sub-linear* number of bits $p' \approx m/(\beta g)$. For the malicious setting, we necessarily assume that $g > 2t/3$, and use simple error-correcting techniques to achieve overhead roughly $t/(3g - 2t)$, with each server returning $p' \approx m/(\beta(3g - 2t))$ bits.

2 Definitions

In this section, we formally define doubly-affine extractors that output affine functions of the random source while tolerating affine leakage. We start by presenting the affine oracle that provides linear access to a truly random string. Afterwards, we define doubly-affine extractors in both the information theoretic and computationally secure settings.

2.1 Affine Leakage Model

In the affine leakage model, there is a uniformly random string $X \in \{0, 1\}^n$. Throughout our work, X is referred to as the *source* or *random source*. The string X is accessed through an affine oracle that receives a n -bit string $Q \in \{0, 1\}^n$ and returns the dot product of $\mathbf{LIN}_X(Q) = Q \cdot X$. In other words, one query to the affine oracle enables retrieving the XOR of a subset of bits of X .

For convenience, multiple queries to the affine oracle may be represented using a single matrix. In particular, q queries may be represented using a $q \times n$ bit-matrix $\mathbf{Q} \in \{0, 1\}^{q \times n}$ such that $\mathbf{LIN}_X(\mathbf{Q}) = \mathbf{Q}X$ where the affine oracle returns the multiplication of \mathbf{Q} and X resulting q bits.

► **Definition 1.** *For any $X \in \{0, 1\}^n$, the affine oracle \mathbf{LIN}_X receives a $q \times n$ binary matrix $\mathbf{Q} \in \{0, 1\}^{q \times n}$ for any $q \geq 1$. Then, $\mathbf{LIN}_X(\mathbf{Q}) = \mathbf{Q}X$.*

³ Technically, S should be encrypted with associated data $P = R \oplus M$.

2.2 Information-Theoretic Doubly-Affine Extractors

The main focus of our work is to construct efficient extractors in the affine leakage model. The goal of a doubly-affine extractor is to utilize a short random seed along with access to the oracle \mathbf{LIN}_X to derive a random string that may be used at higher level applications. For security, the extractor's output should remain random even if an adversary uses the oracle \mathbf{LIN}_X to learn large (but not all) of the underlying random string X .

In more detail, extractors are defined as algorithms that receive a random s -bit seed and output a m -bit random string where $m > s$ (that is, the output random string is larger than the input seed). Extractors are able to perform queries to the oracle \mathbf{LIN}_X to access the uniformly random string X . The output of extractors should remain random to an adversary that has utilized the oracle \mathbf{LIN}_X to learn at most ℓ bits about the random string X .

In this paper, we restrict our attention to extractors that only perform non-adaptive queries to \mathbf{LIN}_X . In other words, the extractor must pick a single query matrix \mathbf{Q} , send it to the \mathbf{LIN}_X and use the response to generate the output. To our knowledge, all prior works also exclusively studied extractors that non-adaptively accessed the underlying random string X . Non-adaptivity may be beneficial in settings where sending queries to \mathbf{LIN}_X may be expensive.

For convenience, we will make the assumption that the output of doubly-affine extractors will simply be the response from the single query to the oracle \mathbf{LIN}_X . We show that this limitation is not important as our constructions will be essentially optimal. With this restriction, the output of doubly-affine extractors will also be affine. This property will be integral in settings when the adversary's leakage consists of previous extractor outputs.

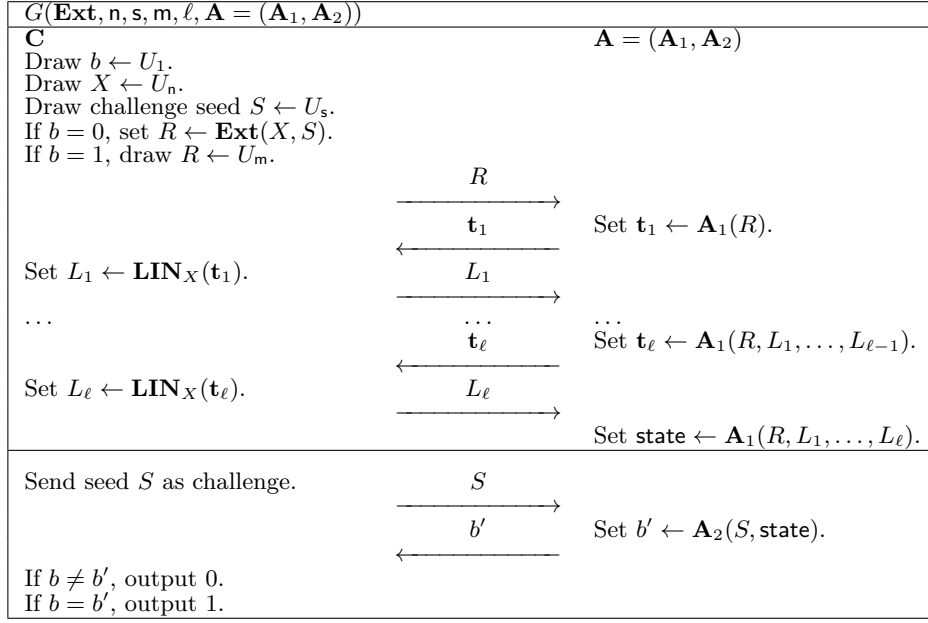
We define extractors as $\mathbf{Ext} : \{0, 1\}^n \times \{0, 1\}^s \rightarrow \{0, 1\}^m$ where the first argument is the n -bit random string, the second argument is s -bit random seed and the output are the m extracted bits. As we consider non-adaptive extractors, we note that all extractors \mathbf{Ext} are uniquely defined by a query algorithm $\mathbf{Pick}_{\mathbf{Ext}} : \{0, 1\}^s \rightarrow \{0, 1\}^{m \times n}$ that outputs a $m \times n$ binary matrix that will be query to the oracle \mathbf{LIN}_X . Since \mathbf{Ext} returns the output from the oracle, we note that $\mathbf{Ext}(X, S) = \mathbf{LIN}_X(\mathbf{Pick}(S))$ for any n -bit random string X and s -bit random seed. We will use \mathbf{Ext} and $\mathbf{Pick}_{\mathbf{Ext}}$ interchangeably in our paper.

The security of doubly-affine extractors are presented in Figure 1. The adversary \mathbf{A} is given a challenge of either a m -bit uniformly random string or the extractor output. Using the oracle \mathbf{LIN}_X , \mathbf{A} may perform ℓ adaptive queries to learn at most ℓ linear functions of X with knowledge of the challenge. Afterwards, \mathbf{A} is given the input seed and must guess the origin of the challenge. We say \mathbf{A} has ε advantage if \mathbf{A} has $1/2 + \varepsilon$ probability of guessing correctly. For ease of presentation, we split \mathbf{A} into stateful adversaries \mathbf{A}_1 and \mathbf{A}_2 that are responsible for generate oracle queries and computing the final guess respectively. We note that both adversaries are computationally unbounded.

We stress that \mathbf{A} is given the challenge prior to performing any queries to the oracle \mathbf{LIN}_X . As a result, we require that doubly-affine extractors provide security against *post-application leakage*. This is a notion that is not achievable in other models with general leakage (we present a counterexample in our full version). By restricting to linear leakage, we enable a significant improvement in security.

► **Definition 2.** A deterministic algorithm $\mathbf{Ext} : \{0, 1\}^n \times \{0, 1\}^s \rightarrow \{0, 1\}^m$ is (ℓ, ε) -secure if for any adversary \mathbf{A} , $\Pr[G(\mathbf{Ext}, n, s, m, \ell, \mathbf{A}) = 1] \leq \frac{1}{2} + \varepsilon$.

We move onto the efficiency of extractors. We define the *probe complexity* of an extractor as the number of bits of X accessed by the oracle in a single extractor execution. We define the *cache complexity* of an extractor as the number of disjoint regions of X accessed by the



■ **Figure 1** Game $G(\text{Ext}, n, s, m, \ell, \mathbf{A})$.

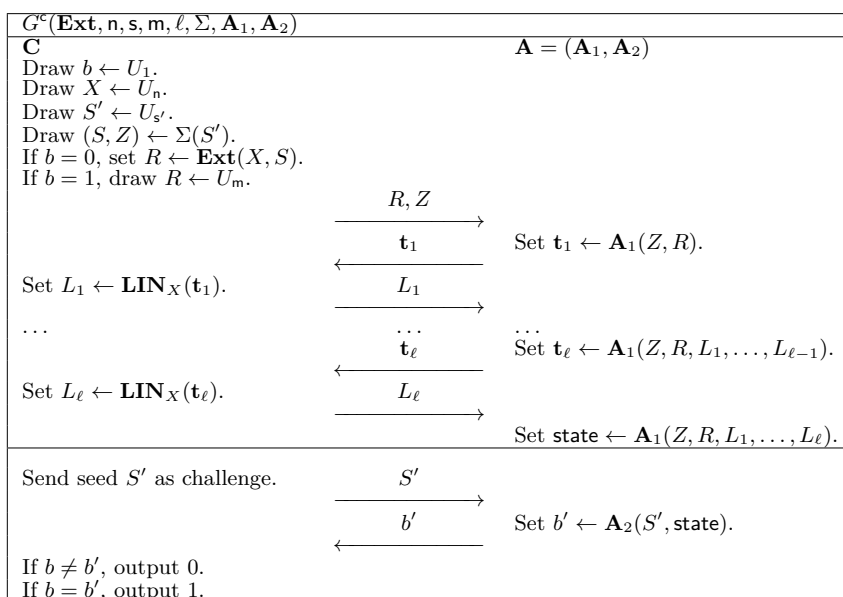
oracle in a single extractor execution. Probe complexity measures the total running time of oracle in a single extractor execution while cache complexity measures the number of cache misses incurred in a single extractor execution. Note that probe complexity may be measured as the number of non-zero columns in the query matrix produced by Pick_{Ext} . Cache complexity corresponds to the number of consecutive groups of non-zero columns found in the query matrix produced by Pick_{Ext} .

► **Definition 3.** Ext is (p, c) -local if for every seed S , $\text{Pick}_{\text{Ext}}(S)$ has at most p non-zero columns and at most c consecutive non-zero column groups.

2.3 Computational Doubly-Affine Extractors

As another advantage of doubly-affine extractors, we show that they may be built even when using seeds that are only computationally-secure. In more detail, suppose the extractor's input seed is computationally-secure with respect to leakage seen by the adversary. Is it possible for the extractor's output to remain information-theoretically random with help from the affine oracle? This is impossible for computationally unbounded adversaries with access to the oracle as the adversary may compute the seed and query the oracle to obtain the extractor's output. Instead, we want computational extractors to produce outputs that are secure against adversaries that are computationally-bounded only when the oracle is available and may become computationally-unbounded afterwards. The ability to handle computationally-secure seeds is a benefit of the affine leakage model that is impossible in general leakage models (see [15, 19]).

The security game for computational extractors is shown in Figure 2. As a major result, we will prove that the security games in Figure 1 and Figure 2 are equivalent (see Section 4). That is, every information-theoretic extractor is also a computational extractor. We consider *hybrid adversaries* $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$ where \mathbf{A}_1 is a stateful PPT adversary and \mathbf{A}_2 is a computationally unbounded adversary. The game uses a computationally-secure



■ **Figure 2** Game $G^c(\mathbf{Ext}, n, s, m, \ell, \Sigma, \mathbf{A}_1, \mathbf{A}_2)$.

protocol Σ that produces a computational seed S as well as leakage Z using a (typically) shorter random seed S' . \mathbf{A}_1 is given both the extractor’s output and the leakage Z of Σ . The role of \mathbf{A}_1 is to adaptively query the oracle \mathbf{LIN}_X to learn ℓ bits about X . \mathbf{A}_2 will use the knowledge gained by \mathbf{A}_1 as well as the original seed S' to distinguish between challenges of either uniformly random strings or extractor outputs.

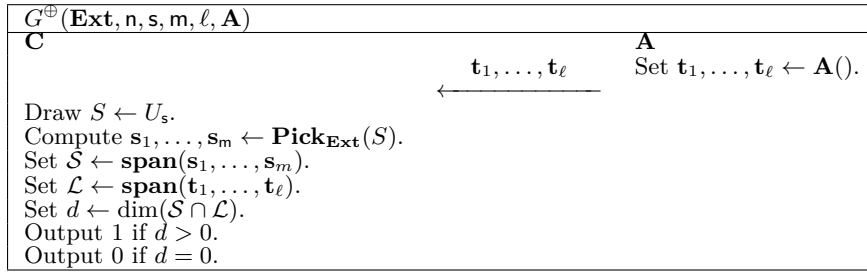
► **Definition 4.** A deterministic algorithm $\mathbf{Ext} : \{0, 1\}^n \times \{0, 1\}^s \rightarrow \{0, 1\}^m$ is $(\ell, \Sigma, \varepsilon)$ -computationally-secure if for any hybrid adversary $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$ such that \mathbf{A}_1 is PPT, $\Pr[G^c(\mathbf{Ext}, n, s, m, \ell, \Sigma, \mathbf{A}) = 1] \leq \frac{1}{2} + \varepsilon$.

3 Information-Theoretic Doubly-Affine Extractors

In this section, we present our constructions for information-theoretic doubly-affine extractors. We start by reducing the security game of doubly-affine extractors to linear algebraic concepts. Afterwards, we show that constructing doubly-affine extractors requires two simpler primitives: *samplers* and *non-local doubly-affine extractors* (or non-local extractor, for short). By presenting efficient samplers and non-local doubly-affine extractors, we obtain our final efficient extractor. We also present various lower bounds for the studied primitives.

3.1 Optimal Doubly-Affine Extractor Adversary

One of the main results in our paper is the ability to reduce the complex security game of doubly-affine extractors to a simpler game. Consider the following adversarial approach to compromise doubly-affine extractors according to the security game in Figure 1. Suppose the adversary has chosen ℓ queries to \mathbf{LIN}_X . Next, the adversary receives the seed S and computes the query matrix $\mathbf{Pick}_{\mathbf{Ext}}(S)$. Next, the adversary checks whether extractor’s output bits is a linear combination of any of the ℓ leakage bits. This is equivalent to checking whether the span of the extractor’s queries intersects the span of the adversary’s queries. In the case of an intersection, the adversary can check whether the output bit matches the



■ **Figure 3** Linear Span Game $G^\oplus(\mathbf{Ext}, n, s, m, \ell, \mathbf{A})$.

linear combination. For real challenges, this is always true. For random challenges, this is only true with probability 1/2. Therefore, the adversary has significant advantage as long as the intersection of query spans is non-empty.

We show that the above adversary is essentially optimal up to choosing the oracle queries. Formally, we prove this by showing the security game in Figure 1 is equivalent to the same simpler game in Figure 3. The game in Figure 3 severely limits the adversary by forcing the adversary to follow the above adversarial approach. The adversary must ignore both ignore the extractor output and non-adaptively query the oracle. Additionally, the adversary loses the ability to post-process the oracle results. Instead, the challenger determines the winner of the game by checking whether the intersection of the adversarial query subspace and the extractor query subspace is non-empty. We show the games in Figures 1 and 3 are identical.

As a caveat, we note that the adversary could also check whether the extractor's outputs bit are linearly independent. If any output bit is a linear combination of the other output bits, then the adversary will already win the game. For real challenges, the linearly dependent output bit must match the linear combination of the other output bits. For random challenges, this only happens 1/2 of the time. Therefore, we will assume that the extractor outputs bits will be linearly independent. In other words, the extractor's oracle queries will always be linearly independent without loss of generality.

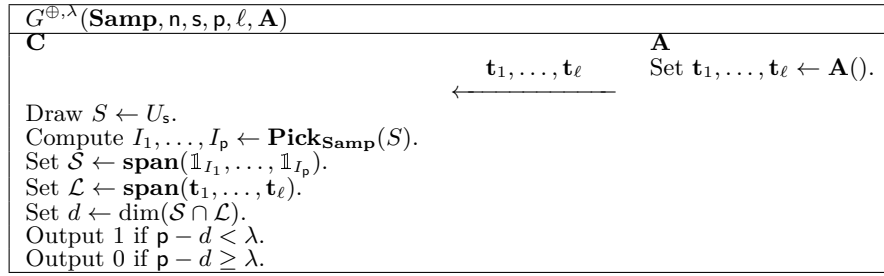
► **Theorem 5.** *Suppose that \mathbf{Ext} is (ℓ, ε) -secure with respect to security game G^\oplus . That is, for any adversary \mathbf{A} , $\Pr[G^\oplus(\mathbf{Ext}, n, s, m, \ell, \mathbf{A}) = 1] \leq \varepsilon$. Then, \mathbf{Ext} is (ℓ, ε) -secure with respect to G according to Definition 2.*

With this theorem, we already see that the main challenge for extractors is to ensure output bits are not a linear combination of the adversarial oracle queries.

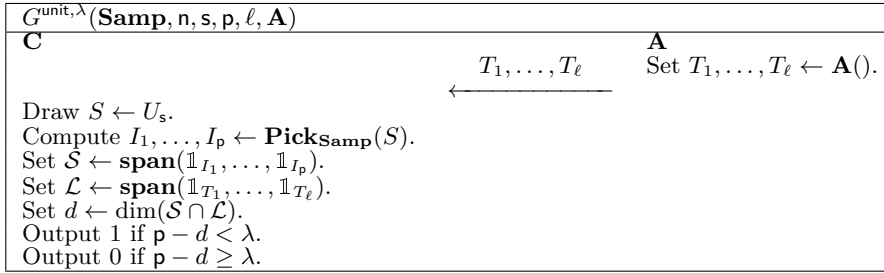
3.2 Sample-then-Extract Paradigm

To construct doubly-affine extractors, we use the sample-then-extract paradigm that was introduced by Vadhan [30]. This paradigm constructs extractors in two steps. First, a subset of the random string X is sampled such that the sample contains a large amount of entropy conditioned on the adversary's leakage. Next, a non-local doubly-affine extractor (that we will also denote as a non-local extractor) is executed on the sampled subset. The non-local extractor is expected to utilize the entirety of the sampled subset to produce as many random bits as possible (since no locality is required, they are denoted as non-local).

We now explain at a high level why the sample-then-extract algorithm results in an extractor. All the sampled bits will not be random after the adversary views leakage bits. If the adversary sees ℓ bits of the n -bit random string X , then we expect only $(1 - \ell/n)$ -fraction of the sampled bits to be random. The role of the non-local extractor is to condense the



■ **Figure 4** Relaxed Linear Span Game $G^{\oplus, \lambda}(\mathbf{Samp}, n, s, p, \ell, \mathbf{A})$.



■ **Figure 5** Relaxed Unit Span Game $G^{\text{unit}, \lambda}(\mathbf{Samp}, n, s, p, \ell, \mathbf{A})$.

mixture of random and non-random sampled bits into a smaller string of truly random bits. We will define and construct samplers and non-local extractors in the upcoming sections.

3.3 Samplers

The notion of *samplers* has been well studied in the past (see [7, 9, 24, 31, 30, 17] as some examples). Prior works studied samplers with respect to general functionalities and/or general leakage. In our work, we define samplers in a narrower manner within the affine leakage model that will be easily composable in the sample-then-extract paradigm.

Samplers are deterministic algorithms $\mathbf{Samp} : \{0, 1\}^n \times \{0, 1\}^s \rightarrow \{0, 1\}^p$ with inputs of a n -bit random string X and a s -bit seed S that outputs p sampled bits of X . The goal is to sample as many random bits as possible in the view of the adversary with leakage bits. As discussed in the previous section, it is unlikely that all sampled bits will be secure. If an adversary has ℓ random leakage bits of X , then only $(1 - \ell/n)p$ sampled bits will be secure in expectation.

For any \mathbf{Samp} , we denote $\mathbf{Pick}_{\mathbf{Samp}}$ as the queries sent to the oracle (similar to extractors). The output of \mathbf{Samp} will also be the response from the oracle. In other words, $\mathbf{Samp}(X, S) = \mathbf{LIN}_X(\mathbf{Pick}_{\mathbf{Samp}}(S))$. As \mathbf{Samp} samples bits, each column of $\mathbf{Pick}_{\mathbf{Samp}}(S)$ is a unit vector. We will use \mathbf{Samp} and $\mathbf{Pick}_{\mathbf{Samp}}$ interchangeably throughout the paper.

We relax the security game of doubly-affine extractors (Figure 3) to obtain a security game for samplers in affine leakage model presented in Figure 4. Recall that the extractor adversary should not compromise any output bits. We modify the definition for samplers so at least λ sampled bits are random in the adversary's view. We chose to immediately define samplers with respect to the optimal adversary for convenience. One could re-define sampler security using a natural game by relaxing the security of doubly-affine extractors in Figure 1.

► **Definition 6.** A deterministic algorithm $\mathbf{Samp} : \{0, 1\}^n \times \{0, 1\}^s \rightarrow \{0, 1\}^p$ is $(\ell, \varepsilon, \lambda)$ -secure if for any adversary \mathbf{A} , $\Pr[G^{\oplus, \lambda}(\mathbf{Ext}, n, s, m, \ell, \mathbf{A}) = 1] \leq \varepsilon$.

13:14 Doubly-Affine Extractors, and Their Applications

Specific to samplers, we immediately show that the adversary may immediately be weakened without loss of generality. Consider a simple adversary for samplers that also samples ℓ bits from X . If the adversary samples $\lambda + 1$ bits identical to the sampler, the adversary will distinguish the real-or-random challenge with high advantage. We prove that this adversary is optimal. In other words, sampler adversaries do not gain advantage by learning linear combinations of X as opposed to sampling single bits of X . To formalize this idea, we modify the sampler game in Figure 4 such that the adversary may only sample physical bits of X . The new game may be found in Figure 5.

► **Theorem 7.** *Suppose that **Samp** is $(\ell, \varepsilon, \lambda)$ -secure with respect to security game $G^{\text{unit}, \lambda}$. That is, for any adversary \mathbf{A} , $\Pr[G^{\text{unit}, \lambda}(\mathbf{Samp}, n, s, m, \ell, \mathbf{A}) = 1] \leq \varepsilon$. Then, **Samp** is $(\ell, \varepsilon, \lambda)$ -secure with respect to $G^{\oplus, \lambda}$ according to Definition 6.*

The above security reductions significantly simplify analyzing sampler security. Before moving on, we highlight that the majority of our efficiency gains are achieved from our improved samplers. In particular, the insight that optimal sampler adversaries in the affine leakage model are limited is the key reason as to why our doubly-affine extractors are more efficient than previous constructions in the general leakage model.

(p, c)-Local Sampler Construction. We start by presenting an efficient (p, c) -local sampler. The idea is to view the n -bit source X as a two-dimensional matrix with c rows and n/c columns and the seed $S = (S_1, \dots, S_c)$ as c integers from the set $[n/c]$ that are represented using $\log(n/c)$ bits. The sampler will sample p/c bits from each row. In the i -th row, the sampler chooses the consecutive p/c bits starting from the S_i -th column. **Samp** is (p, c) -local as a total of p bits are sampled that may be arranged into c consecutive groups (one group per row). Our construction is similar to ones presented in [15], but it was never analyzed or defined as a stand-alone sampler.

► **Theorem 8.** *For any $0 \leq \ell < n$, $c > 0$, $\varepsilon > 0$ and $\lambda > 0$, there exists a (p, c) -local sampler that is $(\ell, \varepsilon, \lambda)$ -secure with seed length $s = c \log(n/c)$ and probe complexity*

$$p = \max \left(c, \frac{\lambda}{1 - \frac{\ell}{n} - \sqrt{\frac{\ln(1/\varepsilon)}{c}}} \right).$$

We note that the efficiency of our sampler is almost optimal. Note that an adversary may always pick ℓ random bits of X as leakage. If p bits are sampled, it is expected that only $(1 - \ell/n)p$ sampled bits are secure. Our construction secure samples $\lambda = (1 - \ell/n - \sqrt{\ln(1/\varepsilon)/c})p$ bits implying that at most $(\sqrt{\ln(1/\varepsilon)/c})$ -fraction of bits are lost beyond the expectation.

(p, p)-Local Sampler Construction. For the setting where cache complexity is irrelevant, we present a (p, p) -local sampler that ends up being more concretely efficient in our full version. Our construction simply picks a subset of p bits from X uniformly at random using a seed S that encodes a random subset using $\log \binom{n}{p}$ bits.

3.4 Non-Local Doubly-Affine Extractors

In the sample-then-extract paradigm, the goal of non-local extractors is to take sampled bits containing both random and non-random bits and produce a truly random string. We abstract the setting to the original doubly-affine extractor security game by assuming that

the non-random bits are leakage viewed by the adversary. In other words, we assume that the random source X has n random bits except that ℓ bits are not random as they have been viewed by the adversary.

The simplest way to build non-local extractors is to use affine universal hash functions and the leftover hash lemma [20]. As the input string X has $n - \ell$ random bits, the leftover hash lemma states that there exists a non-local extractor that outputs $n - \ell - 2\log(1/\varepsilon)$ random bits except with probability ε .

In our work, we improve upon this result by constructing non-local extractors that produces $n - \ell - \log(1/\varepsilon)$ random bits. This reduces the lost entropy by a factor of two. The ability to build improved non-local extractors is another advantage of doubly-affine extractors. To do this, we generically reduce the construction of non-local extractors to a special family of matrices with certain properties. Consider m matrices A_1, \dots, A_m with n rows and s columns. For any non-empty subset $\emptyset \neq \mathcal{I} \subseteq \{1, \dots, m\}$, the matrix $A_{\mathcal{I}} = \sum_{i \in \mathcal{I}} A_i$ has rank n . We present an instantiation using field multiplication.

Special Matrix Family from Toeplitz Matrices. We instantiate the matrix family using Toeplitz matrices to achieve $s = n + m - 1$. Note, as $m \leq n$, the seed length is at most $s \leq 2n$. Toeplitz matrices are the matrices where all diagonals are equal. For any Toeplitz matrix T , it is always the case that $T_{i,j} = T_{i+1,j+1}$. In particular, we will use Toeplitz matrices of dimension $n \times (n + m - 1)$. We define the set T_1, \dots, T_m in the following way. T_i consists of the first $i - 1$ columns only of 0's followed by the $n \times n$ identity matrix occupying the next n columns. All remaining columns will also consist of only 0's. As an example, T_1 consists of the identity matrix occupying the first n columns followed by all 0's. Using this construction, we get that for any seed $S \in \{0, 1\}^{n+m-1}$, $T_i S = (S_i, S_{i+1}, \dots, S_{i+n-1})$. We note that the computational cost of this instantiation is $O(n \log n)$ using FFT (see [25] for more details). In the full version, we present an instantiation from field multiplication with smaller seed lengths, but higher computational costs. We use the Toeplitz matrix instantiation for practical considerations.

► **Theorem 9.** *For any subset $\emptyset \neq \mathcal{I} \subseteq [1, \dots, m]$, $T_{\mathcal{I}} = \sum_{i \in \mathcal{I}} T_i$ has rank n .*

Non-Local Extractor Construction. Our non-local extractor is built using the m matrix family A_1, \dots, A_m described above. The non-local extractor receives a s -bit seed S and a n -bit input string X with m random bits. Then, the output of the non-local extractor is $X \cdot (A_1 S), \dots, X \cdot (A_m S)$. In other words, the i -th output bit is the dot product of X and the matrix-vector multiplication of the A_i and the seed S . Our construction is similar to the one [11], but our security analysis is different, as we extract more bits in our setting.

► **Theorem 10.** *For any $0 \leq \ell < n$ and $\varepsilon > 0$, there exists a non-local extractor that is (ℓ, ε) -secure that outputs $m = n - \ell - \log(1/\varepsilon)$ bits with seed length n .*

We also present a lower bound on the number of extractable bits by non-local extractors that is tight up to an additive constant factor.

► **Theorem 11.** *Non-local extractors extract at most $n - \ell - \log(1/\varepsilon) - O(1)$ bits.*

Seed Length. While our construction extracts an almost optimal number of bits, it requires a large seed length of n . Our seed length is similar to those obtained using the leftover hash lemma resulting in seed lengths of at least $n - \ell + \log(1/\varepsilon) - O(1)$ bits [20]. In our full version, we existentially show there exists a matrix family that would result in a non-local extractor

13:16 Doubly-Affine Extractors, and Their Applications

with seed length $\log(n\ell/\varepsilon)$ while extracting the same number of bits. We also present a seed length lower bound showing that the existential construction is almost optimal. We leave it as an open question to construct such a matrix family explicitly, but remind that: (a) in our setting it is safe to expand the seed computationally (unlike general extractors [3]); (b) in the random oracle model, one can expand the seed using the random oracle as done in [5, 4]; (c) one can use theoretical extractors of [18] which are linear, have seed $O(\log n + \log(1/\varepsilon))$, but double the entropy loss to $2 \log(1/\varepsilon)$.

3.5 Doubly-Affine Extractors

With constructions of both samplers and non-local extractors, we finally construct our local extractors. First, we will formally define the sample-then-extract paradigm and show that it is secure. Afterwards, we plug in our sampler and non-local extractor constructions to obtain our efficient local extractors.

Sample-then-Extract. We formally define the sample-then-extract composition. Suppose we have a sampler **Samp** with seed length S_{Samp} that samples p bits. Note, the corresponding **Pick_{Samp}** algorithm outputs a $p \times n$ query matrix. Additionally, assume we have a non-local extractor **NLExt** with seed length S_{NLExt} that extracts from an p -bit input and produces m -bit outputs. The corresponding **Pick_{NLExt}** algorithm outputs a $m \times p$ query matrix. The resulting local extractor **Ext** is defined by its oracle query function

$$\mathbf{Pick}_{\text{Ext}}(S = (S_{\text{Samp}}, S_{\text{NLExt}})) = \mathbf{Pick}_{\text{NLExt}}(S_{\text{NLExt}})\mathbf{Pick}_{\text{Samp}}(S_{\text{Samp}}).$$

In other words, the oracle query sent by **Ext** is the matrix multiplication of the query matrices chosen by **NLExt** and **Samp**.

► **Theorem 12.** *Let **Samp** be a $(\ell, \varepsilon_1, \lambda)$ -secure sampler with seed length S_1 that samples p bits from an n -bit source and **NLExt** be a $(p - \lambda, \varepsilon_2)$ -secure non-local extractor with seed length S_2 that receives a p -bit source and outputs m bits. Then, there exists an extractor **Ext** that is $(\ell, \varepsilon_1 + \varepsilon_2)$ -secure with seed length $S_1 + S_2$ that outputs m bits. If **Samp** is (p, c) -local, then **Ext** is (p, c) -local.*

Using the above theorem, we present our constructions using our sampler and non-local extractors. The resulting extractors are more efficient than all prior works (see Section 6).

► **Theorem 13** ((p, c) -Local Extractor). *For any $0 \leq \ell < n$, $m > 0$ and $\varepsilon > 0$, there exists an (p, c) -local extractor that is (ℓ, ε) -secure with seed length $s = c \log(n/c) + p$ with probe complexity*

$$p = \max \left(c, \frac{m + \log(2/\varepsilon)}{1 - \frac{\ell}{n} - \sqrt{\frac{\ln(2/\varepsilon)}{c}}} \right)$$

for any cache complexity $c \geq \ln(2/\varepsilon)(1 - \ell/n)^{-2}$.

We note that our extractor is almost optimal in terms of randomness efficiency. Once again, we can consider a simple adversary that samples ℓ leakage bits of X randomly. For any extractor with probe complexity p , only $(1 - \ell/n)$ -fraction of the probed bits will be random in expectation. Our extractor is able to produce $m = (1 - \ell/n - \sqrt{\ln(2/\varepsilon)/c})p - \log(2/\varepsilon)$ bits that is only $(\sqrt{\ln(2/\varepsilon)/c})$ -fraction from the expectation with $\log(2/\varepsilon)$ additive bit loss.

We also present the following lower bound on cache complexity for extractors.

► **Theorem 14.** *Any (p, c) -local extractor that is (ℓ, ε) -secure with one output bit must have cache complexity $c = \Omega(\log(1/\varepsilon)/\log(np/\ell))$.*

In terms of cache complexity, we note that our extractor requires cache complexity at least $\ln(2/\varepsilon)(1 - \ell/n)^{-2}$. Our extractor's cache complexity matches the lower bound in the setting that a constant fraction of bits are leaked, $\ell = \Theta(n)$.

We also present an extractor when cache complexity is ignored ($c = p$). While both constructions are asymptotically identical, our (p, p) -local extractor is more concretely efficient. This construction is also more efficient than all previous schemes (see Section 6).

► **Theorem 15** ((p, p) -Local Extractor). *For any $0 \leq \ell < n$, $m > 0$ and $\varepsilon > 0$, there exists an (p, p) -local extractor that is (ℓ, ε) -secure with seed length $s = p \log(n/p) + p$ with probe complexity p satisfying*

$$\varepsilon \leq 2(m + \log(2/\varepsilon)) \cdot \binom{n - \ell}{m + \log(2/\varepsilon) - 1} \cdot \frac{\binom{\ell}{p - m - \log(2/\varepsilon) + 1}}{\binom{n}{p}}.$$

4 Computational Doubly-Affine Extractors

We move onto constructing extractors that are secure when using computationally-secure seeds. To refresh readers, recall that computational extractors considered security games against hybrid adversaries $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$ where \mathbf{A}_1 is a PPT adversary and \mathbf{A}_2 is computationally unbounded. \mathbf{A}_1 queries the oracle for information about the random source X using the real-or-random challenge and seed leakage. After \mathbf{A}_1 learns ℓ bit of leakage from the oracle, \mathbf{A}_2 is able to use all learned information to distinguish a real-or-random challenge. Note that \mathbf{A}_1 cannot be computationally unbounded as, otherwise, \mathbf{A}_1 could derive the original seed, query the oracle to compute the extractor's output and compare with the challenge. For the full definition, we refer readers back to Section 2.3. We re-iterate that computational seeds are not possible with general leakage [15, 19].

As a major result of our work, we show that any information-theoretic extractor is already a computational extractor. In other words, our constructions from Section 3.5 are also computational extractors with similar parameters. In particular, we prove that the security game in Figure 1 generically implies security with respect to Figure 2.

To prove this result, we will utilize insights similar to the ones presented in Theorem 5. Recall that Theorem 5 proved that after receiving the extractor seed, the optimal adversary for information-theoretic extractors simply checked whether any extractor output is a linear combination of the leakage bits received from the oracle. Note that this post-processing adversary may be executed in PPT adversary, which is the key reason why doubly-affine extractors may utilize computational seeds. This statement is not true in other general leakage models that prevent their usage of computational seeds.

► **Theorem 16.** *If \mathbf{Ext} is information-theoretically (ℓ, ε) -secure, then \mathbf{Ext} is computationally $(\ell, \varepsilon + \varepsilon')$ -secure if the computationally-secure seed generator is a PPT algorithm that is secure against PPT adversaries except with probability ε' .*

Therefore, our efficient information-theoretic extractors from Section 3.5 are also computational extractors that may be used with computational seeds.

5 Applications

In this section, we utilize our doubly-affine extractors to present information-theoretic encryption solutions. We critically leverage both restrictions of affine output and leakage in the following way. Let the random source X be the secret key. To encrypt a message M , one first samples a random seed S and produces the ciphertext $(S, \mathbf{Ext}(X, S) \oplus M)$. To decrypt a ciphertext (S, M') , one can compute the value $M' \oplus \mathbf{Ext}(X, S) = M$. In the above protocol, an adversary with access to encryption/decryption oracles end up only learning extractor outputs that are affine. As a result, the adversary obtains only affine leakage about X . So, doubly-affine extractors end up being the perfect primitive for information-theoretic encryption. While one may also use seeded extractors with general output and leakage, we show our doubly-affine extractors result in better probe/cache complexity (see Section 6).

5.1 Replicated Setting

We consider the problem with $k \geq 2$ parties that need reusable information-theoretic encryption to communicate multiple messages over potentially insecure channels. This may also be referred to as the *group communication* problem. For the case of $k = 2$, we note there is a simple stateful solution. Each party consumes the secret key X as one-time pads starting from different ends and stops when meeting in the middle. However, this solution does not scale well for $k > 2$ parties where all parties do not see all encrypted messages. The natural generalization is to split X into k parts for each party. This is sub-optimal in terms of utilizing X when some parties encrypt infrequently and other parties encrypt frequently. By using doubly-affine extractors, we avoid these issues.

IND-CCA1 Encryption. We show that the simple example presented earlier is already an IND-CCA1 secure for secret key X and messages M .

- $\mathbf{Enc}(X, M) = (S, M \oplus \mathbf{Ext}(X, S))$ with uniformly random S .
- $\mathbf{Dec}(X, (S, M')) = M' \oplus \mathbf{Ext}(X, S)$.

We prove security in a stronger variant of real-or-random challenges denoted by IND-CCA1\$. The adversary submits a challenge message M . The challenger returns either an encryption to M or a random string that must be distinguished by the adversary. Note, this requires that ciphertexts are indistinguishable from random strings.

IND-CCA2 Authenticated Encryption. We present the first application of our computational doubly-affine extractors for constructing IT authenticated encryption (AE) that is IND-CCA2\$ secure. We assume the existence of a standard, computationally-secure AE with associated data (AEAD) scheme, $\mathbf{Enc}_{\text{AEAD}}$ and $\mathbf{Dec}_{\text{AEAD}}$. We assume that the secret key is (X, K_{AEAD}) where K_{AEAD} is an AE secret key.

- $\mathbf{Enc}((X, K_{\text{AEAD}}), M) = (\mathbf{Enc}_{\text{AEAD}}(K_{\text{AEAD}}, S, M'), M')$ with $M' = M \oplus \mathbf{Ext}(X, S)$ and uniformly random S .
- $\mathbf{Dec}((X, K_{\text{AEAD}}), (S', M')) = M' \oplus \mathbf{Ext}(X, \mathbf{Dec}_{\text{AEAD}}(K_{\text{AEAD}}, S', M'))$ and rejecting whenever $\mathbf{Dec}_{\text{AEAD}}(K_{\text{AEAD}}, S', M')$ fails authenticity verification.

It is straightforward to adapt our scheme to an IT AEAD by appending associated data with M' in the above scheme. Our construction also achieves authenticity against any PPT adversaries from the underlying AEAD scheme.

For security, we adapt our real-or-random challenge from the prior sections where the adversary must distinguish between an encryption of an adversarially chosen message and a random string. For IND-CCA2\$, the adversary is able to make both encryption and decryption queries after the challenge. A computationally unbounded adversary in IND-CCA2\$ may trivially break the scheme. For the challenge ciphertext (S', M') , the adversary sends a decryption query for ciphertext (S', M'') where $M'' \neq M'$ to receive $M'' \oplus \mathbf{Ext}(X, S)$ where S is the encrypted seed. Therefore, the adversary computes $\mathbf{Ext}(X, S)$ and may break the scheme. Due to this limitation, we consider adversaries that are PPT when encryption/decryption queries are available but computationally unbounded afterwards.

5.2 Distributed Setting

We consider another setting where the random source X is jointly stored by $t \geq 2$ servers. Each server stores a disjoint subset of X . We will use the standard assumption from information-theoretic cryptography literature where the user will have a private channel with only a subset of $g < t$ trusted servers and the remaining $t - g$ channels may be compromised and/or the servers may be compromised and using faulty randomness. This is a common assumption that appears in many prior important works such as information-theoretic secret sharing [27], multi-party computation [8] and secure message transmission [14]. Note users do not know which servers are compromised. For simplicity, we assume each server stores an equal portion of X (our analysis is trivial to extend when this is not the case). If there are $t - g$ bad servers, a $(t - g)/t$ fraction of X will be leaked. Adversaries may also query the servers to learn affine leakage about the random source X .

We modify our extractors for the multi-server setting. Recall that our extractors first sample bits then apply a non-local extractor on sampled bits. With multiple servers, our extractor first executes the sampler portion with each server individually. Afterwards, the non-local extractor will be executed on all sampled bits to obtain the final extractor output. In terms of efficiency, we note that the virtual random source has only $n := (g/t)|X|$ random bits excluding parts of X stored by the bad servers. Any (p, c) -local extractor will probe $(g/t)p$ bits from good servers and the remaining sampled bits are assumed to be compromised already. Executing a non-local extractor over all sampled bits can output $(g/t)p - \log(1/\varepsilon)$ random bits. By considering the multi-server setting, we must increase the probe complexity of our extractors by a multiplicative factor of (t/g) . In particular, consider any extractor with probe complexity p in the replicated setting that outputs m bits. To obtain m output bits in the multi-server setting, our extractors must instead probe $(t/g)p$ bits.

We emphasize that our doubly-affine extractors are easily amenable to the distributed setting. In particular, the distributed servers do not need any communication or coordination. The seeds that are sent to each server may be computationally-secure. To encrypt/decrypt a message M , the communication and computation from each server is sub-linear in the length of M . Our encryption schemes are easily adaptable to settings where servers may go offline and new servers join as well if servers are malicious. Most of these great properties are not be obtainable by other primitives when moving to distributed settings (such as MPC).

As one requirement, servers must limit the leakage that may be obtained by an adversary. To do this, the servers may keep track of the number of unique bits that are sampled by all users. To limit adversaries to ℓ leakage bits, the servers should stop responding after ℓ/g unique queries. As a result, the g servers return at most ℓ leakage bits. Note that keeping track of the unique sampled bits may be done very efficiently with small storage (using HyperLogLog [21] as an example).

13:20 Doubly-Affine Extractors, and Their Applications

Public-Key IND-CCA2 Encryption. With the server setting, there may be a need for public-key encryption as parties no longer share the secret key. We show that we may build public-key IND-CCA2 encryption schemes using our computational doubly-affine extractors. We re-use our previous IND-CCA2 definition with the only difference that the adversary obtains the public key before any encryption/decryption queries.

We will use any public-key encryption (PKE) scheme $(\mathbf{Enc}_{\mathbf{PK}}, \mathbf{Dec}_{\mathbf{PK}})$ with IND-CCA2 security against PPT adversaries with label support [29] where the label is not private but is required for decryption. Suppose that the PKE has key pair $(\mathbf{pk}, \mathbf{sk})$ and $\mathbf{Ext}(X, S)$ is computed using distributed variant of our doubly-affine extractors.

- $\mathbf{Enc}(X, \mathbf{pk}, M) = (\mathbf{Enc}_{\mathbf{PK}}(\mathbf{pk}, S, M'), M')$ with $M' = M \oplus \mathbf{Ext}(X, S)$ and uniformly random S .
- $\mathbf{Dec}(X, \mathbf{sk}, (S', M')) = M' \oplus \mathbf{Ext}(X, \mathbf{Dec}_{\mathbf{PK}}(\mathbf{sk}, S', M'))$.

We formally define and prove security in our full version.

Computationally-Secure Keys. In the server setting, parties no longer need to meet and share the secret key as they may generate shared randomness through the servers. We show that two parties may utilize information-theoretic encryption schemes without meeting. First, the two parties use key exchange to agree on a computationally-secure seed S . To encrypt a message M , we may use any of the prior constructions where $\mathbf{Ext}(X, S)$ is computed using the servers.

Malicious Servers. In this last section, we consider when the $t - g$ corrupted servers are malicious. Malicious servers are able to answer in an arbitrary manner including no response, wrong responses and responses that are inconsistent across multiple requests. As an example, malicious servers may ignore sampler seeds and return random bits for each request. Therefore, malicious servers can force our extractor outputs to no longer be deterministic. For two queries with the same seeds, the outputs might be different that will be problematic for many applications.

In this section, we present a solution to this problem. The main modification is to utilize Reed-Solomon codes to handle both errors (servers returning wrong responses) and erasures (servers not returning any response). Using a Reed-Solomon code that adds z check bits, the decoding algorithms may handle up to a errors and b erasures such that $2a + b \leq \lfloor z/2 \rfloor$. All \mathbf{p} sampled bits will be encoded using a Reed-Solomon code. If there are at $t - g$ malicious servers, that means at most $(t - g)/t \cdot \mathbf{p}$ errors and/or erasures may occur in the sampled bits. Therefore, we choose to use a Reed-Solomon code with $z \geq 2(t - g)/t \cdot \mathbf{p}$ check bits. In applications, multiple parties will have to share both the same seeds as well as the z check bits to guarantee the same output.

As the z check bits generated by the Reed-Solomon code must be shared between multiple parties, we consider the z check bits to be public and, thus, available to the adversary. One reason we chose to use Reed-Solomon code is that that check bits are linear in the message and, thus, linear in the sampled bits. So, we may consider the leaked z check bits as general linear leakage obtained by the adversary. This requires several modifications to \mathbf{DExt} . Note that our extractors require that the sampled bits from the g good servers have $\mathbf{m} + \log(2/\varepsilon)$ bits of residual entropy as the non-linear extractor will lose $\log(2/\varepsilon)$ bits of entropy. For malicious server setting, the z check bits will be publicly released. Therefore, we need the sampled bits from the good servers to have residual entropy of $\mathbf{m} + z + \log(2/\varepsilon)$ bits as we will lose z bits of entropy for the public check bits and another $\log(2/\varepsilon)$ from the non-linear extractor. Additionally, at most $t/3$ of the servers may be malicious.

■ **Table 1 Numerical examples, comparison with [5].** Both tables consider 100 GB random sources and $\varepsilon = 2^{-64}$. The left table considers 10% leakage and the right table 50% leakage. The leftmost columns denote the probe complexity p . The second and third column denote the number of random bits extracted.

p	Ours	[5]	p	Ours	[5]
500	309	210	500	82	20
1000	734	484	1000	282	104
2000	1638	1031	2000	721	272
4000	3399	2127	4000	1723	608
279	128	88	621	128	40
436	256	174	934	256	93
742	512	342	1527	512	192
351	188	128	1142	343	128
584	381	256	1903	678	256
1052	775	512	3426	1452	512

6 Experimental Evaluation

We now analyze the efficiency of our extractors in several dimensions. We compare our extractors with those that appeared in previous works. The majority of previous works such as [15, 30] focused on asymptotic as opposed to concrete efficiency. We focus on two recent works that present concretely efficient (p, p) -local extractors [5] and (p, c) -local extractors [4]. We caveat that these extractors were built for general leakage as opposed to only linear leakage like our extractors. Therefore, we expect our constructions to be more efficient.

Both [5] and [4] assume the existence of a random oracle. The random oracle is utilized for both seed generation and non-local extraction. To make a fair comparison with our doubly-affine extractors, we replace random oracles with our non-local extractor (see Theorem 10) and require that the requisite seed length is provided as input. With these modifications, our constructions have smaller or identical seed lengths. The (p, c) -local extractor of [4] uses seeds of length $p \log(n/c)$ that is larger than our extractor’s seed length of $c \log(n/c)$ (Theorem 13). Our (p, p) -local extractors in Theorem 15 use $\log \binom{n}{p}$ seeds that are identical to the ones used in [5]. We present numerical comparisons in Tables 1 and 2 verifying that our extractors are more efficient. For our comparison, we use security parameter $\varepsilon = 2^{-64}$. In all settings, our constructions extract more bits compared to both previous works when using the same probe and/or cache complexity.

7 Conclusions

In this paper, we define the notion of *doubly-affine extractors* where both the output and leakage must be affine. Using these two restrictions, we present a series of reductions showing that optimal doubly-affine extractors end up being simple PPT algorithms that check intersection of linear subspaces. Using these insights, we show that doubly-affine extractors may tolerate *post-application leakage* and *computational seeds*, which are impossible in general leakage models. We present extractor constructions that are almost concretely optimal in terms of randomness usage, probe and cache complexity beating prior works. Finally, we show that the doubly-affine restrictions are perfect for the application of information-theoretic encryption. Using our concretely efficient extractors, we obtain state-of-the-art information-theoretic encryption.

■ **Table 2 Numerical examples, comparison with [4].** Both tables consider 100 GB random sources and $\varepsilon = 2^{-64}$. The left table considers 10% leakage and the right table 50% leakage. The leftmost column denotes the cache complexity c . The first row denotes the number of probed bits in each of the c groups. The remaining entries denote the number of random bits extracted.

c	1		8		32		64		512	
	Ours	[4]	Ours	[4]	Ours	[4]	Ours	[4]	Ours	[4]
250	53	53	885	524	3738	695	7542	723	60796	748
500	234	189	2334	1130	9532	1471	19129	1572	153488	1575
1000	622	463	5436	2343	21942	3024	43950	3134	352057	3229
5000	3690	2655	31237	12048	128746	15445	257558	15994	2060924	16464
10000	8263	5395	66565	24178	266455	30972	532976	32068	4264226	33008

c	1		8		32		64		512	
	Ours	[4]	Ours	[4]	Ours	[4]	Ours	[4]	Ours	[4]
250	0	0	85	90	538	146	1142	157	9596	168
500	34	0	734	262	3132	374	6329	396	51088	417
1000	222	84	2236	608	9142	831	18350	874	147257	914
5000	1960	757	16137	3371	64746	4482	129558	4697	1036924	4892
10000	4263	1598	34565	6825	138455	9046	276976	9475	2216266	9864

References

- 1 Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO 2009*, 2009.
- 2 Yonatan Aumann, Yan Zong Ding, and Michael O Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 2002.
- 3 Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, François-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. In *Annual Cryptology Conference*, pages 1–20. Springer, 2011.
- 4 Mihir Bellare and Wei Dai. Defending against key exfiltration: Efficiency improvements for big-key cryptography via large-alphabet subkey prediction. In *CCS*, 2017.
- 5 Mihir Bellare, Daniel Kane, and Phillip Rogaway. Big-key symmetric encryption: Resisting key exfiltration. In *CRYPTO*, 2016.
- 6 Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT*, 2000.
- 7 Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *FOCS'94*, 1994.
- 8 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10, 1988.
- 9 Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 1995.
- 10 Yevgeniy Dodis. Shannon impossibility, revisited. In *ICITS*, 2012.
- 11 Yevgeniy Dodis, Ariel Elbaz, Roberto Oliveira, and Ran Raz. Improved randomness extraction from two independent sources. In *APPROX-RANDOM*, 2004.
- 12 Yevgeniy Dodis, Bhavana Kanukurthi, Jonathan Katz, Leonid Reyzin, and Adam Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. *IEEE Transactions on Information Theory*, 58(9):6207–6222, 2012.
- 13 Yevgeniy Dodis and Kevin Yeo. Doubly-affine extractors, and their applications. Cryptology ePrint Archive, Report 2021/637, 2021. URL: <https://eprint.iacr.org/2021/637>.

- 14 Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *Journal of the ACM (JACM)*, 40(1):17–47, 1993.
- 15 Stefan Dziembowski and Ueli Maurer. Tight security proofs for the bounded-storage model. In *STOC*, 2002.
- 16 Ariel Gabizon and Ran Raz. Deterministic extractors for affine sources over large fields. *Combinatorica*, 2008.
- 17 Oded Goldreich. A sample of samplers: A computational perspective on sampling. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Springer, 2011.
- 18 Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *Journal of the ACM (JACM)*, 56(4):1–34, 2009.
- 19 Danny Harnik and Moni Naor. On everlasting security in the hybrid bounded storage model. In *International Colloquium on Automata, Languages, and Programming*, pages 192–203. Springer, 2006.
- 20 Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 1999.
- 21 Stefan Heule, Marc Nunkesser, and Alexander Hall. Hyperloglog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm. In *EDBT*, 2013.
- 22 Chi-Jen Lu. Encryption against storage-bounded adversaries from on-line strong extractors. *Journal of Cryptology*, 2004.
- 23 Ueli M Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 1992.
- 24 Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 1996.
- 25 Martin Ohsmann. Fast transforms of toeplitz matrices. *Linear algebra and its applications*, 231:181–192, 1995.
- 26 Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 2000.
- 27 Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
- 28 Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 1949.
- 29 Victor Shoup and Rosario Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 2002.
- 30 Salil P Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *Journal of Cryptology*, 17(1):43–77, 2004.
- 31 David Zuckerman. Randomness-optimal oblivious sampling. *Random Structures & Algorithms*, 1997.

Online Linear Extractors for Independent Sources

Yevgeniy Dodis

New York University, New York City, NY, USA

Siyao Guo

New York University Shanghai, Shanghai, China

Noah Stephens-Davidowitz

Cornell University, Ithaca, NY, USA

Zhiye Xie

New York University Shanghai, Shanghai, China

Abstract

In this work, we characterize *linear online extractors*. In other words, given a matrix $A \in \mathbb{F}_2^{n \times n}$, we study the convergence of the iterated process $\mathbf{S} \leftarrow A\mathbf{S} \oplus \mathbf{X}$, where $\mathbf{X} \sim D$ is repeatedly sampled independently from some fixed (but unknown) distribution D with (min)-entropy k . Here, we think of $\mathbf{S} \in \{0, 1\}^n$ as the *state* of an online extractor, and $\mathbf{X} \in \{0, 1\}^n$ as its input.

As our main result, we show that the state \mathbf{S} converges to the uniform distribution for all input distributions D with entropy $k > 0$ if and only if the matrix A has no non-trivial invariant subspace (i.e., a non-zero subspace $V \subsetneq \mathbb{F}_2^n$ such that $AV \subseteq V$). In other words, a matrix A yields a linear online extractor if and only if A has no non-trivial invariant subspace. For example, the linear transformation corresponding to multiplication by a generator of the field \mathbb{F}_{2^n} yields a good linear online extractor. Furthermore, for any such matrix convergence takes at most $\tilde{O}(n^2(k+1)/k^2)$ steps.

We also study the more general notion of *condensing* – that is, we ask when this process converges to a distribution with entropy at least ℓ , when the input distribution has entropy at least k . (Extractors corresponding to the special case when $\ell = n$.) We show that a matrix gives a good condenser if there are relatively few vectors $\mathbf{w} \in \mathbb{F}_2^n$ such that $\mathbf{w}, A^T\mathbf{w}, \dots, (A^T)^{n-k}\mathbf{w}$ are linearly dependent. As an application, we show that the very simple cyclic rotation transformation $A(x_1, \dots, x_n) = (x_n, x_1, \dots, x_{n-1})$ condenses to $\ell = n - 1$ bits for any $k > 1$ if n is a prime satisfying a certain simple number-theoretic condition.

Our proofs are Fourier-analytic and rely on a novel lemma, which gives a tight bound on the product of certain Fourier coefficients of any entropic distribution.

2012 ACM Subject Classification Theory of computation \rightarrow Expander graphs and randomness extractors; Mathematics of computing \rightarrow Information theory

Keywords and phrases feasibility of randomness extraction, randomness condensers, Fourier analysis

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.14

Funding *Yevgeniy Dodis*: Partially supported by gifts from VMware Labs, Facebook and Google, and NSF grants 1314568, 1619158, 1815546.

Siyao Guo: Supported by Shanghai Eastern Young Scholar Program SMEC-0920000169. Parts of this work were done while visiting the Centre for Quantum Technologies, National University of Singapore.

Noah Stephens-Davidowitz: Some of this work was done at MIT supported in part by NSF Grants CNS-1350619, CNS-1414119 and CNS1718161, Microsoft Faculty Fellowship and an MIT/IBM grant. Some of this work was done at the Simons Institute in Berkeley.

1 Introduction

An *extractor* is a deterministic algorithm that takes input $\mathbf{X} \sim D$ sampled from some sufficiently nice distribution D and outputs nearly uniformly random $\mathbf{Y} \in \{0, 1\}^n$. An *online extractor* is a deterministic algorithm with a state $\mathbf{S} \in \{0, 1\}^n$ that takes inputs $\mathbf{X}_1 \sim D_1, \mathbf{X}_2 \sim D_2, \dots, \mathbf{X}_m \sim D_m$ one at a time, updating its state after each input. We



© Yevgeniy Dodis, Siyao Guo, Noah Stephens-Davidowitz, and Zhiye Xie; licensed under Creative Commons License CC-BY 4.0

2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 14; pp. 14:1–14:14



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

14:2 Online Linear Extractors for Independent Sources

say that it *extracts* from D_1, \dots, D_m if the state \mathbf{S} is statistically close to random at the end of this process. This naturally models the idea of “gradually accumulating entropy” from entropic sources.

We are interested in perhaps the simplest possible setting, when the $D_i = D$ are independent and identical but otherwise arbitrary entropic distributions over $\{0, 1\}^n$, and when the extractor is *linear* (over \mathbb{F}_2). In other words, on input $\mathbf{X} \in \{0, 1\}^n$, the state $\mathbf{S} \in \{0, 1\}^n$ is updated by the procedure

$$\mathbf{S} \leftarrow A\mathbf{S} \oplus \mathbf{X}$$

for some fixed linear transformation $A \in \mathbb{F}_2^{n \times n}$.

We then ask the natural question

Which matrices $A \in \mathbb{F}_2^{n \times n}$ are good extractors?

In other words, for which matrices A does the process $\mathbf{S} \leftarrow A\mathbf{S} \oplus \mathbf{X}$ always converge to uniform when \mathbf{X} is sampled independently from *any* distribution with non-zero entropy?

We first notice that there is a natural obstruction that prevents some matrices $A \in \mathbb{F}_2^{n \times n}$ from extracting. As an illustrative example, suppose that A is the “rotation” map defined by $A(x_1, \dots, x_n) = (x_n, x_1, x_2, \dots, x_{n-1})$. Then, A clearly fails to extract from the uniform distribution over $\{0^n, 1^n\}$.

More generally, suppose that there exists a subspace $V \subset \mathbb{F}_2^n$ with dimension $0 < \dim(V) < n$ such that $AV \subseteq V$. Such a subspace is called a *non-trivial invariant subspace*. (The trivial invariant subspaces are $\{0^n\}$ and \mathbb{F}_2^n .) Then, if X is sampled from the uniform distribution over V , it is not hard to see that the distribution of the state \mathbf{S} will itself remain uniform over V after each run of the extractor $\mathbf{S} \leftarrow A\mathbf{S} \oplus \mathbf{X}$. (Here and elsewhere, we assume without loss of generality that the starting state is 0^n .) So, A completely fails to extract from this distribution, even though it clearly has (min-)entropy.

Our main theorem is a proof that this is the only obstruction, i.e., that a matrix A extracts from all entropic distributions *if and only* if A has no non-trivial invariant subspace. In fact, we show that this property implies that A extracts after relatively few samples, just $\tilde{O}(n^2(k+1)/k^2)$ samples. (Notice that n/k samples is the best that one could possibly hope for.)

► **Theorem 1** (Informal, see Theorems 10 and 11). *A matrix $A \in \mathbb{F}_2^{n \times n}$ extracts from arbitrary entropic distributions if and only if A has no non-trivial invariant subspace.*

Specifically, if A has no non-trivial invariant subspace and the input has min-entropy $k > 0$, then the distribution of the state will be 2^{-n} -close to uniform after $m \leq O(n^2(k+1)/k^2 \cdot \log(2n/k))$ steps.

We note that, while the property of having a non-trivial invariant subspace might seem rather opaque, it is efficiently checkable: A has no non-trivial invariant subspace (and thus is a good extractor) if and only if its characteristic polynomial is irreducible [8]. Moreover, there are very sparse matrices A having this property. For example, if A is the linear transformation corresponding to multiplication by a generator of the finite field \mathbb{F}_{2^n} , then A is a good extractor which can be easily implemented in time $O(n)$.¹ Thus, we show very simple linear-time, online linear extractors that work for any (unknown) distribution with non-zero min-entropy.

¹ Indeed, multiplication by the generator corresponds to one cyclic rotation and one conditional XOR with a fixed string corresponding to the coefficients of the irreducible polynomial generating the field.

Our proof of Theorem 1 is Fourier-analytic: the main technical tool is a novel lemma (Lemma 7) concerning certain products of Fourier coefficients of distributions with entropy k . Specifically, for linearly independent $\mathbf{w}_1, \dots, \mathbf{w}_r \in \mathbb{F}_2^n$, we give a tight bound on the product of the associated Fourier coefficients. (The worst case is essentially a linear transformation of the uniform distribution over a Hamming ball.)

Online linear condensers

We also consider a more general question. Recall that a *condenser* is a deterministic algorithm that takes as input $\mathbf{X} \sim D$ sampled from a sufficiently nice distribution and outputs $\mathbf{Y} \in \{0, 1\}^n$ that has relatively large entropy (but is not necessarily close to uniform). In our setting, we are interested in the following question.

For which matrices A does the process $\mathbf{S} \leftarrow A\mathbf{S} \oplus \mathbf{X}$ converge to a distribution with at least ℓ bits of entropy, whenever X is sampled independently from some (unknown) distribution with more than k bits of entropy?

Notice that our extractor question from above corresponds to the special case when $k = 0$ and $\ell = n$.

Here, our result is necessarily a bit more complicated (though the proof is simple and uses the same Fourier-analytic tools). Specifically, we define the *A-rank* of a vector $\mathbf{w} \in \mathbb{F}_2^n$ as the dimension of the subspace spanned by $\mathbf{w}, A\mathbf{w}, \dots, A^{n-1}\mathbf{w}$. Notice that a matrix A has a non-trivial invariant subspace if and only if there is a non-zero vector $\mathbf{w} \in \mathbb{F}_2^n$ with *A-rank* less than n – so that this notion of *A-rank* is naturally related to the idea of non-trivial invariant subspaces discussed above. And, notice that the obstruction that we ran into with rotation arose from the existence of the vector 1^n with rank equal to 1, which can cause our condenser to “get stuck at one bit of entropy.” There is a similar obstruction caused by the uniform distribution over the subspace orthogonal to 1^n (i.e., the subspace of vectors with even Hamming weight) that can cause our condenser to “get stuck at $n - 1$ bits of entropy.”

More generally, a vector with *A-rank* r means that “we can get stuck on distributions with entropy r or entropy $n - r$.” So, if we are going to condense from k bits to ℓ bits, we must have $k > \min\{n - r, r\}$ and $\ell \leq \max\{r, n - r\}$.

We prove that low-rank vectors are essentially the only possible obstruction to condensing. In particular, a matrix A is a good condenser if it has a small number of vectors with small *A-rank*. (Again, while this might seem rather opaque, it is easy to count the vectors with a given *A-rank* by computing the characteristic and minimal polynomials of A [8].) In fact, for technical reasons, it is more natural to study vectors with low A^T -rank, rather than vectors with low *A-rank*. (Since A^T and A have the same characteristic and minimal polynomials, *A-rank* and A^T -rank are closely related.)

► **Theorem 2** (Informal, see Theorem 15). *For any invertible $A \in \mathbb{F}_2^{n \times n}$, if there are at most N vectors in $\{0, 1\}^n$ with A^T -rank less than r , then A condenses any distribution with $k > g := n - r$ bits of min-entropy to a distribution with at least $\ell = n - \log_2 N$ bits of min-entropy. In particular, the state will have entropy at least $\ell - 2^{-n}$ after $m = \tilde{O}(n^2(k - g + 1)/(k - g)^2)$ steps.*

As an application, we show that rotation does in fact condense from $k > 1$ bits of entropy to $n - 1$ bits – and that it only requires $m = \tilde{O}(n^2k/(k - 1)^2)$ steps to do so – when n is a prime satisfying a simple number-theoretic condition.

1.1 Related work

To the best of our knowledge, our question of linear extractors from independent, identically distributed (IID) sources was not explicitly considered by prior work, but several works considered somewhat related models.

The closest such model is our recent prior work [11], which was motivated by a very practical question of analyzing the bit-level complexity of fast entropy accumulation in real-world random number generators (RNGs), such as the Fortuna RNG used by Windows 10 [13]. That work also studied linear online extractors, but only for a specific class of natural distributions that arise in practice and only for hyper-efficient linear transformations A that simply permute the bits of the state. Indeed, in [11], we were primarily concerned with the practical question of optimizing the exact number of samples needed to extract from such distributions for fixed $n \in \{32, 64\}$ using these extremely fast linear transformations.² From a technical point of view, both works use Fourier-analytic techniques, but the details are quite different. The main Fourier-analytic tool in [11] is a bound on the Fourier coefficients of the class of natural distributions that we study there. Here, our main tool is Lemma 7, which applies to arbitrary entropic distributions.

Starting with Chor and Goldreich [7], many papers (see [2, 15, 6] and references therein) studied the much harder question of randomness extraction from several independent (but *not* identical) arbitrary entropic sources. Unlike our work, these extractors *cannot* be linear, and, to the best of our knowledge, no online extractor is known to extract from this general class of sources. However, if one sufficiently restricts the distribution family to be more structured, online extraction is sometimes possible – even by extremely efficient functions. For example, the classical work of Santha and Vazirani [16] showed that simply applying bit-wise XOR is a good extractor for independent (but not necessarily identical) SV-sources. In fact, in some cases online extraction becomes possible even without assuming independence, as long as each new source comes from certain very structured family conditioned on the previous sources [4, 3].

The classical work of von Neumann [18] studied the question of randomness extraction from IID coin flips with an a-priori unknown bias, and his extractor happened to be online. Elias [12] improved the rate of von Neumann’s extractor, but sacrificed the online property to do so.

The works of [9, 10] explicitly considered online extractors in various idealized computational models (such as the random oracle model). These extractors are highly non-linear.

In the setting of so-called “seeded extractors”, where an additional random seed is available for extraction, the power of simple, linear extractors goes back to the leftover hash lemma [14], and the streaming analog of this question (corresponding to a very long source X) was studied by [1].

² In contrast, we are interested in the more theoretical question of extracting from arbitrary entropic sources with arbitrary n . In exchange for this generality, we sacrifice the extreme efficiency achieved in [11] (which was the primary goal of that work). Indeed, in [11] we show that very efficient linear transformations A can extract from a natural class of sources in just a bit more than n/k steps, while it is easy to see that $n - k$ steps are necessary for a linear online extractor to extract from arbitrary entropic sources. Indeed, all of the different linear transformations that we considered in [11] are conjugates of rotation, and are therefore equivalent in our setting of arbitrary entropic sources, while in the model of [11] their convergence rates are quite different. (In [11], we were also happy to converge to at most, e.g., $n - \varepsilon$ bits of entropy, while here we are interested in asymptotic convergence.)

2 Preliminaries

2.1 Entropy and statistical distance

For an integer $n \geq 1$, we write $[n] := \{0, \dots, n-1\}$. For a distribution D over $\{0, 1\}^n$ and $\mathbf{x} \in \{0, 1\}^n$, we write $D(\mathbf{x}) := \Pr_{\mathbf{X} \sim D}[\mathbf{X} = \mathbf{x}]$ for the probability that D assigns to \mathbf{x} . The statistical distance between two distributions D_1 and D_2 over $\{0, 1\}^n$ is

$$\text{SD}(D_1, D_2) := \frac{1}{2} \cdot \sum_{\mathbf{x} \in \{0, 1\}^n} |D_1(\mathbf{x}) - D_2(\mathbf{x})|.$$

We say D_1 is ε -close to D_2 if $\text{SD}(D_1, D_2) \leq \varepsilon$. The *min-entropy* of D is

$$H_{\min}(D) := \min_{x \in \{0, 1\}^n} \log_2(1/D(x)).$$

2.2 Basic Fourier analysis

For a distribution D over $\{0, 1\}^n$ and $\mathbf{w} \in \{0, 1\}^n$, we define the Fourier coefficient of D at \mathbf{w} as

$$\widehat{D}(\mathbf{w}) := \mathbb{E}_{\mathbf{X} \sim D} [(-1)^{\langle \mathbf{X}, \mathbf{w} \rangle}] = \Pr_{\mathbf{X} \sim D} [\langle \mathbf{X}, \mathbf{w} \rangle = 0 \pmod{2}] - \Pr_{\mathbf{X} \sim D} [\langle \mathbf{X}, \mathbf{w} \rangle = 1 \pmod{2}].$$

▷ **Claim 3.** For any distribution D over $\{0, 1\}^n$,

$$H_{\min}(D) \geq n - \log_2 \left(\sum_{\mathbf{w} \in \{0, 1\}^n} |\widehat{D}(\mathbf{w})| \right)$$

and

$$\text{SD}(D, U) \leq \frac{1}{2} \sum_{\mathbf{w} \in \{0, 1\}^n, \mathbf{w} \neq \mathbf{0}} |\widehat{D}(\mathbf{w})|,$$

where U is the uniform distribution over $\{0, 1\}^n$.

Proof. Recall that for any $\mathbf{x} \in \{0, 1\}^n$,

$$D(\mathbf{x}) = \frac{1}{2^n} \sum_{\mathbf{w} \in \{0, 1\}^n} \widehat{D}(\mathbf{w}) (-1)^{\langle \mathbf{x}, \mathbf{w} \rangle} \leq \frac{1}{2^n} \sum_{\mathbf{w} \in \{0, 1\}^n} |\widehat{D}(\mathbf{w})|.$$

Therefore,

$$H_{\min}(D) = \min_{x \in \{0, 1\}^n} \log_2(1/D(\mathbf{x})) \geq n - \log_2 \left(\sum_{\mathbf{w} \in \{0, 1\}^n} |\widehat{D}(\mathbf{w})| \right).$$

Moreover, note that $\widehat{D}(\mathbf{0}) = 1$,

$$\left| D(\mathbf{x}) - \frac{1}{2^n} \right| = \left| \frac{1}{2^n} \sum_{\mathbf{w} \in \{0, 1\}^n, \mathbf{w} \neq \mathbf{0}} \widehat{D}(\mathbf{w}) (-1)^{\langle \mathbf{x}, \mathbf{w} \rangle} \right| \leq \frac{1}{2^n} \sum_{\mathbf{w} \in \{0, 1\}^n, \mathbf{w} \neq \mathbf{0}} |\widehat{D}(\mathbf{w})|.$$

Therefore,

$$\text{SD}(D, U) = \frac{1}{2} \sum_{\mathbf{x} \in \{0, 1\}^n} \left| D(\mathbf{x}) - \frac{1}{2^n} \right| \leq \frac{1}{2} \cdot 2^n \cdot \left(\frac{1}{2^n} \sum_{\mathbf{w} \in \{0, 1\}^n, \mathbf{w} \neq \mathbf{0}} |\widehat{D}(\mathbf{w})| \right) = \frac{1}{2} \sum_{\mathbf{w} \in \{0, 1\}^n, \mathbf{w} \neq \mathbf{0}} |\widehat{D}(\mathbf{w})|. \triangleleft$$

14:6 Online Linear Extractors for Independent Sources

The Fourier coefficients arise naturally in our context because they interact nicely with both convolution and linear transformations, as this next well-known claim shows.

▷ **Claim 4.** For distributions D_1, \dots, D_m over $\{0, 1\}^n$ and linear transformations $A_1, \dots, A_m \in \mathbb{F}_2^{n \times n}$, let D be the distribution given by

$$\Pr_{\mathbf{X} \sim D}[\mathbf{X} = \mathbf{x}] = \Pr_{\mathbf{X}_1 \sim D_1, \dots, \mathbf{X}_m \sim D_m}[A_1 \mathbf{X}_1 \oplus \dots \oplus A_m \mathbf{X}_m = \mathbf{x}],$$

where the \mathbf{X}_i are independent. Then,

$$\widehat{D}(\mathbf{w}) = \widehat{D}_1(A_1^T \mathbf{w}) \cdots \widehat{D}_m(A_m^T \mathbf{w})$$

for any $\mathbf{w} \in \{0, 1\}^n$.

Proof. We have

$$\begin{aligned} \mathbb{E}[(-1)^{\langle \mathbf{w}, \mathbf{X} \rangle}] &= \mathbb{E}[(-1)^{\langle \mathbf{w}, A_1 \mathbf{X}_1 \oplus \dots \oplus A_m \mathbf{X}_m \rangle}] \\ &= \mathbb{E}[(-1)^{\langle \mathbf{w}, A_1 \mathbf{X}_1 \rangle}] \cdots \mathbb{E}[(-1)^{\langle \mathbf{w}, A_m \mathbf{X}_m \rangle}] \\ &= \mathbb{E}[(-1)^{\langle A_1^T \mathbf{w}, \mathbf{X}_1 \rangle}] \cdots \mathbb{E}[(-1)^{\langle A_m^T \mathbf{w}, \mathbf{X}_m \rangle}] \\ &= \widehat{D}_1(A_1^T \mathbf{w}) \cdots \widehat{D}_m(A_m^T \mathbf{w}). \quad \blacktriangleleft \end{aligned}$$

For a distribution D over $\{0, 1\}^n$, integer $\ell \geq 1$, and linear transformation $A : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, we write $D_A^{(\ell)}$ for the distribution obtained by sampling $\mathbf{X}_1, \dots, \mathbf{X}_\ell$ independently and returning $\mathbf{X}_1 \oplus A \mathbf{X}_2 \oplus \dots \oplus A^{\ell-1} \mathbf{X}_\ell$.

2.3 Properties of (near)-uniform distribution over the Hamming ball

The (near)-uniform distribution over the Hamming ball with a given min-entropy plays an important role in our analysis.

► **Definition 5.** For $r, n \in \mathbb{N}, k \in \mathbb{R}$, suppose $1 \leq r \leq n$, and $n - r < k \leq n$, we define $D_{r,k}^*$ over $\{0, 1\}^n$ as follows,

$$D_{r,k}^*(\mathbf{x}) := \begin{cases} 2^{-k} & \sum_{i=1}^r x_i < d^* \\ p^* & \sum_{i=1}^r x_i = d^* \\ 0 & \text{otherwise,} \end{cases}$$

where $d^* := \min\{0 \leq d \leq r : 2^{n-r} \cdot ((\binom{r}{0} + \binom{r}{1} + \dots + \binom{r}{d}) \geq 2^k)\}$, and

$$p^* := \frac{1}{\binom{r}{d^*}} \cdot \left(2^{-(n-r)} - 2^{-k} \cdot \left(\binom{r}{0} + \binom{r}{1} + \dots + \binom{r}{d^*-1} \right) \right).$$

(I.e., d^* and p^* are chosen to make $D_{r,k}^*$ a probability distribution.)

► **Lemma 6.** Let $1 \leq r \leq n$ and $n - r < k \leq n$, and let $D_{r,k}^*$ be defined as above. Then, for $1 \leq i \leq r$,

$$\widehat{D_{r,k}^*}(\mathbf{e}_i) \leq 1 - \frac{c \cdot d^*}{r} \leq \left(1 - \frac{c(r+k-n)}{6r \log(2r/(r+k-n))} \right),$$

where $c := 1 - 2^{-(r+k-n)} \geq \min(\frac{1}{2}, \frac{r+k-n}{2})$.

Proof. By symmetry, for $1 \leq i \leq r$, $j \in \mathbb{N}$,

$$r \cdot \widehat{D}_{r,k}^*(e_i) = \sum_{i'=1}^r \widehat{D}_{r,k}^*(e_{i'}) = \sum_{i'=1}^r (1 - 2 \Pr_{\mathbf{x} \sim D_{r,k}^*} [x_{i'} = 1]) = r - 2 \mathbb{E}_{\mathbf{x} \sim D_{r,k}^*} \left[\sum_{i'=1}^r x_{i'} \right].$$

Let $p_j := \Pr_{\mathbf{x} \sim D_{r,k}^*} [\sum_{i'=1}^r x_{i'} = j]$. We have that

$$p_j := \begin{cases} 2^{n-r-k} \binom{r}{j} & 0 \leq j \leq d^* - 1 \\ 2^{n-r} \binom{r}{d^*} \cdot p^* & j = d^* \\ 0 & \text{otherwise.} \end{cases}$$

For $1 \leq j \leq d^* - 1$, it holds that

$$j \cdot p_j + (d^* - j) \cdot p_{d^*-j} \geq (p_j + p_{d^*-j}) \cdot (d^*/2)$$

because $p_j \leq p_{d^*-j}$ if and only if $j \leq d^* - j$ ³. Hence,

$$\begin{aligned} 2 \mathbb{E}_{\mathbf{x} \sim D_{r,k}^*} \left[\sum_{i'=1}^r x_{i'} \right] &= \sum_{j=0}^{d^*} (j \cdot p_j + (d^* - j) \cdot p_{d^*-j}) \\ &\geq \sum_{j=1}^{d^*-1} (p_j + p_{d^*-j}) \cdot (d^*/2) + 2d^* \cdot p_{d^*} \\ &= d^* \cdot \sum_{i=0}^{d^*} p_i + d^* \cdot (p_{d^*} - p_0) \\ &= d^* (1 + p_{d^*} - p_0) \\ &\geq d^* \cdot c \end{aligned}$$

where the last inequality is due to $p_{d^*} \geq 0$. Hence

$$\widehat{D}_{r,k}^*(e_i) = 1 - \frac{2 \mathbb{E}_{\mathbf{x} \sim D_{r,k}^*} [\sum_{i'=1}^r x_{i'}]}{r} \leq 1 - \frac{c \cdot d^*}{r}.$$

The first inequality in the theorem statement follows.

To finish the proof, we prove that for $k \in \mathbb{R}$, $n - r < k \leq n$,

$$d^* \geq \frac{r + k - n}{6 \log(2r/(r + k - n))}.$$

We rely on some basic facts about binary entropy function listed in Appendix A. For $p \in (0, 1)$, the binary entropy function is $H(p) := p \log_2(1/p) + (1 - p) \log_2(1/(1 - p))$. By Fact 19, we have

$$2^{r+k-n} \leq \sum_{i=0}^{d^*} \binom{r}{i} \leq 2^{rH(d^*/r)}.$$

If $k \leq n - 1$, then $d^* \leq r/2$. The desired conclusion follows by instantiating $rH(d^*/r) \geq r + k - n$ in Claim 20. If $n - 1 < k \leq n$, then $d^* > r/2 > \frac{r+k-n}{6 \log(2r/(r+k-n))}$ because $\frac{r+k-n}{6 \log(2r/(r+k-n))} \leq r/6$ for all $k \leq n$. \blacktriangleleft

³ Note that $p_j = 2^{n-r-k} \cdot \binom{r}{j}$, $p_{d^*-j} = 2^{n-r-k} \cdot \binom{r}{d^*-j}$ for $1 \leq j \leq d^* - 1$. If $p_j \leq p_{d^*-j}$, it implies $\binom{r}{j} \leq \binom{r}{d^*-j}$. Since $(j + d^* - j)/2 = d^*/2 \leq r/2$, it implies $j \leq d^* - j$. Conversely, if $j \leq d^* - j$, by the same reason it implies $\binom{r}{j} \leq \binom{r}{d^*-j}$ and thus $p_j \leq p_{d^*-j}$.

3 Our main lemma

► **Lemma 7.** For $r, n \in \mathbb{N}, k \in \mathbb{R}$, suppose $1 \leq r \leq n$, $n - r < k \leq n$, \mathbb{F}_2 -linearly independent vectors $\mathbf{w}_1, \dots, \mathbf{w}_r \in \{0, 1\}^n$, and a distribution D over $\{0, 1\}^n$ with at least min-entropy k , we have

$$\prod_{i=1}^r |\widehat{D}(\mathbf{w}_i)| \leq 2^{-c(r+k-n)/6 \log_2(2r/(r+k-n))}.$$

where $c = 1 - 2^{-(r+k-n)}$.

Proof of Lemma 7. Let $D_{r,k}^*$ be defined as in Definition 5. We show that products of Fourier coefficients at independent vectors is maximized by the products of Fourier coefficients at basis vectors for $D_{r,k}^*$.

▷ **Claim 8.** For \mathbb{F}_2 -linearly independent vectors $\mathbf{w}_1, \dots, \mathbf{w}_r \in \{0, 1\}^n$ and any distribution D over $\{0, 1\}^n$ with min-entropy $k \leq n$, we have

$$\prod_{i=1}^r |\widehat{D}(\mathbf{w}_i)| \leq \prod_{i=1}^r \widehat{D_{r,k}^*}(\mathbf{e}_i),$$

where $\mathbf{e}_i \in \{0, 1\}^n$ is the i th standard basis vector.

Combining with Lemma 6, we have

$$\prod_{i=1}^r |\widehat{D}(\mathbf{w}_i)| \leq \prod_{i=1}^r \widehat{D_{r,k}^*}(\mathbf{e}_i) \leq \left(1 - \frac{c(r+k-n)}{6r \log_2(2r/(r+k-n))}\right)^r.$$

The desired conclusion follows (notice $(1-x)^r \leq 2^{-rx}$ for $x \geq 0$). ◀

Proof of Claim 8. Let $A \in \mathbb{F}_2^{n \times n}$ be an invertible linear transformation such that $A^T \mathbf{w}_i = \mathbf{e}_i$ for all i . Then, $H_{\min}(AD) = H_{\min}(D)$ and $\widehat{AD}(\mathbf{e}_i) = \widehat{D}(\mathbf{w}_i)$. So, by applying the linear transformation A , we may assume without loss of generality that $\mathbf{w}_i = \mathbf{e}_i$. By possibly flipping some bits, we may also assume that $\widehat{D}(\mathbf{e}_i) \geq 0$, so that it suffices to prove that

$$\prod_{i=1}^r \widehat{D}(\mathbf{e}_i) \leq \prod_{i=1}^r \widehat{D_{r,k}^*}(\mathbf{e}_i),$$

For $1 \leq i < j \leq r$, let $\pi : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be the map that swaps the i th and j th coordinates and leaves all other coordinates untouched. Let D' be the distribution given by $D'(\mathbf{x}) = (D(\mathbf{x}) + D(\pi(\mathbf{x}))) / 2$. Notice that $H_{\min}(D') \geq H_{\min}(D)$. Furthermore,

$$\prod_{k=1}^r \widehat{D'}(\mathbf{e}_k) = \frac{(\widehat{D}(\mathbf{e}_i) + \widehat{D}(\mathbf{e}_j))^2}{4} \cdot \prod_{k \notin \{i,j\}} \widehat{D}(\mathbf{e}_k) \geq \prod_{k=1}^r \widehat{D}(\mathbf{e}_k),$$

where the last inequality follows from the fact that $(a+b)/2 \geq \sqrt{ab}$ for $a, b \geq 0$. Therefore, we may assume without loss of generality that $D(\mathbf{x}) = D(\pi(\mathbf{x}))$. By a similar argument, we may assume that $D(\mathbf{x}) = D(\mathbf{x}')$ for any $\mathbf{x}, \mathbf{x}' \in \{0, 1\}^n$ with $\sum_{i=1}^r x_i = \sum_{i=1}^r x'_i$.

Now, suppose that there exists a vector $\mathbf{x} \in \{0, 1\}^n$ and an index $1 \leq i \leq r$ such that $x_i = 1$, $D(\mathbf{x}) > 0$ and $D(\mathbf{x} \oplus \mathbf{e}_i) < 2^{-k}$. Then, let D' be the distribution that is identical to D except that $D'(\mathbf{x}) = D(\mathbf{x}) - p$ and $D'(\mathbf{x} \oplus \mathbf{e}_i) = D(\mathbf{x} \oplus \mathbf{e}_i) + p$, where $0 < p \leq \min\{D(\mathbf{x}), 2^{-k} - D(\mathbf{x} \oplus \mathbf{e}_i)\}$. Clearly, $H_{\min}(D') \geq k$ and $\prod_{i=1}^r \widehat{D'}(\mathbf{e}_i) > \prod_{i=1}^r \widehat{D}(\mathbf{e}_i)$.

So, by replacing D with D' , we may assume without loss of generality that no such \mathbf{x} and i exist. Together with the above assumption that $D(\mathbf{x}) = D(\mathbf{x}')$ whenever $\sum_{i=1}^r x_i = \sum_{i=1}^r x'_i$, this uniquely characterizes the distribution D . I.e., $D = D_{r,k}^*$. The result follows. ◀

4 Extractability

In this section, we characterize the matrices A that yield online extractors.

► **Definition 9.** We say that a subspace $S \subseteq \mathbb{F}_2^n$ is an invariant subspace of $A \in \mathbb{F}_2^{n \times n}$ or A -invariant if for every $\mathbf{w} \in S$, $A\mathbf{w} \in S$. We say S is non-trivial if $S \neq \{\mathbf{0}\}$ and $S \neq \mathbb{F}_2^n$.

There is a rich theory of invariant subspaces that is beyond the scope of this work. (See, e.g., [8].) For our purposes, it suffices to note simply that the invariant subspaces can be computed efficiently. In particular, the invariant subspaces correspond to factors of the characteristic and minimal polynomials of A , and A has no non-trivial invariant subspace if and only if the characteristic polynomial of A is irreducible.

Invariant subspaces arise naturally in this context. Indeed, if $S \subset \mathbb{F}_2^n$ is a non-trivial invariant subspace of A , then A will completely fail to extract from the uniform distribution over S . We make this observation formal in Theorem 10.

► **Theorem 10.** For $A \in \mathbb{F}_2^{n \times n}$, if there exists a non-trivial A -invariant subspace with dimension r , then there exists a distribution D over $\{0, 1\}^n$ with min-entropy r such that $D_A^{(m)} = D$ for all m .

Proof. Let S be an A -invariant subspace with dimension r . Let D be the uniform distribution over S with min-entropy r . Recall that D_A^m is the distribution obtained by sampling $\mathbf{X}_1, \dots, \mathbf{X}_\ell$ independently from D and returning $\mathbf{X}_1 \oplus A\mathbf{X}_2 \oplus \dots \oplus A^{m-1}\mathbf{X}_m$. Because S is A -invariant, it holds that $\mathbf{y} := A\mathbf{X}_2 \oplus \dots \oplus A^{m-1}\mathbf{X}_m$ is in the subspace S , and $\mathbf{X}_1 \oplus \mathbf{y}$ is uniformly distributed over S for an independent $\mathbf{y} \in S$. Therefore for all m , $D_A^{(m)}$ is the uniform distribution over S . ◀

Perhaps more surprisingly, the next theorem shows that this is the only restriction. In particular, if A has no non-trivial invariant subspace, then A extracts from any source with min-entropy k after $\tilde{O}(n^2(k+1)/k^2)$ steps.

► **Theorem 11.** For $A \in \mathbb{F}_2^{n \times n}$, if A has no non-trivial invariant subspace, then for $k > 0$, and any distribution D over $\{0, 1\}^n$ with min-entropy at least k ,

$$\text{SD}(D_A^{(m)}, U) \leq 2^{n-1-\lfloor m/n \rfloor \cdot \frac{ck}{6 \log_2(2n/k)}}$$

where $c = 1 - 2^{-k}$.

Proof. Because the orthogonal subspace of an A -invariant subspace is A^T -invariant, A^T also has no non-trivial invariant subspace. For any non-zero \mathbf{w} , it must therefore be the case that $\mathbf{w}_1 := \mathbf{w}, \mathbf{w}_2 := \dots, \mathbf{w}_n := (A^T)^{n-1}\mathbf{w}$, are linearly independent. Otherwise the span of $\mathbf{w}_1, \dots, \mathbf{w}_n$ would be a non-trivial A^T -invariant subspace. By applying Lemma 7 with $r = n$, we obtain

$$\prod_{i=0}^{n-1} |\widehat{D}((A^T)^i \mathbf{w})| = \prod_{i=1}^n |\widehat{D}(\mathbf{w}_i)| \leq 2^{-ck/6 \log_2(2n/k)}, \quad (1)$$

where $c = 1 - 2^{-k}$. Therefore, for any non-zero \mathbf{w} ,

$$|\widehat{D_A^{(m)}}(\mathbf{w})| = \prod_{i=0}^{m-1} |\widehat{D}((A^T)^i \mathbf{w})| = \prod_{j=0}^{\lfloor m/n \rfloor - 1} \prod_{i=0}^{n-1} |\widehat{D}((A^T)^{jn+i} \mathbf{w})| \leq 2^{-\lfloor m/n \rfloor \cdot \frac{ck}{6 \log_2(2n/k)}}$$

where the last inequality is due to $(A^T)^{jn}\mathbf{w} \neq 0$ and (1).

14:10 Online Linear Extractors for Independent Sources

By applying Claim 3,

$$\text{SD}(D_A^{(m)}, U) \leq \frac{1}{2} \cdot \sum_{\mathbf{w} \in \{0,1\}^n, \mathbf{w} \neq \mathbf{0}} |\widehat{D_A^{(m)}}(\mathbf{w})| \leq 2^{n-1 - \lfloor m/n \rfloor \cdot \frac{ck}{6 \log_2(2n/k)}},$$

as desired. \blacktriangleleft

We remark that a better upper bound can be obtained by using $\text{SD}(D_A^{(m)}, U) \leq \frac{1}{2} \cdot \sqrt{\sum_{\mathbf{w} \in \{0,1\}^n, \mathbf{w} \neq \mathbf{0}} \widehat{D_A^{(m)}}(\mathbf{w})^2}$ instead of Claim 3. Because it yields the same (asymptotic) upper bound on the number of required steps, we do not include it for the sake of simplicity.

We note in passing that the matrix A corresponding to multiplication by a generator of a finite field is a particularly nice example satisfying the condition of Theorem 11. That is, if we interpret $\mathbf{y} = (y_1, \dots, y_n) \in \{0, 1\}^n$ as the polynomial $y_1 + y_2t + \dots + y_nt^{n-1} \in \mathbb{F}_2[t]/p(t)$ for some irreducible polynomial $p(t) \in \mathbb{F}_2[t]$ of degree n . Then, the matrix A corresponding to multiplication by t has no non-trivial invariant subspace⁴ and thus yields a good extractor. This matrix has the convenient property that it is quite sparse – with all columns except the last having a single non-zero entry.

5 Condensibility

We now turn our attention to linear online condensers. Our results will be in terms of the concept of the A -rank of a vector $\mathbf{w} \in \mathbb{F}_2^n$, defined below.

► **Definition 12.** For any $A \in \mathbb{F}_2^{n \times n}$, the A -orbit of a vector $\mathbf{w} \in \{0, 1\}^n$ is the set $\{A^k \mathbf{w}\}_{k=0}^\infty$. The linear orbit $[\mathbf{w}]$ of \mathbf{w} is the subspace spanned by A -orbit of \mathbf{w} .

► **Definition 13.** For any $A \in \mathbb{F}_2^{n \times n}$, the A -rank of a vector $\mathbf{w} \in \{0, 1\}^n$ is the maximal integer r such that the set of vectors $\{\mathbf{w}, A\mathbf{w}, \dots, A^{r-1}\mathbf{w}\}$ is linearly independent. We use $\text{rank}_A(\mathbf{w})$ ⁵ to denote A -rank of \mathbf{w} .

One can efficiently compute the number of vectors with a given A -rank by computing the minimal polynomial of A [8].

► **Proposition 14.** For $A \in \mathbb{F}_2^{n \times n}$, $\mathbf{w} \in \{0, 1\}^n$ with the A -rank r , the linear orbit $[\mathbf{w}]$ is an invariant subspace of dimension r . Moreover,

$$[\mathbf{w}] = \text{span}(\mathbf{w}, A\mathbf{w}, \dots, A^{r-1}\mathbf{w}).$$

The above proposition shows that the A -rank of \mathbf{w} characterizes the minimal invariant subspace V containing \mathbf{w} : if the A -rank of \mathbf{w} is r , then the first r vectors in the A -orbit are linear independent and thus generate V . In particular, if A has no non-trivial invariant subspace, then every $\mathbf{w} \in \mathbb{F}_2^n \setminus \{0^n\}$ has A -rank n .

Our next theorem gives a partial characterization of matrices A that yield good linear online condensers in terms of A^T -rank and the number of vectors with small A^T -rank. This yields a natural generalization of Theorem 11.

⁴ To see this, suppose for contradiction that there exists a non-trivial t -invariant subspace $V \subset \mathbb{F}_2[t]/p(t)$. Then, for any $x \in V$, we must have that $x, tx, \dots, t^{n-1}x$ are linearly dependent (since otherwise V is either not invariant or $V = \mathbb{F}_2^n$ is non-trivial). Since $\mathbb{F}_2[t]/p(t)$ is a field, if $V \neq \{0\}$, we must also have that $1, t, \dots, t^{n-1}$ are linearly independent. This means that t is a root of a polynomial with degree at most $n-1$, contradicting the assumption that p is irreducible.

⁵ In linear algebra, our notation $\text{rank}_A(\mathbf{w})$ is the same as the maximal dimension of a Krylov subspace generated by A and \mathbf{w} .

► **Theorem 15.** For any invertible $A \in \mathbb{F}_2^{n \times n}$, if there are at most N vectors in $\{0, 1\}^n$ with A^T -rank less than r , then for any real number $g := n - r < k \leq n$ and any distribution D over $\{0, 1\}^n$ with min-entropy at least k ,

$$H_{\min}(D_A^{(m)}) \geq n - \log_2(N + 2^{n - \lfloor m/n \rfloor \cdot \frac{c(k-g)}{6 \log_2(2^r/(k-g))}})$$

where $c = 1 - 2^{-(k-g)}$.

Proof. For any $\mathbf{w} \in \{0, 1\}^n$ with A^T -rank at least r , then there are at least r -linear independent vectors among $\mathbf{w}, \dots, (A^T)^{n-1}\mathbf{w}$, denoted as $\mathbf{w}_1, \dots, \mathbf{w}_r$. By Lemma 7, it implies

$$\prod_{i=0}^{n-1} |\widehat{D}((A^T)^i \mathbf{w})| \leq \prod_{i=1}^r |\widehat{D}(\mathbf{w}_i)| \leq 2^{-\frac{c(k-g)}{6 \log_2(2^r/(k-g))}},$$

where $c = 1 - 2^{-(k-g)}$. Moreover, because A is invertible, $(A^T)^n \mathbf{w}$ has the same A^T -rank as \mathbf{w} . We have that,

$$|\widehat{D}_A^{(m)}(\mathbf{w})| = \prod_{i=0}^{m-1} |\widehat{D}((A^T)^i \mathbf{w})| \leq \prod_{j=0}^{\lfloor m/n \rfloor - 1} \prod_{i=0}^{n-1} |\widehat{D}((A^T)^{jn+i} \mathbf{w})| \leq 2^{-\lfloor m/n \rfloor \cdot \frac{c(k-g)}{6 \log_2(2^r/(k-g))}}$$

Because there are at most N vectors with A^T -rank less than r and $|\widehat{D}_A^{(m)}(\mathbf{w})| \leq 1$ for every \mathbf{w} , it holds that

$$\sum_{\mathbf{w} \in \{0, 1\}^n} |\widehat{D}_A^{(m)}(\mathbf{w})| \leq N \cdot 1 + \sum_{\mathbf{w}: \text{rank}_{A^T}(\mathbf{w}) \geq r} |\widehat{D}_A^{(m)}(\mathbf{w})| \leq N + 2^n \cdot 2^{-\lfloor m/n \rfloor \cdot \frac{c(k-g)}{6 \log_2(2^r/(k-g))}}.$$

By applying Claim 3,

$$H_{\min}(D) = n - \log_2 \left(\sum_{\mathbf{w} \in \{0, 1\}^n} |\widehat{D}(\mathbf{w})| \right) \geq n - \log_2(N + 2^{n - \lfloor m/n \rfloor \cdot \frac{c(k-g)}{6 \log_2(2^r/(k-g))}}),$$

as desired. ◀

Theorem 15 implies that *any* distribution with $> n - r$ bits of min-entropy can be condensed into at least $n - \log_2 N$ bits. Notice that Theorem 15 is non-vacuous if $N < 2^r$. Moreover, the constraint $k > n - r$ is tight. If there exists a vector with A^T -rank r , then there is an A^T -invariant subspace V of dimension r , which in particular contains 2^r vectors of A^T -rank at most r . Then, by Theorem 10 the distribution D that is uniform over the subspace orthogonal to V has min-entropy $n - r$ but $D_A^{(m)} = D$ for all m .

Rotation

Finally, as an application of this result, we show that rotation yields a good condenser for some n . (Moreover, if we assume an additional minor condition on the distribution D , we actually get an extractor.)

We write rot_n for the linear transformation over $\{0, 1\}^n$ which rotates the coordinates of a vector \mathbf{x} by 1. In other words,

$$\text{rot}_n((x_1, \dots, x_n)) := (x_2, x_3, \dots, x_n, x_1).$$

Our first observation is that $\{\mathbf{x} : x_i = x_{i+d}, \forall 1 \leq i \leq n-d\}$ is an invariant subspace of any rotation when $d < n$ is a divisor of n . By Theorem 10, rot_n therefore cannot extract from sources with min-entropy d for $d < n$ a divisor of n . Moreover, rotations in general cannot condense from a single bit of randomness because of the invariant subspace $\{0^n, 1^n\}$ and cannot condense beyond $n-1$ bits of randomness because of the invariant subspace $\{\mathbf{x} : x_1 \oplus \dots \oplus x_n = 0\}$. Therefore, the best we can hope for is to condense from $k > 1$ bits of entropy to $n-1$ bits of entropy for n prime.

We show that rot_n does in fact achieve this as long as n is a prime satisfying a natural number-theoretic condition. Indeed, this follows from Theorem 15 together with the following lemma due to Vazirani [17].

► **Lemma 16** ([17]). *If n is a prime such that 2 generates \mathbb{Z}_n^* (e.g., 5, 29, 37), then all $\mathbf{w} \in \{0, 1\}^n \setminus \{1^n, 0^n\}$ have rot_n -rank at least $n-1$.*

Plugging into Theorem 15 yields the following. In particular, for such primes, rot_n condenses from $k > 1$ bits to $n-1$ bits in at most $m = \tilde{O}(n^2 k / (k-1)^2)$ steps.

► **Corollary 17.** *If n is a prime with 2 is a primitive root for \mathbb{Z}_n^* , then for any real number $1 < k \leq n$, and distribution D over $\{0, 1\}^n$ with at least min-entropy k ,*

$$H_{\min}(D_{\text{rot}_n}^{(m)}) \geq n - \log_2(2 + 2^{n - \lfloor m/n \rfloor \cdot \frac{c(k-1)}{6 \log_2(2(n-1)/(k-1))}}).$$

where $c = 1 - 2^{-(k-1)}$.

Finally, we note that our proof of Theorem 15 actually yields a statement about extraction as well, which we present here in the special case of rotation. Specifically, in the proof of Theorem 15, we used the trivial bound of $|\tilde{D}(\mathbf{w})| \leq 1$ for low-rank \mathbf{w} . If we instead happen to know a better bound on the Fourier coefficient explicitly for the single non-zero low-rank vector for rotation, 1^n , we see that we can actually extract.

► **Theorem 18.** *For primes n such that 2 generates \mathbb{Z}_n^* , and for $1 < k \leq n$, a distribution D over $\{0, 1\}^n$ with at least min-entropy k ,*

$$\text{SD}(D_{\text{rot}_n}^{(m)}, U) \leq \frac{1}{2} \cdot \left(|\widehat{D}(1^n)|^m + 2^{n - \lfloor m/n \rfloor \cdot \frac{c(k-1)}{6 \log_2(2(n-1)/(k-1))}} \right)$$

where $c = 1 - 2^{-(k-1)}$.

Theorem 18 implies that for such primes n , rotation yields a good online linear extractor for distributions D with small $|\widehat{D}(1^n)|$ and min-entropy strictly larger than one. Notice that the two counterexamples that we discussed in the definition – the uniform distribution over $\{0^n, 1^n\}$, and the uniform distribution over all strings with even Hamming weight – show that one of these conditions alone is not enough.

References

- 1 Ziv Bar-Yossef, Luca Trevisan, Omer Reingold, and Ronen Shaltiel. Streaming computation of combinatorial objects. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity, Montréal, Québec, Canada, May 21-24, 2002*, pages 165–174. IEEE Computer Society, 2002. doi:10.1109/CCC.2002.1004352.
- 2 Boaz Barak, Russell Impagliazzo, and Avi Wigderson. Extracting randomness using few independent sources. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 384–393. IEEE Computer Society, 2004. doi:10.1109/FOCS.2004.29.

- 3 Salman Beigi, Andrej Bogdanov, Omid Etesami, and Siyao Guo. Optimal deterministic extractors for generalized santha-vazirani sources. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, volume 116 of *LIPICs*, pages 30:1–30:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICs.APPROX-RANDOM.2018.30.
- 4 Salman Beigi, Omid Etesami, and Amin Gohari. Deterministic randomness extraction from generalized and distributed santha-vazirani sources. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 143–154. Springer, 2015. doi:10.1007/978-3-662-47672-7_12.
- 5 Andrej Bogdanov and Siyao Guo. Sparse extractor families for all the entropy. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 553–560. ACM, 2013. doi:10.1145/2422436.2422496.
- 6 Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 670–683. ACM, 2016. doi:10.1145/2897518.2897528.
- 7 Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988. doi:10.1137/0217015.
- 8 Pete L. Clark. Linear algebra: Invariant subspaces, 2013. URL: http://alpha.math.uga.edu/~pete/invariant_subspaces.pdf.
- 9 Sandro Coretti, Yevgeniy Dodis, Harish Karthikeyan, and Stefano Tessaro. Seedless fruit is the sweetest: Random number generation, revisited. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 205–234. Springer, 2019. doi:10.1007/978-3-030-26948-7_8.
- 10 Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the cbc, cascade and HMAC modes. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 494–510. Springer, 2004. doi:10.1007/978-3-540-28628-8_30.
- 11 Yevgeniy Dodis, Siyao Guo, Noah Stephens-Davidowitz, and Zhiye Xie. No time to hash: On superefficient entropy accumulation. Cryptology ePrint Archive, Report 2021/523, 2021. URL: <https://eprint.iacr.org/2021/523>.
- 12 Peter Elias. The Efficient Construction of an Unbiased Random Sequence. *The Annals of Mathematical Statistics*, 43(3):865–870, 1972. doi:10.1214/aoms/1177692552.
- 13 Niels Ferguson. The windows 10 random number generation infrastructure. <https://www.microsoft.com/security/blog/2019/11/25/going-in-depth-on-the-windows-10-random-number-generation-infrastructure/>, 2019.
- 14 Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. doi:10.1137/S0097539793244708.
- 15 Jesse Kamp, Anup Rao, Salil P. Vadhan, and David Zuckerman. Deterministic extractors for small-space sources. *J. Comput. Syst. Sci.*, 77(1):191–220, 2011. doi:10.1016/j.jcss.2010.06.014.

14:14 Online Linear Extractors for Independent Sources

- 16 Miklos Santha and Umesh V. Vazirani. Generating quasi-random sequences from semi-random sources. *J. Comput. Syst. Sci.*, 33(1):75–87, 1986. doi:10.1016/0022-0000(86)90044-9.
- 17 Umesh V. Vazirani. Efficiency considerations in using semi-random sources. In *STOC*, pages 160–168, 1987. doi:10.1145/28395.28413.
- 18 John von Neumann. Various techniques used in connection with random digits. In *Monte Carlo Method*, pages 36–38. National Bureau of Standards Applied Mathematics Series, 12, 1951.

A Facts about binary entropy function

► **Fact 19.** For $0 \leq d \leq \ell/2$,

$$\sum_{i=0}^d \binom{\ell}{i} \leq 2^{\ell H(d/\ell)}.$$

▷ **Claim 20.** [5] For every $p \in (0, 1/2]$,

$$\frac{H(p)}{6 \log_2(2/H(p))} \leq p \leq \frac{H(p)}{\log_2(1/H(p))}.$$

We include the proof from [5] for completeness.

Proof. The upper bound on p follows from the inequality $H(p) \geq p \log_2 1/p$. Applying twice we obtain

$$\frac{1}{p} \geq \frac{1}{H(p)} \log_2 \frac{1}{p} \geq \frac{1}{H(p)} \log_2 \left(\frac{1}{H(p)} \log_2 \frac{1}{p} \right) \geq \frac{1}{H(p) \log_2 \frac{1}{H(p)}}$$

because $1/p \geq 2$. For the lower bound, we apply $H(p) \leq 2p \log_2 1/p$ twice to obtain

$$\frac{1}{p} \leq \frac{2}{H(p)} \log_2 \frac{1}{p} \leq \frac{2}{H(p)} \log \left(\frac{2}{H(p)} \log_2 \frac{1}{p} \right).$$

Now $2/H(p) \geq (1/p) \log_2(1/p) \geq \sqrt{\log_2(1/p)}$, which is true for every $p \in (0, 1]$. Therefore,

$$\frac{1}{p} \leq \frac{2}{H(p)} \log_2 \left(\frac{8}{H(p)^3} \right) = \frac{6}{H(p)} \log_2 \left(\frac{2}{H(p)} \right). \quad \triangleleft$$

Code Offset in the Exponent

Luke Demarest  

University of Connecticut, Storrs, CT, USA

Benjamin Fuller  

University of Connecticut, Storrs, CT, USA

Alexander Russell  

University of Connecticut, Storrs, CT, USA

Abstract

Fuzzy extractors derive stable keys from noisy sources. They are a fundamental tool for key derivation from biometric sources. This work introduces a new construction, code offset in the exponent. This construction is the first reusable fuzzy extractor that simultaneously supports structured, low entropy distributions with correlated symbols and confidence information. These properties are specifically motivated by the most pertinent applications – key derivation from biometrics and physical unclonable functions – which typically demonstrate low entropy with additional statistical correlations and benefit from extractors that can leverage confidence information for efficiency.

Code offset in the exponent is a group encoding of the code offset construction (Juels and Wattenberg, CCS 1999). A random codeword of a linear error-correcting code is used as a one-time pad for a sampled value from the noisy source. Rather than encoding this directly, code offset in the exponent encodes by exponentiation of a generator in a cryptographically strong group. We introduce and characterize a condition on noisy sources that directly translates to security of our construction in the generic group model. Our condition requires the inner product between the source distribution and all vectors in the null space of the code to be unpredictable.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques; Security and privacy → Biometrics

Keywords and phrases fuzzy extractors, code offset, learning with errors, error-correction, generic group model

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.15

Related Version *Full Version: Cryptology ePrint Archive* [12]

Funding *Luke Demarest:* Funded in part by NSF Grants No. 1849904 and 1547399.

Benjamin Fuller: Funded in part by NSF Grants No. 1849904 and 1547399. This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via Contract No. 2019-19020700008. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

Alexander Russell: This material is based upon work supported by the National Science Foundation under Grant No. 1801487.

Acknowledgements The authors give special thanks to reviewer comments and feedback. The authors thank James Bartusek, Ryann Cartor, Fermi Ma, and Mark Zhandry and their helpful discussions of their work.



© Luke Demarest, Benjamin Fuller, and Alexander Russell;
licensed under Creative Commons License CC-BY 4.0

2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 15; pp. 15:1–15:23



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Fuzzy extractors [13] permit derivation of a stable key from a noisy *source*. Specifically, given a reading \mathbf{e} from the noisy source, the fuzzy extractor produces a pair (key, pub) , consisting of a derived key and a public value; the public value pub must then permit key to (only) be recovered from any \mathbf{e}' that is sufficiently close to \mathbf{e} in Hamming distance. Fuzzy extractors are the emblematic technique for robust, secure key derivation from biometrics and physical unclonable functions. These applications place special emphasis on the source distribution and for this reason a principal goal of fuzzy extractor design is to precisely identify those distributions over \mathbf{e} for which extraction is possible and, moreover, produce efficient constructions for these distributions.

Despite years of work, existing constructions do not simultaneously secure practical sources while retaining efficient recovery. Canetti et al.’s construction [8, 9] is secure for the widest variety of sources. However, Simhadri et al.’s [31] implementation for the iris estimates only 32 bits of security with algorithms that take ≈ 10 seconds on a 32-core machine.

The fuzzy extraction problem is well-understood in the information-theoretic setting, where the fundamental quantity of interest is the *fuzzy min-entropy* [18, 19] of the distribution of \mathbf{e} ; this measures the total weight of an arbitrarily centered ball of radius t in the probability distribution over \mathbf{e} . While this measure is sufficient for determining the feasibility of information-theoretic fuzzy extraction for a distribution, it doesn’t indicate whether it is possible in polynomial time [18, 34]. In the information-theoretic setting, it is not possible to build an information-theoretic fuzzy extractor that simultaneously works for all distributions [18, 17, 19]. That is, a fuzzy extractor exists for each distribution with fuzzy min-entropy but no construction can secure all such distributions.

One can hope to sidestep these limitations by providing only computational security [15, 16]. However, even in this more favorable setting no universal theory has emerged without resorting to general purpose obfuscation. Two known fuzzy extractors use “computational” tools¹ to correct errors, they are:

- Canetti et al.’s [8, 9] construction explicitly places random subsets of \mathbf{e} in a *digital locker* [7] and records the indices used in each subset. To recover, one attempts to open each digital locker with subsets of the value \mathbf{e}' . Canetti et al.’s construction is secure when a random subset of locations is hard to predict (Definition 10). However, Simhadri’s implementation for the iris provides poor security (32 bits) in order to run in 10 seconds and still requires millions of digital lockers [31].
- Fuller et al. [15, 16] modify the *code-offset* construction [24]. The code-offset construction is determined by a linear error-correcting code $\mathbf{A} \in \mathbb{F}_q^{n \times k}$ and a secret, uniformly random $\mathbf{x} \in \mathbb{F}_q^k$; given a sample $\mathbf{e} \in \mathbb{F}_q^n$ from the noisy source, the construction publishes the pair $\text{pub} = (\mathbf{A}, \mathbf{Ax} + \mathbf{e})$. All operations are carried out over the field with q elements. To *reproduce* the value \mathbf{e} note that with a second sample \mathbf{e}' from the source – which we assume has small Hamming distance from \mathbf{e} ² – the difference $(\mathbf{Ax} + \mathbf{e}) - \mathbf{e}' = \mathbf{Ax} + (\mathbf{e} - \mathbf{e}')$ is evidently close to the codeword \mathbf{Ax} . By decoding the error correcting code one can recover \mathbf{x} (and \mathbf{e}).³ Security analysis of the code offset treats \mathbf{Ax} as a biased one time pad, proving that $\mathbf{Ax} + \mathbf{e}$ leaks no more than $(n - k) \log q$

¹ Multiple computational fuzzy extractors retain the information-theoretic core and analyze it using standard information-theory techniques [32, 33]; these works are subject to the above limitations.

² It is also possible to consider other distances between \mathbf{e} and \mathbf{e}' . However the error correction techniques required are different. We consider Hamming error in this work.

³ Applying a randomness extractor [25] on either \mathbf{x} or \mathbf{e} yields a uniform key.

bits about \mathbf{e} . However, many real distributions have entropy less than $(n - k) \log q$, which we call *low entropy*, for which this analysis provides no security guarantee. To support low entropy distributions, Fuller et al. instantiate this construction with \mathbf{A} being randomly distributed and show security whenever the distribution over \mathbf{e} yields a secure learning with errors (LWE) instance. Known LWE error distributions consider i.i.d. symbols (discretized Gaussian [28] and uniform interval [14]).

The digital locker construction supports more distributions (i.i.d. symbols implies that all subsets have entropy). Both constructions use information set decoding [26], that is, repeated selection of random subsets of coordinates with the hope to find a subset with no errors.

The digital locker construction comes with an important drawback. Many physical sources are sampled along with correlated side information that is called *confidence*. Confidence information is a secondary probability distribution \mathbf{z} (correlated with the reading \mathbf{e}) that can predict the error rate in a symbol \mathbf{e}_i . When \mathbf{z}_i is large this indicates that the symbol of \mathbf{e}_i is less likely to differ. Examples include the magnitude of a convolution in the iris [31] and the magnitude of the difference between two circuit delays in ring oscillator PUFs [22]. By considering bits with high confidence it is possible to reduce the effective error rate from $t = n/10$ to $t = 3n/10^6$ [22]. For a subset size of 128 and $t = n/10$ unlocking with 95% probability requires testing approximately $2 \cdot 10^6$ subsets while $t = 3n/10^6$ requires testing a single subset. This confidence information cannot be used in the digital locker construction as subsets are specified at enrollment time whereas confidence information is determined when \mathbf{e} is drawn. The LWE construction can use this information [23] as it allows on-the-fly testing of all large enough subsets. Confidence information is critical: fuzzy extractors that secure low entropy distributions do not support $t = \Theta(n)$ which is demonstrated in practice, leading to inefficient implementations. Because constructions are used with sources beyond their designed error tolerance any reduction in error rate has a drastic impact on efficiency (see Subsection 3.2).

Our contributions

This work introduces the *code offset in the exponent* construction. Code offset in the exponent yields the first reusable fuzzy extractor that simultaneously

- allows the symbols of \mathbf{e} to be correlated,
- supports structured but low entropy distributions over \mathbf{e} (less than $(n - k) \log q$), and
- allows the use of confidence information for improved efficiency.

This work introduces the *Code Offset in the Exponent* problem:

Distinguish $r^{\mathbf{A}\mathbf{x}+\mathbf{e}}$, given (\mathbf{A}, r) , from a random tuple of group elements, where r is a random generator of a prime order group, \mathbf{A} is a suitable linear code, and \mathbf{x} is a uniform dimension k vector.

A natural fuzzy extractor constructor exists when $r^{\mathbf{A}\mathbf{x}+\mathbf{e}}$ has such pseudorandom properties. We show that when the group effectively limits the adversary to linear operations – by adopting the generic group model – the resulting fuzzy extractor is secure for many low entropy distributions while retaining the ability to use confidence information. This allows code offset in the exponent to benefit from the efficiency gains of using confidence information while remaining secure for a large family of distributions. Specifically, we present three contributions:

Sec 1.1 We define the code offset in the exponent construction and show that it yields a reusable fuzzy extractor if the distribution on \mathbf{e} is *good enough*.

Sec 1.2 We define and describe what constitutes *good enough* in the generic group model with a novel information-theoretic sufficient condition we call MIPURS.

Sec 1.3 We characterize MIPURS, establishing containment relations between MIPURS and the secured distributions in Canetti et al. [8] and Fuller et al. [15] (see Figure 1).

We then review further related work and offer a table of comparisons (Sec 1.4). Section 2 covers definitions and preliminaries including the MIPURS condition. Section 3 details the code offset in the exponent construction. Section 4 characterizes MIPURS distributions.

1.1 Code offset in the exponent

Code offset in the exponent is motivated by the observation that *reproduction* of \mathbf{e} in the LWE construction uses only linear operations. Thus, we explore an adaptation of the code offset construction that effectively limits the adversary to linear operations by translating all relevant arithmetic into a “hard” group. Specifically, we introduce *code offset in the exponent*: If r is a random generator for a cyclic group \mathbb{G} of prime order q , we consider $\text{pub} = (\mathbf{A}, r, r^{\mathbf{Ax}+\mathbf{e}})$ where we adopt the shorthand notation $r^{\mathbf{v}}$, for a vector $\mathbf{v} = (v_1, \dots, v_n)^\top \in (\mathbb{Z}_q)^n$, to indicate the vector $(r^{v_1}, \dots, r^{v_n})^\top$. This construction possesses strong security properties under natural cryptographic assumptions on the group \mathbb{G} . We focus on code-offset in the exponent with a random linear code (given by \mathbf{A}) and adopt the generic group model [30] to reflect the cryptographic properties of the underlying group. As stated above, the goal is to characterize the distributions on \mathbf{e} for which $r^{\mathbf{Ax}+\mathbf{e}}$ given (\mathbf{A}, r) is pseudorandom. Pseudorandomness suffices to show security of a fuzzy extractor that leaks nothing about \mathbf{e} . Analysis of this construction is most natural when \mathbf{e} has symbols over a large alphabet, but binary \mathbf{e} can be amplified (see Section 3.1).

Looking ahead, if one uses a random generator in each enrollment the construction allows multiple (noisy) enrollments of \mathbf{e} , known as a reusable fuzzy extractor [6]. The reusability proof uses the details of the generic group proof, while the one time analysis is just based on pseudorandomness. See the full version of this work for details of that proof.

1.2 When is code offset in the exponent hard?

In the generic group model, we establish that distinguishing code offset in the exponent from a random vector of group elements is hard (Theorem 4) for any error distribution \mathbf{e} where the following game is hard to win for any information-theoretic adversary \mathcal{A} :⁴

Experiment $\mathbb{E}_{\mathcal{A}, \mathbf{e}}^{\text{MIPURS}}(n, k)$:
 $\psi \leftarrow \mathbf{e}; A \xleftarrow{\$} \mathbb{F}_q^{n \times k}$.
 $(b, g) \leftarrow \mathcal{A}(A, \mathbf{e})$.
 If $b \in \text{null}(A)$, $b \neq \vec{0}$ and $\langle b, \psi \rangle = g$ output 1.
 Output 0.

Observe that the role of the random matrix A in the game above is merely to define a random subspace of (typical) dimension k .

We call this condition on an error distribution MIPURS or *maximum inner product unpredictable over random subspace*. Specifically, a random variable \mathbf{e} over \mathbb{F}_q^n is (k, β) -MIPURS if for all \mathcal{A} (which knows the distribution of \mathbf{e} but not the sampled value ψ), $\Pr[\mathbb{E}_{\mathcal{A}, \mathbf{e}}^{\text{MIPURS}}(n, k) = 1] \leq \beta$.

⁴ We use boldface to represent random variables, capitals to represent random variables over matrices, and plain letters to represent samples. We use ψ to represent samples from \mathbf{e} to avoid conflict with Euler’s number.

When \mathbf{e} is a $(k - \Theta(1), \beta)$ -MIPURS distribution for a code with dimension k and $\beta = \text{ngl}(n)$ then code-offset in the exponent yields a fuzzy extractor in the generic group model (Theorem 5). Showing this requires one additional step of key extraction; we use a result of Akavia, Goldwasser, and Vaikuntanathan [1, Lemma 2] which states that dimensions of \mathbf{x} become *hardcore* once there are enough dimensions for LWE to be indistinguishable. This reduction is entirely linear and holds in the generic group setting.

MIPURS is necessary. When \mathcal{A} is information theoretic, for all distributions \mathbf{e} that are not MIPURS one can find a nonzero vector b in the null space of A whose inner product with \mathbf{e} is predictable, thus predicting $\langle b, \mathbf{Ax} + \mathbf{e} \rangle = \langle b, \mathbf{e} \rangle \stackrel{?}{=} g$. This is not the case for a uniform distribution, \mathbf{U} : the value $\langle b, \mathbf{U} \rangle$ is uniform (and thus is $\langle b, \mathbf{U} \rangle = g$ with small probability if the size of q is super polynomial). Thus the vector b serves as a way to distinguish $\mathbf{Ax} + \mathbf{e}$ from \mathbf{U} .

Beullens and Wee [3] recently introduced the KOALA assumption which roughly assumes that an adversary’s only mechanism for distinguishing a vector from a subspace from random is by outputting a vector that is likely to be the null space of the provided vector. This can be seen as specializing [11, Assumption 5] that vectors can only be distinguished by fixed inner products.

The adversary has more power in the MIPURS setting (than in KOALA) in three ways. First, the distribution \mathbf{e} and thus $\mathbf{Ax} + \mathbf{e}$ is not linear, second the adversary doesn’t have to “nullify” all subspaces – only a single vector, and third, the adversary can predict any inner product, not just 0. One can view MIPURS as an assumption on a group: Whenever an adversary can distinguish the (nonlinear) vector $\mathbf{Ax} + \mathbf{e}$ from uniform that there is another adversary that can choose some b and predict $\langle b, \mathbf{Ax} + \mathbf{e} \rangle$ (in our setting this choice of b is after seeing A). Theorem 4 can be interpreted as the MIPURS “assumption” holding in the generic group model.

1.3 Supported Distributions

Our technical work characterizes the MIPURS property (summarized in Figure 1). The most involved relationship is showing that all high entropy sources are MIPURS. To provide intuition for our results, we summarize this result here.

For any $d = \text{poly}(n)$ there is an efficiently constructible distribution \mathbf{e} whose entropy is approximately $\log(dq^{n-k-1})$ where the MIPURS game is winnable by an efficient adversary with noticeable probability: For $1 \leq i \leq d$, sample some d random linear spaces \mathbf{B}_i of dimension $n - k - 1$ and define \mathbf{E}_i to be all points in a random coset g_i of \mathbf{B}_i . Consider the following distribution \mathbf{e} : Pick $i \leftarrow \{1, \dots, d\}$ for some polynomial size d then output a random element of \mathbf{E}_i . The support size of this distribution is approximately dq^{n-k-1} . Then since $\text{null}(\mathbf{A})$ has dimension at least $n - k$, $\exists b_i \neq \vec{0}$ such that $b_i \in \text{null}(\mathbf{A}) \cap \text{null}(\mathbf{B}_i)$ (since $\dim(\text{null}(\mathbf{A})) + \dim(\text{null}(\mathbf{B}_i)) > n$). The adversary can calculate these b_i ’s. Then the adversary just picks a random i and predicts (b_i, g_i) .⁵ This result is nearly tight: all distributions whose entropy is greater than $\log(\text{poly}(n)q^{n-k})$ are MIPURS. Note this is a factor of q away from matching the size of our counterexample for a random code. Informally, this yields the following (see Corollary 25):

► **Theorem 1 (Informal).** *Let $n, k \in \mathbb{Z}$ be parameters. Let $q = q(n)$ be a large enough prime. For all $\mathbf{e} \in \mathbb{Z}_q^n$ whose minentropy is at least $\omega(\log n) + \log(q^{n-k})$, there exists some $\beta = \text{ngl}(n)$ for which \mathbf{e} is (k, β) -MIPURS.*

⁵ If \mathbf{A} is some fixed code (chosen before adversary specifies \mathbf{e}), then \mathbf{E}_i can directly be a coset of \mathbf{A} and one can increase the size of \mathbf{E} to dq^{n-k} .

As mentioned above, information theoretic analysis of code offset provides a key of length $\omega(\log n)$ when the initial entropy of \mathbf{e} is at least $\omega(\log n) + (n-k)\log(q)$. However, information theoretic analysis of code offset reduces the entropy of \mathbf{e} which may allow prediction of sensitive attributes. In the generic group analysis no predicate of \mathbf{e} is leaked. The generic group analysis also allows the construction to be safely reused multiple times (with independent generators).

Proof Intuition

Suppose in the above game the adversary generated \mathbf{e} as the span of a linear space \mathbf{E} with the goal that $\text{null}(\mathbf{A}) \cap \text{null}(\mathbf{E}) \supset \{\vec{0}\}$. For a random, independent $\mathbf{B} \stackrel{\text{def}}{=} \text{null}(\mathbf{A})$, the probability of \mathbf{B} and $\text{null}(\mathbf{E})$ overlapping is noticeable only if the sum of the dimensions is more than n (Lemma 19). This creates an upper bound on the dimension of \mathbf{E} of $n - k$ (ignoring the unlikely case when \mathbf{A} is not full rank).

Our proof is dedicated to showing that the general case (where \mathbf{E} is not linear) does not provide the adversary with more power. First we upper bound the size of a set E where each vector is predictable in the MIPURS game. We show for a random sample from E to have a large intersection with a low dimensional space requires E to have size at least that of the low dimensional space (Lemma 18). In Lemma 20, we switch from measuring the size of intersection of a sample of E with respect to the worst case subspace to how “linear” E is with respect to the worst vector in an average case subspace. This result thus controls an “approximate” algebraic structure in the sense of additive combinatorics. We show the adversary can’t do much better on a single vector b as long as it is chosen from a random \mathbf{B} .

The above argument considers the event that the adversary correctly predicts an inner product of 0; this can be transformed to an arbitrary inner product by a compactness argument which introduces a modest loss in parameters (Theorem 22). Once we have a bound on how large a predictable set E can be, another superlogarithmic factor guarantees that all distributions \mathbf{e} with enough minentropy are not predictable.

1.4 Further Related Work

We have already introduced the work of Canetti et al. [8] and Fuller et al. [16]. Canetti et al. [8] explicitly place some subsets into a digital locker, for security they require that an average subset has average min-entropy, which we call *average subsets have entropy*.

Lemma 11 shows that the MIPURS condition is contained in average subsets have entropy. This containment is proper, we actually show that there are distributions where all subsets have entropy that are not MIPURS. Suppose that \mathbf{e} is a Reed-Solomon code, then all subsets of \mathbf{e} have entropy but as long as the dimension of the code $< n - k - 1$ then the null space of \mathbf{e} is likely to intersect with $\text{null}(\mathbf{A})$ (Prop. 14).

There are also MIPURS sources where not all subsets have entropy. Consider a uniform distribution over $n-k$ coordinates with a fixed value in the remaining k coordinates (Prop. 13). Since $\text{null}(\mathbf{A})$ is unlikely to have non zero coordinates only at these fixed k coordinates, predicting the inner product remains difficult. Fortunately, multiplying a binary source where all subsets have entropy by a random vector produces a location source which is contained in MIPURS. It is this transformation we recommend for actual biometrics, see Section 3.1.

One can additionally build a good fuzzy extractor assuming a variant of multilinear maps [5]. Concurrent work of Galbraith and Zobernig [20] introduces a new subset sum assumption to build a secure sketch that is able to handle $t = \Theta(n)$ errors; they conjecture hardness for all securable distributions. A secure sketch is the error correction component in

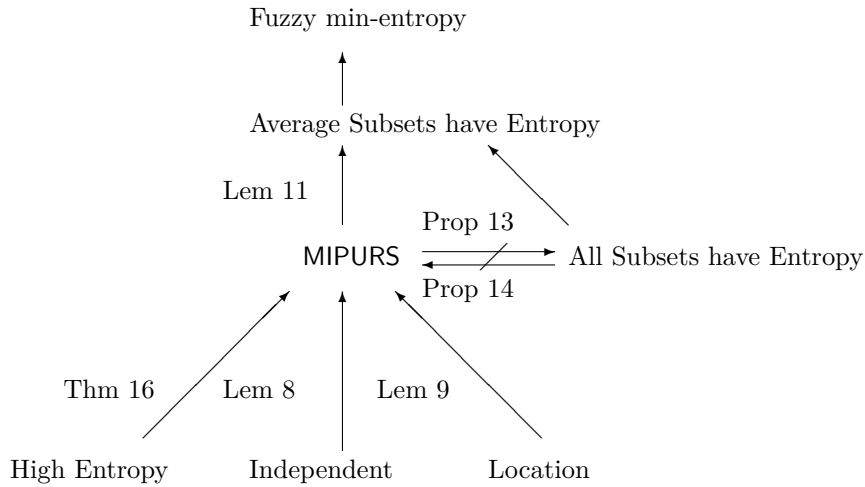


Figure 1 Implications between different types of supported distributions for fuzzy extraction. Arrows are implications. All shown implications are proper. Location sources are those that have random group elements in some locations with zeroes in other locations but it is hard to find a subset of all zero locations. A location source can be produced as the component wise product of a binary source where all subsets have entropy and a random vector of group elements. We consider this type of distribution in Section 3.1.

most fuzzy extractors. Their assumption is security of the cryptographic object and deserves continued study. A line of works [32, 33] use information-theoretic tools for error correction and computational tools to achieve additional properties. Those constructions embed a variant of the code offset. Table 1 summarizes constructions that use computational tools for the “correction” component and the traditional information theoretic analysis of the code offset construction.

2 Notation and Preliminaries

2.1 Notation

We use boldface to represent random variables, capitals to represent random variables over matrices or sets, and corresponding plain letters to represent samples. As one notable exception, we use ψ to represent samples from \mathbf{e} to avoid conflict with Euler’s number. We denote the exponential function with $\exp(\cdot)$. When defining ranges for parameters, we use $[$ and $]$ to indicate ranges inclusive of indicated values and $($ and $)$ to indicate ranges exclusive of the indicated values. For random variables \mathbf{x}_i over some alphabet \mathcal{Z} we denote the tuple by $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$. For a vector \mathbf{v} we denote the i th entry as \mathbf{v}_i . For a set of indices J , \mathbf{x}_J denotes the restriction of \mathbf{x} to the indices in J . For $m \in \mathbb{N}$, we let $[m] = \{1, \dots, m\}$, so that $[0] = \emptyset$. We use the notation $\text{span}(S)$ to denote the linear span of a set S of vectors and apply the notation to sequences of vectors without any special indication: If $F = (f_1, \dots, f_m)$ is a sequence of vectors, $\text{span}(F) = \text{span}(\{f_i \mid i \in [m]\})$. The *min-entropy* of a random variable \mathbf{x} is $H_\infty(\mathbf{x}) = -\log(\max_x \Pr[\mathbf{x} = x])$.

We consider the Hamming metric. Let \mathcal{Z} be a finite set and consider elements of \mathcal{Z}^n ; then we define $\text{dis}(x, y) = |\{i \mid x_i \neq y_i\}|$. U_n denotes the uniformly distributed random variable on $\{0, 1\}^n$. Logarithms are base 2. We denote the vector of all zero elements as 0 . We let \cdot_c

■ **Table 1** Comparison of computational techniques for fuzzy extractors. Many schemes [32, 33] use information theoretic techniques for information reconciliation and these are grouped together. These techniques all inherit the information theoretic analysis on the strength of information reconciliation. Reuse is denoted as \circ if reuse is supported with some assumption about how multiple readings are correlated and \bullet if no assumption is made. See Figure 1 for relations between supported distributions. The LWE works considered the setting when $k = \Theta(n)$ which leads to $t = \Theta(\log n)$. If one sets $k = \omega(\log n)$ one can achieve error tolerance of $o(n)$ using the analysis in this work, we thus present the more favorable regime for the above comparison.

Construction	Supported low entropy dist.	Reuse	Error rate (t)	Weakness
Code Offset [13]	-	\bullet	$\Theta(n)$	
LWE [2, 15]	Independent	\bullet	$o(n)$	
Subset sum [20]	Fuzzy min-ent.	\circ	$\Theta(n)$	Assumes security
Grey box obf. [5]	Fuzzy min-ent.	\circ	$\Theta(n)$	Multilinear maps
Digital Locker [8]	Average Subsets have Ent.	\bullet	$o(n)$	No <i>confidence</i> info
This work	MIPURS	\bullet	$o(n)$	

denote component-wise multiplication. In our theorems we consider a security parameter γ , when we use the term negligible and super polynomial, we assume other parameters are functions of γ . We elide the notational dependence of other parameters on γ .

2.2 Fuzzy Extractors

Our motivating application is a new fuzzy extractor that performs error correction “in the exponent.” A fuzzy extractor is a pair of algorithms designed to extract stable keys from a physical randomness source that has entropy but is noisy. If repeated readings are taken from the source one expects these readings to be close in an appropriate distance metric but not identical. We consider a generic group version of security (computational security is defined in [15], information-theoretic security in [13]).

Before introducing the definition, we review some notation from the generic group model; the model is reviewed in detail in the full version of this work. Let \mathbb{G} be a group of prime order q . For each element $r \in \mathbb{G}$ in the standard game, rather than receiving r , the adversary receives a handle $\sigma(r)$ where σ is a random function with a large range. The adversary is given access to an oracle, which we denote as $\mathcal{O}_{\mathbb{G}}^{\sigma}$, which given $x = \sigma(r_1), y = \sigma(r_2)$ computes $\sigma(\sigma^{-1}(x) + \sigma^{-1}(y))$; when σ can be inferred from context, we write $\mathcal{O}_{\mathbb{G}}$. Since the adversary receives random handles they cannot infer anything about the underlying group elements except using the group operation and testing equality. We assume throughout that the range of σ is large enough that the probability of a collision is statistically insignificant (that is $\ll 1/q$).

We overload the notation $\sigma(\cdot)$ to apply to tuples and, furthermore, adopt the convention that $\sigma(\cdot)$ is the identity on non-group elements; thus, it can be harmlessly applied to all inputs provided to the adversary. Specifically, when $z \stackrel{\text{def}}{=} z_1, \dots, z_n$ then $\sigma(z)$ only passes z_i through σ if $z_i \in \mathbb{G}_q$. For example, if $z = (r, \mathbf{A}, r^{\mathbf{Ax}+\mathbf{w}})$, then $\sigma(z) = (\sigma(r), \mathbf{A}, \sigma(r^{\mathbf{Ax}+\mathbf{w}}))$.

► **Definition 2.** Let \mathcal{E} be a family of probability distributions over the metric space $(\mathcal{M}, \text{dis})$. A pair of procedures $(\text{Gen} : \mathcal{M} \rightarrow \{0, 1\}^{\kappa} \times \{0, 1\}^*, \text{Rep} : \mathcal{M} \times \{0, 1\}^* \rightarrow \{0, 1\}^{\kappa})$ is an $(\mathcal{M}, \mathcal{E}, \kappa, t)$ -fuzzy extractor that is $(\epsilon_{\text{sec}}, m)$ -hard with error δ if Gen and Rep satisfy the following properties:

- Correctness: if $\text{dis}(\psi, \psi') \leq t$ and $(\text{key}, \text{pub}) \leftarrow \text{Gen}(\psi)$, then

$$\Pr[\text{Rep}(\psi', \text{pub}) = \text{key}] \geq 1 - \delta.$$

- Security: for any distribution $\mathbf{e} \in \mathcal{E}$, the string key is close to random conditioned on pub for all \mathcal{A} making at most m queries to the group oracle $\mathcal{O}_{\mathbb{G}}$, that is

$$|\Pr[\mathcal{A}^{\mathcal{O}_{\mathbb{G}}}(\sigma(\text{key}, \text{pub})) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\mathbb{G}}}(\sigma(U, \text{pub})) = 1]| \leq \epsilon_{\text{sec}}.$$

Where the probability of the statement is taken over $\sigma \xleftarrow{\$} \Sigma$ and $(\text{key}, \text{pub}) \leftarrow \text{Gen}(\mathbf{e})$.

We also assume that the adversary receives $\sigma(1)$. The errors are chosen before pub : if the error pattern between ψ and ψ' depends on the output of Gen , then there is no guarantee about the probability of correctness.

2.3 The MIPURS condition

In this section, we introduce our novel *Maximum Inner Product Unpredictable over Random Subspace* (MIPURS) condition.

► **Definition 3.** Let \mathbf{e} be a random variable taking values in \mathbb{F}_q^n and let \mathbf{A} be uniformly distributed over $\mathbb{F}_q^{n \times k}$ and independent of \mathbf{e} . We say that \mathbf{e} is a (k, β) -MIPURS distribution if for all random variables $\mathbf{b} \in \mathbb{F}_q^n, \mathbf{g} \in \mathbb{F}_q^k$ independent of \mathbf{e} (but depending arbitrarily on \mathbf{A} and each other)

$$\mathbb{E}_{\mathbf{A}} [\Pr [\langle \mathbf{b}, \mathbf{e} \rangle = \mathbf{g} \text{ and } \mathbf{b} \in \text{null}(\mathbf{A}) \setminus \vec{0}]] \leq \beta.$$

To see the equivalence between this definition and the game presented in the introduction, the random variables \mathbf{b} and \mathbf{g} can be seen as encoding the “adversary” and quantifying over all (\mathbf{b}, \mathbf{g}) is equivalent to considering all information-theoretic adversaries.

► **Theorem 4.** Let γ be a security parameter. Let q be a prime and $n, k \in \mathbb{Z}^+$ with $k \leq n \leq q$. Let $\mathbf{A} \in \mathbb{F}_q^{n \times k}$ and $\mathbf{x} \in \mathbb{F}_q^k$ be uniformly distributed. Let \mathbf{e} be a (k, β) -MIPURS distribution. Let $\mathbf{u} \in (\mathbb{F}_q)^n$ be uniformly distributed. Let Σ be the set of random functions with domain of size q and range of size q^3 . Then for all adversaries \mathcal{D} making at most m queries

$$\left| \Pr_{\sigma \xleftarrow{\$} \Sigma} [\mathcal{D}^{\mathcal{O}_{\mathbb{G}}}(\mathbf{A}, \sigma(\mathbf{A}\mathbf{x} + \mathbf{e})) = 1] - \Pr[\mathcal{D}^{\mathcal{O}_{\mathbb{G}}}(\mathbf{A}, \sigma(\mathbf{u})) = 1] \right| < \mu \left(\frac{3}{q} + \beta \right)$$

for $\mu = ((m + n + 2)(m + n + 1))^2 / 2$. If $1/q = \text{ngl}(\gamma), n, m = \text{poly}(\gamma)$, and $\beta = \text{ngl}(\gamma)$ then the statistical distance between the two cases is $\text{ngl}(\gamma)$.

In the above, the adversary is provided the code directly in the group, not its image in the handle space. The proof of Theorem 4 is a relatively straightforward application of the simultaneous oracle game introduced by Bishop et al. [4, Section 4]; this proof appears in the full version of this work.

3 A Fuzzy Extractor from Hardness of Code Offset in the Exponent

One can directly build a fuzzy extractor out of any \mathbf{e} that satisfies the MIPURS condition. To do so, one instantiates the code-offset construction “in the exponent” and then uses hardcore elements of \mathbf{x} as the key.

15:10 Code Offset in the Exponent

► **Construction 1.** Let γ be a security parameter, t be a distance, $k = \omega(\log \gamma)$, $\alpha \in \mathbb{Z}^+$, $\ell \in \mathbb{Z}^+$, let q be a prime and let \mathbb{G}_q be a cyclic group of order q . Let \mathbb{F}_q be the field with q elements. Suppose that \mathbf{e} and $\mathbf{e}' \in \mathbb{F}_q^n$, and let dis be the Hamming metric. Define (Gen, Rep) as follows:

<p>$\text{Gen}(\psi = \psi_1, \dots, \psi_n)$</p> <ol style="list-style-type: none"> 1. Sample generator r of \mathbb{G}_q. 2. Sample $A \leftarrow \mathbb{F}_q^{n \times (k+\alpha)}$, $x \leftarrow \mathbb{F}_q^{k+\alpha}$. 3. For $i = 1, \dots, n$: set $r^{c_i} = r^{A_i \cdot x + \psi_i}$. 4. Set $\text{key} = r^{x_0}, \dots, r^{x_{\alpha-1}}$. 5. Set $\text{pub} = (r, A, \{r^{c_i}\}_{i=1}^n)$. 6. Output (key, pub). 	<p>$\text{Rep}(\psi', \text{pub} = (r, A, r^{c_1} \dots r^{c_n}))$</p> <ol style="list-style-type: none"> 1. For $i = 1, \dots, n$, set $r^{c'_i} = r^{c_i} / r^{\psi'_i}$. 2. For $i = 1, \dots, \ell$: <ol style="list-style-type: none"> (i) Sample $J_i \subseteq \{1, \dots, n\}$ where $J_i = k + \alpha$. (ii) If $A_{J_i}^{-1}$ does not exist go to 2. (iii) Compute $r^s = r^{A_{J_i}^{-1} c'_{J_i}}$. (iv) Compute $r^{c''_i} = r^{A^s}$. (v) If $\text{dis}(r^{c'_i}, r^{c''_i}) \leq t$, output $r^{s_0}, \dots, r^{s_{\alpha}}$. 3. Output \perp.
---	--

► **Theorem 5.** Let c be a constant. Let all parameters be as in Construction 1. Let \mathcal{E} be the set of all (k, β) -MIPURS distributions. Suppose that

- $k' \stackrel{\text{def}}{=} k + \alpha = o(n)$ and $k' = \omega(\log n)$,
- t is such that $tk' \leq cn \log n$ for some constant c , which with the above implies $t = o(n)$,
- Let $\delta' > 0$ be some value,
- Let $\eta > 0$ be some constant and let $\ell = n^{2(1+\eta)c} \log \frac{1}{\delta'}$, and
- Let δ be some value such that $\delta \leq \delta' + \exp(-\Omega(n))$.

Then (Gen, Rep) is a $(\mathbb{F}_q^n, \mathcal{E}, |\mathbb{F}_q^\alpha|, t)$ -fuzzy extractor that is $(\epsilon_{\text{sec}}, m)$ -hard that is correct with probability $1 - \delta$ for all adversaries in the generic group model (making at most m queries) where

$$\epsilon_{\text{sec}} = \left(\frac{((m+n+2)(m+n+1))^2}{2} \right) \left(\frac{3}{q} + \beta \right).$$

The proof of Theorem 5 is shown in the full version of this work [12].

3.1 Handling binary sources

In this section we show one way to transform binary sources to a good MIPURS distribution and consider the associated impact on correctness. Assume that the source \mathbf{e} takes binary values and all subsets of \mathbf{e} are hard to predict, one can form a MIPURS distribution by multiplying by an auxiliary random and uniform random variable $\mathbf{r} \in \mathbb{F}_q^n$. This has the effect of placing random errors in the locations where $\mathbf{e}_i = 1$. Since decoding finds a subset without errors (it does not rely on the magnitude of errors) we can augment errors into random errors. We prove that this augmented vector is MIPURS in Section 4.

However, this transform creates a problem with decoding. When bits of \mathbf{e} are 1, denoted $\mathbf{e}_i = 1$ we cannot use location i for decoding as it is a random value (even if $\mathbf{e}'_j = 1$ as well). When one amplifies a binary \mathbf{e} , we recommend using another uniform random variable $\mathbf{y} \in \{0, 1\}^n$ and check when $\mathbf{y}_i \neq \mathbf{e}_i$ to indicate when to include a random error. Then in reproduction the algorithm should restrict to locations where $\mathbf{y}_i = \mathbf{e}_i$. Using Chernoff bounds one can show this subset is big enough and the error rate in this subset is not much higher than the overall error rate (except with negligible probability). If $k + \alpha$ is just barely $\omega(\log n)$ one can support error rates that are just barely $o(n)$.

To introduce the construction we first need to formalize the required property of the distribution \mathbf{e} . We introduce a notion called *all subsets have entropy*:

► **Definition 6.** Let a source $\mathbf{e} = \mathbf{e}_1, \dots, \mathbf{e}_n$ consist of n -bit binary strings. For some parameters k and β we say that the source \mathbf{e} is a source where **all k -subsets have entropy β** if $H_\infty(\mathbf{e}_{j_1}, \dots, \mathbf{e}_{j_k}) \geq \beta$ for any $1 \leq j_1, \dots, j_k \leq n$, $j_a \neq j_b$ for $a \neq b$.

► **Construction 2.** Let γ be a security parameter, t be a distance, $k = \omega(\log \gamma)$, $\alpha \in \mathbb{Z}^+$, q be a prime and let \mathbb{G}_q be some cycle group of order q . Let \mathbb{F}_q be the field with q elements. Let $\mathcal{E} \in \{0, 1\}^n$ and let dis be the Hamming metric. Let $\tau = \max(0.01, t/n)$. Define (Gen, Rep) as follows:

<p>Gen($\psi = \psi_1, \dots, \psi_n$)</p> <ol style="list-style-type: none"> 1. Sample random generator r of \mathbb{G}_q. 2. Sample $A \leftarrow (\mathbb{F}_q)^{n \times (k+\alpha)}$, 3. Sample $x \leftarrow (\mathbb{F}_q)^{k+\alpha}$. 4. Sample $y \xleftarrow{\\$} \{0, 1\}^n$. 5. For $i = 1, \dots, n$: <ol style="list-style-type: none"> (i) If $\psi_i = y_i$, set $r^{c_i} = r^{A_i \cdot x}$. (ii) Else set $r^{c_i} \xleftarrow{\\$} \mathbb{G}_q$. 6. Set $\text{key} = r^{x_0 \dots x_{\alpha-1}}$. 7. Set $\text{pub} = (r, y, A, \{r^{c_i}\}_{i=1}^n)$. 8. Output (key, pub). 	<p>Rep($\psi', \text{pub} = (r, y, A, r^{c_1} \dots r^{c_\ell})$)</p> <ol style="list-style-type: none"> 1. Let $\mathcal{I} = \{i \psi'_i = y_i\}$. 2. For $i = 1, \dots, \ell$: <ol style="list-style-type: none"> (i) Choose random $J_i \subseteq \mathcal{I}$, with $J_i = k$. (ii) If $A_{J_i}^{-1}$ does not exist, output \perp. (iii) Compute $r^s = r^{A_{J_i}^{-1} c_{J_i}}$. (iv) Compute $r^{c'} = r^{A(A_{J_i}^{-1} c_{J_i})}$. (v) If $\text{dis}(c_{\mathcal{I}}, c'_{\mathcal{I}}) \leq 2 c_{\mathcal{I}} \tau$, output $r^{s_0}, \dots, r^{s_{\alpha-1}}$. 3. Output \perp.
--	---

► **Theorem 7.** Let all parameters be as in Construction 2. Let $\gamma \in \mathbb{N}$ and let \mathcal{E} be the set of all sources where all $(k - \gamma)$ -subsets have entropy β (Definition 6) over $\{0, 1\}^n$. Then (Gen, Rep) is a $(\{0, 1\}^n, \mathcal{E}, |\mathbb{F}_q^\alpha|, t)$ -fuzzy extractor that is $(\epsilon_{\text{sec}}, m)$ -hard for all adversaries in the generic group model (making at most m queries) where

$$\epsilon_{\text{sec}} = \left(\frac{((m+n+2)(m+n+1))^2}{2} \right) \left(\frac{4}{q} + 2^{-\beta} + \left(\frac{(k-\gamma) \binom{n}{k-\gamma-1}}{q^{\gamma+1}} \right) \right).$$

Furthermore, suppose that

- $k' \stackrel{\text{def}}{=} k + \alpha = o(n)$ and $k' = \omega(\log n)$,
- t is such that $tk' \leq cn \log n$ for some constant c , which with the above implies $t = o(n)$,
- Let $\delta' > 0$ be some value.
- Let $\eta > 0$ be some constant.
- Let $\ell = n^{2(1+\eta)c} \log \frac{1}{\delta'}$, (if $tk' = o(n \log n)$ setting $\ell = n \log 1/\delta'$ suffices)

Then there is some function negligible $\text{ngl}(n)$ such that the Rep is correct with probability $1 - \delta' - \text{ngl}(n)$.

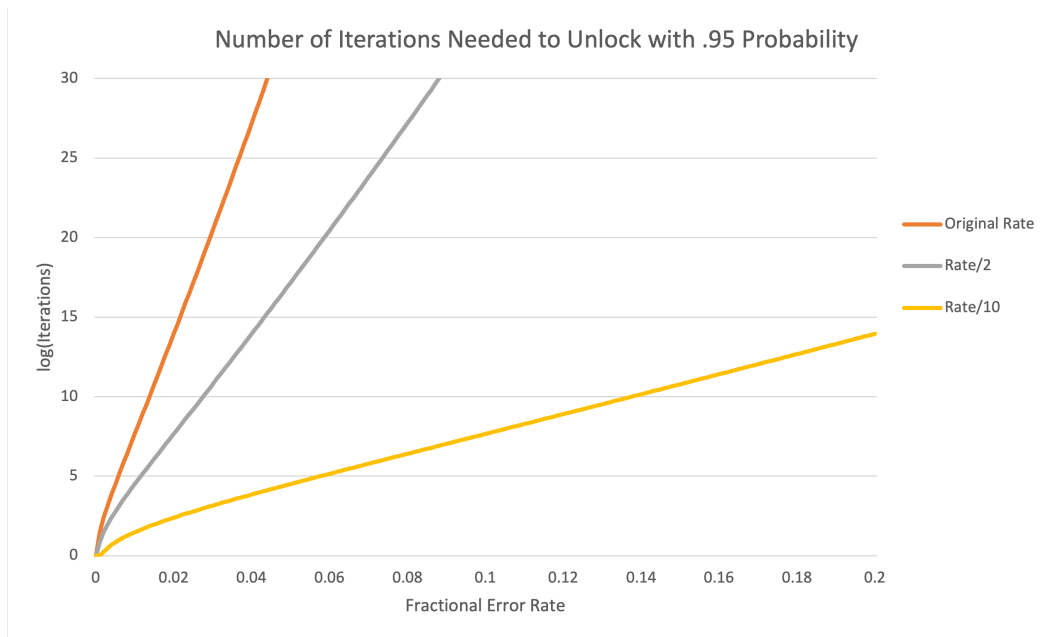
We defer proving Theorem 7 to the full version of this work [12].

3.2 The power of confidence information

Most PUFs and biometrics demonstrate a constant error rate $\tau = t/n$. This is higher than the correction capacity of our construction and Canetti et al.'s digital locker construction [8]. However, existing fuzzy extractors that support constant τ do not support the low entropy distributions found in practice.

While code offset in the exponent is not designed for constant error rates it is efficient for small constant τ . As described in the introduction for the case of PUFs and biometrics, using confidence information can lead to a multiplicative decrease in the effective error rate of bits chosen for information set decoding. The important tradeoff is between the fractional error rate $\tau = t/n$ and the number of required iterations.

15:12 Code Offset in the Exponent



■ **Figure 2** Expected number of iterations ℓ to have Rep output a value with .95 probability across error rate. Three lines represent original error rate t and two reduced error rates of $t/2$ and $t/10$ that may be achievable by using confidence information. Note that the y-axis is in log scale.

We observe, for practical parameters, multiplicative changes in τ lead to exponential changes in the required iterations ℓ . To demonstrate we consider the following parameters: a source of length $n = 1024$ (common for the iris), a subset size of $k = 128$, and an output key of a single group element ($\alpha = 1$). Figure 2 shows how $\log \ell$ increases for different τ . Three lines represent the original error rate and two potential reduced error rates (multiplicative decreases of 2 and 10 respectively). Figure 2 considers τ steps of .001. Between 0 and .06, each step of .001 increases $\log \ell$ by .667 (r^2 value of .999).

As mentioned in the introduction, Canetti et al. [8] digital locker⁶ condition for security is that *average* subsets have entropy. A distribution satisfying MIPURS implies that average subsets have entropy (see Section 4.3). Since code offset in the exponent allows the adversary to test any subset, average subsets having entropy does not suffice (see Section 1.4). Section 3.1 showed how to handle distributions where *all* subsets have entropy by multiplying by a random error vector. Unfortunately, as we show in Section 4.4, MIPURS and all subsets have entropy are incomparable notions creating a barrier to removing this random vector in Construction 2.

Reusability

Reusability is the ability to support multiple independent enrollments of the same value, allowing users to reuse the same biometric or PUF, for example, with multiple noncooperating providers. More precisely, the algorithm Gen may be run multiple times on correlated readings e^1, \dots, e^ρ of a given source. Each time, Gen will produce a different pair of values

⁶ Intuitively, a digital locker is a symmetric encryption that is semantically secure even when instantiated with keys that are correlated and only have entropy [10].

$(\text{key}^1, \text{pub}^1), \dots, (\text{key}^\rho, \text{pub}^\rho)$. Security for each extracted string key^i should hold even in the presence of all the helper strings $(\text{pub}^1, \dots, \text{pub}^\rho)$. The reproduction procedure Rep at the i th provider still obtains only a single \mathbf{e}' close to \mathbf{e}^i and uses a single helper string pub_i . Because providers may not trust each other key_i should be secure even when all key_j for $j \neq i$ are also given to the adversary. In the full version of this work [12] we show that Construction 1 is reusable if a random generator is used with each enrollment.

4 Characterizing MIPURS

Definition 3 of MIPURS is admittedly unwieldy. It considers a property of a distribution $\mathbf{e} \in \mathbb{F}_q^n$ with respect to a random matrix. We turn to characterizing distributions that satisfy MIPURS. We begin with easier distributions and conclude with the general entropy case in Section 4.5. Throughout, we consider a prime order group \mathbb{G} of prime size q , a random linear code $\mathbf{A} \in \mathbb{F}_q^{n \times k}$ and the null space $\mathbf{B} \stackrel{\text{def}}{=} \text{null}(\mathbf{A})$.

4.1 Independent Sources \subset MIPURS

In most versions of LWE, each error coordinate is independently distributed and contributes entropy. Examples include the discretized Gaussian introduced by Regev [28, 29], and a uniform interval introduced by Döttling and Müller-Quade [14]. We show that these distributions fit within our MIPURS characterization.

► **Lemma 8.** *Let $\mathbf{e} = \mathbf{e}_1, \dots, \mathbf{e}_n \in \mathbb{F}_q^n$ be a distribution where each \mathbf{e}_i is independently sampled. Let $\alpha = \min_{1 \leq i \leq n} H_\infty(\mathbf{e}_i)$. For any $k \leq n$, \mathbf{e} is a (k, β) -MIPURS distribution for $\beta = 2^{-\alpha}$.*

Proof of Lemma 8. Consider a fixed element $b \neq 0$ in \mathbf{B} . Since the components of \mathbf{e} are independent, predicting $\langle b, \mathbf{e} \rangle$ is at least as hard as predicting \mathbf{e}_i for each i such that $\mathbf{b}_i \neq 0$. This can be seen by fixing b and \mathbf{e}_j for $j \neq i$ and noting that the value of \mathbf{e}_i then uniquely determines $\langle b, \mathbf{e} \rangle$. Since $b \neq 0$ there exists at least one such i . Thus,

$$\Pr_{\mathbf{B}} \left[\max_g \max_{b \in \mathbf{B} \setminus \{0\}} \Pr_{\mathbf{e}}[\langle b, \mathbf{e} \rangle = g] \right] \leq 2^{-\alpha} \stackrel{\text{def}}{=} \beta. \quad \blacktriangleleft$$

4.2 Location Sources \subset MIPURS

Next, we consider \mathbf{e}' given by the coordinatewise product of a uniform vector $\mathbf{r} \in \mathbb{F}_q^n$ and a “selection vector” $\mathbf{e} \in \{0, 1\}^n$: that is, $\mathbf{e}'_i = \mathbf{r}_i \cdot_c \mathbf{e}_i$ where all large enough subsets of \mathbf{e} are unpredictable (\cdot_c is component-wise multiplication). Location sources are important for applications (see Section 3).

► **Lemma 9.** *Let $\gamma \in \mathbb{N}$ and $k \in \mathbb{Z}^+$. Let $\mathbf{e} \in \{0, 1\}^n$ be a distribution where all $(k - \gamma)$ -subsets have entropy α . Define the distribution \mathbf{e}' as the coordinatewise product of a uniform vector $\mathbf{r} \in \mathbb{F}_q^n$ and \mathbf{e} : that is, $\mathbf{e}'_i = \mathbf{e}_i \cdot_c \mathbf{r}_i$. Then the distribution \mathbf{e}' is a (k, β) -MIPURS distribution for $\beta = 2^{-\alpha} + ((k - \gamma) \binom{n}{k - \gamma - 1}) / q^{\gamma + 1}$.*

Proof of Lemma 9. We use $\mathbf{A} \in \mathbb{F}_q^{n \times k}$ to represent the random matrix from the definition of a MIPURS distribution and let $\mathbf{B} \in \mathbb{F}_q^{n \times n - k}$ represent its null space. We start by bounding the “minimum distance” of \mathbf{B} , that is, the minimum weight of a non-zero element of $\mathbf{B} = \text{null}(\mathbf{A})$. Observe that the number of vectors in \mathbb{F}_q^n of weight less than $k - \gamma$ is

$$\sum_{j=0}^{k-\gamma-1} \binom{n}{j} q^j \leq (k - \gamma) \binom{n}{k - \gamma - 1} q^{k - \gamma - 1}.$$

15:14 Code Offset in the Exponent

The probability that any fixed, nonzero vector lies x in \mathbf{B} is q^{-k} , as it must annihilate k independent, uniform linear equations. That is, $\sum_i x_i \mathbf{A}_{is} = 0$ for each $1 \leq s \leq k$. Thus

$$\mathbb{E}[|\{w \in \text{null}(\mathbf{A}) \setminus 0 \mid \text{wt}(w) < k - \gamma\}|] \leq (k - \gamma) \binom{n}{k - \gamma - 1} q^{-\gamma - 1}. \quad (1)$$

By Markov's inequality, the probability that there is at least one such small weight vector in $\text{null}(\mathbf{A})$ is no more than the expected number of such vectors. Hence

$$\Pr[\exists w \in \text{null}(\mathbf{A}) \setminus 0, \text{wt}(w) < k - \gamma] \leq (k - \gamma) \binom{n}{k - \gamma - 1} q^{-\gamma - 1}.$$

For some \mathbf{b} in the span of \mathbf{B} with weight at least $k - \gamma$, consider the product $\langle \mathbf{b}, \mathbf{e}' \rangle = \sum_{i=1}^n \mathbf{b}_i \cdot \mathbf{e}_i \cdot \mathbf{r}_i$. Define \mathcal{I} as the set of nonzero coordinates in \mathbf{b} . With probability at least $1 - 2^{-\alpha}$ there is some nonzero coordinate in $\mathcal{e}_{\mathcal{I}}$. Conditioned on this fact this means that at least one value \mathbf{r}_i is included in the inner product. Thus, the entropy of the inner product is bounded below by the entropy of $\mathbf{e}_i \cdot \mathbf{r}_i$ which since $\mathbf{e}_i \neq 0$ is bounded by the entropy of \mathbf{r}_i . In this case, the prediction probability of an inner product (and therefore a single element of) is $1/q$. The argument concludes by assuming perfect predictability when there exists \mathbf{b} in \mathbf{B} with weight of at most $k - \gamma - 1$. ◀

4.3 MIPURS \subset Average Subsets Have Entropy

As mentioned in the Introduction, Canetti et al. [8] showed a fuzzy extractor construction for all sources where an average subset has entropy:⁷

► **Definition 10** ([8] average subsets have entropy). *Let the source $\mathbf{e} = \mathbf{e}_1, \dots, \mathbf{e}_n$ consist of strings of length n over some arbitrary alphabet \mathcal{Z} . We say that the source \mathbf{e} is a source with a k -average subsets have entropy β if*

$$\mathbb{E}_{j_1, \dots, j_k \stackrel{\$}{\leftarrow} [1, \dots, n], j_\alpha \neq j_\gamma} \left(\max_z \{ \Pr[(\mathbf{e}_{j_1}, \dots, \mathbf{e}_{j_k}) = z \mid j_1, \dots, j_k] \} \right) \leq \beta.$$

We now show that a MIPURS distribution also has that average subsets have entropy.

► **Lemma 11.** *Let $\mathbf{e} = \mathbf{e}_1, \dots, \mathbf{e}_n$ be a source over alphabet \mathcal{Z} such that \mathbf{e} is (k, β) -MIPURS. Then \mathbf{e} has (k', β') -entropy samples for any k' and*

$$\beta' = \frac{\beta}{\left(1 - \frac{(q^{k' - (k+1)})}{(2^{k'} \binom{n}{k'})} \right)}.$$

Proof of Lemma 11. We proceed by contradiction, that is suppose that \mathbf{e} does not have k', β' entropy samples. That is,

$$\mathbb{E}_{j_1, \dots, j'_k \stackrel{\$}{\leftarrow} [1, \dots, n], j_\alpha \neq j_\gamma} \left(\max_z \{ \Pr[(\mathbf{e}_{j_1}, \dots, \mathbf{e}_{j_k}) = z \mid j_1, \dots, j'_k] \} \right) > \beta'.$$

We consider the following definition of \mathbf{b}, \mathbf{g} in the MIPURS game:

1. Receive input \mathbf{A} , compute $\mathbf{B} = \text{null}(\mathbf{A})$.

⁷ We make a small modification to their definition changing to sampling without replacement.

2. Select random $\mathbf{b} \in \mathbf{B}$ such that $\text{wt}(\mathbf{b}) \leq k'$, $\mathbf{b} \neq \mathbf{0}$. If no such \mathbf{b} exists output $\mathbf{b} = \mathbf{0}$, $\mathbf{g} = \mathbf{0}$.
3. Define \mathcal{I} as the set of nonzero locations in \mathbf{b} . If $|\mathcal{I}| < k'$ insert random distinct locations until $|\mathcal{I}| = k'$.
4. Compute $z = \arg \max_z \{\Pr[\mathbf{e}_{\mathcal{I}} = z \mid \mathcal{I}]\}$.
5. Output $\mathbf{g} = \langle \mathbf{b}, z \rangle$.

If z is the correct prediction for $\mathbf{e}_{\mathcal{I}}$ then $\mathbf{g} = \langle \mathbf{b}, z \rangle = \langle \mathbf{b}, \mathbf{e} \rangle$. As noted above, the probability of any particular value nonzero \mathbf{b} being in \mathbf{B} is q^{-k} . Thus, conditioned on finding a good \mathbf{b} , the distribution of the random variable \mathbf{b} is exactly that of a uniform weight k' value. This implies that $\mathbb{E}_{\mathbf{b}}[\max_z \{\Pr[(\mathbf{e}_{\mathcal{I}}) = z \mid \mathcal{I}]\}] > \beta'$. It remains to analyze the probability that \mathbf{B} contains no vectors of weight k' . Here we derive an elementary bound, asymptotic formulations exist in the information theory literature [21, Theorem 1.1].

► **Lemma 12.** *Let V denote a random subspace of \mathbb{F}_q^n of dimension κ . Let W_ℓ denote the subset of \mathbb{F}_q^n consisting of all vectors with weight ℓ , then*

$$\Pr[V \cap W_\ell = \emptyset] \leq \frac{(q^n - 1)}{\binom{n}{\ell}(q-1)^{\ell-1}(q^\kappa - 1)}.$$

Proof of Lemma 12. We begin by noting that $|W_\ell| = \binom{n}{\ell}(q-1)^\ell$. For a vector $\vec{v} \in W_\ell$, define $X_{\vec{v}} = 1$ if $\vec{v} \in V$ and 0 otherwise. Then

$$\mathbb{E} \left[\sum_{\vec{v} \in W_\ell} X_{\vec{v}} \right] = \binom{n}{\ell} (q-1)^\ell \frac{q^\kappa - 1}{q^n - 1}.$$

We wish to compute the second moment of the sum $\sum X_{\vec{v}}$. We have

$$\begin{aligned} \mathbb{E} \left[\sum_{\vec{v}, \vec{w} \in W_\ell} X_{\vec{v}} X_{\vec{w}} \right] &= \mathbb{E} \left[\sum_{\substack{\vec{v}, \vec{w} \in W_\ell \\ \vec{v}, \vec{w} \text{ independent}}} X_{\vec{v}} X_{\vec{w}} \right] + \mathbb{E} \left[\sum_{\substack{\vec{v}, \vec{w} \in W_\ell \\ \vec{v}, \vec{w} \text{ dependent}}} X_{\vec{v}} X_{\vec{w}} \right] \\ &\leq \binom{n}{\ell} (q-1)^\ell \left(\binom{n}{\ell} (q-1)^\ell - (q-1) \right) \max_{\text{indep. } \vec{v}, \vec{w}} \Pr[\vec{v}, \vec{w} \in V] \\ &\quad + \binom{n}{\ell} (q-1)^{\ell+1} \max_{\text{dependent } \vec{v}, \vec{w}} \Pr[\vec{v}, \vec{w} \in V] \\ &\leq \underbrace{\left(\binom{n}{\ell} (q-1)^\ell \right)^2 \frac{(q^\kappa - 1)(q^{\kappa-1} - 1)}{(q^n - 1)(q^{n-1} - 1)}}_{(\ddagger)} + \binom{n}{\ell} (q-1)^{\ell+1} \frac{q^\kappa - 1}{q^n - 1}. \end{aligned}$$

Note that $(m-t)/(n-t) < m/n$ assuming that $t \leq m < n$ and hence that that

$$(\ddagger) = \left(\binom{n}{\ell} (q-1)^\ell \right)^2 \frac{(q^\kappa - 1)(q^\kappa - q)}{(q^n - 1)(q^n - q)} \leq \left(\binom{n}{\ell} (q-1)^\ell \right)^2 \frac{(q^\kappa - 1)^2}{(q^n - 1)^2} \leq \mathbb{E} \left[\sum X_{\vec{v}} \right]^2.$$

It follows that

$$\text{Var} \left[\sum X_{\vec{v}} \right] = \mathbb{E} \left[\left(\sum X_{\vec{v}} \right)^2 \right] - \mathbb{E} \left[\sum X_{\vec{v}} \right]^2 \leq \binom{n}{\ell} (q-1)^{\ell+1} \frac{q^\kappa - 1}{q^n - 1}.$$

Then using Chebyshev's inequality with a constant of

$$\alpha = \sqrt{\left(\binom{n}{\ell} (q-1)^\ell \frac{q^\kappa - 1}{q^n - 1} \right)^2 / \left(\binom{n}{\ell} (q-1)^{\ell+1} \frac{q^\kappa - 1}{q^n - 1} \right)}$$

15:16 Code Offset in the Exponent

one finds:

$$\Pr \left[\sum X_{\vec{v}} = 0 \right] \leq \frac{\text{Var}[\sum X_{\vec{v}}]}{\mathbb{E}[\sum X_{\vec{v}}]^2} \leq \frac{\binom{n}{\ell}(q-1)^{\ell+1} \frac{q^\kappa - 1}{q^{n-1}}}{\left(\binom{n}{\ell}(q-1)^\ell \frac{q^\kappa - 1}{q^{n-1}} \right)^2} \leq \frac{(q^n - 1)}{\left(\binom{n}{\ell}(q-1)^{\ell-1}(q^\kappa - 1) \right)}.$$

This completes the proof of Lemma 12. \blacktriangleleft

Thus, for $\dim(\mathbf{B}) \geq n - k$ it is true that for any k' :

$$\Pr[\mathbf{B} \cap W_{k'} = 0] \leq \frac{(q^n - 1)}{\binom{n}{k'}(q-1)^{k'-1}(q^{n-k} - 1)} \leq \frac{q^n}{2^{k'} \binom{n}{k'} q^{n-k+k'-1}} = \frac{q^{k'-(k+1)}}{2^{k'} \binom{n}{k'}}.$$

We note that the overall success of prediction of \mathbf{b}, \mathbf{g} in the MIPURS game is bounded below by $\Pr[\mathbf{B} \cap W_{k'} = 0] * 0 + (1 - \Pr[\mathbf{B} \cap W_{k'} = 0]) * \beta' = \beta$. This completes the proof of Lemma 11. \blacktriangleleft

4.4 MIPURS and all subsets have entropy

We now consider the relationship between MIPURS and all subsets have entropy. Recall, that we showed that for a distribution \mathbf{e} where all subsets have entropy multiplying by a random vector produced a MIPURS distribution. With two simple examples, we show that MIPURS is not contained by all subsets have entropy and all subsets have entropy is not contained by MIPURS.

► **Proposition 13** (MIPURS $\not\rightarrow$ all subsets have entropy). *Define $\mathbf{e} \in \mathbb{F}_q^n$ as the distribution that is fixed in the first k positions and uniform in all other positions. Clearly for any $\beta > 0$ it does not hold that all k -subsets have entropy. Furthermore, \mathbf{e} is (k, β) -MIPURS for $\beta \geq (1 - \frac{1}{q^k} - \frac{k}{q^{n-k}}) \log q$.*

To show the above proposition, assume perfect predictability in the MIPURS game in the case when \mathbf{A} is not full rank or when $1^k || 0^{n-k}$ is in $\text{null}(\mathbf{A})$. Otherwise, full entropy results from the same argument as Lemma 8.

For the second direction we assume that \mathbf{e} is a Reed-Solomon [27] code (the counterexample is similar to the one presented in Section 1.3). For the field \mathbb{F}_q of size q , a message length k , and code length n , such that $k \leq n \leq q$, define the Vandermonde matrix \mathbf{V} where the i th row, $\mathbf{V}_i = [i^0, i^1, \dots, i^k]$. The Reed Solomon Code $\mathbb{RS}(n, k, q)$ is the set of all points $\mathbf{V}\mathbf{x}$ where $\mathbf{x} \in \mathbb{F}_q^k$.

► **Proposition 14** (all subsets have entropy $\not\rightarrow$ MIPURS). *Let $k < n/2$ and let \mathbf{e} be the uniform distribution over $\mathbb{RS}(n, n-k-1, q)$ then all k subsets of \mathbf{e} have entropy $k \log q$. Furthermore, \mathbf{e} is **not** (k, β) -MIPURS for any $\beta < 1$.*

Note $\dim(\text{null}(\mathbf{A})) \geq n - k$ and thus $\text{null}(\mathbf{A})$ and $\text{null}(\mathbb{RS}(n, n-k-1, q))$ are guaranteed to have a nontrivial intersection. The result follows by picking some \mathbf{b} in this intersection and setting $g = 0$.

4.5 High entropy \subset MIPURS

We now turn to the general entropy condition: MIPURS is hard for all distribution where the min-entropy exceeds $\log q^{n-k}$ (by a super logarithmic amount). For conciseness, we introduce $\kappa \stackrel{\text{def}}{=} n - k$.

The adversary is given a generating matrix of the code, \mathbf{A} ; this determines $\mathbf{B} = \text{null}(\mathbf{A})$. Our proof is divided into three parts. Denote by E a set of possible error vectors.

1. Theorem 16: We show that the number of vectors $\psi \in E$ that are likely to have 0 inner product with an adversarially chosen vector in \mathbf{B} is small. Intuitively, we show that this set is “not much larger than a κ -dimensional subspace.”
2. Theorem 22: We then show it is difficult to predict the value of the inner product: even if the adversary may select arbitrarily coupled \mathbf{b} and \mathbf{g} , it is difficult to achieve $\langle \mathbf{b}, \psi \rangle = \mathbf{g}$.
3. Lemma 24: We show that any distribution \mathbf{e} with sufficient entropy cannot lie in the set of *predictable* error vectors E with high probability.

We codify the set of possible adversarial strategies by introducing a notion of κ -induced random variables. For the moment, we assume that \mathbf{B} is a uniformly selected subspace of dimension exactly κ ; at the end of the proof we remove this restriction to apply these results when \mathbf{B} has the distribution given by $\text{null}(\mathbf{A})$ (Corollary 25).

► **Definition 15.** Let \mathbf{b} be a random variable taking values in \mathbb{F}_q^n . We say that \mathbf{b} is κ -induced if there exists a (typically dependent) random variable \mathbf{B} , uniform on the collection of κ -dimensional subspaces of \mathbb{F}_q^n , so that $\mathbf{b} \in \mathbf{B}$ and $\mathbf{b} \neq \vec{0}$ with certainty: $\Pr[\mathbf{b} \in \mathbf{B} \wedge \mathbf{b} \neq \vec{0}] = 1$. Note, in fact, that the random variables \mathbf{B} and \mathbf{b} are necessarily dependent unless $n = \kappa$.

It suffices to consider the maximum probability in Definition 3 with respect to κ -induced random variables. This is because for any \mathbf{b} that is not κ -induced we can find another \mathbf{b} that is κ induced that does no worse in the game in Definition 3. For example when \mathbf{b} is not in \mathbf{B} or is the zero vector, one can replace \mathbf{b} with a random element in the span of \mathbf{B} .

We now show that if the set E is large enough there is no strategy for \mathbf{b} that guarantees $\langle \mathbf{b}, \psi \rangle = 0$ with significant probability. The next theorem (Thm. 22) will, more generally, consider prediction of the inner product itself. For a κ induced random variable \mathbf{b} , define

$$E_\epsilon^{(\mathbf{b}, 0)} = \left\{ f \in \mathbb{F}_q^n \mid \Pr_{\mathbf{b}}[\langle \mathbf{b}, f \rangle = 0] \geq \epsilon \right\}.$$

When \mathbf{b} can be inferred from context, we simply refer to this set as E_ϵ . Then define $P_{\kappa, \epsilon} = \max_{\mathbf{b}} |E_\epsilon^{(\mathbf{b}, 0)}|$ where the maximum is over all κ -induced random variables in \mathbb{F}_q^n .

► **Theorem 16.** Let q be a prime and let $d > 1$, $\kappa, m, \eta \in \mathbb{Z}^+$ be parameters for which $\kappa \leq n$. Then assuming $P_{\kappa, \epsilon} > d \cdot q^\kappa$ we must have

$$\epsilon \leq \binom{\kappa + \eta}{m} + \binom{m}{\kappa} \left(\binom{m}{\eta} \left(\frac{1}{d} \right)^\eta + \left(\frac{2}{q} \right) \right).$$

Before proving Theorem 16, we introduce and prove two combinatorial lemmas (18 and 20). We then proceed with the proof of Theorem 16. The major challenge is that the set E_ϵ (for a particular \mathbf{b}) is typically not a linear subspace; these results show that is has reasonable “approximate linear” structure. We begin with the notion of *linear density* to measure, intuitively, how close the set is to linear.

► **Definition 17.** The ℓ -linear density of a sequence of vectors $F = (f^1, \dots, f^m)$, with each $f^i \in \mathbb{F}_q^n$, is the maximum number of entries that are covered by a subspace of dimension ℓ . Formally,

$$\Delta^\ell(F) = \max_{V, \dim(V)=\ell} |\{i \mid f^i \in V\}|.$$

► **Lemma 18.** Let q be a prime and let $n, \ell \in \mathbb{Z}^+$ satisfy $\ell \leq n$. Let $E \subset \mathbb{F}_q^n$ satisfy $|E| \geq q^\ell$ and let $\mathbf{F} = (\mathbf{f}^1, \dots, \mathbf{f}^m)$ be a sequence of uniformly and independently chosen elements of E . Define d so that $|E| = dq^\ell$; then for any $\eta \geq 0$,

$$\Pr_{\mathbf{F}}[\Delta^\ell(\mathbf{F}) \geq \ell + \eta] \leq \binom{m}{\ell} \binom{m - \ell}{\eta} \left(\frac{1}{d} \right)^\eta.$$

15:18 Code Offset in the Exponent

Proof of Lemma 18. By the definition of linear density, if $\Delta^\ell(\mathbf{F}) \geq \ell + \eta$ there must be at least one subset of $\ell + \eta$ indices $I \subset [m]$ so that $\{\mathbf{f}^i \mid i \in I\}$ is contained in a subspace of dimension ℓ . In order for a subset I to have this property, there must be a partition of I into a disjoint union $S \cup L$, where S has cardinality ℓ and T indexes the remaining η “lucky” vectors that lie in the span of the vectors given by S . Formally, $\forall t \in T, \mathbf{f}^t \in \text{span}(\{\mathbf{f}^s \mid s \in S\})$.

Fix, for the moment, ℓ indices of \mathbf{F} to identify a candidate subset of vectors to play the role of S and η indices of \mathbf{F} to identify a candidate set T . The probability that each of the η vectors indexed by T lie in the space spanned by S is clearly no more than $(q^\ell/|E|)^\eta \leq (1/d)^\eta$. Taking the union bound over these choices of indices completes the argument: The probability of a sequence is no more than $\binom{m}{\ell} \binom{m-\ell}{\eta} d^{-\eta}$, as desired. \blacktriangleleft

Before introducing our second combinatorial lemma (Lem 20), we need a Lemma bounding the probability of a fixed subspace having a nontrivial intersection with a random subspace.

► **Lemma 19.** *Let q be a prime and $\kappa, n \in \mathbb{N}$ with $\kappa \leq n$. Let \mathbf{V} be a random variable uniform on the set of all κ -dimensional subspaces of \mathbb{F}_q^n . Let W be a fixed subspace of dimension ℓ . Then*

$$\Pr[\mathbf{V} \cap W \neq \{0\}] \leq q^{\kappa+\ell-(n+1)} \cdot \left(\frac{q}{q-1} \right).$$

Proof of Lemma 19. Let \mathcal{L} denote the set of all 1-dimensional subspaces in W . Each 1-dimensional subspace is described by an equivalence class of $q-1$ vectors under the relation $x \sim y \Leftrightarrow \exists \lambda \in \mathbb{F}_q^*, \lambda x = y$. Thus $|\mathcal{L}| = (q^\ell - 1)/(q-1) \leq q^{\ell-1}(q/(q-1))$. Then

$$\Pr[\mathbf{V} \cap W \neq \{0\}] \leq \sum_{L \in \mathcal{L}} \Pr[L \subset \mathbf{V}] \leq |\mathcal{L}| \max_{v \in \mathbb{F}_q^n \setminus \{0\}} \Pr[v \in \mathbf{V}] \leq q^{\kappa+\ell-(n+1)} \left(\frac{q}{q-1} \right),$$

where we recall the fact that for any particular fixed nonzero vector v , $\Pr[v \in \mathbf{V}] = \frac{q^\kappa - 1}{q^n - 1} \leq q^{\kappa-n}$. \blacktriangleleft

► **Lemma 20.** *Let q be a prime, let $\ell, \kappa, n \in \mathbb{Z}^+$ satisfy $\ell, \kappa \leq n$. Let $F = (f^1, \dots, f^m)$ be a sequence of elements of \mathbb{F}_q^n with $\dim(\text{span}(F)) \geq \ell$. Then, for any κ -induced random variable \mathbf{b} taking values in \mathbb{F}_q^n ,*

$$\Pr_{\mathbf{b}}[|\{i \mid \langle \mathbf{b}, f^i \rangle = 0\}| \geq \Delta^\ell(F)] \leq \binom{m}{\ell} q^{\kappa-\ell-1} \left(\frac{q}{q-1} \right) \leq 2 \binom{m}{\ell} q^{\kappa-\ell-1}.$$

Proof of Lemma 20. Let \mathcal{V}_F denote the collection of all ℓ -dimensional subspaces of \mathbb{F}_q^n spanned by subsets of elements in the sequence F . That is,

$$\mathcal{V}_F = \{V \mid V = \text{span}(\{f^i \mid i \in I\}), I \subset [m], \dim(V) = \ell\}.$$

Then $|\mathcal{V}_F| \leq \binom{m}{\ell}$, as each such subspace is spanned by at least one subset of F of size ℓ . As $\dim(\text{span}(F)) \geq \ell$, the set \mathcal{V}_F is nonempty.

Observe that if $I \subset [m]$ has cardinality at least $\Delta^\ell(F)$ then, by definition, $\dim(\text{span}(\{f^i \mid i \in I\})) \geq \ell$; otherwise, an additional element of F could be added to the set indexed by I to yield a set of size exceeding $\Delta^\ell(F)$ which still lies in a subspace of dimension ℓ (contradicting the definition of Δ^ℓ). Note in the case that $m = \ell$ (and there is no element to add) then $\Delta^\ell(F) = \ell = \dim(\text{span}(\{f^i \mid i \in I\}))$. Thus, if $I \subset [m]$ has cardinality at least $\Delta^\ell(F)$, there must be some $V \in \mathcal{V}_F$ for which $V \subset \text{span}(\{f^i \mid i \in I\})$. In particular

$$\begin{aligned} \Pr_{\mathbf{b}}[|\{f^i \in F \mid \langle \mathbf{b}, f^i \rangle = 0\}| \geq \Delta^\ell(F)] &\leq \Pr_{\mathbf{b}}[\exists V \in \mathcal{V}_F, \forall v \in V, \langle v, \mathbf{b} \rangle = 0] \\ &\leq \sum_{V \in \mathcal{V}_F} \Pr_{\mathbf{b}}[\forall v \in V, \langle v, \mathbf{b} \rangle = 0] = \sum_{V \in \mathcal{V}_F} \Pr_{\mathbf{b}}[\mathbf{b} \in V^\perp], \end{aligned}$$

where we have adopted the notation $V^\perp = \{w \mid \forall v \in V, \langle v, w \rangle = 0\}$. Recall that when V is a subspace of dimension ℓ , V^\perp is a subspace of dimension $n - \ell$. To complete the proof, we recall that \mathbf{b} is κ -induced, so that there is an associated random variable \mathbf{B} , uniform on dimension κ subspaces, for which $\mathbf{b} \in \mathbf{B}$ with certainty; applying Lemma 19 we may then conclude

$$\sum_{V \in \mathcal{V}_F} \Pr_{\mathbf{b}}[\mathbf{b} \in V^\perp] \leq \sum_{V \in \mathcal{V}_F} \Pr_{\mathbf{B}}[\mathbf{B} \cap V^\perp \neq \{\vec{0}\}] \leq \binom{m}{\ell} q^{\kappa - \ell - 1} \left(\frac{q}{q-1} \right).$$

This completes the proof of Lemma 20. \blacktriangleleft

Proof of Theorem 16. Now we analyze the relationship between our two parameters of interest: ϵ and d . Fix some $\epsilon > 0$. Let \mathbf{b} be a κ -induced random variable for which $|E_\epsilon^{(\mathbf{b},0)}| = P_{\kappa,\epsilon}$ and let \mathbf{B} be the coupled variable, uniform on subspaces, for which $\mathbf{b} \in \mathbf{B}$.

For the purposes of analysis we consider a sequence of m vectors chosen independently and uniformly from $E_\epsilon = E_\epsilon^{(\mathbf{b},0)}$ with replacement; we let $\mathbf{F} = (\mathbf{f}^1, \dots, \mathbf{f}^m)$ denote the set of vectors so chosen. We study the expectation of the number of vectors in \mathbf{F} that are orthogonal to \mathbf{b} . We first give an immediate lower bound by linearity of expectation and the definition of E_ϵ : $\mathbb{E}_{\mathbf{b},\mathbf{F}}[|\{\mathbf{f}^i \in F \mid \langle \mathbf{b}, \mathbf{f}^i \rangle = 0\}|] \geq \epsilon \cdot m$.

We now infer an upper bound on this expectation using Lemmas 18 and 20. We say that the samples \mathbf{F} from E_ϵ are *compact* if $\Delta^\kappa(\mathbf{F}) \geq \kappa + \eta$. The probability of this *compact* event is no more than $\binom{m}{\kappa} \binom{m-\kappa}{\eta} \left(\frac{1}{d}\right)^\eta$ by Lemma 18. For *compact* selections, we crudely upper bound the expectation by m ; for *spread* selections we further split the expectation based on the random variable \mathbf{B} . We say that \mathbf{B} is *susceptible* (for a fixed $F = (f^1, \dots, f^m)$) if there exists some $b \in \mathbf{B}$ such that $|\{f^i \in F \mid \langle b, f^i \rangle = 0\}| \geq \Delta^\kappa(F)$. Otherwise, \mathbf{B} is *resistant*. The probability of a *susceptible* selection of \mathbf{B} is bounded above by $(2/q) \binom{m}{\kappa}$ in light of Lemma 20 (applied with $\ell = \kappa$). In the pessimistic case (that \mathbf{B} is *susceptible*), we again upper bound the expectation by m . Then if the experiment is neither *compact* nor *susceptible*, we may clearly upper bound the expectation by $\kappa + \eta$. So, for any $\eta > 0$ we conclude that

$$\mathbb{E}_{\mathbf{b},\mathbf{B},\mathbf{F}}[|\{f_i \in F \mid \langle \mathbf{b}, f_i \rangle = 0\}|] \leq (\kappa + \eta) + m \left(\binom{m}{\kappa} \binom{m-\kappa}{\eta} \left(\frac{1}{d}\right)^\eta + \frac{2}{q} \binom{m}{\kappa} \right)$$

and hence that

$$\epsilon \leq \left(\frac{\kappa + \eta}{m} \right) + \binom{m}{\kappa} \left(\binom{m}{\eta} \left(\frac{1}{d}\right)^\eta + \frac{2}{q} \right).$$

This completes the proof of Theorem 16. \blacktriangleleft

► Corollary 21. Let κ and n be parameters satisfying $1 \leq \kappa < n$ and let q be a prime such that $q \geq 2^{4\kappa}$. Then for $\epsilon \geq 5eq^{-1/(2(\kappa+1))}$ we have $P_{\kappa,\epsilon} \leq 5eq^\kappa/\epsilon$. In particular, for such ϵ and any κ -induced \mathbf{b} , the set $|E_\epsilon^{(\mathbf{b},0)}| \leq 5eq^\kappa/\epsilon$.

Proof of Corollary 21. Consider parameters for Theorem 16 that satisfy the following:

$$1 < d \leq q^{1/(2(\kappa+1))}, \quad m = \frac{d\eta}{2e}, \quad \text{and} \quad \eta = \log q.$$

First note that $\kappa < 4\kappa \leq \log q = \eta$ (as $q \geq 2^{4\kappa}$). Then, consider a set $E_\epsilon^{(\mathbf{b},0)}$ for some \mathbf{b} . We have

$$\epsilon \leq \left(\frac{\kappa + \eta}{m} \right) + \binom{m}{\kappa} \left(\left(\frac{me}{\eta d} \right)^\eta + \frac{2}{q} \right) \leq \left(\frac{2\eta}{m} \right) + 3 \binom{m}{\kappa} q^{-1} \leq \underbrace{\left(\frac{4e}{d} \right) + 3 \binom{d\eta/2e}{\kappa}}_{(\dagger)} q^{-1}.$$

15:20 Code Offset in the Exponent

Since $q \geq 2^{4\kappa}$, we may write $q = 2^{2\alpha\kappa}$ for some $\alpha \geq 2$ and it follows that $\left(\frac{\log q}{\kappa}\right)^\kappa = (2\alpha)^\kappa \leq (2^\alpha)^\kappa = \sqrt{q}$ because $2\alpha \leq 2^\alpha$ for all $\alpha \geq 2$. In light of this, consider the second term in the expression (†) above:

$$3 \binom{d\eta/2e}{\kappa} q^{-1} \leq 3 \left(\frac{d\eta}{2\kappa}\right)^\kappa q^{-1} \leq \frac{3}{2} \left(\frac{d\eta}{\kappa}\right)^\kappa q^{-1} \leq \frac{3}{2} \left(\frac{d^\kappa}{\sqrt{q}}\right) \left(\left(\frac{\log q}{\kappa}\right)^\kappa \frac{1}{\sqrt{q}}\right) \leq \frac{3}{2d} \leq \frac{e}{d}.$$

We conclude that for any $1 < d \leq q^{1/(2(\kappa+1))}$, $P_{\kappa,\epsilon} \geq dq^k \implies \epsilon \leq 5e/d$. Observe then that for any $\epsilon > 5e/q^{1/(2(\kappa+1))}$ we may apply the argument above to $P_{\kappa,\epsilon}$ with $d = 5e/\epsilon$ and conclude that $P_{\kappa,\epsilon} \leq 5eq^\kappa/\epsilon$. ◀

Predicting Arbitrary Values

We now show that the adversary cannot do much better than Theorem 16 even if the task is predicting an arbitrary inner product (not just zero).

► **Theorem 22.** *Let \mathbf{b} be a κ -induced random variable in \mathbb{F}_q^n and let \mathbf{g} be a random variable over \mathbb{F}_q (arbitrarily dependent on \mathbf{b}). For $\epsilon > 0$ we generalize the notation above so that*

$$E_\epsilon^{(\mathbf{b},\mathbf{g})} = \left\{ f \in \mathbb{F}_q \mid \Pr_{\mathbf{b},\mathbf{g}}[\langle \mathbf{b}, f \rangle = \mathbf{g}] \geq \epsilon \right\}. \quad \text{then} \quad |E_{\epsilon^2/8}^{(\mathbf{b},0)}| \geq \frac{\epsilon^2}{8} |E_\epsilon^{(\mathbf{b},\mathbf{g})}|.$$

Proof of Theorem 22. For an element $\psi \in E_\epsilon^{(\mathbf{b},\mathbf{g})}$, define $F_\psi = \{(x, \langle x, \psi \rangle) \mid x \in \mathbb{F}_q^n\}$. Note that $\Pr_{\mathbf{b},\mathbf{g}}[(\mathbf{b}, \mathbf{g}) \in F_\psi] \geq \epsilon$ by assumption. For any $\delta < \epsilon$, there is a subset $F^* \subset E_\epsilon^{(\mathbf{b},\mathbf{g})}$ for which $|F^*| \leq 1/\delta$ and for any $\psi \in E_\epsilon^{(\mathbf{b},\mathbf{g})}$, $\Pr_{\mathbf{b},\mathbf{g}}[(\mathbf{b}, \mathbf{g}) \in (F_\psi \cap (\bigcup_{f' \in F^*} F_{f'}))] \geq \epsilon - \delta$. To see this, consider incrementally adding elements of $E_\epsilon^{(\mathbf{b},\mathbf{g})}$ into F^* so as to greedily increase $\Pr_{\mathbf{b},\mathbf{g}}[(\mathbf{b}, \mathbf{g}) \in \bigcup_{f' \in F^*} F_{f'}]$. If this process is carried out until no $\psi \in E_\epsilon^{(\mathbf{b},\mathbf{g})}$ increases the total probability by more than δ , then it follows that every F_ψ intersects with the set with probability mass at least $\epsilon - \delta$, as desired. Note also that this termination condition is achieved after including no more than $1/\delta$ sets. It follows that for any $\psi \in E_\epsilon^{(\mathbf{b},\mathbf{g})}$,

$$\mathbb{E}_{f' \in F^*} \Pr_{\mathbf{b}}[\langle \mathbf{b}, \psi \rangle = \langle \mathbf{b}, f' \rangle] \geq (\epsilon - \delta)\delta \quad \text{and} \quad \mathbb{E}_{f' \in F^*} \mathbb{E}_{\psi \in E_\epsilon^{(\mathbf{b},\mathbf{g})}} \Pr_{\mathbf{b}}[\langle \mathbf{b}, \psi \rangle = \langle \mathbf{b}, f' \rangle] \geq (\epsilon - \delta)\delta.$$

Then there exists an f^* for which

$$\mathbb{E}_{\psi \in E_\epsilon^{(\mathbf{b},\mathbf{g})}} \Pr[\langle \mathbf{b}, \psi \rangle = \langle \mathbf{b}, f^* \rangle] \geq (\epsilon - \delta)\delta.$$

Setting $\delta = \epsilon/2$ and we see that

$$\mathbb{E}_{\psi \in E_\epsilon^{(\mathbf{b},\mathbf{g})}} \Pr[\langle \mathbf{b}, \psi \rangle = \langle \mathbf{b}, f^* \rangle] = \Pr_{\mathbf{b}, \psi \in E_\epsilon^{(\mathbf{b},\mathbf{g})}}[\langle \mathbf{b}, \psi \rangle = \langle \mathbf{b}, f^* \rangle] \geq \frac{\epsilon^2}{4}.$$

Using this expectation (of a probability), we bound the probability it is greater than $1/2$ its mean. As the inner product is bi-linear,

$$\Pr_{\psi \in E_\epsilon^{(\mathbf{b},\mathbf{g})}} \left[\Pr_{\mathbf{b}}[\langle \mathbf{b}, \psi - f^* \rangle = 0] \geq \frac{\epsilon^2}{8} \right] \geq \frac{\epsilon^2}{8}.$$

Thus, an $\epsilon^2/8$ fraction of the set $\{\psi - f^* \mid \psi \in E_\epsilon^{(\mathbf{b},\mathbf{g})}\}$ must be a subset of $E_{\epsilon^2/8}^{(\mathbf{b},0)}$: The claim of the theorem follows, that $|E_{\epsilon^2/8}^{(\mathbf{b},0)}| \geq (\epsilon^2/8)|E_\epsilon^{(\mathbf{b},\mathbf{g})}|$. ◀

With the language and settings of this last Theorem, applying Corollary 21 to appropriately control $|E_{\epsilon^2/8}^{(\mathbf{b},0)}|$ yields the following bound on $|E_{\epsilon}^{(\mathbf{b},\mathbf{g})}|$.

► **Corollary 23.** *Let κ and n be parameters satisfying $1 \leq \kappa < n$ and let q be a prime such that $q \geq 2^{4\kappa}$. Let \mathbf{b} be any κ -induced random variable in \mathbb{F}_q^n and \mathbf{g} any random variable in \mathbb{F}_q . Then for any $\epsilon \geq 11q^{-1/(4(\kappa+1))}$ it holds that $|E_{\epsilon}^{(\mathbf{b},\mathbf{g})}| \leq (320eq^{\kappa})/(\epsilon^4)$.*

This implies all high min-entropy distributions are not predictable in the above game.

► **Lemma 24.** *Let \mathbf{b} be a κ -induced random variable in \mathbb{F}_q^n . Let \mathbf{g} be an arbitrary random variable in \mathbb{F}_q . Let \mathbf{e} be a random variable with $H_{\infty}(\mathbf{e}) = s$. Let $E_{\epsilon}^{(\mathbf{b},\mathbf{g})}$ be as defined in Theorem 22. Then for $\epsilon > 0$, $\Pr_{\psi \leftarrow \mathbf{e}, \mathbf{b}, \mathbf{g}} [\langle \mathbf{b}, \psi \rangle = \mathbf{g}] \leq 2^{-s} |E_{\epsilon}^{(\mathbf{b},\mathbf{g})}| + \epsilon$.*

Proof of Lemma 24. Our predictable set $E_{\epsilon} = E_{\epsilon}^{(\mathbf{b},\mathbf{g})}$ gives us no guarantee on the instability of the inner product. If $\psi \in E_{\epsilon}$ then we upper bound the probability by 1. Because \mathbf{e} has min-entropy s , we know that no element is selected with probability greater than 2^{-s} , thus the probability of a lying inside a set of size $|E_{\epsilon}|$ is at most $|E_{\epsilon}|/2^s$. Outside of our predictable set, we know that the probability of a stable inner product cannot be greater than ϵ by definition of E_{ϵ} . Therefore if ψ does not fall in the predictable set, we bound the probability by ϵ (for simplicity, we ignore the multiplicative term less than 1). ◀

► **Corollary 25.** *Let k and n be parameters with $n > k$ and let q be a prime such that $q \geq 2^{4(n-k)}$. Let $\epsilon \geq 11q^{-1/(4(n-k+1))}$ be a parameter. Then for all distributions $\mathbf{e} \in \mathbb{F}_q^n$ such that $H_{\infty}(\mathbf{e}) \geq \log(320eq^{n-k}\epsilon^{-5})$, it holds that (for any \mathbf{b} and \mathbf{g} above)*

$$\Pr_{\mathbf{b}, \mathbf{g}, \mathbf{e}} [\langle \mathbf{b}, \mathbf{e} \rangle = \mathbf{g}] \leq 2\epsilon + k/q^{n-k}$$

and thus \mathbf{e} is $(k, 2\epsilon + k/q^{n-k})$ – MIPURS.

The additional k/q^{n-k} term is due to the probability that \mathbf{A} may not be full rank, all of the above analysis was conditioned on \mathbf{A} being full rank. The corollary then follows by replacing $\kappa = n - k$.

References

- 1 Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *Theory of cryptography conference*, pages 474–495. Springer, 2009.
- 2 Daniel Apon, Chongwon Cho, Karim Eldefrawy, and Jonathan Katz. Efficient, reusable fuzzy extractors from LWE. In *International Conference on Cyber Security Cryptography and Machine Learning*, pages 1–18. Springer, 2017.
- 3 Ward Beullens and Hoeteck Wee. Obfuscating simple functionalities from knowledge assumptions. In *IACR International Workshop on Public Key Cryptography*, pages 254–283. Springer, 2019.
- 4 Allison Bishop, Lucas Kowalczyk, Tal Malkin, Valerio Pastro, Mariana Raykova, and Kevin Shi. A simple obfuscation scheme for pattern-matching with wildcards. In *Annual International Cryptology Conference*, pages 731–752. Springer, 2018.
- 5 Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On virtual grey box obfuscation for general circuits. *Algorithmica*, 79(4):1014–1051, 2017.
- 6 Xavier Boyen. Reusable cryptographic fuzzy extractors. In *Proceedings of the 11th ACM conference on Computer and Communications Security*, pages 82–91, 2004.

15:22 Code Offset in the Exponent

- 7 Ran Canetti and Ronny Ramzi Dakdouk. Obfuscating point functions with multibit output. In *Advances in Cryptology–EUROCRYPT 2008*, pages 489–508. Springer, 2008.
- 8 Ran Canetti, Benjamin Fuller, Omer Paneth, Leonid Reyzin, and Adam Smith. Reusable fuzzy extractors for low-entropy distributions. In *Advances in Cryptology – EUROCRYPT*, pages 117–146. Springer, 2016.
- 9 Ran Canetti, Benjamin Fuller, Omer Paneth, Leonid Reyzin, and Adam Smith. Reusable fuzzy extractors for low-entropy distributions. *Journal of Cryptology*, 34(1):1–33, 2021.
- 10 Ran Canetti, Yael Tauman Kalai, Mayank Varia, and Daniel Wichs. On symmetric encryption and point obfuscation. In *Theory of Cryptography Conference*, pages 52–71. Springer, 2010.
- 11 Ran Canetti, Guy N Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *Theory of Cryptography Conference*, pages 72–89. Springer, 2010.
- 12 Luke Demarest, Benjamin Fuller, and Alexander Russell. Code offset in the exponent. *Cryptology ePrint archive*, 2018. URL: <https://eprint.iacr.org/2018/1005>.
- 13 Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008.
- 14 Nico Döttling and Jörn Müller-Quade. Lossy codes and a new variant of the learning-with-errors problem. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 18–34. Springer, 2013.
- 15 Benjamin Fuller, Xianrui Meng, and Leonid Reyzin. Computational fuzzy extractors. In *Advances in Cryptology-ASIACRYPT 2013*, pages 174–193. Springer, 2013.
- 16 Benjamin Fuller, Xianrui Meng, and Leonid Reyzin. Computational fuzzy extractors. *Information and Computation*, page 104602, 2020.
- 17 Benjamin Fuller and Lowen Peng. Continuous-source fuzzy extractors: source uncertainty and insecurity. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2952–2956. IEEE, 2019.
- 18 Benjamin Fuller, Leonid Reyzin, and Adam Smith. When are fuzzy extractors possible? In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 277–306. Springer, 2016.
- 19 Benjamin Fuller, Leonid Reyzin, and Adam Smith. When are fuzzy extractors possible? *IEEE Transactions on Information Theory*, 2020.
- 20 Steven D Galbraith and Lukas Zobernig. Obfuscated fuzzy hamming distance and conjunctions from subset product problems. In *Theory of Cryptography Conference*, pages 81–110. Springer, 2019.
- 21 Jing Hao, Han Huang, Galyna Livshyts, and Konstantin Tikhomirov. Distribution of the minimum distance of random linear codes. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 114–119. IEEE, 2020.
- 22 Charles Herder, Ling Ren, Marten van Dijk, Meng-Day Yu, and Srinivas Devadas. Trapdoor computational fuzzy extractors and stateless cryptographically-secure physical unclonable functions. *IEEE Transactions on Dependable and Secure Computing*, 2016.
- 23 Chenglu Jin, Charles Herder, Ling Ren, Phuong Ha Nguyen, Benjamin Fuller, Srinivas Devadas, and Marten Van Dijk. Fpga implementation of a cryptographically-secure puf based on learning parity with noise. *Cryptography*, 1(3):23, 2017.
- 24 Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 28–36, 1999.
- 25 Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.
- 26 Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.
- 27 Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

- 28 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.
- 29 Oded Regev. The learning with errors problem. *Invited survey in CCC*, 7(30):11, 2010.
- 30 Victor Shoup. Lower bounds for discrete logarithms and related problems. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 256–266. Springer, 1997.
- 31 Sailesh Simhadri, James Steel, and Benjamin Fuller. Cryptographic authentication from the iris. In *International Conference on Information Security*, pages 465–485. Springer, 2019.
- 32 Yunhua Wen and Shengli Liu. Robustly reusable fuzzy extractor from standard assumptions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 459–489. Springer, 2018.
- 33 Yunhua Wen, Shengli Liu, and Dawu Gu. Generic constructions of robustly reusable fuzzy extractor. In *IACR International Workshop on Public Key Cryptography*, pages 349–378. Springer, 2019.
- 34 Joanne Woodage, Rahul Chatterjee, Yevgeniy Dodis, Ari Juels, and Thomas Ristenpart. A new distribution-sensitive secure sketch and popularity-proportional hashing. In *Annual International Cryptology Conference*, pages 682–710. Springer, 2017.

P_4 -free Partition and Cover Numbers & Applications

Alexander R. Block ✉

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Simina Brânzei ✉

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Hemanta K. Maji ✉

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Himanshi Mehta ✉

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Tamalika Mukherjee ✉

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Hai H. Nguyen ✉

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Abstract

P_4 -free graphs—also known as cographs, complement-reducible graphs, or hereditary Dacey graphs—have been well studied in graph theory. Motivated by computer science and information theory applications, our work encodes (flat) joint probability distributions and Boolean functions as bipartite graphs and studies bipartite P_4 -free graphs. For these applications, the graph properties of edge partitioning and covering a bipartite graph using the minimum number of these graphs are particularly relevant. Previously, such graph properties have appeared in leakage-resilient cryptography and (variants of) coloring problems.

Interestingly, our covering problem is closely related to the well-studied problem of product (a.k.a., Prague) dimension of loopless undirected graphs, which allows us to employ algebraic lower-bounding techniques for the product/Prague dimension. We prove that computing these numbers is NP-complete, even for bipartite graphs. We establish a connection to the (unsolved) Zarankiewicz problem to show that there are bipartite graphs with size- N partite sets such that these numbers are at least $\epsilon \cdot N^{1-2\epsilon}$, for $\epsilon \in \{1/3, 1/4, 1/5, \dots\}$. Finally, we accurately estimate these numbers for bipartite graphs encoding well-studied Boolean functions from circuit complexity, such as set intersection, set disjointness, and inequality.

For applications in information theory and communication & cryptographic complexity, we consider a system where a setup samples from a (flat) joint distribution and gives the participants, Alice and Bob, their portion from this joint sample. Alice and Bob's objective is to non-interactively establish a shared key and extract the left-over entropy from their portion of the samples as independent private randomness. A genie, who observes the joint sample, provides appropriate assistance to help Alice and Bob with their objective. Lower bounds to the minimum size of the genie's assistance translate into communication and cryptographic lower bounds. We show that (the \log_2 of) the P_4 -free partition number of a graph encoding the joint distribution that the setup uses is equivalent to the size of the genie's assistance. Consequently, the joint distributions corresponding to the bipartite graphs constructed above with high P_4 -free partition numbers correspond to joint distributions requiring more assistance from the genie.

As a representative application in non-deterministic communication complexity, we study the communication complexity of nondeterministic protocols augmented by access to the equality oracle at the output. We show that (the \log_2 of) the P_4 -free cover number of the bipartite graph encoding a Boolean function f is equivalent to the minimum size of the nondeterministic input required by the parties (referred to as the communication complexity of f in this model). Consequently, the functions corresponding to the bipartite graphs with high P_4 -free cover numbers have high communication complexity. Furthermore, there are functions with communication complexity close to the naïve protocol where the nondeterministic input reveals a party's input. Finally, the access to the equality



© Alexander R. Block, Simina Brânzei, Hemanta K. Maji, Himanshi Mehta, Tamalika Mukherjee, and Hai H. Nguyen;

licensed under Creative Commons License CC-BY 4.0

2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 16; pp. 16:1–16:25



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

oracle reduces the communication complexity of computing set disjointness by a constant factor in contrast to the model where parties do not have access to the equality oracle. To compute the inequality function, we show an exponential reduction in the communication complexity, and this bound is optimal. On the other hand, access to the equality oracle is (nearly) useless for computing set intersection.

2012 ACM Subject Classification Security and privacy → Mathematical foundations of cryptography; Security and privacy → Information-theoretic techniques; Theory of computation → Communication complexity; Mathematics of computing → Graph theory

Keywords and phrases Secure keys, Secure private randomness, Gray-Wyner system, Cryptographic complexity, Nondeterministic communication complexity, Leakage-resilience, Combinatorial optimization, Product dimension, Zarankiewicz problem, Algebraic lower-bounding techniques, P_4 -free partition number, P_4 -free cover number

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.16

Related Version *Full Version:* <https://eprint.iacr.org/2020/1605.pdf>

Funding Alexander R. Block is supported in part by NSF CCF #1910659. Alexander R. Block, Hemanta K. Maji, Tamalika Mukherjee, and Hai H. Nguyen are supported in part by an NSF CRII Award CNS-1566499, NSF SMALL Awards CNS-1618822 and CNS-2055605, the IARPA HECTOR project, MITRE Innovation Program Academic Cybersecurity Research Awards (2019–2020, 2020–2021), a Purdue Research Foundation (PRF) Award, and The Center for Science of Information, an NSF Science and Technology Center, Cooperative Agreement CCF-0939370.

1 Introduction

A graph is P_4 -free if no four vertices induce a path of length three. Since the 1970s, P_4 -free graphs – also known as cographs, complement-reducible graphs, or hereditary Dacey graphs from empirical logic [22] – have been widely studied in graph theory [45, 46, 36, 60, 62]. Motivated by computer science and information theory applications, our work encodes joint probability distributions and Boolean functions as bipartite graphs and studies *bipartite* P_4 -free graphs.¹ For these applications, the graph properties of edge *partitioning* and *covering* a bipartite graph using the minimum number of these graphs are particularly relevant.²

The P_4 -free *partition number* of a bipartite graph G is the minimum number of P_4 -free subgraphs partitioning G 's edges, denoted by $P_4\text{-fp}(G)$. Similarly, the P_4 -free *cover number* of a bipartite graph G is the minimum number of P_4 -free subgraphs covering G 's edges, denoted by $P_4\text{-fc}(G)$. The definition extends to general graphs; however, our study focuses on bipartite graphs. We are given a bipartite graph as input, and the objective is to partition or cover its edges using P_4 -free bipartite graphs. P_4 -free partition and cover numbers are natural extensions of fundamental graph properties, such as product/Prague dimension, equivalence cover number, biclique partition, and cover numbers, arboricity, and star arboricity (refer to [63] for definitions). In turn, these graph properties have applications to theoretical computer science, information theory, and combinatorial optimization; for a discussion of these connections, see Appendix E in the full version.

¹ A bipartite P_4 -free graph is a disjoint union of *bicliques*.

² In contrast, [31] introduced the *vertex* partitioning a graph into different color-classes so that the vertices of any color-class induces a P_4 -free graph.

In addition to being motivated by intellectual curiosity, our work illustrates that the P_4 -free partition and cover numbers appear in diverse computer science and information theory problems (refer to problems A and B in Section 1.1). Section 1.2 presents the equivalence between the P_4 -free partition number and Problem A, and the consequences of the graph theory results for problem A. Next, Section 1.3 demonstrates the equivalence of Problem B and the P_4 -free cover number, and the implications of the graph results for problem B. Interestingly, we prove that the P_4 -free cover number of a bipartite graph is either identical to or one less than the well-studied product/Prague dimension [54, 55] of the complement graph (interpreted as a loopless undirected graph). Our work proves the following graph theory results (refer to Section 2 for formal statements).

1. Determining the P_4 -free partition and cover numbers of general graphs, even bipartite ones, is NP-complete.
2. There are bipartite graphs with size- N partite sets whose P_4 -free partition and cover numbers are at least $\epsilon \cdot N^{1-2\epsilon}$, for constant $\epsilon \in \{1/3, 1/4, 1/5, \dots\}$. Furthermore, Erdős-Rényi graphs (with constant parameter) have P_4 -free partition and cover numbers $\geq N/\log N$, asymptotically almost surely.
3. Finally, we encode the Boolean set intersection and disjointness functions, and the inequality function as bipartite graphs. We present tight estimates of the P_4 -free partition and cover numbers of these graphs.

1.1 Motivating Problems

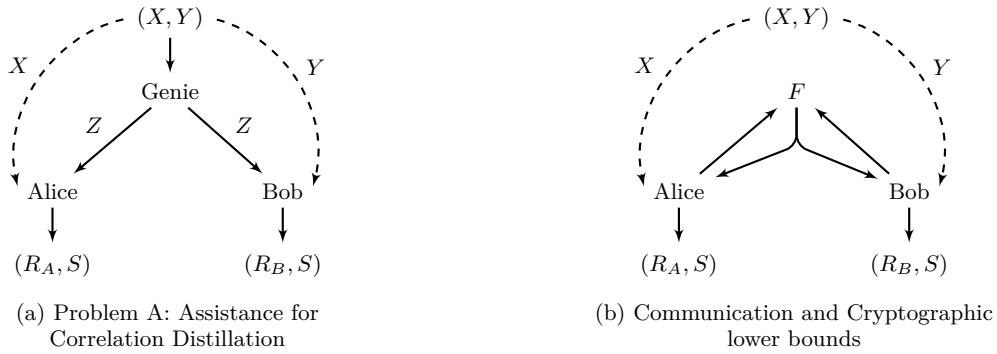
We encode joint probability distributions and Boolean functions as equivalent bipartite graphs and study the P_4 -free partition and cover numbers of these graphs. Leveraging this connection, we present representative applications of these graph properties and their estimates to information theory and circuit complexity. In particular, consider the following illustrative representative problems from information theory and communication & cryptographic complexity motivating this study.

1.1.1 Problem A. Assistance for Correlation Distillation

Extracting randomness [32, 56], establishing secret keys [49, 50, 51, 1, 2], and performing general secure computation [16, 17, 40, 41, 19, 42, 18, 64, 65, 39, 13] with maximum efficiency and resilience from noise sources is fundamental to theoretical computer science and information theory. Towards that objective, we study the communication and cryptographic complexity of parties to agree on a shared secret and extract private local randomness from a source.

A setup (see part (a) of Figure 1), the only source of randomness in the system, samples (x, y) according to the joint probability distribution p_{XY} , and (privately) sends x to Alice and y to Bob. Alice and Bob's objective is to agree on a shared secret key and private (independent) randomness without any additional public communication. A genie, who observes the sample (x, y) , provides a public k -bit assistance z to Alice and Bob to facilitate their efforts. We emphasize that all agents Alice, Bob, and the genie are deterministic. After that, Alice and Bob locally compute the shared key s from their respective local views (x, z) and (y, z) . Finally, Alice extracts the left-over entropy from x (conditioned on (s, z)) as her local private randomness r_A . Similarly, Bob extracts his local private randomness r_B from the left-over entropy of y .

For the security of Bob's local randomness, an honest but curious Alice cannot obtain any additional information on r_B beyond what is already revealed by z and s . Analogously, Bob's view should contain no additional information on Alice's view conditioned on z and s .



■ **Figure 1** Part (a). A pictorial summary of the system in our motivating problem A. Part (b). The setup samples (x, y) according to the distribution p_{XY} and sends x to Alice and y to Bob. Alice and Bob use F adaptively multiple times to communicate with each other; F delivers its output to both Alice and Bob. The functionality F may be a communication protocol (i.e., a message forwarding functionality), or help Alice and Bob evaluate any (possibly, a stateful) functionality of their inputs. The objective of Alice and Bob is to generate a shared secret key s at the end of the protocol and extract the left-over entropy in their shares as independent local randomness.

Intuitively, conditioned on the genie’s assistance Z , Alice-Bob samples’ joint distribution splits into shared randomness and local independent randomness.

What is the *minimum* length k of the genie’s assistance sufficient for Alice and Bob to agree on a shared key and obtain secure private randomness? In particular, which distributions p_{XY} need no assistance at all?

Mutual information and other common information variants (refer to Appendix D in the full version for discussion) cannot accurately measure this information-theoretic property; thus, motivating our study. This problem is equivalent to computing the P_4 -free partition number of a bipartite graph encoding the (flat) joint probability distribution p_{XY} . In particular, lower bounds to k translates into lower bounds on (interactive) communication and cryptographic complexity (see part (b) of Figure 1).

1.1.2 Problem B. Nondeterministic Communication Complexity relative to the Equality Oracle

The nondeterministic communication complexity of the equality function is high [44]. However, what is the additional utility of an oracle call to the equality function in computing other functions?

Suppose Alice has input $x \in X$, Bob has input $y \in Y$, and are interested in computing the Boolean function $f: X \times Y \rightarrow \{0, 1\}$ of their private inputs. They have access to an *equality oracle* $\text{EQ}: \{0, 1\}^* \times \{0, 1\}^* \rightarrow \{0, 1\}$ defined by $\text{EQ}(a, b) = 1$ if and only if $a = b$. They are interested in computing $f(x, y)$ using this equality oracle and a k -bit nondeterministic input *without any additional communication*.

The functions $A: X \times \{0, 1\}^k \rightarrow \{0, 1\}^*$ and $B: Y \times \{0, 1\}^k \rightarrow \{0, 1\}^*$ satisfying the following constraints define a *nondeterministic protocol* for f relative to the equality oracle.

1. For every input-pair $(x, y) \in X \times Y$ such that the output $f(x, y) = 1$, there exists a nondeterministic input $z \in \{0, 1\}^k$ ensuring $\text{EQ}(A(x, z), B(y, z)) = 1$.
2. For every input-pair $(x, y) \in X \times Y$ such that the output $f(x, y) = 0$, for all nondeterministic inputs $z \in \{0, 1\}^k$, we have $\text{EQ}(A(x, z), B(y, z)) = 0$.

The *communication complexity* of this protocol is k , i.e., the length of the nondeterministic input. What is the *minimum* communication complexity k of the function f ?

Intuitively, we are augmenting the nondeterministic communication protocols with an equality oracle at the output. If the EQ oracle is useful to compute a function f , then its communication complexity in our model shall be significantly lower than where the parties cannot access the EQ oracle. We show that this problem is identical to the P_4 -free cover number of a bipartite graph encoding the Boolean function f . Our results show that the access to the equality oracle reduces the communication complexity of computing set disjointness by a constant factor compared to the model where parties do not have access to the equality oracle. To compute the inequality function, perhaps surprisingly, we show an *exponential* reduction in the communication complexity. On the other hand, access to the equality oracle is virtually useless to computing the set intersection. Section 1.3 provides the details.

1.1.3 Additional Applications and History

In Appendix F of the full version, we present a representative scheduling problem that naturally reduces to computing P_4 -free partition/cover numbers. Beyond the applications above, this example highlights the innate ability of P_4 -free graphs to encode scheduling problems that are amenable to *parallelization*.

Edge-partitioning graphs using the minimum number of P_4 -free graphs have found applications in *leakage-resilient cryptography* [9]. In particular, if k -bits of genie's assistance suffices for the setup in problem A, then k -bits of leakage also suffices for the adversary to destroy the possibility of performing general secure computation. Identifying a large P_4 -free subgraph of a given graph is studied in clustering. For example, an *exclusive row and column bichuster* [48, 37] is identical to a P_4 -free graph, with applications in analyzing biological data. [15] used P_4 -free partition and cover numbers to approach a coloring conjecture (a variant of Ryser's conjecture) for bipartite graphs.

1.1.4 Related graph properties: Equivalence Cover Number and Product/Prague Dimension

The following discussion is specific to *loopless undirected graphs*. An *equivalence graph* is a (disjoint) union of cliques. The *equivalence cover number* of a graph G is the minimum number d of equivalence sub-graphs that cover the edges of G [54, 55]. Note that the P_4 -free cover number is an extension of this concept to bipartite graphs. Furthermore, the equivalence cover number of G is identical to the *product/Prague dimension* of the complement of the graph G [63, 30], the minimum $d \in \mathbb{N}$ such that the complement of the graph G is an induced subgraph of $K_{\mathbb{N}}^d$ (the d -fold product of the infinite complete graph $K_{\mathbb{N}}$). Computing the equivalence cover number or the product dimension of a graph is NP-complete [54].

The P_4 -free cover number (for bipartite graphs) has a close connection to the product (a.k.a., Prague) dimension.

► **Proposition 1.** *If a redundancy-free³ bipartite graph $G = (L, R, E)$ has a size- d P_4 -free edge-covering, then the complement bipartite graph $\overline{G} := (L, R, L \times R \setminus E)$ is an induced subgraph of $K_2 \times K_{\mathbb{N}}^d$.*

³ A graph is redundancy-free if no two vertices have an identical neighborhood.

The converse of the proposition does not hold exactly (refer to Section 7). However, if \overline{G} is an induced subgraph of $K_2 \times K_{\mathbb{N}}^d$, then G has a size- $(d + 1)$ P_4 -free cover. We prove that $\text{P}_4\text{-fc}(G) \in \{\text{pdim}(H), \text{pdim}(H) - 1\}$, where $G = (L, R, E \subseteq L \times R)$ is a bipartite graph, $H = (L \cup R, L \times R \setminus E)$ is the loopless undirected graph representing the complement of the bipartite graph G , and $\text{pdim}(H)$ is the product/Prague dimension of H (refer to Corollary 34 in Section 7). Figure 7 presents a graph showing the necessity of this slack in the characterization. However, for most applications, an additive slack of one should be acceptable. This proposition facilitates lower-bounding the $\text{P}_4\text{-fc}(G)$ using the algebraic lower-bounding techniques for the product/Prague dimension [47, 4, 63, 5].

Despite this similarity, extremal properties of the equivalence cover number and product/Prague dimension need not translate into extremal properties of the P_4 -free cover number. For example, an N -vertex star has an equivalence cover number $(N - 1)$ [63]. On the other hand, the P_4 -free cover number of any bipartite graph with size- N partite sets is at most its star arboricity (because star forests are P_4 -free), which is at most (roughly) $N/2$ [3]. The bottleneck here is that the $\text{P}_4\text{-fc}(G)$ is close to $\text{pdim}(H)$, where H represents a bipartite graph, i.e., the graph H is structured (triangle-free in this particular case). The graphs realizing the extremal properties for equivalence cover number and product/Prague dimension need not have this structure. In particular, the construction of bipartite graphs with high P_4 -free cover and partition numbers turns out to be non-trivial, and our work establishes a connection to the well-known (unsolved) Zarankiewicz problem [11] and relies on probabilistic techniques to demonstrate their existence.

Section 7 also presents a variant of the product/Prague dimension to estimate the P_4 -free partition number (see Corollary 37). A lower bound for the P_4 -free partition number is non-trivial if it is not already a lower bound to the P_4 -free cover number. Unfortunately, no non-trivial lower-bounding techniques for general graphs are known for this new graph embedding property. When non-trivial lower bounds for this variant of the product/Prague dimension is proven, they shall transfer to the P_4 -free partition number.

Among several notions of product dimension for graphs [30], most of which are unrelated to the property we wish to capture,⁴ the graph property mentioned above is the closest and most relevant.

1.2 P_4 -free Partition Number

We reduce problem A to computing the P_4 -free partition number. We present the reduction's highlight. A bipartite graph G naturally represents a (flat) joint distribution p_{XY} , where the edge-set is the support of p_{XY} (see Figure 2 for examples). If G is already P_4 -free, then Alice and Bob need no assistance from the genie; the connected component's identity is their shared key s , and (conditioned on the identity of the shared key) their samples $r_A = (x|s)$ and $r_B = (y|s)$ are independent private randomness. If G is not P_4 -free, the genie decomposes G into G_1, \dots, G_d such that each G_i is P_4 -free and the edge sets $E(G_1), \dots, E(G_d)$ partition the edge set $E(G)$. For a joint sample $(u, v) \in E(G)$, the genie reveals the (unique) $z = i$ such that $(u, v) \in E(G_i)$. Conditioning on the genie's assistance $z = i$, Alice-Bob's samples come from the joint distribution G_i , which is P_4 -free, so they agree on their shared key and secure private randomness as above. To minimize the genie's assistance, one needs to minimize $d \in \mathbb{N}$, identical to $\text{P}_4\text{-fp}(G)$.

⁴ Even the notions of dimension that are deceptively similar sounding, for example, the “product dimension of bipartite graphs” introduced by [59], are unrelated to the graph properties that this paper studies.

	00	11	01	10
00	1	1	0	0
11	1	1	0	0
01	0	0	1	1
10	0	0	1	1

	00	01	10	11
00	1	1	0	0
01	0	1	1	0
10	0	0	1	1
11	1	0	0	1

(a) Forward or flip.

(b) Noisy typewriter.

■ **Figure 2** Pictorial representation of the probability distributions (a) forward or flip, and (b) noisy typewriter distributions, for $n = 2$. Rows correspond to Alice samples, and columns correspond to Bob samples. The (i, j) -th entry of a matrix being 1 represents that (i, j) is in the support of the distribution. The distribution is a uniform distribution over all the elements in the support. Let G_a be the bipartite graph whose adjacency matrix is defined by the matrix representation of the forward and flip distribution. The graph G_a is a disjoint union of 2^{n-1} copies of the $K_{2,2}$ biclique. Note that G_a is P_4 -free, and, hence, $\mathsf{P}_4\text{-fp}(G_a) = 1$. Let G_b be the bipartite graph whose adjacency matrix is defined by the matrix representation of the noisy typewriter distribution. The graph G_b is a cycle of length 2^{n+1} . Note that G_b is *not* P_4 -free, and $\mathsf{P}_4\text{-fp}(G_b) = 2$ (the graph decomposes into two matchings).

1.2.1 Discussion on Problem A

We begin by expanding how lower-bounding the information-theoretic measure in problem A translates into communication and cryptographic lower bounds (as in [8]). Suppose, in our model, one proves that the genie's assistance must be $k \geq k^*$ bits. Now consider the setting in part (b) of Figure 1 where there is no genie; however, the parties have access to a functionality F . The functionality F may be an arbitrary *communication protocol* or multiple calls to arbitrary *interactive stateful functionalities* that receive adaptive inputs from Alice and Bob. In particular, F may be multiple copies of the NAND-functionality, which is sufficient for general secure computation [68, 27, 42]. Observe that the genie can simulate the functionality F 's entire output with access to (x, y) . Consequently, we have the following result.

► **Proposition 2.** *If p_{XY} needs $k \geq k^*$ bits of assistance from the genie in our model, then Alice and Bob need to receive at least k^* bits from F in the Figure 1 part (b) model to establish a shared key s and extract the left-over entropy in their sample as independent private randomness.*

In information theory, Gray-Wyner systems/networks are well-studied [66]. However, existing measures like mutual information and various notions of common information are inadequate to capture the information-theoretic property in Problem A accurately. For example, there are two joint distributions with identical (Shannon's) mutual information [61]; however, one needs no assistance while the other needs one-bit assistance.⁵ Refer to Figure 2 for the following discussion. Consider the first distribution (namely, the *forward or flip distribution*), where Alice gets i.i.d. uniformly random bits $x = (x_1, x_2, \dots, x_n)$, and Bob either (with probability half) gets $y = x$ or $y = (\bar{x}_1, \dots, \bar{x}_n)$, i.e., every bit of x is flipped. In the second distribution (the *noisy typewriter distribution*), Alice gets a uniformly random sample $x \in \{0, 1, \dots, 2^n - 1\}$, and Bob either gets $y = x$ or $y = (x + 1) \bmod 2^n$ with probability half. The bipartite graph corresponding to the forward or flip distribution is, indeed, P_4 -free, and the bipartite graph corresponding to the noisy typewriter distribution

⁵ By tensorizing the distributions, one can increase the gap in the necessary assistance arbitrarily.

has P_4 -free partition number 2 (i.e., one-bit assistance is necessary and sufficient). Both distributions have $(n - 1)$ bits of mutual information; however, the first distribution needs no assistance, but the second distribution needs one-bit assistance⁶ to agree on a secret key.

Wyner's common information [66] estimates the minimum assistance that removes any dependence between Alice-Bob samples. This quantity is a significant overestimation (for example, in the forward or flip distribution, it needs $(n-1)$ -bits of assistance $z = (x_1, \dots, x_{n-1})$), and Wyner's assistance eliminates the possibility of Alice and Bob agreeing on a secret key, which defeats the objective of this problem. Gács-Körner common information [25] estimates the length of the secret key that Alice and Bob can generate without any assistance from the genie, which results in pessimistic estimates. For example, starting with samples from the noisy typewriter distribution, Alice and Bob cannot even agree on a one-bit secret; however, appropriate one-bit assistance would help them generate an $(n - 1)$ -bit secret. Likewise, non-interactive correlation distillation [53, 52] enables parties to agree on a secret non-interactively *without any assistance*. However, even without the necessity to generate independent local randomness, strong hardness of computation results are known [53, 52, 67, 10, 14].

Refer to Appendix D in the full version for additional discussion on various forms of common information.

1.2.2 Our results for Problem A

Observe that the naïve assistance that reveals the XOR of the parties' inputs suffices; however, the minimum assistance may be exponentially smaller. Our work relies on suitably encoding (flat) joint distributions as bipartite graphs. We prove in Theorem 5 that ascertaining the minimum assistance is, in general, difficult. Furthermore, there are joint distributions where the minimum assistance that is needed is close to the naïve assistance mentioned above, yielding lower bounds in communication and cryptographic complexity. In other words, we obtain the following as a corollary to Theorem 6.

► **Corollary 3.** *Let $\Omega_X = \Omega_Y = \{0, 1\}^n$. Fix $t \in \mathbb{N}$. There are joint distributions over the sample space $\Omega_X \times \Omega_Y$ that require Alice and Bob to (each) receive at least $(1 - \frac{2}{t+2})n$ bits of communication in the model in Figure 1 part (b).*

Finally, we upper-bound the minimum assistance needed for a few well-studied probability distributions i.e. when p_{XY} is the INT_n ⁷ or the DISJ_n ⁸ joint distribution, then $\lceil n/2 \rceil$ -bit assistance suffices (we explicitly provide the assistance that the genie provides and it is efficient to compute, see Theorem 8). For INEQ_N , where $N = 2^n$, the genie needs to provide $\lceil \log n \rceil$ bits of assistance. The assistance for INEQ_N is optimal because we prove a matching lower bound. In general, $\min\{\log_2 N, \frac{1}{2} \log_2 |\text{Supp}(p_{XY})|\}$ bits of assistance suffices.⁹

⁶ The genie notifies the parties whether $y = x$ or not.

⁷ Alice receives random $X \subseteq \{1, 2, \dots, n\}$, and Bob receives random $Y \subseteq \{1, 2, \dots, n\}$ conditioned on $X \cap Y \neq \emptyset$.

⁸ Alice receives random $X \subseteq \{1, 2, \dots, n\}$, and Bob receives random $Y \subseteq \{1, 2, \dots, n\}$ conditioned on $X \cap Y = \emptyset$.

⁹ Because, $\text{P}_4\text{-fp}(G) \leq \text{sa}(G) \leq \mathcal{O}\left(\sqrt{|E(G)|}\right)$. The last bound on the star arboricity of G follows from an averaging argument and the bound of [3].

1.3 P_4 -free Cover Number

We reduce Problem B to the P_4 -free cover number. Boolean functions naturally encode a bipartite graph's adjacency matrix; an input-pair that evaluates to 1 denotes an edge in the graph. If the graph G (of a function f) is P_4 -free, then parties need no nondeterministic input; they can evaluate f using the EQ oracle.¹⁰ Otherwise, decompose G into P_4 -free G_1, \dots, G_d such that the union of the edge-sets of G_1, \dots, G_d is the edge-set of G . For input (x, y) such that $f(x, y) = 1$, the nondeterministic input is $i \in \{1, \dots, d\}$, where the edge-set of G_i contains the edge (x, y) . Next, given this nondeterministic input, parties can evaluate f . For input (x, y) such that $f(x, y) = 0$, no nondeterministic input can make Alice and Bob output 1. One minimizes $d \in \mathbb{N}$ to minimize the nondeterministic communication complexity, which is identical to P_4 -fc(G).

1.3.1 Discussion on Problem B

The equality function in the *standard* nondeterministic communication complexity model (where parties *do not* have access to the EQ oracle) has high nondeterministic communication complexity. Determining the minimum nondeterministic input is equivalent to covering the input-pairs where the output is 1 using a minimum number of *combinatorial rectangles*, a.k.a., the *biclique cover number* [35]. The motivating problem's objective is to characterize the utility of oracle access to the EQ function in computing other functions. If the EQ oracle is useful, then the nondeterministic communication complexity relative to the EQ oracle shall be lower than without accessing the EQ oracle. The particular notion of "reduction" considered above is similar to Karp-reduction [38], which permits only one call to the oracle and no post-processing of the oracle's output. Similarly, in circuit complexity, it is typical to augment a circuit class with a more expressive gate at the output that is not computable by circuits in that class. For example, one studies the effects of augmenting AC^0 circuits with a MAJ (majority) gate or a THR (threshold) gate at the output [7, 26, 33, 29], enabling a controlled exploration of the gap between the power of AC^0 and TC^0 circuits.

1.3.2 Our results for Problem B

Similar to the result for P_4 -free partition number, we prove that computing the P_4 -free cover number is difficult (see Theorem 5), and there are functions that need nondeterministic input (roughly) the size of the parties' inputs, in other words, we obtain the following as a corollary to Theorem 6.

► **Corollary 4.** *Fix $t \in \mathbb{N}$. There are Boolean functions $f: \{1, 2, \dots, N\} \times \{1, 2, \dots, N\} \rightarrow \{0, 1\}$ requiring at least $(1 - \frac{2}{t+2}) \log_2 N$ bits of nondeterministic input in the communication complexity model where parties have access to the EQ oracle.*

These functions are analogs of the "fooling sets" in our communication model. In the standard nondeterministic communication model, the EQ function is hard-to-compute and needs n -bits of nondeterministic input. The "fooling set" lower-bounding technique draws inspiration from this result. For a general f , this argument demonstrates pairs of Alice and Bob's input-sets where only the diagonal elements are 1; and the rest are 0. That is, the function f has an embedded EQ function. The size of this "embedded EQ" (a.k.a., the fooling set) in

¹⁰ Parties compute the connected component where their private input belongs. Then, they use the EQ oracle to test if they belong to the same connected component.

16:10 P_4 -free Partition and Cover Numbers & Applications

f suffices to prove lower bounds on the nondeterministic input needed to compute f . In our setting, these functions that require $(1 - \frac{2}{t+2})n$ -bit nondeterministic input serve as “fooling sets” in the nondeterministic communication complexity model where parties can access the EQ oracle.

Next, we provide estimates for some well-known functions in communication complexity (see Theorem 8). We prove that the P_4 -free cover number of DISJ_n is (roughly) $\leq \sqrt{N}$. That is, only $n/2$ bits of nondeterministic input suffices to compute this function. Recall that, in the standard model, the function DISJ_n requires n -bit nondeterministic input because $\{(X, \{1, 2, \dots, n\} \setminus X)\}_{X \subseteq \{1, 2, \dots, n\}}$ is a fooling set. Consequently, our result demonstrates a linear gap in the number of bits needed in our model, which indicates that the EQ oracle is non-trivially useful to compute DISJ_n . We prove a lower bound showing that $0.085n$ -bit assistance is necessary.

Next, we prove that the P_4 -free cover number of INT_n is between n and $n(1 - \frac{\log_2(n)}{n})$. Observe that the nondeterministic communication complexity of INT_n (without access to the EQ oracle) is already $\lceil \log_2 n \rceil$ bits. Consequently, EQ oracle’s access is practically useless because the difference between the ceiling of the log of the lower and the upper bounds is at most 1 (asymptotically).

Finally, we show that INEQ_N needs only $\log_2 \log_2 N$ bit nondeterministic input using the EQ oracle. Intuitively, if $N = 2^{2^s}$ and all inputs are 2^s -bit binary strings, then the nondeterministic input is the s -bit index where the parties’ input differ. Recall that in the standard model (without access to the EQ oracle), INEQ_N requires $\log_2 N$ -bit nondeterministic input, which is exponentially higher. Furthermore, using the algebraic technique of [47, 63], we prove a matching lower bound to the P_4 -free cover number of INEQ_N . Observe that we prove that $P_4\text{-fp}(\text{INEQ}_N)$, not just $P_4\text{-fc}(\text{INEQ}_N)$, matches the lower bound for the $P_4\text{-fc}(\text{INEQ}_N)$.

2 Our Contribution

We prove the NP-completeness of determining the P_4 -free partition and cover numbers of a bipartite graph.

► **Theorem 5** (Hardness of P_4 -free Partition and Cover). *The following languages are NP-complete.*

$$\begin{aligned} P_4\text{-FREE-PART} &= \{ \langle G \rangle \mid G \text{ is a bipartite graph and } P_4\text{-fp}(G) \leq 2 \}, \\ P_4\text{-FREE-COV} &= \{ \langle G \rangle \mid G \text{ is a bipartite graph and } P_4\text{-fc}(G) \leq 2 \}. \end{aligned}$$

Similar problems, for example, calculating the biclique partition number/cover [57] and star arboricity [34] (even for bipartite graphs) are NP-complete.

Next, we prove that there are graphs G with large P_4 -free partition and cover numbers. Note that for a bipartite graph $G = (L, R, E)$, we have $P_4\text{-fc}(G) \leq P_4\text{-fp}(G) \leq \min\{|L|, |R|\}$ by decomposing the graph into stars rooted at vertices of the smaller partite set. Towards understanding the tightness of this naïve upper-bound, we show that, for any $N \in \mathbb{N}$ and constant $\epsilon \in \{1/3, 1/4, \dots\}$, there are bipartite graphs with size- N partite sets and $P_4\text{-fp}(G) \geq P_4\text{-fc}(G) \geq \Omega(\epsilon \cdot N^{1-2\epsilon})$ (roughly).

► **Theorem 6** (High P_4 - Free Partition and Cover Numbers). *Let C be an appropriate positive absolute constant and $t \in \mathbb{N}$ be a parameter. There exists $N_0 \in \mathbb{N}$ such that for all $N \in \mathbb{N}$ and $N \geq N_0$, there is a graph $G_{N,t} = (L, R, E)$ such that (1) $|L| = |R| = N$, and (2) $P_4\text{-fp}(G_{N,t}) \geq P_4\text{-fc}(G_{N,t}) \geq C \cdot \frac{1}{t} \cdot N^{1 - \frac{2}{t+2}}$.*

Our constructions rely on extremal bipartite graphs that avoid $K_{t+1,t+1}$ -subgraphs (the unsolved Zarankiewicz problem [11]), for which only probabilistic constructions are known (refer to the discussion in Section 4). Explicit constructions are known only for very specialized values of t . However, the P_4 -free partition and cover numbers of $G_{N,t}$ cannot be too large. For any sparse bipartite graph G , using an averaging argument, its star-arboricity has the upper bound $\text{sa}(G) \leq \mathcal{O}\left(\sqrt{|E(G)|}\right)$ [3]. Since star forests are P_4 -free and $G_{N,t}$ has $\mathcal{O}\left(N^{2-\frac{2}{t+1}}\right)$ edges, it implies that $\text{P}_4\text{-fp}(G_{N,t}) \leq \mathcal{O}\left(N^{1-\frac{1}{t+2}}\right)$.

In problem A, the joint distributions corresponding to these bipartite graphs require a lot of assistance from the genie. Consequently, these lower bounds translate into communication and cryptographic complexity lower bounds. The functions corresponding to these bipartite graphs are difficult to compute for parties with nondeterministic input and access to the EQ oracle. If these functions are embedded in another function, then that function must have high nondeterministic communication complexity as well.

As a corollary (of the proof technique presented above), we prove the following result for dense bipartite graphs drawn from the Erdős-Rényi distribution with (constant) parameter $p \in (0, 1)$. Graphs drawn from $\text{ER}(N, N, p)$ avoid bicliques with size- $(2 \log_a N)$ partite sets. Therefore, we have the following result.

► **Corollary 7** (High P_4 -Free Partition and Cover Number of Erdős-Rényi Graphs). *Let $p \in (0, 1)$ be a constant parameter. Let $\text{ER}(N, N, p)$ represent the distribution over the sample space of all bipartite graphs over size- N partite sets that includes every edge into the graph independently with probability p . Then, for $a = 1/p$, we have*

$$\Pr \left[\text{P}_4\text{-fp}(G) \geq \text{P}_4\text{-fc}(G) \geq \frac{pN}{4 \log_a N} \cdot (1 - o(1)) : G \stackrel{\$}{\leftarrow} \text{ER}(N, N, p) \right] \geq 1 - o(1).$$

Upper bounds to the P_4 -free cover and partition numbers for bipartite Erdős-Rényi graphs is potentially an extremely challenging problem. Upper-bounding the P_4 -free partition number of Erdős-Rényi bipartite graphs remains open.

Finally, we estimate the P_4 -free partition and cover numbers for the graphs INT_n , DISJ_n , and INEQ_N that are well-studied functions from communication theory and are defined below.

1. **The Intersection Graph.** For $n \in \mathbb{N}$, let $\text{INT}_n = (\{0, 1\}^n, \{0, 1\}^n, E)$ be the bipartite graph defined as follows. For any $u, v \in \{0, 1\}^n$, we have $(u, v) \in E$ if and only if the set $U \subseteq \{1, 2, \dots, n\}$ indicated by u , intersects the set $V \subseteq \{1, 2, \dots, n\}$ indicated by v .
2. **The Disjointness Graph.** For $n \in \mathbb{N}$, let $\text{DISJ}_n = (\{0, 1\}^n, \{0, 1\}^n, E)$ be the bipartite graph defined as follows. For any $u, v \in \{0, 1\}^n$, we have $(u, v) \in E$ if and only if the set $U \subseteq \{1, 2, \dots, n\}$ indicated by u , is disjoint from the set $V \subseteq \{1, 2, \dots, n\}$ indicated by v .
3. **The Inequality Graph.** For $N \in \mathbb{N}$, let $\text{INEQ}_N = (\{1, 2, \dots, N\}, \{1, 2, \dots, N\}, E)$ be the bipartite graph defined as follows. For any $u, v \in \{1, 2, \dots, N\}$, we have $(u, v) \in E$ if and only if $u \neq v$.

► **Theorem 8** (Estimates for Particular Graphs). *For all $n, N \in \mathbb{N}$, the following statements hold.*

1. $n - \frac{1}{2} \lg(n) - \mathcal{O}(1) \leq \text{P}_4\text{-fc}(\text{INT}_n) \leq n$, and $\text{P}_4\text{-fp}(\text{INT}_n) \leq \begin{cases} 2 \cdot 2^{n/2} - 2, & \text{even } n, \text{ and} \\ 3 \cdot 2^{(n-1)/2} - 2, & \text{odd } n. \end{cases}$
2. $2^{0.085n} \leq \text{P}_4\text{-fc}(\text{DISJ}_n) \leq \text{P}_4\text{-fp}(\text{DISJ}_n) \leq 2^{\lceil n/2 \rceil}$.
3. $\text{P}_4\text{-fc}(\text{INEQ}_N) = \text{P}_4\text{-fp}(\text{INEQ}_N) = \lceil \log_2 N \rceil$.

Recall that for any Boolean function f , parties can calculate it with $\lceil \log_2 \text{P}_4\text{-fc}(G(f)) \rceil$ -bit nondeterministic input and one call to the EQ oracle, where $G(f)$ is the bipartite graph representing the Boolean function f . Therefore, the bounds above translate into communication bounds.

Observe the exponential gap between the upper bounds on the P_4 -free cover and partition numbers of INT_n . We conjecture that similar to the exponential gaps in the biclique cover and partition number of some graphs [58], INT_n is a candidate bipartite graph witnessing an exponential gap in its P_4 -free cover and partition numbers. Currently, the authors are unaware of any general non-trivial lower bounding technique for the partition number that is not a lower bound to the cover number for this problem.

Lower-bounding the P_4 -free cover numbers of INEQ_N and INT_n relies on Proposition 1 and the algebraic technique of [47, 63]. Furthermore, the P_4 -free cover and partition numbers of INEQ_N are exact, previously unknown for the partition number. Finally, the lower bound on the P_4 -free cover number of DISJ_n uses a new counting strategy.

3 Hardness of P_4 -free Partition and Cover Numbers

In this section, we will prove Theorem 5. Our proof of hardness for both partition and cover number is based on a result from [28], which shows that computing the edge partition of a bipartite planar graph into two star forests is NP-complete.

► **Definition 9.** A star is a tree with one internal node, in other words, a biclique in which either the left partite set or the right partite set has size one. A star forest is a forest whose connected components are stars. The star arboricity of a graph, represented by $\text{sa}(G)$, is the minimum number of star forests that a graph can be partitioned into.

► **Imported Theorem 10** (Gonçalves and Ochem [28]). For any $g > 3$, deciding whether a bipartite planar graph G with girth¹¹ at least g and maximum degree 3 satisfies $\text{sa}(G) \leq 2$ is NP-complete.

Proof of Theorem 5. First we show the decision problem is in NP, that is, given a partition of the edge set of G into ≤ 2 components we can verify in polynomial time whether it is a P_4 -free partition of size ≤ 2 of G or not. This can be done in polynomial time by checking if any set of four vertices (two in the left set and two in the right set) in each component is P_4 -free.

Next we show that the decision problem from Theorem 10 is polynomial-time reducible to the P_4 -free partition and cover number on bipartite graphs. The decision problem in Theorem 10 is NP-complete for any bipartite planar graph of girth at least $g > 3$; in particular, it holds for $g \geq 6$. Suppose we have a bipartite planar graph G with girth $g \geq 6$ and maximum degree 3. Since G has girth at least 6, there are no cycles of length less than 6 in G . It implies that $K_{2,2}$ is not a subgraph of G . Therefore, any disjoint union of bicliques in G is a star forest. This implies that $\text{sa}(G) = \text{P}_4\text{-fp}(G) = \text{P}_4\text{-fc}(G)$, since $K_{2,2}$ -free graphs have the property that the P_4 -free partition and cover numbers are both identical to the star arboricity. Thus, the star arboricity of G is ≤ 2 if and only if the P_4 -free partition number of G is ≤ 2 . ◀

¹¹The girth of an undirected graph is the length of the shortest cycle in the graph.

4 High P_4 -free Partition and Cover Numbers

We shall prove Theorem 6 and Corollary 7 in this section. We begin with some terminologies in extremal graph theory. Fix a graph H . A classical problem in graph theory is to find the maximum number of edges in a graph on N vertices that does not contain a copy of H .

► **Definition 11** (Turán number). *Turán number denoted by $ex(N, H)$ is the maximum number of edges in an N -vertex graph that does not contain a copy of H .*

A sub-problem of special interest is when H is a complete bipartite graph, this problem is commonly referred to as the Zarankiewicz problem.

► **Definition 12** (Zarankiewicz function). *Zarankiewicz function, denoted by $z(M, N; s, t)$, is the maximum number of edges in a bipartite graph $G = (L, R, E)$, where $|L| = M$ and $|R| = N$, that does not contain a sub-graph of the form $K_{s,t}$.*

The Zarankiewicz function is well-studied [24]. The best general lower bound obtained by the probabilistic method [20] yields the following bound.

► **Imported Theorem 13** (Erdős and Spencer [20]). *For all $a, b \in \mathbb{N}$, we have $ex(N, K_{a,b}) \geq C \cdot N^{2 - \frac{a+b-2}{ab-1}}$, where C is a positive absolute constant.*

An explicit construction for $K_{t+1,t+1}$ -avoiding graphs for $t = 2$ is known [12], which has $\frac{1}{2}N^{\frac{5}{3}} + o(N^{\frac{5}{3}})$ edges.¹² Using *norm graphs*, constructions of $K_{t,s}$ -avoiding graphs for fixed $t \geq 2$ and $s > (t-1)!$ are known as well [43, 6]. Note that the latter set of constructions do not apply to our setting for $t > 3$. Considering the adjacency matrix of a $K_{a,b}$ -free graph on n vertices, we get $z(N, N, a, b) \geq 2ex(N, K_{a,b})$.

Let $G = (L, R, E)$ be a bipartite graph. A *combinatorial rectangle* is a set of the form $A \times B$, where $A \subseteq L$ and $B \subseteq R$. Observe that a combinatorial rectangle corresponds to a biclique if we restrict ourselves to rectangles of the form $\{A \times B : (u, v) \in A \times B \iff (u, v) \in E\}$. We shall use this fact in the sequel to show that the P_4 -free partition number of a $K_{t+1,t+1}$ -free bipartite graph is high.

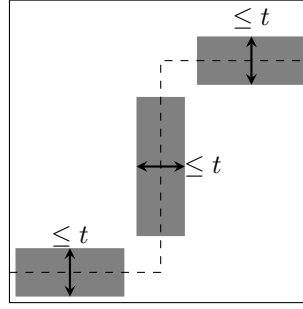
► **Lemma 14.** *For a bipartite graph $G = (L, R, E)$ such that $|L| = |R| = N$, if G is $K_{t+1,t+1}$ -free for some $t > 0$, then $P_4\text{-fp}(G) \geq \frac{e(G)}{2Nt}$.*

Proof. Consider the adjacency matrix of the bipartite graph G . A biclique in G can be represented as a combinatorial rectangle in the adjacency matrix of G (as explained above). The *width* of this combinatorial rectangle is the smaller of its two dimensions, and the *length* of this combinatorial rectangle is the larger of the two dimensions. Observe that any P_4 -free bipartite graph is the union of non-intersecting combinatorial rectangles.

Let G' be a P_4 -free bipartite sub-graph of G . It is instructive to refer to Figure 3. For any combinatorial rectangle in G' , *length* $\leq 2N$ and *width* $\leq t$, since if *width* $= t+1 \leq$ *length*, then there exists a $K_{t+1,t+1}$ -subgraph in G . This observation implies that $e(G') < 2Nt$, and consequently $P_4\text{-fp}(G) \geq \frac{e(G)}{2Nt}$. ◀

The proof of Theorem 6 follows from the fact about Zarankiewicz function of $K_{t+1,t+1}$ -free bipartite graphs and Lemma 14.

¹²For $t = 1$, Levi graph of a finite projective plane yields an explicit construction.



■ **Figure 3** Let $t \in \mathbb{N}$ be a parameter. Proof intuition underlying the fact that a $K_{t+1,t+1}$ -free bipartite graph cannot have a dense P_4 -free subgraph.

Proof of Theorem 6. We construct a bipartite graph $G = (L, R, E)$ such that $|L| = |R| = N$ and it is $K_{t+1,t+1}$ -free. By Imported Theorem 13,

$$e(G) = z(N, N; t+1, t+1) \geq 2ex(N, K_{t+1,t+1}) \geq 2CN^{2-\frac{2}{t+2}},$$

where C is a positive absolute constant. By Lemma 14, we get that

$$P_4\text{-fp}(G) \geq \frac{e(G)}{2Nt} = \frac{2CN^{2-\frac{2}{t+2}}}{2Nt} = C \cdot \frac{1}{t} \cdot N^{1-\frac{2}{t+2}}. \quad \blacktriangleleft$$

Similarly, to prove that $\text{ER}(N, N, p)$ have high P_4 -free partition and cover numbers (Corollary 7), we rely on the following two observations.

1. The number of edges in a bipartite graph $G \stackrel{\$}{\leftarrow} \text{ER}(N, N, p)$ is at least $pN^2 \cdot (1 - o(1))$, with probability $1 - o(1)$.
2. Furthermore, $G \stackrel{\$}{\leftarrow} \text{ER}(N, N, p)$ is $K_{t+1,t+1}$ -avoiding with high probability, where $t+1 = \lceil 2 \log_a N \rceil$.

The proof of the second observation follows from the standard outline for first moment techniques, see, for example, [23] Chapter 7.2. More concretely, let $t+1 = \lceil 2 \log_a N \rceil$. Let \mathbb{N}_{t+1} be the random variable counting the number of $K_{t+1,t+1}$ bicliques in G . Then, we have

$$\begin{aligned} \mathbb{E}[\mathbb{N}_{t+1}] &= \binom{N}{t+1}^2 p^{(t+1)^2} \leq \left(\frac{eN}{t+1} \right)^{2(t+1)} p^{(t+1)^2} = \left(\frac{eNp^{\frac{t+1}{2}}}{t+1} \right)^{2(t+1)} \\ &\leq \left(\frac{eN \cdot \frac{1}{N}}{t+1} \right)^{2(t+1)} = o(1) \end{aligned}$$

Therefore, with probability $1 - o(1)$, there are no $K_{t+1,t+1}$ bicliques in G .

5 Upper Bounds for INT_n , DISJ_n , and INEQ_N

In this section, we establish the upper bounds for DISJ_n , INT_n , and INEQ_N as stated in Theorem 8. We also exhibit a non-trivial gap between the star arboricity, and the P_4 -free partition number of DISJ_n (see Eq. 2 of Theorem 19).

5.1 P_4 -free Partition/Cover Number and Graph Products

First, we introduce the notion of a graph product, and state some properties regarding the behavior of P_4 -free partition/cover number on graph products. These concepts are used to solve recurrence relations for DISJ_n and INT_n in the sequel.

► **Definition 15** (Graph Product). Let $G_1 = (L_1, R_1, E_1)$ and $G_2 = (L_2, R_2, E_2)$ be two bipartite graphs. Let G denote the tensor product of the two bipartite graphs G_1 , and G_2 , represented by $G_1 \times G_2$. The partite sets of G are $L_1 \times L_2$ and $R_1 \times R_2$, and the edge set is $E(G) := \{((u, a), (v, b)) : (u, v) \in E_1, (a, b) \in E_2\}$.

▷ **Claim 16** (Product of P_4 -free bipartite graphs is P_4 -free). Let G and H be two P_4 -free bipartite graphs, then $G \times H$ is also P_4 -free.

▷ **Claim 17** (Sub-multiplicativity of the P_4 -free Partition Number). Let G and H be two bipartite graphs, then the following holds for their graph product.

$$P_4\text{-fp}(G \times H) \leq P_4\text{-fp}(G) \cdot P_4\text{-fp}(H)$$

Similarly, the P_4 -free cover number is also sub-multiplicative.

▷ **Claim 18** (Sub-multiplicativity of the P_4 -free Cover Number). Let G and H be two bipartite graphs, then the following holds for their graph product.

$$P_4\text{-fc}(G \times H) \leq P_4\text{-fc}(G) \cdot P_4\text{-fc}(H)$$

5.2 Bound on DISJ_n

We show an upper bound for $P_4\text{-fp}(\text{DISJ}_n)$ using the fact that DISJ_n is the tensor product $\text{DISJ}_1^{\times n}$, and we show a lower bound for $\text{sa}(\text{DISJ}_n)$, thus exhibiting a gap between the two measures.

► **Theorem 19.** For any $n \in \mathbb{N}$, the following bounds hold.

1. $P_4\text{-fp}(\text{DISJ}_n) = P_4\text{-fp}(\text{DISJ}_1^n) \leq 2^{\lceil n/2 \rceil}$, and
2. $\text{sa}(\text{DISJ}_n) > \lceil (3/2)^n \rceil = \lceil 2.25^{n/2} \rceil$.

Proof. For the first bound, the proof proceeds by induction on n . For the base cases, observe that $P_4\text{-fp}(\text{DISJ}_1) = P_4\text{-fp}(\text{DISJ}_2) = 2$. Next, for any $2 < n \in \mathbb{N}$, we have

$$\begin{aligned} P_4\text{-fp}(\text{DISJ}_n) &= P_4\text{-fp}(\text{DISJ}_{n-2} \times \text{DISJ}_2) \\ &\leq P_4\text{-fp}(\text{DISJ}_{n-2}) \cdot P_4\text{-fp}(\text{DISJ}_2) && \text{(Claim 17)} \\ &\leq 2^{\lceil n-2/2 \rceil} \cdot 2 && \text{(Inductive Hypothesis)} \\ &= 2^{\lceil n/2 \rceil} \end{aligned}$$

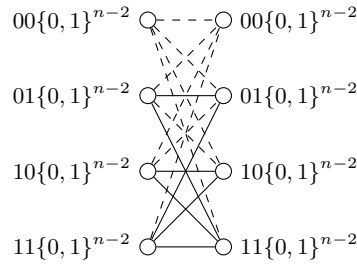
This observation completes the inductive proof.

For the second bound, note that a star forest over partite sets L and R has $< |L| + |R| = 2 \cdot 2^n$ edges in it. Note that $e(\text{DISJ}_n) = 3^n$. Therefore, one needs $> \lceil (3/2)^n \rceil$ star forests to partition the edges of DISJ_n . ◀

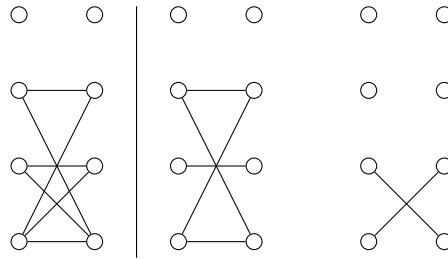
5.3 Bound on INT_n

First, we show that $P_4\text{-fc}(\text{INT}_n) \leq n$. Let $[n]$ denote the set $\{1, 2, \dots, n\}$. For each $1 \leq i \leq n$, construct a subgraph $G_i = (L_i, R_i, E_i)$ of INT_n that connect all sets that contain the element i in $[n]$. More formally, $L_i = R_i = \{S \subseteq [n] : S \ni i\}$, and $E_i = \{(S, T) : S \in L_i, T \in R_i\}$. Note that G_i is a biclique and it has 4^{n-1} edges. Note also that every edge in INT_n is covered by at least one graph G_i , for some $i \in [n]$ that witnesses the intersection of the two sets. It implies that G_1, G_2, \dots, G_n is a P_4 -free cover of INT_n . Therefore, it holds that $P_4\text{-fc}(\text{INT}_n) \leq n = \lg N$.

16:16 P_4 -free Partition and Cover Numbers & Applications

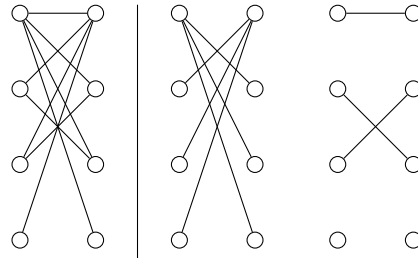


■ **Figure 4** Partition of edges of INT_n into two sets.



■ **Figure 5** Partition of G_1 in Lemma 20 into two P_4 -free graphs.

Next, we prove the upper bound for $P_4\text{-fp}(\text{INT}_n)$. Before we discuss our result, it is instructive to see that $P_4\text{-fp}(\text{INT}_n) \leq P_4\text{-fp}(\text{INT}_{n-1}) + P_4\text{-fp}(\text{DISJ}_{n-1})$, and by working out this recurrence relation we could have obtained a worse bound of $P_4\text{-fp}(\text{INT}_n) \leq 3 \cdot 2^{n/2} - 3$.



■ **Figure 6** Partition of H_1 in Lemma 20 into two P_4 -free graphs.

► **Lemma 20.** For all $n \in \mathbb{N}$ and $n \geq 3$, $P_4\text{-fp}(\text{INT}_n) \leq 2P_4\text{-fp}(\text{INT}_{n-2}) + 2$

Proof. Consider the graph INT_n . We partition the edges of INT_n into two sets. Consider an edge (u, v) where $u, v \in \{0, 1\}^n$. Let $u' \in \{0, 1\}^2$ represent the two most significant bits in u , define v' similarly. Let b_{uv} be an indicator variable that takes value 1 when u' and v' intersect, and 0 otherwise.

If for the edge (u, v) , $b_{uv} = 1$, then we add the edge to the “bold” set. When $b_{uv} = 0$, we add the edge in the “dashed” set (refer to Figure 4). Let G be the subgraph induced by the bold edges, and let H be the subgraph induced by the dashed edges.

Next, we note that $G = K_{2^{n-2}, 2^{n-2}} \times G_1$ where G_1 is a graph with P_4 -free partition number 2. See Figure 5 for an illustration. Similarly, $H = \text{INT}_{n-2} \times H_1$ where H_1 has P_4 -free partition number 2. See Figure 6 for an illustration. Combing the above observations, we get that

$$\begin{aligned} \text{P}_4\text{-fp}(\text{INT}_n) &\leq \text{P}_4\text{-fp}(G) + \text{P}_4\text{-fp}(H) \\ &\leq \text{P}_4\text{-fp}(K_{2^{n-2}, 2^{n-2}} \times G_1) + \text{P}_4\text{-fp}(\text{INT}_{n-2} \times H_1) \\ &\leq \text{P}_4\text{-fp}(K_{2^{n-2}, 2^{n-2}}) \cdot \text{P}_4\text{-fp}(G_1) + \text{P}_4\text{-fp}(\text{INT}_{n-2}) \cdot \text{P}_4\text{-fp}(H_1) \quad (\text{Claim 17}) \\ &\leq 2 + 2\text{P}_4\text{-fp}(\text{INT}_{n-2}) \quad \blacktriangleleft \end{aligned}$$

Applying Lemma 20 inductively, we have the following result as a consequence.

► **Theorem 21.** $\text{P}_4\text{-fp}(\text{INT}_n) \leq \begin{cases} 2 \cdot 2^{n/2} - 2, & \text{for even } n, \\ 3 \cdot 2^{(n-1)/2} - 2, & \text{for odd } n. \end{cases}$

5.4 Bound on INEQ_N

In fact, we prove a more general result.

▷ Claim 22 (Complement of a P_4 -free graph has a small P_4 -free partition number). Let H be a P_4 -free bipartite graph with $c \in \mathbb{N}$ connected components. Let G be the complement of H . Then, the following bound holds.

$$\text{P}_4\text{-fc}(G) \leq \text{P}_4\text{-fp}(G) \leq \begin{cases} \lceil \log_2 c \rceil, & \text{if } H \text{ has no isolated vertex,} \\ \lceil \log_2 c \rceil + 1, & \text{if } H \text{ has isolated vertices and } c > 1, \text{ and} \\ 2, & \text{if } H \text{ has isolated vertices and } c = 1. \end{cases}$$

Proposition 1 (along with a suitable embedding φ) implies the upper bound $\text{P}_4\text{-fc}(G) \leq \lceil \log_2 c \rceil$. However, we prove the stronger result that $\text{P}_4\text{-fp}(G) \leq \lceil \log_2 c \rceil$.

Our objective is to demonstrate a P_4 -free partition for G of size $\lceil \log_2 c \rceil$. The proof starts by kernelizing the graph G using the rules in [21]. Essentially, without loss of generality, one can assume that H is a matching. For simplicity assume that H is a matching with c edges and assume that it has c vertices in each partite set (i.e., there are no isolated vertices).

Next, the idea is to break the problem into half the size while including only one P_4 -free graph in the partition of G . Assume, without loss of generality, that the partite sets are $L = \{1, \dots, c\}$ and $R = \{1, \dots, c\}$, and the edges in H are (i, i) , for $1 \leq i \leq c$.

Define $L_0 := \{1, \dots, \lfloor c/2 \rfloor\}$ and $L_1 := L \setminus L_0$. Similarly, define $R_0 := \{1, \dots, \lfloor c/2 \rfloor\}$ and $R_1 := R \setminus R_0$. Observe the following.

1. The edges induced by (L_0, R_1) and (L_1, R_0) in G are disjoint bicliques. Together, they shall form one P_4 -free subgraph of G .
2. Next, the edges induced by (L_0, R_0) and (L_1, R_1) in G are disjoint and complements of matchings as well; albeit the matchings are of size $\lfloor c/2 \rfloor$ and $\lceil c/2 \rceil$, respectively. We recursively partition the disjoint union of these graphs.

Hence, Claim 22 is proved. Applying this claim for $G = \text{INEQ}_N$ and $H = \text{EQ}_N$, we have the following result.

► **Theorem 23.** *For any $N \in \mathbb{N}$, it holds that $\text{P}_4\text{-fp}(\text{INEQ}_N) \leq \log_2 N$.*

6 Lower Bounds for INT_n , DISJ_n , and INEQ_N

This section presents the proofs of the lower bounds in Theorem 8.

6.1 Bound for INEQ_N

We begin with a lower bound on $\text{P}_4\text{-fc}(\text{INEQ}_N)$ by outlining the proof of Proposition 1 below. Given a size- d P_4 -free cover $\{G_1, \dots, G_d\}$ of a bipartite graph $G = (L, R, E)$ consider the following function $\varphi: L \cup R \rightarrow \{1, 2\} \times \mathbb{N}^d$. For $i \in \{0, 1, \dots, d\}$, $\varphi(u)_i$ refers to the i -th coordinate of the mapping $\varphi(u)$. Define $\varphi(u)_0 := 1$ if $u \in L$; otherwise, if $u \in R$, define $\varphi(u)_0 := 2$. If the edge $(u, v) \in E$ is covered in the G_i by the k -th connected component, then define $\varphi(u)_i = \varphi(v)_i := k$. Since each connected component of G_i is a biclique, there are no inconsistencies introduced in defining the mapping φ . All remaining undefined coordinates of the mapping φ are completed with unique entries.

Observe that the mapping φ has the following property. For any $u \in L$ and $v \in R$, we have $(u, v) \in E$ if and only if $\varphi(u)_0 \neq \varphi(v)_0$, and there exists $i \in \{1, \dots, d\}$ such that $\varphi(u)_i = \varphi(v)_i$. Equivalently, by taking the negation, one concludes that $(u, v) \in L \times R \setminus E$ if and only if, for all $i \in \{0, 1, \dots, d\}$, we have $\varphi(u)_i \neq \varphi(v)_i$. Therefore, the complement of the bipartite graph G is a subgraph of $K_2 \times K_{\mathbb{N}}^d$, if φ is injective. Note that a redundancy-free graph cannot have $\varphi(u) = \varphi(v)$, for distinct vertices u and v . Consequently, we have Proposition 1. The other direction of the proposition does not hold because the first coordinate of the mapping φ need not be constant restricted over the vertices in L or R . However, given φ one can prepend a coordinate that is 1 for the vertices in L and 2 for the vertices in R . Therefore, if \overline{G} is an induced subgraph of $K_2 \times K_{\mathbb{N}}^d$, then G has a size- $(d + 1)$ P_4 -free cover.

For deriving the lower bound, consider $G = \text{INEQ}_N$, i.e., $\overline{G} = \text{EQ}_N$. Using the algebraic lower-bounding technique of [47, 4, 63], one concludes $d \geq \lceil \log_2 N \rceil$. Therefore, we have the following result.

► **Theorem 24.** *For any $N \in \mathbb{N}$, it holds that $\text{P}_4\text{-fc}(\text{INEQ}_N) \geq \lceil \log_2 N \rceil$.*

6.2 Bound on DISJ_n

We rely on a counting technique to obtain this lower bound. Intuitively, existing algebraic technique are useful to obtain logarithmic lower bounds. However, in this problem, we seek to prove a polynomial lower bound.

► **Theorem 25.** *For all $n \in \mathbb{N}$, the following bound holds.*

$$\text{P}_4\text{-fp}(\text{DISJ}_n) \geq \text{P}_4\text{-fc}(\text{DISJ}_n) \geq N^{\log_2 3 - 3/2} \approx N^{0.085}$$

The following lemma is the key for the proof of Theorem 25.

► **Lemma 26.** *Any P_4 -free subgraph of DISJ_n has at most $N\sqrt{N}$ edges.*

To prove Lemma 26, we shall use the following claims (see full version for their proofs).

▷ **Claim 27.** Any biclique subgraph of DISJ_n has at most N edges.

▷ **Claim 28.** Let $\{(a_i, b_i)\}_{i \in \mathbb{N}}$ be a sequence of non-negative numbers. Then,

$$\sum_{i \in \mathbb{N}} a_i b_i \leq \sqrt{\left(\max_{i \in \mathbb{N}} a_i b_i \right) \left(\sum_{i \in \mathbb{N}} a_i \right) \left(\sum_{i \in \mathbb{N}} b_i \right)}.$$

Furthermore, equality holds if and only if (a) for all $i \in \mathbb{N}$, one has $a_i > 0$ iff $b_i > 0$. (b) all positive a_i are constant, and (c) all positive b_i are constant.

Proof of Lemma 26. Suppose G is a P_4 -free subgraph of DISJ_n . Let $K_{a_1, b_1}, K_{a_2, b_2}, \dots, K_{a_m, b_m}$ be the (biclique) connected components of G , where $a_i \in \mathbb{N}, b_i \in \mathbb{N}$ for every $1 \leq i \leq m$ and $m \in \mathbb{N}$. The total number of edges in G is $\sum_{i=1}^m a_i b_i$. We shall show that $\sum_{i=1}^m a_i \cdot b_i \leq N\sqrt{N}$. By Claim 27, it holds that $a_i \cdot b_i \leq N$ for every $1 \leq i \leq m$. Since all the left partite sets of $K_{a_1, b_1}, K_{a_2, b_2}, \dots, K_{a_m, b_m}$ are disjoint, it holds that $\sum_{i=1}^m a_i \leq N$. Similarly, $\sum_{i=1}^m b_i \leq N$. Therefore, applying Claim 28, the following inequality holds.

$$\begin{aligned} \sum_{i=1}^m a_i b_i &\leq \sqrt{\left(\max_i a_i b_i\right) \left(\sum_{i=1}^m a_i\right) \left(\sum_{i=1}^m b_i\right)} \\ &\leq \sqrt{N \cdot N \cdot N} = N^{3/2} \end{aligned}$$

Thus, any P_4 -free subgraph of DISJ_n has at most $N^{3/2}$ edges. \blacktriangleleft

Now, we are ready to prove Theorem 25.

Proof of Theorem 25. First, observe that there are 3^n edges in DISJ_n . By Lemma 26, any P_4 -free subgraph of DISJ_n has at most $N\sqrt{N}$ edges. Therefore, we have

$$\text{P}_4\text{-fp}(\text{DISJ}_n) \geq \text{P}_4\text{-fc}(\text{DISJ}_n) \geq \frac{3^n}{N\sqrt{N}} = N^{\log_2 3 - 3/2} \approx N^{0.085}$$

as desired. \blacktriangleleft

6.3 Bounds on P_4 -free Cover Number of INT_n

We shall prove the following lower bound on the P_4 -free cover number of INT_n .

► **Theorem 29.** *For all $n \in \mathbb{N}$, the following bounds hold.*

$$n - \frac{1}{2} \left(\lg \pi + \lg \left(\frac{n+1}{2} + \frac{1}{4} + \frac{1}{64(n+1)} \right) \right) \leq \text{P}_4\text{-fc}(\text{INT}_n).$$

First, we state claims needed for the proof of Theorem 29 (see full version for their proofs).

▷ **Claim 30.** For every $n \in \mathbb{N}$, the following bound holds.

$$\lg \binom{n}{\lfloor n/2 \rfloor} \geq n - \frac{1}{2} \left(\lg \pi + \lg \left(\frac{n+1}{2} + \frac{1}{4} + \frac{1}{64(n+1)} \right) \right)$$

▷ **Claim 31.** Let G be a bipartite graph. Then, for every induced subgraph H of G , the following inequality holds.

$$\text{P}_4\text{-fc}(H) \leq \text{P}_4\text{-fc}(G)$$

Proof of Theorem 29. Consider the induced subgraph $G = (L', R', E')$ of INT_n , where $L' = \{S \subseteq [n] : |S| = \lfloor \frac{n}{2} \rfloor\}$, $R' = \{T \subseteq [n] : |T| = \lceil \frac{n}{2} \rceil\}$. Observe that each vertex $S \in L'$ is connected to every $T \in R'$ except when $T = [n] \setminus S$. Thus, graph G is the complement of a matching of size M , where $M = \binom{n}{\lfloor n/2 \rfloor}$. Using the algebraic lower-bounding technique of [47] and Proposition 1, one concludes that

$$\text{P}_4\text{-fc}(G) \geq \lceil \lg M \rceil \geq n - \frac{1}{2} \left(\lg \pi + \lg \left(\frac{n+1}{2} + \frac{1}{4} + \frac{1}{64(n+1)} \right) \right),$$

where the last inequality follows from Claim 30. Finally, by Claim 31, $\text{P}_4\text{-fc}(G) \leq \text{P}_4\text{-fc}(\text{INT}_n)$. Therefore, we have

$$n - \frac{1}{2} \left(\lg \pi + \lg \left(\frac{n+1}{2} + \frac{1}{4} + \frac{1}{64(n+1)} \right) \right) \leq \text{P}_4\text{-fc}(\text{INT}_n),$$

as desired. \blacktriangleleft

7 Relation to Graph Embedding

This section presents the connection between P_4 -free partition/cover number and product/-Prague dimension.

7.1 P_4 -free Cover Number

▷ **Claim 32.** If a bipartite graph $G = (L, R, E)$ has a size- d P_4 -free covering, then the complement bipartite graph $\overline{G} = (L, R, L \times R \setminus E)$ is an induced subgraph of $K_2 \times K_{\mathbb{N}}^d$.

Proof. Let G_1, \dots, G_d be a size- d P_4 -free cover of G . Define a vertex mapping $\varphi: L \cup R \rightarrow K_2 \times K_{\mathbb{N}}^d$ as follows. Let $\varphi(u)_i$ denote the i -th coordinate of the mapping $\varphi(u)$. Define $\varphi(u)_0 = 0$, for all $u \in L$, and $\varphi(v)_0 = 1$, for all $v \in R$. For $i \in \{1, \dots, d\}$, define $\varphi(u)_i = \varphi(v)_i = k$, for every edge (u, v) in the k -th connected component of G_i . All remaining entries of φ are filled with unique values. One can verify that $(u, v) \in L \times R \setminus E$ if and only if $\varphi(u)$ and $\varphi(v)$ differ in every coordinate, that is, $\varphi(u)_i \neq \varphi(v)_i$ for every $i \in \{0, 1, \dots, d\}$. Therefore, the complement bipartite graph \overline{G} is an induced subgraph of $K_2 \times K_{\mathbb{N}}^d$.

We emphasize that the vertex mapping φ has the additional property that $\varphi(u)$ and $\varphi(v)$ have t identical coordinates if and only if the edge (u, v) is covered in t P_4 -free graphs among G_1, \dots, G_d . This property shall be useful in the proof of Claim 35. ◁

▷ **Claim 33.** If a loopless undirected graph $H = (L \cup R, E)$ is an induced subgraph of $K_{\mathbb{N}}^d$ and $E \subseteq L \times R$, then the bipartite graph $H' = (L, R, L \times R \setminus E)$ has a size- d P_4 -free covering.

Proof. Suppose a loopless undirected graph $H = (L \cup R, E)$ is an induced subgraph of $K_{\mathbb{N}}^d$ and $E \subseteq L \times R$. Then, there exists a vertex mapping $\varphi: L \cup R \rightarrow \mathbb{N}^d$ such that $(u, v) \in E$ if and only if there exists $i \in \{1, 2, \dots, d\}$ such that $\varphi(u)_i = \varphi(v)_i$. Define a new vertex mapping $\varphi^+: L \cup R \rightarrow \{1, 2\} \times \mathbb{N}^d$ as follows.

$$\varphi^+(u) = \begin{cases} (1, \varphi(u)), & \text{if } u \in L \\ (2, \varphi(u)), & \text{otherwise.} \end{cases}$$

For $i \in \{1, 2, \dots, d\}$, define $G_i = (L, R, E_i)$ such that E_i is the set of all $u \in L$ and $v \in R$ such that $\varphi^+(u)_i = \varphi^+(v)_i$. Observe that the set of vertices $u \in L$ such that $\varphi^+(u)_i = k$ and the set of vertices $v \in R$ such that $\varphi^+(v)_i = k$ for some $k \in \mathbb{N}$ form a biclique, and each E_i is a disjoint union of bicliques. Furthermore, an edge $(u, v) \in E$ if and only if there exists an $i \in \{1, 2, \dots, d\}$ such that $\varphi(u)_i = \varphi(v)_i$ which is equivalent to $\varphi^+(u)_i = \varphi^+(v)_i$. This implies that E_i cover the edge (u, v) . Therefore, E_1, E_2, \dots, E_d is a P_4 -free cover of H .

The G_1, \dots, G_d have the property that if an edges (u, v) is covered t times by these P_4 -free graphs, then $\varphi^+(u)$ intersects $\varphi^+(v)$ in exactly t coordinates. This property of the vertex mapping shall be useful in the proof of Claim 36. ◁

The following result is a consequence of Claim 32 and Claim 33.

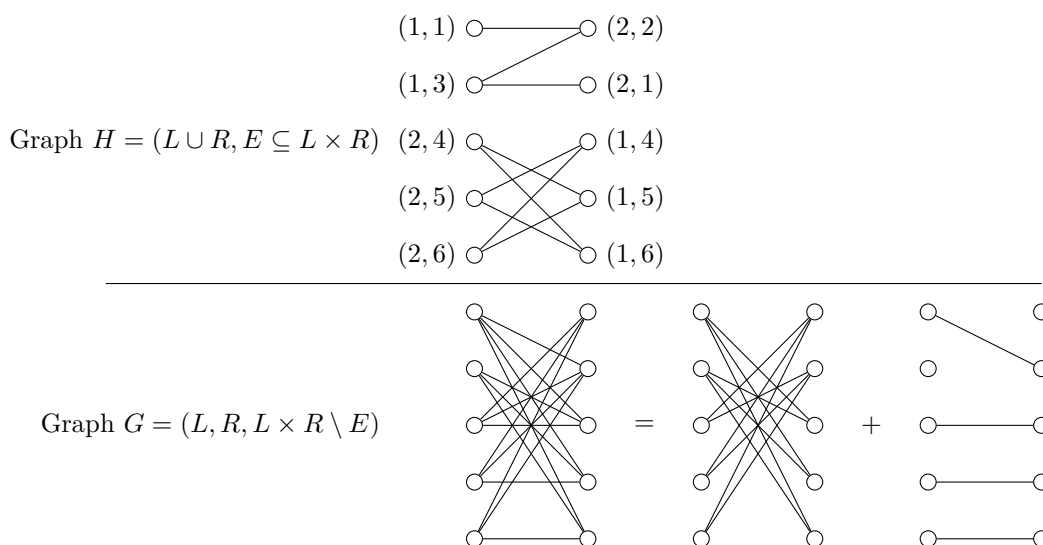
► **Corollary 34.** *Let $G = (L, R, E)$ be a bipartite graph and $H = (L \cup R, E)$ be a loopless undirected graph. Then, the following identity holds.*

$$\text{pdim}(H) \in \{\text{P}_4\text{-fc}(G), \text{P}_4\text{-fc}(G) + 1\},$$

or, equivalently,

$$\text{P}_4\text{-fc}(G) \in \{\text{pdim}(H) - 1, \text{pdim}(H)\}.$$

Note that the additive slack of 1 in Corollary 34 is necessary. Figure 7 gives an example.



■ **Figure 7** Example for the tightness of Corollary 34. Note that the loopless undirected graph $H = (L \cup R, E) = P_4 + C_6$, where $E \subseteq L \times R$, is an induced subgraph of $K_2 \times K_N$. The (partition) vertex mapping of each vertex is explicitly mentioned next to it. However, the bipartite graph $G = (L, R, L \times R \setminus E)$ is not P_4 -free and, hence, $P_4\text{-fc}(G) \geq 2$; in fact, we have $P_4\text{-fc}(G) = P_4\text{-fp}(G) = 2$. The edges of G partition into $K_{2,3} + K_{3,2}$ and $4K_{1,1}$.

7.2 P_4 -free Partition Number

Suppose a graph H is an induced subgraph of $K_{\mathbb{N}}^d$ via a vertex mapping $\varphi: V(H) \rightarrow \mathbb{N}^d$. The vertex mapping φ is a *partition* if the following conditions are satisfied.

1. If $(u, v) \in E(H)$, then $\varphi(u)_i \neq \varphi(v)_i$, for all $i \in \{1, 2, \dots, d\}$.
 2. If $(u, v) \notin E(H)$, then there exists a *unique* $i \in \{1, 2, \dots, d\}$ such that $\varphi(u)_i = \varphi(v)_i$.
- We emphasize that in an unrestricted vertex mapping, instead of (2) above, we insist that there exists an $i \in \{1, 2, \dots, d\}$ (not necessarily a *unique* i). Let $\text{pdim}^*(H)$ represent the minimum $d \in \mathbb{N}$ such that H is an induced subgraph of $K_{\mathbb{N}}^d$ via a partition vertex mapping.

▷ **Claim 35.** If a bipartite graph $G = (L, R, E)$ has a size- d P_4 -free partitioning, then the complement bipartite graph $\overline{G} = (L, R, L \times R \setminus E)$ is an induced subgraph of $K_2 \times K_{\mathbb{N}}^d$ via a partition vertex mapping.

▷ **Claim 36.** If a loopless undirected graph $H = (L \cup R, E)$ is an induced subgraph of $K_{\mathbb{N}}^d$ via a partition vertex mapping and $E \subseteq L \times R$, then the bipartite graph $H' = (L, R, L \times R \setminus E)$ has a size- d P_4 -free partitioning.

The proofs of Claim 35 and Claim 36 are identical to the proofs of Claim 32 and Claim 33, respectively, utilizing the fact that the vertex mapping is a partition. As a consequence of Claim 35 and Claim 36, we have the following result.

► **Corollary 37.** Let $G = (L, R, E)$ be a bipartite graph and $H = (L \cup R, E)$ be a loopless undirected graph. Then, the following identity holds.

$$\text{pdim}^*(H) \in \{P_4\text{-fp}(G), P_4\text{-fp}(G) + 1\},$$

or equivalently

$$P_4\text{-fp}(G) \in \{\text{pdim}^*(H) - 1, \text{pdim}^*(H)\}.$$

References

- 1 Rudolf Ahlswede and Imre Csiszár. Common randomness in information theory and cryptography - I: secret sharing. *IEEE Trans. Inf. Theory*, 39(4):1121–1132, 1993. doi:10.1109/18.243431.
- 2 Rudolf Ahlswede and Imre Csiszár. Common randomness in information theory and cryptography - part II: CR capacity. *IEEE Trans. Inf. Theory*, 44(1):225–240, 1998. doi:10.1109/18.651026.
- 3 I. Algor and Noga Alon. The star arboricity of graphs. *Discret. Math.*, 75(1-3):11–22, 1989. doi:10.1016/0012-365X(89)90073-3.
- 4 Noga Alon. Covering graphs by the minimum number of equivalence relations. *Combinatorica*, 6(3):201–206, 1986.
- 5 Noga Alon and Ryan Alweiss. On the product dimension of clique factors. *European Journal of Combinatorics*, 86:103097, 2020.
- 6 Noga Alon, Lajos Rónyai, and Tibor Szabó. Norm-graphs: Variations and applications. *J. Comb. Theory, Ser. B*, 76(2):280–290, 1999. doi:10.1006/jctb.1999.1906.
- 7 James Aspnes, Richard Beigel, Merrick L. Furst, and Steven Rudich. The expressive power of voting polynomials. In *23rd Annual ACM Symposium on Theory of Computing*, pages 402–409, New Orleans, LA, USA, May 6–8 1991. ACM Press. doi:10.1145/103418.103461.
- 8 Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 317–342, San Diego, CA, USA, February 24–26 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-54242-8_14.
- 9 Alexander R. Block, Hemanta K. Maji, and Hai H. Nguyen. Secure computation based on leaky correlations: High resilience setting. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 20–24 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-63715-0_1.
- 10 Andrej Bogdanov and Elchanan Mossel. On extracting common random bits from correlated sources. *IEEE Trans. Inf. Theory*, 57(10):6351–6355, 2011. doi:10.1109/TIT.2011.2134067.
- 11 Béla Bollobás. *Extremal graph theory*. Courier Corporation, 2004.
- 12 W. G. Brown. On graphs that do not contain a thomsen graph. *Canadian Mathematical Bulletin*, 9(3):281–285, 1966. doi:10.4153/CMB-1966-036-2.
- 13 Ignacio Cascudo, Ivan Damgård, Felipe Lacerda, and Samuel Ranellucci. Oblivious transfer from any non-trivial elastic noisy channel via secret key agreement. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 204–234, Beijing, China, October 31 – November 3 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53641-4_9.
- 14 Siu On Chan, Elchanan Mossel, and Joe Neeman. On extracting common random bits from correlated sources on large alphabets. *IEEE Trans. Inf. Theory*, 60(3):1630–1637, 2014. doi:10.1109/TIT.2014.2301155.
- 15 G. Chen, S. Fujita, A. Gyarfás, J. Lehel, and A. Toth. Around a biclique cover conjecture, 2012. arXiv:1212.6861.
- 16 Claude Crépeau and Joe Kilian. Achieving oblivious transfer using weakened security assumptions (extended abstract). In *29th Annual Symposium on Foundations of Computer Science*, pages 42–52, White Plains, NY, USA, October 24–26 1988. IEEE Computer Society Press. doi:10.1109/SFCS.1988.21920.
- 17 Claude Crépeau and Joe Kilian. Weakening security assumptions and oblivious transfer (abstract). In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO’88*, volume 403 of *Lecture Notes in Computer Science*, pages 2–7, Santa Barbara, CA, USA, August 21–25 1990. Springer, Heidelberg, Germany. doi:10.1007/0-387-34799-2_1.

- 18 Claude Crépeau, Kirill Morozov, and Stefan Wolf. Efficient unconditional oblivious transfer from almost any noisy channel. In Carlo Blundo and Stelvio Cimato, editors, *SCN 04: 4th International Conference on Security in Communication Networks*, volume 3352 of *Lecture Notes in Computer Science*, pages 47–59, Amalfi, Italy, September 8–10 2005. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-30598-9_4.
- 19 Ivan Damgård, Joe Kilian, and Louis Salvail. On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT’99*, volume 1592 of *Lecture Notes in Computer Science*, pages 56–73, Prague, Czech Republic, May 2–6 1999. Springer, Heidelberg, Germany. doi:10.1007/3-540-48910-X_5.
- 20 Paul Erdős and Joel Spencer. *Probabilistic methods in combinatorics*, volume 17. Academic Press New York, 1974.
- 21 Herbert Fleischner, Egbert Mujuni, Daniël Paulusma, and Stefan Szeider. Covering graphs with few complete bipartite subgraphs. *Theoretical Computer Science*, 410(21):2045–2053, 2009. doi:10.1016/j.tcs.2008.12.059.
- 22 DJ Foulis. Empirical logic, xeroxed course notes. *University of Massachusetts, Amherst, Massachusetts (1969-1970)*, 1969.
- 23 Alan Frieze and Michał Karoński. *Introduction to random graphs*. Cambridge University Press, 2016.
- 24 Zoltán Füredi and Miklós Simonovits. The history of degenerate (bipartite) extremal graph problems. In *Erdős Centennial*, pages 169–264. Springer, 2013.
- 25 Peter Gács and János Körner. Common information is far less than mutual information. *Problems of Control and Information Theory*, 2(2):149–162, 1973.
- 26 Mikael Goldmann. On the power of a threshold gate at the top. *Inf. Process. Lett.*, 63(6):287–293, 1997. doi:10.1016/S0020-0190(97)00141-5.
- 27 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27 1987. ACM Press. doi:10.1145/28395.28420.
- 28 Daniel Gonçalves and Pascal Ochem. On star and caterpillar arboricity. *Discret. Math.*, 309(11):3694–3702, 2009. doi:10.1016/j.disc.2008.01.041.
- 29 Parikshit Gopalan and Rocco A. Servedio. Learning and lower bounds for ac^0 with threshold gates. In Maria J. Serna, Ronen Shaltiel, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, volume 6302 of *Lecture Notes in Computer Science*, pages 588–601. Springer, 2010. doi:10.1007/978-3-642-15369-3_44.
- 30 Richard Hammack, Wilfried Imrich, and Sandi Klavžar. *Handbook of product graphs*. CRC press, 2011.
- 31 Chinh T. Hoàng and Van Bang Le. P_4 -Colorings and P_4 -Bipartite Graphs. *Discrete Mathematics and Theoretical Computer Science*, 4(2):109–122, 2001. URL: <https://hal.inria.fr/hal-00958951>.
- 32 Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *21st Annual ACM Symposium on Theory of Computing*, pages 12–24, Seattle, WA, USA, May 15–17 1989. ACM Press. doi:10.1145/73007.73009.
- 33 Jeffrey C. Jackson, Adam Klivans, and Rocco A. Servedio. Learnability beyond AC^0 . In *34th Annual ACM Symposium on Theory of Computing*, pages 776–784, Montréal, Québec, Canada, May 19–21 2002. ACM Press. doi:10.1145/509907.510018.
- 34 Minghui Jiang. Trees, paths, stars, caterpillars and spiders. *Algorithmica*, 80(6):1964–1982, 2018.
- 35 Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*. Springer Publishing Company, Incorporated, 2012.

- 36 Heinz A Jung. On a class of posets and the corresponding comparability graphs. *Journal of Combinatorial Theory, Series B*, 24(2):125–133, 1978.
- 37 Sebastian Kaiser. *Biclustering: methods, software and application*. PhD thesis, lmu, 2011.
- 38 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.
- 39 Dakshita Khurana, Hemanta K. Maji, and Amit Sahai. Secure computation from elastic noisy channels. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 184–212, Vienna, Austria, May 8–12 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49896-5_7.
- 40 Joe Kilian. Founding cryptography on oblivious transfer. In *20th Annual ACM Symposium on Theory of Computing*, pages 20–31, Chicago, IL, USA, May 2–4 1988. ACM Press. doi:10.1145/62212.62215.
- 41 Joe Kilian. A general completeness theorem for two-party games. In *23rd Annual ACM Symposium on Theory of Computing*, pages 553–560, New Orleans, LA, USA, May 6–8 1991. ACM Press. doi:10.1145/103418.103475.
- 42 Joe Kilian. More general completeness theorems for secure two-party computation. In *32nd Annual ACM Symposium on Theory of Computing*, pages 316–324, Portland, OR, USA, May 21–23 2000. ACM Press. doi:10.1145/335305.335342.
- 43 János Kollár, Lajos Rónyai, and Tibor Szabó. Norm-graphs and bipartite turán numbers. *Combinatorica*, 16(3):399–406, 1996. doi:10.1007/bf01261323.
- 44 Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.
- 45 H Lerchs. On cliques and kernels. *Department of Computer Science, University of Toronto*, 1971.
- 46 H Lerchs. On the clique-kernel structure of graphs. *Dept. of Computer Science, University of Toronto*, 1972.
- 47 László Lovász, J Nešetřil, and Ales Pultr. On a product dimension of graphs. *Journal of Combinatorial Theory, Series B*, 29(1):47–67, 1980.
- 48 Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM transactions on computational biology and bioinformatics*, 1(1):24–45, 2004.
- 49 Ueli M. Maurer. Perfect cryptographic security from partially independent channels. In *23rd Annual ACM Symposium on Theory of Computing*, pages 561–571, New Orleans, LA, USA, May 6–8 1991. ACM Press. doi:10.1145/103418.103476.
- 50 Ueli M. Maurer. A universal statistical test for random bit generators. *Journal of Cryptology*, 5(2):89–105, January 1992. doi:10.1007/BF00193563.
- 51 Ueli M. Maurer. Secret key agreement by public discussion from common information. *IEEE Trans. Inf. Theory*, 39(3):733–742, 1993. doi:10.1109/18.256484.
- 52 Elchanan Mossel and Ryan O’Donnell. Coin flipping from a cosmic source: On error correction of truly random bits. *Random Structures & Algorithms*, 26(4):418–436, 2005. doi:10.1002/rsa.20062.
- 53 Elchanan Mossel, Ryan O’Donnell, Oded Regev, Jeffrey E Steif, and Benny Sudakov. Non-interactive correlation distillation, inhomogeneous markov chains, and the reverse bonami-beckner inequality. *Israel Journal of Mathematics*, 154(1):299–336, 2006.
- 54 J Nešetřil and Ales Pultr. A dushnik-miller type dimension of graphs and its complexity. In *International Conference on Fundamentals of Computation Theory*, pages 482–493. Springer, 1977.

- 55 Jaroslav Nešetřil and Vojtěch Rödl. A simple proof of the galvin-ramsey property of the class of all finite graphs and a dimension of a graph. *Discrete Mathematics*, 23(1):49–55, 1978.
- 56 Noam Nisan and David Zuckerman. More deterministic simulation in logspace. In *25th Annual ACM Symposium on Theory of Computing*, pages 235–244, San Diego, CA, USA, May 16–18 1993. ACM Press. doi:10.1145/167088.167162.
- 57 James Orlin. Contentment in graph theory: Covering graphs with cliques. *Indagationes Mathematicae (Proceedings)*, 80(5):406–424, 1977. doi:10.1016/1385-7258(77)90055-5.
- 58 Trevor Pinto. Biclique covers and partitions. *arXiv preprint arXiv:1307.6363*, 2013.
- 59 Svatopluk Poljak, D Rödl, and Ales Pultr. On a product dimension of bipartite graphs. *Journal of graph theory*, 7(4):475–486, 1983.
- 60 Dieter Seinsche. On a property of the class of n-colorable graphs. *Journal of Combinatorial Theory, Series B*, 16(2):191–193, 1974.
- 61 Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.
- 62 David P Sumner. Dacey graphs. *Journal of the Australian Mathematical Society*, 18(4):492–502, 1974.
- 63 Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, NJ, 1996.
- 64 Jürg Wullschleger. Oblivious-transfer amplification. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 555–572, Barcelona, Spain, May 20–24 2007. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-72540-4_32.
- 65 Jürg Wullschleger. Oblivious transfer from weak noisy channels. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 332–349. Springer, Heidelberg, Germany, March 15–17 2009. doi:10.1007/978-3-642-00457-5_20.
- 66 Aaron Wyner. The common information of two dependent random variables. *IEEE Transactions on Information Theory*, 21(2):163–179, 1975. doi:10.1109/TIT.1975.1055346.
- 67 Ke Yang. On the (im)possibility of non-interactive correlation distillation. In Martin Farach-Colton, editor, *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium*, volume 2976 of *Lecture Notes in Computer Science*, pages 222–231, Buenos Aires, Argentina, April 5–8 2004. Springer, Heidelberg, Germany.
- 68 Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, November 3–5 1982. IEEE Computer Society Press. doi:10.1109/SFCS.1982.38.

Replacing Probability Distributions in Security Games via Hellinger Distance

Kenji Yasunaga  

Graduate School of Information Science and Technology, Osaka University, Japan

Abstract

Security of cryptographic primitives is usually proved by assuming “ideal” probability distributions. We need to replace them with approximated “real” distributions in the real-world systems without losing the security level. We demonstrate that the Hellinger distance is useful for this problem, while the statistical distance is mainly used in the cryptographic literature. First, we show that for preserving λ -bit security of a given security game, the closeness of $2^{-\lambda/2}$ to the ideal distribution is sufficient for the Hellinger distance, whereas $2^{-\lambda}$ is generally required for the statistical distance. The result can be applied to both search and decision primitives through the bit security framework of Micciancio and Walter (Eurocrypt 2018). We also show that the Hellinger distance gives a tighter evaluation of closeness than the max-log distance when the distance is small. Finally, we show that the leftover hash lemma can be strengthened to the Hellinger distance. Namely, a universal family of hash functions gives a strong randomness extractor with optimal entropy loss for the Hellinger distance. Based on the results, a λ -bit entropy loss in randomness extractors is sufficient for preserving λ -bit security. The current understanding based on the statistical distance is that a 2λ -bit entropy loss is necessary.

2012 ACM Subject Classification Security and privacy → Cryptography; Mathematics of computing → Probability and statistics

Keywords and phrases Security proof, Hellinger distance, randomness extractor, entropy loss

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.17

Funding This work was supported in part by JSPS Grants-in-Aid for Scientific Research Numbers 16H01705, 17H01695, and 18K11159.

1 Introduction

Security of cryptographic primitives relies on the use of randomness sources. Secret keys and random bits are usually assumed to be sampled from uniform distributions. Various probability distributions other than uniform ones appear in cryptography. In lattice-based cryptography, discrete Gaussian distributions are used for the hardness of the Learning with Errors (LWE) problem [32, 31, 21, 27] and the tight reductions for the Short Integer Solution (SIS) problem [24, 23]. Adding noise from Laplace distributions enables data privacy of statistical databases in differential privacy [15, 14, 16].

To ensure the security of primitives, we usually define a security game played by an adversary and show that the adversary’s success probability is sufficiently close to some value. In the proof, we assume we can use “ideal” probability distributions. We need to replace them with approximated “real” distributions in real-world systems. For example, in a security game of an encryption scheme, the adversary receives a ciphertext and tries to guess which of the two plaintexts were encrypted. The scheme is secure if the success probability is sufficiently close to $1/2$. A secret key and random coins for encryption are assumed to be sampled from uniform distributions. One may employ the output of a randomness extractor [33, 11] as a randomness source since the output distribution is sufficiently close to the uniform distribution. However, the distance to the ideal distribution may affect the security level of primitives. A question is which closeness measure of distributions should be used when replacing distributions in security games.



© Kenji Yasunaga;
licensed under Creative Commons License CC-BY 4.0
2nd Conference on Information-Theoretic Cryptography (ITC 2021).
Editor: Stefano Tessaro; Article No. 17; pp. 17:1–17:15



Leibniz International Proceedings in Informatics
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

17:2 Replacing Probability Distributions in Security Games via Hellinger Distance

In cryptographic literature, we mainly employ the *statistical distance* (a.k.a. the *total variation distance*) to measure distribution closeness. The main reason is that it enables a straightforward analysis of the resulting security levels. The statistical distance is defined as the maximum difference of probabilities of events between two distributions. By employing a distribution P that is close to ideal Q within ϵ in the statistical distance, we can guarantee that the adversary's success probability only increases by at most ϵ . However, there may not be any other reason for using the statistical distance.

Also, achieving security by the statistical distance has some limitations. Radhakrishnan and Ta-Shma [30] showed a lower bound on the entropy loss of randomness extractors. Roughly, the result implies that to extract a uniformly random string from an entropy source, we need to lose $2 \log(1/\epsilon)$ of entropy, where ϵ is the distance to the uniform distribution. Based on this result, if we extract a random string from a source of 120-bit entropy by ensuring 50-bit security, the output bit should be of length at most $120 - 2 \cdot 50 = 20$. This loss of entropy is crucial when using biometric data as entropy sources [11, 8], where a limited amount of entropy can be used. Randomness extraction (or key derivation) from weak sources arises in many situations of cryptography, including Diffie-Hellman key exchange [17, 20] and random number generators from physical sources [5, 4], to name a few.

Our Contribution

In this work, we propose to use the *Hellinger distance* for replacing distributions in security games. Roughly speaking, we show that the closeness of $2^{-\lambda/2}$ in the Hellinger distance is sufficient to preserve λ -bit security. When using the statistical distance, the closeness of $2^{-\lambda}$ is, in general, necessary to achieve the same security level.

To discuss the *bit security*, we use the framework of Micciancio and Walter [26]. Their framework can smoothly connect the bit security between *search* and *decision* primitives. Their definition is the same as the standard one for search primitives, where the secret is chosen from a sufficiently large space. For decision primitives, in which the attacker tries to guess a secret bit, the definition of the advantage is different from the standard one. See Section 3 for the details. We show that the distance closeness of $2^{-\lambda/2}$ in the Hellinger distance is sufficient for preserving the bit security for both search and decision primitives.

Next, we show that the Hellinger distance gives a tighter evaluation of closeness than the *max-log distance*, the probability metric introduced in [25, 26]. The work showed that the closeness of $2^{-\lambda/2}$ in the max-log distance is sufficient for preserving λ -bit security. We proved that the Hellinger distance is bounded above by the max-log distance as long as the max-log distance is at most $\sqrt{2} - 1$. Also, we present a concrete example of a distribution pair such that their Hellinger distance is exponentially small, while their max-log distance is a constant.

Finally, we demonstrate the usefulness of using the Hellinger distance in the problem of randomness extraction (or information-theoretic key derivation). We show that the leftover hash lemma [6, 19] can be strengthened to the Hellinger distance without losing the security level. Namely, a *universal* family of hash functions gives a strong randomness extractor with optimal entropy loss even when measuring in the Hellinger distance. We can conclude that the entropy loss of λ -bit is sufficient for preserving λ -bit security. In general, the entropy loss of 2λ -bit is necessary to preserve bit security when using the statistical distance.

Techniques

We describe a technical overview of our results. Although the actual proofs seem different from the below, it reflects the difference between the statistical distance and the Hellinger distance. Let $P = (P_1, P_2, \dots)$ and $Q = (Q_1, Q_2, \dots)$ be a pair of probability distribution ensembles such that each P_i is close to Q_i . Let ϵ_A^Q be the probability that an adversary A succeeds in the security game in which samples from Q is used. We want to bound the probability ϵ_A^P , which is the success probability when using P instead of Q .

For $\ell \in \mathbb{N}$, we define the probability μ_ℓ^Q that A succeeds in at least one out of ℓ independent plays of G_A^Q . As long as ℓ is small compared to $1/\epsilon_A^P$, it holds that $\mu_\ell^P \approx \ell \cdot \epsilon_A^P$. Since the number of sample queries in each game is bounded above by the running time T_A of A , $\mu_\ell^P \leq \mu_\ell^Q + \text{SD}(P^\ell, Q^\ell) \leq \mu_\ell^Q + \ell T_A \cdot \max_i \text{SD}(P_i, Q_i)$, where $\text{SD}(P_i, Q_i)$ is the statistical distance between P_i and Q_i , and P^ℓ is the ℓ -fold product of P . Note that we use the relation $\text{SD}(P^\ell, Q^\ell) \leq \ell T_A \cdot \max_i \text{SD}(P_i, Q_i)$. Now, it holds that $\epsilon_A^P \approx \ell^{-1} \cdot \mu_\ell^P \leq \ell^{-1} \cdot (\mu_\ell^Q + \ell T_A \cdot \max_i \text{SD}(P_i, Q_i)) \approx \epsilon_A^Q + T_A \cdot \max_i \text{SD}(P_i, Q_i)$. Thus, if the primitive has λ -bit security, i.e., $\epsilon_A^Q/T_A \leq 2^{-\lambda}$, then $\epsilon_A^P/T_A \leq 2^{-\lambda} + \max_i \text{SD}(P_i, Q_i)$. It implies that $\max_i \text{SD}(P_i, Q_i) \leq 2^{-\lambda}$ is required for preserving bit security. For the Hellinger distance $\text{HD}(P_i, Q_i)$, we provide a technical lemma (Lemma 1) showing that $\text{SD}(P^\ell, Q^\ell) \leq \sqrt{2\ell T_A} \cdot \max_i \text{HD}(P_i, Q_i)$. Therefore, we have $\epsilon_A^P \leq \epsilon_A^Q + \sqrt{2\ell^{-1} T_A} \cdot \max_i \text{HD}(P_i, Q_i)$. Hence, if the primitive has λ -bit security, $\epsilon_A^P/T_A \leq 2^{-\lambda} + \sqrt{2(\ell T_A)^{-1}} \cdot \max_i \text{HD}(P_i, Q_i)$, implying that, by choosing $\ell = 1/\epsilon_A^P$, it suffices to satisfy $\max_i \text{HD}(P_i, Q_i) \leq 2^{-\lambda/2}$ for preserving bit security.

The leftover hash lemma essentially gives an upper bound on the *collision probability* of the hash functions chosen from a universal family. If the collision probability is bounded, it is close to uniform in the Hellinger distance. This relation was provided by Chung and Vadhan [9] using Hölder's inequality. Based on the relation, we show that a universal family of hash functions gives a strong randomness extractor for the Hellinger distance. Notably, we can achieve the same parameters as in the case of the statistical distance. Thus, the optimal entropy loss is achieved by universal hash functions.

Related Work

Barak et al. [3] initiated the study on improving the leftover hash lemma for a limited class of primitives. The work of [3, 13] showed that the bound of [30] could be improved for the search primitives and the square-friendly decision primitives, including stateless encryption schemes and weak pseudorandom functions. Specifically, the entropy loss of λ is sufficient for square-friendly primitives. For search primitives, Dodis, Pietrzak, and Wichs [12] achieved the entropy loss of $O(\log \lambda)$ in randomness extraction with $O(\lambda)$ -wise independent hash functions. Matsuda et al. [22] generalized the results of [13] by using the Rényi divergence for capturing the case that the ideal distribution is not uniform. Skorski [34] showed that being square-friendly is necessary to reduce entropy loss. Compared with the above work, our results for reducing entropy loss do not build on a specific class of primitives but need to rely on the bit security framework of [26], especially for the decision primitives.

In lattice-based cryptography, several probability metrics other than the statistical distance have been employed for improving the analysis of security proofs [28, 2, 25, 29, 36]. The metrics used in these work include the Kullback-Leibler divergence, the Rényi divergence, the max-log distance, and the relative error.

Micciancio and Walter [26] introduced a new framework of bit security that can smoothly connect the search primitives and the decision primitives quantitatively. A feature is that it allows the adversary to declare an attack failure. With their framework, we can say that a

λ -bit secure pseudorandom generator (a decision primitive) is also a λ -bit secure one-way function (a search primitive). In the conventional definition, a $\lambda/2$ -bit secure pseudorandom generator strangely yields a λ -bit secure one-way function. While they showed that the max-log distance is beneficial in their framework, we show that the Hellinger distance has the same effect and gives a tighter evaluation of closeness.

Distances/divergences between distributions other than the statistical distance have appeared in other cryptographic literature. Chung and Vadhan [9] gave a tight analysis of hashing block sources using the Hellinger distance as a key tool. Agrawal [1] introduced the notion of randomness extractors for the Kullback-Leibler divergence and gave explicit/non-explicit constructions with almost the same parameters as standard extractors. Steinberger [35] used the Hellinger distance for the improved analysis of key-alternating ciphers. Dai, Hoang, and Tessaro [10] used the chi-square divergence to analyze the information-theoretic indistinguishability proofs. Berman et al. [7] studied the polarization lemma for various distance notions such as the triangular discrimination and the Jensen-Shannon divergence to extend the region of polarization.

2 Preliminaries

We define the distances for distributions used in this work. The basic properties and general relationships of various distances/divergences can be found in [18]. We also present a useful lemma for the Hellinger distance, which will be used later.

Let P and Q be probability distributions over a finite set Ω . For a distribution P over Ω and $A \subseteq \Omega$, we denote by $P(A)$ the probability of event A , which is equal to $\sum_{x \in A} P(x)$. The *statistical distance* (a.k.a. *total variation distance*) between P and Q is

$$\text{SD}(P, Q) = \max_{A \subseteq \Omega} |P(A) - Q(A)|.$$

The *data processing inequality* guarantees that for any function $f: \Omega \rightarrow \{0, 1\}^*$, we have

$$\text{SD}(f(P), f(Q)) \leq \text{SD}(P, Q). \quad (1)$$

The *Hellinger distance* between P and Q is

$$\text{HD}(P, Q) = \sqrt{\frac{1}{2} \sum_{x \in \Omega} \left(\sqrt{P(x)} - \sqrt{Q(x)} \right)^2} = \sqrt{1 - \sum_{x \in \Omega} \sqrt{P(x) \cdot Q(x)}},$$

which takes values in $[0, 1]$. It holds that

$$\text{HD}(P, Q)^2 \leq \text{SD}(P, Q) \leq \sqrt{2} \cdot \text{HD}(P, Q). \quad (2)$$

The *Hellinger affinity* is defined as

$$\text{HA}(P, Q) = 1 - \text{HD}(P, Q)^2 = \sum_{x \in \Omega} \sqrt{P(x) \cdot Q(x)},$$

which is also known as the Bhattacharyya coefficient or fidelity.

The Hellinger distance has the following useful property, which is weaker than the *Pythagorean probability preservation* defined in [25, 26].

► **Lemma 1.** *Let $Q = (Q_1, \dots, Q_\ell)$ and $P = (P_1, \dots, P_\ell)$ be probability distribution ensembles over a finite support $\prod_i \Omega_i$. Then,*

$$\text{SD}(P, Q) \leq \sqrt{2\ell} \cdot \max_{a_i \in \prod_{j < i} \Omega_j} \text{HD}(P_i|a_i, Q_i|a_i).$$

Proof. Let $\epsilon = \max_{a_i \in \prod_{j < i} \Omega_j} \text{HD}(P_i|a_i, Q_i|a_i)$. Then, $\text{HA}(P_i|a_i, Q_i|a_i) = 1 - \text{HD}(P_i|a_i, Q_i|a_i)^2 \geq 1 - \epsilon^2$ for any i and $a_i \in \prod_{j < i} \Omega_j$. It holds that

$$\begin{aligned} \text{HA}(P, Q) &= \sum_{b_1, \dots, b_\ell \in \prod_i \Omega_i} \sqrt{P(b_1, \dots, b_\ell) \cdot Q(b_1, \dots, b_\ell)} \\ &= \sum_{b_1 \in \Omega_1} \sqrt{P_1(b_1) \cdot Q_1(b_1)} \cdot \left(\sum_{b_2 \in \Omega_2} \sqrt{P_2(b_2|P_1 = b_1) \cdot Q_2(b_2|Q_1 = b_1)} \cdot \left(\dots \right. \right. \\ &\quad \left. \left. \cdot \left(\sum_{b_\ell \in \Omega_\ell} \sqrt{P_\ell(b_\ell|P_{1,\ell-1} = (b_1, \dots, b_{\ell-1})) \cdot Q_\ell(b_\ell|Q_{1,\ell-1} = (b_1, \dots, b_{\ell-1}))} \right) \dots \right) \right) \\ &\geq (1 - \epsilon^2)^\ell \geq 1 - \ell\epsilon^2, \end{aligned}$$

where $P_{1,\ell-1} = (P_1, \dots, P_{\ell-1})$ and $Q_{1,\ell-1} = (Q_1, \dots, Q_{\ell-1})$. Thus, $\text{HD}(P, Q) = \sqrt{1 - \text{HA}(P, Q)} \leq \sqrt{\ell}\epsilon$. The statement follows from (2). \blacktriangleleft

3 Replacing Distributions in Security Games

We consider replacing probability distributions in security games. Let $Q = (Q_\theta)_\theta$ be an ideal distribution ensemble in a security game. We want to replace Q with an approximated distribution ensemble $P = (P_\theta)_\theta$ without compromising security. We define a general security game by following the definitions of [26, 25].

An n -bit security game G_A is a game played by an adversary A interacting with a challenger C . At the beginning of the game, the challenger chooses a uniformly random secret $x \in \{0, 1\}^n$, represented by the random variable X . At the end of the game, A outputs some value v , represented by the random variable V . The goal of the adversary is to output v such that $R(x, v) = 1$, where R is a Boolean function. The adversary may output a special symbol \perp such that $R(x, \perp) = 0$ for any x . During the game, A or C may obtain a sample from a distribution Q_θ by querying θ . The success probability of A is $\epsilon_A^Q = \Pr[R(X, V) = 1]$, where the probability is taken over the randomness of the entire security game, including the randomness of A . We may denote the game by G_A^Q since we intend to replace Q with another distribution ensemble.

Micciancio and Walter [26] defined the bit security based on an *advantage* that is different from most of the literature for the case $n = 1$. We use their framework for evaluating the security loss by replacing distributions in security games.

► **Definition 2** (Bit Security of [26]). *Let Π be a primitive for which an n -bit security game G_A^Q is defined. Let X and V be random variables representing a random secret $x \in \{0, 1\}^n$ and an output value v of A in G_A^Q , respectively. We define the output probability $\alpha_A = \Pr[V \neq \perp]$ and the conditional success probability $\beta_A = \Pr[R(X, V) = 1 \mid V \neq \perp]$. The advantage of A is defined to be*

$$\text{adv}_A = \begin{cases} \alpha_A \beta_A & n > 1 \\ \alpha_A (2\beta_A - 1)^2 & n = 1 \end{cases}.$$

The bit security of Π is defined to be

$$\min_A \log_2 \frac{T_A}{\text{adv}_A},$$

where T_A is the running time of A . We say the primitive is λ -bit secure if its bit security is at least λ .

17:6 Replacing Probability Distributions in Security Games via Hellinger Distance

We say Π is a *search* primitive if its n -bit security game G is defined for $n > 1$, and a *decision* primitive if G is a 1-bit security game.

For search primitives, it is not difficult to see that $Q = (Q_\theta)_\theta$ can be replaced with $P = (P_\theta)_\theta$ if their statistical distance between P_θ and Q_θ is sufficiently small and the number of queries is not so much. Specifically, if a search primitive Π^Q is λ -bit secure and $\text{SD}(P_\theta, Q_\theta) \leq 2^{-\lambda}$, then Π^P is $(\lambda - \log q)$ -bit secure, where we denote by Π^Q a primitive for which a security game G_A^Q is defined and q is the number of queries. This fact implies that it is sufficient to choose P that is close to Q within $2^{-\lambda}$ in the statistical distance for preserving the bit security.

Micciancio and Walter [25, 26] demonstrated that if distributions are close in the *max-log distance*, the closeness requirement may be relaxed. The max-log distance between distributions P and Q over Ω with the same support $S \subseteq \Omega$ is

$$\text{ML}(P, Q) = \max_{x \in S} |\ln P(x) - \ln Q(x)|.$$

They showed that the closeness of $2^{-\lambda/2}$ is sufficient to preserve the bit security for search primitives in [25] and decision primitives in [26].

► **Lemma 3** ([25, 26]). *Let $Q = (Q_i)_i$ and $P = (P_i)_i$ be distribution ensembles over the support $\prod_i \Omega_i$ satisfying $\text{ML}(P_i|a_i, Q_i|a_i) \leq 2^{-\lambda/2} \leq 1/4$ for any i and $a_i \in \prod_{j < i} \Omega_j$. If a search primitive Π^Q is λ -bit secure, then Π^P is $(\lambda - 3)$ -bit secure. If a decision primitive Π^Q is λ -bit secure, then Π^P is $(\lambda - 8)$ -bit secure.*

They showed the above results for a more general class of λ -efficient divergences [25, 26]. We demonstrate that similar effects can be obtained by using the Hellinger distance.

3.1 Security for Search Primitives

Let $Q = (Q_i)_i$ and $P = (P_i)_i$ be distribution ensembles over the same support $\prod_i \Omega_i$. We consider P and Q satisfying $\text{HD}(P_i|a_i, Q_i|a_i) \leq 2^{-\lambda/2}$ for any i and $a_i \in \prod_{j < i} \Omega_j$. We call such a pair (P, Q) a $2^{-\lambda/2}$ -Hellinger close pair. We show that this closeness is sufficient for preserving bit security.

► **Theorem 4.** *Let Π^Q be a primitive for which an n -bit security game G_A^Q is defined for $n > 1$. For any $2^{-\lambda/2}$ -Hellinger close pair (P, Q) , if Π^Q is λ -bit secure, then Π^P is $(\lambda - 2.973)$ -bit secure.*

Proof. Let ϵ_A^Q be the success probability of an adversary A in G_A^Q , and T_A the running time of A . Since Π is λ -bit secure, it holds that $\epsilon_A^Q/T_A \leq 2^{-\lambda}$ for any A . It is sufficient to show that $\epsilon_A^P/T_A < 2^{-(\lambda-2.973)}$, where ϵ_A^P is the success probability of A in G_A^P .

We consider ℓ independent plays of G_A^Q and define μ_ℓ^Q to be the probability that A succeeds in at least one out of ℓ plays of G_A^Q . Namely, $\mu_\ell^Q = 1 - (1 - \epsilon^Q)^\ell$. We define μ_ℓ^P analogously. Since the number of queries to the distribution ensemble is at most T_A in each play, it holds that

$$\left| \mu_\ell^P - \mu_\ell^Q \right| \leq \text{SD}(P^\ell, Q^\ell) \leq \sqrt{2\ell T_A} \cdot 2^{-\lambda/2},$$

where P^ℓ is the ℓ -fold product of P , the first inequality is by the data processing inequality, and the second inequality follows from Lemma 1. Thus,

$$(1 - \epsilon_A^Q)^\ell \leq \sqrt{2\ell T_A} \cdot 2^{-\lambda/2} + (1 - \epsilon_A^P)^\ell.$$

By the fact that $(1-x)^\ell \geq 1-\ell x$ for $x \in [0, 1]$ and setting $\ell = 1/\epsilon_A^P$, it holds that

$$1 - \frac{\epsilon_A^Q}{\epsilon_A^P} \leq \sqrt{\frac{2T_A \cdot 2^{-\lambda}}{\epsilon_A^P}} + (1 - \epsilon_A^P)^{1/\epsilon_A^P} < \sqrt{\frac{2T_A \cdot 2^{-\lambda}}{\epsilon_A^P}} + e^{-1},$$

where we use the relation that $(1-1/x)^x < e^{-1}$ for $x > 0$. By rewriting the inequality,

$$\left(\sqrt{\epsilon_A^P} - \frac{\sqrt{T_A \cdot 2^{-\lambda}}}{\sqrt{2(1-e^{-1})}} \right)^2 < \frac{\epsilon_A^Q}{1-e^{-1}} + \frac{T_A \cdot 2^{-\lambda}}{2(1-e^{-1})^2}.$$

It holds that

$$\sqrt{\epsilon_A^P} < \sqrt{\frac{\epsilon_A^Q}{1-e^{-1}} + \frac{T_A \cdot 2^{-\lambda}}{2(1-e^{-1})^2}} + \frac{\sqrt{T_A \cdot 2^{-\lambda}}}{\sqrt{2(1-e^{-1})}}.$$

Squaring both sides gives that

$$\frac{\epsilon_A^P}{T_A} < \frac{\epsilon_A^Q}{(1-e^{-1})T_A} + \frac{2^{-\lambda}}{(1-e^{-1})^2} + \frac{\sqrt{2} \cdot 2^{-\lambda/2}}{1-e^{-1}} \sqrt{\frac{\epsilon_A^Q}{(1-e^{-1})T_A} + \frac{2^{-\lambda}}{2(1-e^{-1})^2}}.$$

Since $\epsilon_A^Q/T_A \leq 2^{-\lambda}$, we have $\epsilon_A^P/T_A < 7.851 \cdot 2^{-\lambda} < 2^{2.973} \cdot 2^{-\lambda}$. Therefore, the statement follows. \blacktriangleleft

3.2 Security for Decision Primitives

Next, we show that the closeness of $2^{-\lambda/2}$ in the Hellinger distance is sufficient for preserving λ -bit security even for decision primitives.

► **Theorem 5.** *Let Π^Q be a primitive for which a 1-bit security game G_A^Q is defined. For any $2^{-\lambda/2}$ -Hellinger close pair (P, Q) , if Π^Q is λ -bit secure, then Π^P is $(\lambda - 6.847)$ -bit secure.*

Proof. Suppose for contradiction that Π^P is not $(\lambda - 6.667)$ -bit secure. Namely, there exists an adversary A for Π^P with running time T_A such that $\alpha_A^P(2\beta_A^P - 1)^2 > T_A/2^{\lambda-6.847}$, where α_A^Q and β_A^Q are the output probability and the conditional success probability of A . Since Π^Q is λ -bit secure, we have $\alpha_A^Q(2\beta_A^Q - 1)^2 \leq T_A/2^\lambda$, where α_A^Q and β_A^Q are the corresponding probabilities for Π^Q . Let $\alpha = \min\{\alpha_A^Q, \alpha_A^P\}$.

We define the games \tilde{G}_A^Q and \tilde{G}_A^P such that they are the same as G_A^Q and G_A^P with the difference that the adversary can restart the game with fresh randomness at any time. Consider the adversary B that runs A repeatedly until either the output value is different from \perp or B runs A in total $1/\alpha$ times, and outputs the same value as A does in the former and \perp in the latter. Let α_B^Q and β_B^Q be the output probability and the conditional success probability, respectively when playing \tilde{G}_B^Q . We also define α_B^P and β_B^P analogously. Then, it holds that $\beta_B^Q = \beta_A^Q$ and $\beta_B^P = \beta_A^P$. The running time of B satisfies $T_B \leq T_A/\alpha$. It follows from the data processing inequality and Lemma 1 that

$$\beta_B^P - \beta_B^Q \leq \sqrt{2T_B} \cdot 2^{-\lambda/2}.$$

Hence, we have

$$2\beta_B^P - 1 \leq 2\beta_B^Q - 1 + \sqrt{\frac{8T_B}{2^\lambda}}.$$

17:8 Replacing Probability Distributions in Security Games via Hellinger Distance

Since $\beta_B^Q = \beta_A^Q$, it holds that

$$2\beta_B^Q - 1 = 2\beta_A^Q - 1 \leq \sqrt{\frac{T_A}{\alpha 2^\lambda}}.$$

It follows from the above inequalities that

$$2\beta_A^P - 1 = 2\beta_B^P - 1 \leq \sqrt{\frac{8T_B}{2^\lambda}} + \sqrt{\frac{T_A}{\alpha 2^\lambda}} \leq (\sqrt{8} + 1) \sqrt{\frac{T_A}{\alpha 2^\lambda}}.$$

Then, we have

$$\frac{T_A}{\alpha (2\beta_A^P - 1)^2} \geq \frac{2^\lambda}{(\sqrt{8} + 1)^2} > 2^{\lambda - 3.874}.$$

If $\alpha = \alpha_A^P$, the above inequality implies that Π^P is $(\lambda - 3.874)$ -bit secure. Otherwise, we have

$$\alpha_A^Q = \alpha < \frac{T_A}{2^{\lambda - 3.874} (2\beta_A^P - 1)^2} < \frac{2^{\lambda - 6.847} \cdot \alpha_A^P}{2^{\lambda - 3.874}} = 2^{-2.973} \cdot \alpha_A^P,$$

where the last inequality follows from the assumption. In the proof of Theorem 4, if we consider the event that A outputs values other than \perp instead of the event that A succeeds, it implies that $\alpha_A^P < 2^{2.973} \cdot \alpha_A^Q$ for $2^{-\lambda/2}$ -Hellinger close pair (P, Q) . This contradicts the above inequality. Therefore, the statement follows. \blacktriangleleft

Limitation of the Statistical Distance

We show that a similar result to Theorem 5 does not hold for the statistical distance. Namely, the closeness of $2^{-\lambda/2}$ in the statistical distance is not sufficient for preserving security.

As a concrete example, we consider a modified one-time pad encryption scheme Π^Q . The probabilistic encryption function for messages over $\{0, 1\}^\lambda$ is defined to be

$$\text{Enc}_k(m) = \begin{cases} (1, m) & \text{with probability } 2^{-\lambda} \\ (0, m \oplus k) & \text{with probability } 1 - 2^{-\lambda} \end{cases},$$

where $k \in \{0, 1\}^\lambda$ is a key sampled according to a distribution Q . Here we assume that Q is the uniform distribution over $\{0, 1\}^\lambda$. Consider a distinguishing game in which, for a random secret $b \in \{0, 1\}$, an attacker tries to predict b given m_0, m_1 , and $\text{Enc}_k(m_b)$. The attacker can easily find the corresponding message if the first bit of the ciphertext is 1. Otherwise, the scheme is perfectly secure, and thus the attacker has no advantage in the distinguishing game. Let A be an attacker such that given m_0, m_1 , $\text{Enc}_k(m_b) = (c_1, c_2)$, where $c_1 \in \{0, 1\}, c_2 \in \{0, 1\}^\lambda$, A outputs b such that $c_2 = m_b$ if $c_1 = 1$, and \perp otherwise. Then,

$$\frac{T_A}{\alpha_A^Q (2\beta_A^Q - 1)^2} \geq \frac{T_A}{2^{-\lambda}} \geq 2^\lambda.$$

Since other adversaries cannot achieve a higher advantage than $2^{-\lambda}$, Π^Q has λ -bit security.

Let P be a distribution over $\{0, 1\}^\lambda$ such that

$$P(x) = \begin{cases} 2^{-\lambda} + 2^{-\lambda/2} & x = 0^\lambda \\ 0 & x \in S \\ 2^{-\lambda} & \text{otherwise} \end{cases},$$

where $|S| = 2^{\lambda/2}$. One may consider S a set of strings starting with $1^{\lambda/2}$. It holds that $\text{SD}(P, Q) = 2^{-\lambda/2}$. Consider an adversary A' such that when $c_1 = 1$, A' outputs b satisfying $c_2 = m_b$. When $c_1 = 0$, A' outputs b such that $c_2 = m_b$ if $c_2 \in \{m_0, m_1\}$, and \perp if $c_2 \notin \{m_0, m_1\}$. For this adversary A' , it is not difficult to see that $\alpha_{A'}^P = 2^{-\lambda} + (1 - 2^{-\lambda})(2^{-\lambda} + 2^{-\lambda/2}) \geq 2^{-\lambda/2}$ and $\beta_{A'}^P = 1$. Thus, the bit security of Π^P is at most $\lambda/2$. This indicates that the closeness of $2^{-\lambda/2}$ in the statistical distance may reduce the bit security by half.

4 Relations between Max-Log Distance and Hellinger Distance

We show that the Hellinger distance is bounded above by the max-log distance when the max-log distance is less than $\sqrt{2} - 1$. Namely, the Hellinger distance gives a tighter evaluation of closeness when the distance is small.

► **Proposition 6.** *Let P and Q be distributions over Ω with the same support $S \subseteq \Omega$. Then, $\text{HD}(P, Q) \leq \text{ML}(P, Q)$ as long as $\text{ML}(P, Q) \leq \sqrt{2} - 1$.*

Proof. It follows from the relation between the Hellinger distance and the chi-square divergence (cf. [18]) that

$$\text{HD}(P, Q) \leq \sqrt{\frac{1}{2} \sum_{x \in S} \frac{(P(x) - Q(x))^2}{Q(x)}},$$

where $S \subseteq \Omega$ is the support of P and Q . Then,

$$\begin{aligned} \text{HD}(P, Q) &\leq \sqrt{\frac{1}{2} \sum_{x \in S} Q(x) \left(\frac{P(x)}{Q(x)} - 1 \right)^2} \\ &\leq \sqrt{\frac{1}{2} \sum_{x \in S} Q(x) \cdot \max_{x \in S} \left| \frac{P(x)}{Q(x)} - 1 \right|^2} \\ &= \frac{1}{\sqrt{2}} \max_{x \in S} \left| \frac{P(x)}{Q(x)} - 1 \right|. \end{aligned}$$

Let $\text{ML}(P, Q) = \epsilon$. By definition, for any $x \in S$,

$$e^{-\epsilon} \leq \frac{P(x)}{Q(x)} \leq e^{\epsilon}.$$

Since we have the relations $e^y - 1 \leq y + y^2$ and $1 - e^{-y} \leq y + y^2$ for $y \in [0, \sqrt{2}]$, it holds that

$$\text{HD}(P, Q) \leq \frac{1}{\sqrt{2}}(\epsilon + \epsilon^2) \leq \epsilon,$$

where the last inequality holds for $0 \leq \epsilon \leq \sqrt{2} - 1$. ◀

Next, we give a concrete example of distributions for which an exponential gap exists. We show that, for a uniform distribution Q over $\{0, 1\}^n$, there is a distribution P such that $\text{ML}(P, Q) = 0.6$ and $\text{HD}(P, Q) \leq 0.6 \cdot 2^{-n/2}$.

► **Proposition 7.** *Let Q be the uniform distribution over Ω with $|\Omega| \geq 4$. There is a distribution P over Ω such that*

$$\text{ML}(P, Q) = \epsilon \quad \text{and} \quad \text{HD}(P, Q) \leq \sqrt{\frac{3(\epsilon + \epsilon^2)}{8|\Omega|}}$$

for any $\epsilon \in [0, 0.618]$.

17:10 Replacing Probability Distributions in Security Games via Hellinger Distance

Proof. Let $M = |\Omega|$. We define P such that

$$P(x) = \begin{cases} M^{-1} \cdot e^\epsilon & x = x_0 \\ M^{-1} \cdot e^{-\epsilon} & x = x_1 \\ (M-2)^{-1} \cdot (1 - M^{-1}(e^\epsilon + e^{-\epsilon})) & x \notin \{x_0, x_1\} \end{cases}.$$

First we show that $\text{ML}(P, Q) = \epsilon$. It is clear from the definition that $|\ln P(x) - \ln Q(x)| = \epsilon$ for $x \in \{x_0, x_1\}$. For $x \notin \{x_0, x_1\}$, we need to show that

$$M^{-1} \cdot e^{-\epsilon} \leq (M-2)^{-1} \cdot (1 - M^{-1}(e^\epsilon + e^{-\epsilon})) \leq M^{-1} \cdot e^\epsilon,$$

which can be rewritten as

$$M \geq \max \left\{ \frac{e^\epsilon - e^{-\epsilon}}{1 - e^{-\epsilon}}, \frac{e^\epsilon - e^{-\epsilon}}{e^\epsilon - 1} \right\}.$$

Since the right-hand side is at most 4 for $\epsilon \geq 0$, we have $\text{ML}(P, Q) = \epsilon$.

Next, we give an upper bound on $\text{HD}(P, Q)$. Recall that $\text{HD}(P, Q) = \sqrt{1 - \text{HA}(P, Q)}$ and $\text{HA}(P, Q) = \sum_{x \in \Omega} \sqrt{P(x) \cdot Q(x)}$. For $x \notin \{x_0, x_1\}$,

$$P(x) = \frac{1}{M-2} \left(1 - \frac{1}{M}(e^\epsilon + e^{-\epsilon}) \right) \geq \frac{1}{M-2} \left(1 - \frac{1}{M}(2 + \epsilon + \epsilon^2) \right) = \frac{1}{M} \left(1 - \frac{\epsilon + \epsilon^2}{M-2} \right),$$

where the inequality follows from the fact that $e^x + e^{-x} \leq 2 + x + x^2$ for $0 \leq x \leq 1$. By using the relation that $e^x + e^{-x} \geq 2$ for $0 \leq x \leq 1$, we have

$$\begin{aligned} \text{HA}(P, Q) &= \sqrt{P(x_0)Q(x_0)} + \sqrt{P(x_1)Q(x_1)} + \sum_{x \in \Omega \setminus \{x_0, x_1\}} \sqrt{P(x)Q(x)} \\ &\geq \frac{2}{M} + \frac{M-2}{M} \cdot \sqrt{1 - \frac{\epsilon + \epsilon^2}{M-2}}. \end{aligned}$$

Thus,

$$\begin{aligned} \text{HD}(P, Q)^2 &\leq 1 - \frac{2}{M} - \frac{M-2}{M} \cdot \sqrt{1 - \frac{\epsilon + \epsilon^2}{M-2}} \\ &\leq 1 - \frac{2}{M} - \frac{M-2}{M} \left(1 - \frac{\epsilon + \epsilon^2}{2(M-2)} - \frac{1}{2} \left(\frac{\epsilon + \epsilon^2}{M-2} \right)^2 \right) \\ &= \frac{\epsilon + \epsilon^2}{2M} \left(1 + \frac{\epsilon + \epsilon^2}{2(M-2)} \right) \leq \frac{3(\epsilon + \epsilon^2)}{8M}, \end{aligned}$$

where the second and the last inequalities follow from $\sqrt{1-x} \geq 1 - x/2 - x^2/2$ for $0 \leq x \leq 1$ and $(\epsilon + \epsilon^2)/(2(M-2)) \leq 1/4$ for $\epsilon \in [0, 0.618]$ and $M \geq 4$, respectively. Hence, the statement follows. \blacktriangleleft

5 Randomness Extraction for Hellinger Distance

We focus on the problem of randomness extraction from entropy sources. The *min-entropy* of random variable X over $\{0, 1\}^n$ is $H_{\min}(X) = \min_{x \in \{0, 1\}^n} \log_2(1/\Pr[X = x])$. *Randomness extractors* are usually defined as a seeded function that maps any entropy source to a distribution that is close to the uniform distribution in the statistical distance. For $n \in \mathbb{N}$, we denote by U_n the uniform distribution over $\{0, 1\}^n$.

► **Definition 8** (Randomness Extractor). *A function $\text{Ext}: \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ is said to be a (k, ϵ) -(strong) extractor if for every distribution X over $\{0, 1\}^n$ of $H_{\min}(X) \geq k$, it holds that $\text{SD}((\text{Ext}(X, U_d), U_d), U_{m+d}) \leq \epsilon$, where X and U_d are independent.*

For (strong) extractors, the input entropy is $k + d$, and the output length is $m + d$. The difference $(k + d) - (m + d) = k - m$ is called the *entropy loss* of extractors. The entropy loss is unavoidable. Radhakrishnan and Ta-Shma [30] showed that it must be at least $2 \log(1/\epsilon) - O(1)$.

It is known that a *universal family of hash functions* gives an extractor with optimal entropy loss. A random hash function $H: \{0, 1\}^n \rightarrow \{0, 1\}^m$ from a family \mathcal{H} of hash functions is called *universal* if for any distinct $x, x' \in \{0, 1\}^n$, $\Pr[H(x) = H(x')] \leq 2^{-m}$. Specifically, let $|\mathcal{H}| = 2^d$ and $m = k - 2 \log(1/\epsilon)$. Then, extractor Ext defined by $\text{Ext}(x, H) = H(x)$ is a $(k, \epsilon/2)$ -strong extractor. This result is known as the *leftover hash lemma* [6, 19]. The main technical lemma is a bound on the *collision probability*. For a random variable X , the collision probability of X is

$$\text{cp}(X) = \Pr[X = X'] = \sum_x \Pr[X = x]^2,$$

where X' is an independent copy of X .

► **Lemma 9** (The Leftover Hash Lemma [6, 19]). *Let X be a random variable over $\{0, 1\}^n$ with $H_{\min}(X) \geq k$. Let $H: \{0, 1\}^n \rightarrow \{0, 1\}^m$ be a random hash function from a universal family \mathcal{H} . Then, $\text{cp}(H(X), H) \leq 2^{-d} \cdot (2^{-m} + 2^{-k})$.*

We define a notion of extractors for which the output distribution is close to uniform in the Hellinger distance.

► **Definition 10** (Hellinger extractor). *A function $\text{Ext}: \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ is said to be a (k, ϵ) -(strong) Hellinger extractor if for every distribution X over $\{0, 1\}^n$ of $H_{\min}(X) \geq k$, it holds that $\text{HD}((\text{Ext}(X, U_d), U_d), U_{m+d}) \leq \epsilon$, where X and U_d are independent.*

It follows from (2) that if Ext is a (k, ϵ) -Hellinger extractor, then it is also a $(k, \sqrt{2}\epsilon)$ -extractor.

We use the following useful lemma of Chung and Vadhan [9] for proving a leftover hash lemma for the Hellinger distance.

► **Lemma 11** ([9, Lemma 3.12]). *Let X be a random variable over $\{0, 1\}^n$. If $\text{cp}(X) \leq \alpha/2^n$, then $\text{HA}(X, U_n) \geq \sqrt{1/\alpha}$.*

We show that a universal family of hash functions gives a Hellinger extractor with optimal entropy loss.

► **Theorem 12** (Leftover Hash Lemma for Hellinger). *Let $H: \{0, 1\}^n \rightarrow \{0, 1\}^m$ be a random hash function from a universal family \mathcal{H} with $|\mathcal{H}| = 2^d$, $m = k + 1 - 2 \log(1/\epsilon)$. Then, function $\text{Ext}: \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ defined by $\text{Ext}(x, H) = H(x)$ is a (k, ϵ) -Hellinger extractor.*

Proof. Let X be a random variable over $\{0, 1\}^n$ with $H_{\min}(X) \geq k$. It follows from Lemma 9 that

$$\text{cp}(H(X), H) \leq 2^{-d} \cdot (2^{-m} + 2^{-k}) = \frac{\alpha}{2^{m+d}},$$

17:12 Replacing Probability Distributions in Security Games via Hellinger Distance

where $\alpha = 1 + 2^{m-k}$. By Lemma 11, we have that $\text{HA}((H(X), H), U_{m+d}) \geq \alpha^{-1/2}$. Then, it holds that

$$\begin{aligned} \text{HD}((H(X), H), U_{m+d}) &= \sqrt{1 - \text{HA}((H(X), H), U_{m+d})^2} \\ &\leq \sqrt{1 - \alpha^{-1/2}} = \sqrt{1 - (1 + 2^{m-k})^{-1/2}} \\ &\leq \sqrt{1 - (1 - 2^{m-k-1})} = \sqrt{2^{m-k-1}} = \epsilon, \end{aligned}$$

where the last inequality follows from the fact that $(1+x)^{-1/2} \geq 1-x/2$ for $x \geq 0$. Hence, the statement follows. \blacktriangleleft

Since we have the relation that $\text{SD}(P, Q) \leq \sqrt{2} \cdot \text{HD}(P, Q)$, the lower bound of [30] implies that the entropy loss of Theorem 12 is also optimal.

Entropy Loss of Randomness Extractors in Security Games

We consider the situations in which a uniform distribution is employed in security games, and we would like to replace it with an output of randomness extractors. Let Π be a primitive with an n -bit security game G_A^Q such that the uniform distribution $Q = U_m$ is employed. Suppose that Π has λ -bit security.

Theorems 4 and 5 imply that for preserving the bit security when replacing Q with P , it is enough to hold $\text{HD}(P, Q) \leq 2^{-\lambda/2}$. Regarding the statistical distance, the closeness of $2^{-\lambda}$ is sufficient for preserving security.

A universal family of hash functions can achieve the security of extractors for both distances. When using the statistical distance, the entropy loss for achieving $\text{SD}(P, Q) \leq 2^{-\lambda}$ is $k - m = 2(\lambda - 1)$. By Theorem 12, the entropy loss for $\text{HD}(P, Q) \leq 2^{-\lambda/2}$ is $k - m = \lambda - 1$. Thus, by analyzing security games via the Hellinger distance, the entropy loss for preserving λ -bit security can be reduced by half.

References

- 1 Rohit Agrawal. Samplers and extractors for unbounded functions. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA.*, volume 145 of *LIPICs*, pages 59:1–59:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.APPROX-RANDOM.2019.59.
- 2 Shi Bai, Tancrede Lepoint, Adeline Roux-Langlois, Amin Sakzad, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the rényi divergence rather than the statistical distance. *J. Cryptology*, 31(2):610–640, 2018. doi:10.1007/s00145-017-9265-9.
- 3 Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, François-Xavier Standaert, and Yu Yu. Leftover hash lemma, revisited. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011. doi:10.1007/978-3-642-22792-9_1.
- 4 Boaz Barak and Shai Halevi. A model and architecture for pseudo-random generation with applications to /dev/random. In Vijay Atluri, Catherine A. Meadows, and Ari Juels, editors, *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11, 2005*, pages 203–212. ACM, 2005. doi:10.1145/1102120.1102148.

- 5 Boaz Barak, Ronen Shaltiel, and Eran Tromer. True random number generators secure in a changing environment. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 2003. doi:10.1007/978-3-540-45238-6_14.
- 6 Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. How to reduce your enemy's information (extended abstract). In Hugh C. Williams, editor, *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 468–476. Springer, 1985. doi:10.1007/3-540-39799-X_37.
- 7 Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. Statistical difference beyond the polarizing regime. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:38, 2019. URL: <https://ecc.ecc.weizmann.ac.il/report/2019/038>.
- 8 Xavier Boyen, Yevgeniy Dodis, Jonathan Katz, Rafail Ostrovsky, and Adam D. Smith. Secure remote authentication using biometric data. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 147–163. Springer, 2005. doi:10.1007/11426639_9.
- 9 Kai-Min Chung and Salil P. Vadhan. Tight bounds for hashing block sources. In Ashish Goel, Klaus Jansen, José D. P. Rolim, and Ronitt Rubinfeld, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*, volume 5171 of *Lecture Notes in Computer Science*, pages 357–370. Springer, 2008. doi:10.1007/978-3-540-85363-3_29.
- 10 Wei Dai, Viet Tung Hoang, and Stefano Tessaro. Information-theoretic indistinguishability via the chi-squared method. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 497–523. Springer, 2017. doi:10.1007/978-3-319-63697-9_17.
- 11 Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008. doi:10.1137/060651380.
- 12 Yevgeniy Dodis, Krzysztof Pietrzak, and Daniel Wichs. Key derivation without entropy waste. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 93–110. Springer, 2014. doi:10.1007/978-3-642-55220-5_6.
- 13 Yevgeniy Dodis and Yu Yu. Overcoming weak expectations. In Amit Sahai, editor, *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, volume 7785 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2013. doi:10.1007/978-3-642-36594-2_1.
- 14 Cynthia Dwork. Differential privacy: A survey of results. In Manindra Agrawal, Ding-Zhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation, 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings*, volume 4978 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2008. doi:10.1007/978-3-540-79228-4_1.
- 15 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006. doi:10.1007/11681878_14.

17:14 Replacing Probability Distributions in Security Games via Hellinger Distance

- 16 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014. doi:10.1561/04000000042.
- 17 Rosario Gennaro, Hugo Krawczyk, and Tal Rabin. Secure hashed diffie-hellman over non-ddh groups. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 361–381. Springer, 2004. doi:10.1007/978-3-540-24676-3_22.
- 18 Alison L. Gibbs and Francis Edward Su. On choosing and bounding probability metrics. *INTERNAT. STATIST. REV.*, pages 419–435, 2002.
- 19 Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 12–24. ACM, 1989. doi:10.1145/73007.73009.
- 20 Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 631–648. Springer, 2010. doi:10.1007/978-3-642-14623-7_34.
- 21 Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010. doi:10.1007/978-3-642-13190-5_1.
- 22 Takahiro Matsuda, Kenta Takahashi, Takao Murakami, and Goichiro Hanaoka. Improved security evaluation techniques for imperfect randomness from arbitrary distributions. In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography - PKC 2019 - 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part I*, volume 11442 of *Lecture Notes in Computer Science*, pages 549–580. Springer, 2019. doi:10.1007/978-3-030-17253-4_19.
- 23 Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2013. doi:10.1007/978-3-642-40041-4_2.
- 24 Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. doi:10.1137/S0097539705447360.
- 25 Daniele Micciancio and Michael Walter. Gaussian sampling over the integers: Efficient, generic, constant-time. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 455–485. Springer, 2017. doi:10.1007/978-3-319-63715-0_16.
- 26 Daniele Micciancio and Michael Walter. On the bit security of cryptographic primitives. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 3–28. Springer, 2018. doi:10.1007/978-3-319-78381-9_1.
- 27 Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342. ACM, 2009. doi:10.1145/1536414.1536461.
- 28 Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic*

- Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 353–370. Springer, 2014. doi:10.1007/978-3-662-44709-3_20.
- 29 Thomas Prest. Sharper bounds in lattice-based cryptography using the rényi divergence. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 347–374. Springer, 2017. doi:10.1007/978-3-319-70694-8_13.
- 30 Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM J. Discrete Math.*, 13(1):2–24, 2000. doi:10.1137/S0895480197329508.
- 31 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005. doi:10.1145/1060590.1060603.
- 32 Oded Regev. The learning with errors problem (invited survey). In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, USA, June 9-12, 2010*, pages 191–204. IEEE Computer Society, 2010. doi:10.1109/CCC.2010.26.
- 33 Ronen Shaltiel. An introduction to randomness extractors. In Luca Aceto, Monika Henzinger, and Jirí Sgall, editors, *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, volume 6756 of *Lecture Notes in Computer Science*, pages 21–41. Springer, 2011. doi:10.1007/978-3-642-22012-8_2.
- 34 Maciej Skorski. Lower bounds on key derivation for square-friendly applications. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPICs*, pages 57:1–57:12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPICs.STACS.2017.57.
- 35 John P. Steinberger. Improved security bounds for key-alternating ciphers via hellinger distance. *IACR Cryptology ePrint Archive*, 2012:481, 2012. URL: <http://eprint.iacr.org/2012/481>.
- 36 Michael Walter. Sampling the integers with low relative error. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje-eddine Rachidi, editors, *Progress in Cryptology - AFRICACRYPT 2019 - 11th International Conference on Cryptology in Africa, Rabat, Morocco, July 9-11, 2019, Proceedings*, volume 11627 of *Lecture Notes in Computer Science*, pages 157–180. Springer, 2019. doi:10.1007/978-3-030-23696-0_9.

Differentially Private Approximations of a Convex Hull in Low Dimensions

Yue Gao 

Department of Computing Science, University of Alberta, Edmonton, Canada

Or Sheffet  

Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel

Abstract

We give the first differentially private algorithms that estimate a variety of geometric features of points in the Euclidean space, such as diameter, width, volume of convex hull, min-bounding box, min-enclosing ball, etc. Our work relies heavily on the notion of *Tukey-depth*. Instead of (non-privately) approximating the convex-hull of the given set of points P , our algorithms approximate the geometric features of $D_P(\kappa)$ – the κ -Tukey region induced by P (all points of Tukey-depth κ or greater). Moreover, our approximations are all bi-criteria: for any geometric feature μ our (α, Δ) -approximation is a value “sandwiched” between $(1 - \alpha)\mu(D_P(\kappa))$ and $(1 + \alpha)\mu(D_P(\kappa - \Delta))$.

Our work is aimed at producing a (α, Δ) -kernel of $D_P(\kappa)$, namely a set \mathcal{S} such that (after a shift) it holds that $(1 - \alpha)D_P(\kappa) \subset \text{CH}(\mathcal{S}) \subset (1 + \alpha)D_P(\kappa - \Delta)$. We show that an analogous notion of a bi-criteria approximation of a directional kernel, as originally proposed by [1], *fails* to give a kernel, and so we result to subtler notions of approximations of projections that do yield a kernel. First, we give differentially private algorithms that find (α, Δ) -kernels for a “fat” Tukey-region. Then, based on a private approximation of the min-bounding box, we find a transformation that does turn $D_P(\kappa)$ into a “fat” region *but only if* its volume is proportional to the volume of $D_P(\kappa - \Delta)$. Lastly, we give a novel private algorithm that finds a depth parameter κ for which the volume of $D_P(\kappa)$ is comparable to the volume of $D_P(\kappa - \Delta)$. We hope our work leads to the further study of the intersection of differential privacy and computational geometry.

2012 ACM Subject Classification Theory of computation \rightarrow Theory of database privacy and security

Keywords and phrases Differential Privacy, Computational Geometry, Tukey Depth

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.18

Related Version *Full Version:* <http://arxiv.org/abs/2007.08110>

Funding Much of this work was done when the first author was advised by the second author, and was supported by grant #2017–06701 of the Natural Sciences and Engineering Research Council of Canada (NSERC). In addition, this work was partially done when O.S. was a participant of the Simons’ Institute for the Theory of Computing program of Data-Privacy. O.S. is supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister’s Office, and by ISF grant no. 2559/20.

Acknowledgements Both authors thank the anonymous reviewers for many helpful suggestions in improving this paper.

1 Introduction

With modern day abundance of data, there are numerous datasets that hold the sensitive and personal details of individuals, yet collect only a few features per user. Examples of such *low-dimensional* datasets include locations (represented as points on the $2D$ -plane), medical data composed of only a few measurements (e.g., [25, 27]), or high-dimensional data restricted to a small subset of features. It is therefore up to us to make sure that the analyses of such sensitive datasets do not harm the privacy of their participants. Differentially private algorithms [10, 12] alleviate such privacy concerns as they guarantee that the presence or absence of any single individual in the dataset has only a limited affect on any outcome.



© Yue Gao and Or Sheffet;

licensed under Creative Commons License CC-BY 4.0

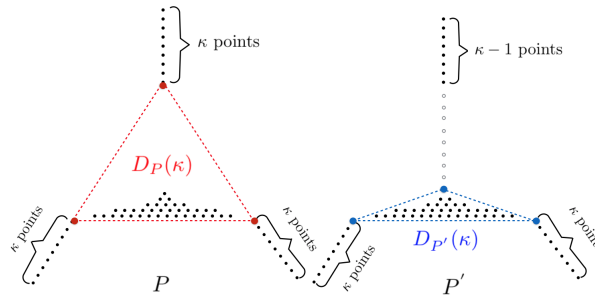
2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 18; pp. 18:1–18:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** An example showing that $D_P(\kappa)$'s volume, width and min-enclosing triangle can be greatly affected by a single point in the input.

Often (usually motivated by data-visualization), understanding the geometric features of such low-dimensional datasets is a key step in their analysis. Yet, to this day, very little work has been done to establish differentially private algorithms that approximate the data's geometrical features. This should not come as a surprise seeing as most geometric features – such as diameter, width,¹ volume of convex-hull, min-bounding ball radius, etc. – are highly sensitive to the presence / absence of a single datum. Moreover, while it is known that differential privacy generalizes [11, 2], geometrical properties often do not: if the dataset P is composed on n i.i.d. draws from a distribution \mathcal{P} then it might still be likely that, say, $\text{diam}(P)$ and $\text{diam}(\mathcal{P})$ are quite different.²

But differential privacy has already overcome the difficulty of large sensitivity in many cases, the leading example being the median – despite the fact that the median may vary greatly by the presence/absence of a single datum, we are still capable of privately approximating the median. The crux in differentially private median approximations [21, 4] is that the quality of the approximation is not measured by the actual distance between the true input-median and the result of the algorithm, but rather by the probability mass of the input's CDF “sandwiched” between the true median and the output of the private algorithm. A similar effect takes place in our work. While we deal with geometric concepts that exhibit large sensitivity, we formulate *robust* approximation guarantees of these concepts, guarantees that do generalize when the data is drawn i.i.d. from some unknown distribution. Yet unlike the private median approximations, our private kernel-approximation algorithm does not *always* return an output. It first verifies that certain niceness assumptions about the input hold; if they don't hold, it is capable of finding a sufficiently “deep” portion of the input which can be privately approximated. Details to follow.

Much like in previous works in differential privacy [3, 19], our approximation rely heavily on the notion of the *depth* of a point. Specifically, our approximation guarantees are with respect to *Tukey depth* [26]. Roughly speaking (see Section 2), a point x has Tukey depth κ w.r.t. a dataset P , denoted $\text{TD}(x, P) = \kappa$, if the smallest set $S \subset P$ one needs to remove from P so that some hyperplane separates x from $P \setminus S$ has cardinality κ . This also allows us to define the κ -*Tukey region* $D_P(\kappa) = \{x \in \mathbb{R}^d : \text{TD}(x, P) \geq \kappa\}$. So, for example, $D_P(0) = \mathbb{R}^d$ and $D_P(1) = \text{CH}(P)$ (the convex-hull of P). It follows from the definition that for any $1 \leq \kappa_1 \leq \kappa_2$ we have $\text{CH}(P) = D_P(1) \supset D_P(\kappa_1) \supset D_P(\kappa_2)$. It is known that for any dataset

¹ The min gap between two hyperplanes that “sandwich” the data.

² For example, consider \mathcal{P} as a uniform distribution over $2n$ discrete points whose diameter greatly shrinks unless two specific points are drawn into P .

P and depth κ the Tukey-region $D_P(\kappa)$ is a convex polytope, and moreover (see [14]) that for any P of size n it holds that $D_P(n/(d+1)) \neq \emptyset$. Moreover, there exists efficient algorithms (in low-dimensions) that find $D_P(\kappa)$.

One pivotal property of the Tukey depth, which enables differentially private approximations, is that it exhibits low-sensitivity at any given point. As noted by [3], it follows from the very definition of Tukey-depth that if we add to / remove from P any single datum, then the depth of any given $x \in \mathbb{R}^d$ changes by no more than 1. And so, in this work, we give bi-criteria approximations of key geometric features of $D_P(\kappa)$ – where the quality of the approximation is measured both by a multiplicative factor and with respect to a *shallower* Tukey region. Given a measure μ of the convex polytope $D_P(\kappa)$, such as diameter, width, volume etc., we return a (α, Δ) -approximation of μ – a value lower bounded by $(1-\alpha)\mu(D_P(\kappa))$ and upper-bounded by $(1+\alpha)\mu(D_P(\kappa-\Delta))$. This implies that the quality of the approximation depends on *both* the approximation parameters fed into the algorithm and also on the “niceness” properties of the data. For datasets where $\mu(D_P(\kappa-\Delta)) \approx \mu(D_P(\kappa))$, our (α, Δ) -approximation is a good approximation of $\mu(D_P(\kappa))$; but for datasets where $\mu(D_P(\kappa-\Delta)) \gg \mu(D_P(\kappa))$ our guarantee is rather weak. Note that no differentially private algorithm can correctly report for all P whether $\mu(D_P(\kappa))$ and $\mu(D_P(\kappa-\Delta))$ are / are-not similar seeing as, as Figure 1 shows, such proximity can be highly affected by the existence of a single datum in P . Again, this is very much in line with private approximations of the median [21, 4].³

Our main goal in this work is to produce an *kernel* for $D_P(\kappa)$. Non privately, a α -kernel [1] of a dataset P is a set $\mathcal{S} \subset P$ where for any direction u it holds that $(1-\alpha) \max_{p,q \in P} \langle p-q, u \rangle \leq \max_{p,q \in \mathcal{S}} \langle p-q, u \rangle \leq \max_{p,q \in P} \langle p-q, u \rangle$. Agarwal et al. [1] showed that for any P there exists such a kernel whose size is $(1/\alpha)^{O(d)}$. (We thus assume $|P| \gg (1/\alpha)^{O(d)}$ for otherwise the non-private algorithm can trivially output P itself.) More importantly, the fact that \mathcal{S} is a α -kernel implies that $(1-O(\alpha))\text{CH}(P) \subset \text{CH}(\mathcal{S}) \subset \text{CH}(P)$. It is thus tempting to define an analogous notion of (α, Δ) -kernel as “for any direction u we have $(1-\alpha) \max_{p,q \in D_P(\kappa)} \langle p-q, u \rangle \leq \max_{p,q \in \mathcal{S}} \langle p-q, u \rangle \leq (1+\alpha) \max_{p,q \in D_P(\kappa-\Delta)} \langle p-q, u \rangle$ ” and hope that it yields that $(1-O(\alpha))D_P(\kappa) \subset \text{CH}(\mathcal{S}) \subset D_P(\kappa-\Delta)$. Alas, that is *not* the case. Having $\mathcal{S} \subset D_P(\kappa)$ turns out to be a crucial component in arguing about the containment of the convex-hulls, and the argument breaks without it. We give a counter example in a later discussion (in Section 3). Therefore, viewing this directional-width approximation property as means to an end, we define the notion of (α, Δ) -kernel directly w.r.t. the containment of the convex bodies.

► **Definition 1.** *Given a dataset P and a parameter κ , a set \mathcal{S} is called a (α, Δ) -kernel for $D_P(\kappa)$ if there exist two points c_1, c_2 such that $(1-\alpha)(D_P(\kappa) - c_1) \subset \text{CH}(\mathcal{S}) - c_1$ and $\text{CH}(\mathcal{S}) - c_2 \subset (1+\alpha)(D_P(\kappa-\Delta) - c_2)$.*

Non privately, the “center” points c_1 and c_2 may just as well be the origin, since we can shift the points so that the origin is in the convex-hull; but privately we cannot make such an assumption as it differentiates between two neighboring datasets. Note that in particular, a (α, Δ) -kernel gives the (α, Δ) -approximation of the projection along every direction u proposed earlier (in quotation-marks above). In fact, a (α, Δ) -kernel yields

³ In particular, in the case where P is drawn from a distribution \mathcal{P} , it is known that $\forall x \in \mathbb{R}^d$, $|\frac{1}{n}\text{TD}(x, P) - \text{TD}(x, \mathcal{P})| = O(\sqrt{\frac{d \log(n)}{n}})$ [8], where $\text{TD}(x, \mathcal{P})$ denotes the smallest measure \mathcal{P} places on any halfspace containing x . Thus, if $D_P(\kappa)$ and $D_P(\kappa-\Delta)$ vary drastically, then it follows that the distribution \mathcal{P} is “volatile” at depth $\frac{\kappa}{n}$.

(α, Δ) -approximations of numerous properties of $D_P(\kappa)$, like volume, min-bounding box, min-enclosing / max-enclosed ball radius, surface area, etc. Our work is the first to give a private approximation of *any* of these concepts.

The main caveat of our work is that we are able to output a (α, Δ) -kernel of $D_P(\kappa)$ only when $D_P(\kappa)$ satisfies some “niceness” properties. We briefly describe the structure of our work to better explain these properties and how they relate. We begin with preliminaries in Section 2. In Section 3 we give our algorithm for finding a kernel, which works under the premise that the width of $D_P(\kappa)$ is large. This means that our goal is complete if we are able to assert, using a private algorithm, that $D_P(\kappa)$ has large width (we design a heuristics for this purpose, but it is deferred to the full version of this work); or if we can find a value of κ for which $D_P(\kappa)$ can be privately transformed into a region with large width – a complicated task for which we require multiple “stepping stones” that are detailed in the following sections.

In Sections 4 and 5 we establish some basic privacy-preserving algorithms for tasks we require later. In Section 6, we give a private $(O(1), \Delta)$ -approximation of the min-bounding box of $D_P(\kappa)$; and show that this box yields a transformation that turns $D_P(\kappa)$ into a region of large width, *but only if* the volumes of $D_P(\kappa)$ and $D_P(\kappa - \Delta)$ are comparable. So finally, in Section 7, we give an algorithm that finds a value of κ for which this premise about the volumes of $D_P(\kappa)$ and $D_P(\kappa - \Delta)$ holds, rendering us capable of privately finding a (α, Δ) -kernel for this particular $D_P(\kappa)$.

Providing further details about the private approximation algorithms we introduce in this work requires that we first delve into some background details and introduce some parameters.

The Setting: Low-Dimension and Small Granularity

Differential privacy deals with the trade-offs between the privacy parameters, ϵ and δ , and an algorithm’s utility guarantee. Unlike the majority of works in differential privacy, we don’t express these trade-offs based on the size n of the data.⁴ Instead, in our work we upper bound the Δ -term of a private (α, Δ) -approximation as a function of the privacy- and accuracy-parameters, as well as additional two parameters. These two parameters are (i) the dimension, d , which we assume to be constant and so $n^{\text{poly}(d)}$ is still considered efficient for our needs; and (ii) the granularity of the grid on which the data resides. In differential privacy, it is impossible to provide useful algorithms for certain basic tasks [6] when the universe of possible entries is infinite. Therefore, we assume that the given input P lies inside the hypercube $[0, 1]^d$ and moreover – that its points reside on a grid \mathcal{G}^d whose granularity is denoted as Υ . This means that each coordinate of a point $p \in P$ can be described using $v = \log_2(1/\Upsilon)$ many bits. We assume here that $1/\Upsilon$ is large (say, all numbers are ints in \mathbb{C} , so $\Upsilon = 2^{-32}$), too large for the grid to be efficiently traversed. And so, for each (ϵ, δ) -differentially private algorithm we present, an algorithm that returns with a high probability of $1 - \beta$ a (α, Δ) -approximation of some geometric feature of $D_P(\kappa)$, we upper bound the Δ -term as a function of $(\alpha, \beta, \epsilon, \delta, d, v)$. (Of course, we must also have that $\kappa > \Delta$ otherwise the algorithm can simply return $[0, 1]^d$.) In addition, any algorithm with runtime of $(n \cdot v \cdot \epsilon^{-1} \cdot \alpha^{-1} \cdot \log(1/\beta\delta))^{\text{poly}(d)}$ is considered efficient.

⁴ Though n comes into play in our work, both in requiring that for large enough κ we have that $D_P(\kappa) \neq \emptyset$ and in bounding Δ , since if $\Delta > n$ then it is trivial to give a (α, Δ) -kernel. Moreover, ideally we would have that $\Delta \leq \sqrt{dn \log(n)}$ so that both $D_P(\kappa)$ and $D_P(\kappa - \Delta)$ (roughly) represent the same Tukey-depth region w.r.t to the distribution the dataset was drawn from, based on the above-mentioned bounds of [8].

Lastly, as pre-processing we apply the algorithm of Kaplan et al. [19] that asserts that for sufficiently large κ it holds that $D_P(\kappa)$ is non-empty and non-degenerate (doesn't have 0-volume).

Detailed Contribution and Organization

First, in Section 2 we survey some background in differential privacy and geometry. Our contributions are detailed in the remaining section and are as follows.

- In Section 3 we give our private (α, Δ) -kernel approximation, that much like its non-private equivalent [1], requires some “fatness” condition. In fact, we have two somewhat different conditions. Our first algorithm requires a known (constant) lower bound on $\text{width}(D_P(\kappa))$, and our second algorithm requires a known (constant) lower bound on the ratio $\frac{\text{width}(D_P(\kappa))}{\text{diam}(D_P(\kappa-\Delta))}$. More importantly, the resulting sets from each algorithm do not satisfy an analogous property to the non-private kernel definition of [1], but rather more intricate properties regarding projections along any direction. Thus, Section 3 begins by discussing these two properties and proving that they are sufficient for finding a (α, Δ) -kernel. Due to brevity, we provide here only the high-level ideas of the first (and very simply) algorithm, whereas its full details, as well as the second algorithm and a heuristic that may allow us to tell if a region is “fat”,⁵ are all deferred to the full version of this work. The remainder of the work presents multiple tools designed in order to privately find a transformation that turns $D_P(\kappa)$ into a fat Tukey-region, each of which may be of independent interest.
- Beimel et al. [3] constructed a function for Tukey-Depth Completion (TDC): given a prefix of $0 \leq i < d$ coordinates, each $x \in \mathbb{R}$ is mapped to the max Tukey-depth of a point whose first $i + 1$ coordinates are the given prefix concatenated with x . Beimel et al. showed that this TDC-function is quasi-concave (details in Section 4), so (i) by off-the-shelf private approximation algorithms for quasi-concave functions [4, 7] we can find x with high $\text{TDC}(x)$ -value; and (ii) repeating this process d times returns a point with high TD. So our first tool is detailed in Section 4 where we present a simple and efficient implementation of the TDC-function in low-dimensions. We also introduce a function that takes an additional parameter ℓ and maps x to $\min\{\text{TDC}(x), \text{TDC}(x + \ell)\}$, which is also quasi-concave and can also be computed efficiently. The two functions play an important role in the construction of *all* following algorithms – we often rotate the space so that some direction v aligns with first axis and then apply TDC to find a good extension of a particular coordinate along v into a point inside $D_P(\kappa)$. While we highlight the main ideas, the full details of this section appear in the full version of this work.
- In Section 5 we give a second batch of rudimentary tools – our efficient private algorithms for $(\alpha, \Delta^{\text{diam}})$ -diameter approximation and $(\alpha, \Delta^{\text{width}})$ -width approximation. These algorithms are quite standard and rely on the Sparse-Vector Technique; thus their formal descriptions are deferred to the full version of this work.
- In Section 6 we turn our attention to *asserting* that the fatness condition required for the kernel-approximation algorithm holds. We present a private (c, Δ) -approximation of the min bounding box problem – it returns a box \mathbf{B} that (a) contains $D_P(\kappa)$ and (b) with volume upper bounded by $c \cdot \text{vol}(D_P(\kappa - \Delta))$. We then show that if $\text{vol}(D_P(\kappa)) \geq \frac{\text{vol}(D_P(\kappa-\Delta))}{2}$ then by affinely mapping \mathbf{B} to $[0, 1]^d$ we turn $D_P(\kappa)$ into a fat Tukey region.

⁵ This heuristic allows us to take κ as input to our algorithm: if the heuristic returns “OK” then the niceness conditions hold and we can return a (α, Δ) -approximation of $D_P(\kappa)$; o/w the algorithms of Sections 6 and 7 allow us to replace the value of the given κ with a different value, one for which we can return a (α, Δ) -approximation of $D_P(\kappa)$.

- In Section 7 we give a private algorithm for finding a “good” depth-parameter κ , one for which it does hold that $\text{vol}(D_P(\kappa)) \geq \text{vol}(D_P(\kappa - \Delta))/2$. We formulate a certain query q where any κ for which $q_P(\kappa)$ is large must also be a good κ , and then give a private algorithm for finding a κ with a large $q_P(\kappa)$ -value. The ε -differentially private algorithm we give is actually rather novel – it is based on a combination of the Exponential-Mechanism with additive Laplace noise. Its privacy is a result of arguing that for any neighboring P and P' where $P' = P \cup \{x\}$ we can *match* κ with $\kappa + 1$ so that $|q_P(\kappa) - q_{P'}(\kappa + 1)| \leq 1$, and then using a few more observations that establish pure ε -differential privacy (rather than (ε, δ) -DP). Again, due to space considerations, the full-details and proofs from Sections 6 and 7 appear in the full version of this work.

Our work thus culminates in the following theorem.

► **Theorem 2.** *There exists an efficient (ε, δ) -differentially private algorithm, that for any sufficiently large dataset P , where $|P| \geq \tilde{\Omega}(d^4 v \cdot \Delta)$, with probability $\geq 1 - \beta$ finds a “good” depth parameter κ and a set \mathcal{S} such that \mathcal{S} is a (α, Δ) -kernel of $D_P(\kappa)$ where $\Delta = O\left(\frac{f(d)}{\varepsilon} \cdot \left(\frac{1}{\alpha}\right)^{\frac{d}{2}} \sqrt{\log\left(\frac{1}{\delta}\right) \log\left(\frac{1}{\alpha\beta}\right)}\right)$ for some function $f(d) = 2^{d^2 \text{poly} \log(d)}$.*

In fact, it is also required that $\Delta \geq \Delta^{\text{BB}}(d, v, \varepsilon, \delta, \beta)$ where Δ^{BB} is guarantee of the private min-bounding-box algorithm, as detailed in Theorem 16; yet this lower-bound holds under a very large regime of parameters.

Additional Works

In addition to the two works [3, 19] that privately find a point inside a convex hull, it is also worth mentioning the works regarding privately approximating the diameter [23, 22] (they return a $O(1)$ -approximation of the diameter that may miss a few points) as well as the recent work of [15] which can also be used to approximate the diameter; and the work of [18] that privately approximates a k -edges polygon yet requires a dataset of points where many lie inside the polygon and many lie *outside* the polygon. No additional works that we know of lie in the intersection of differential privacy and computational geometry. Computational geometry, of course, is a rich field of computer science replete with many algorithms for numerous tasks in geometry. Our work only give private analogs to (a few of) the algorithms of [9, 1], but there are far many more algorithms to be privatized and the reader is referred to [16] for a survey of the field. Many works deal with computing the Tukey-depth and the Tukey region [24, 20], and others give statistical convergence rates for the Tukey-depth when the data is composed of i.i.d. draws from a distribution [28, 8, 5].

2 Preliminaries

Geometry

In this work we use $\langle \cdot, \cdot \rangle$ to denote the inner-product between two vectors in \mathbb{R}^d . We use e_j to denote the nature basis element with 1 on j th coordinate and zeros elsewhere. A closed half-space is defined by a vector u and a scalar λ and it is the set $\{x \in \mathbb{R}^d : \langle x, u \rangle \leq \lambda\}$. A polytope, which is a convex body, is the intersection of finitely many closed half-spaces. For a polytope P and a point x we define $P - x$ as the shift of P by x (namely $z \in P - x$ iff $\exists y \in P$ s.t. $z = y - x$), and we define by cP the blow-up of P by a scalar c . An inner product $\langle x, u \rangle = \|x\| \|u\| \cos(\angle(x, u))$ is also known a projection of x onto the subspace spanned by u . A projection onto a subspace Π^V maps any $x \in \mathbb{R}^d$ to its closest point in the subspace V . The following fact is well-known.

► **Fact 3.** Let S be a convex body. Let u be any vector and let $\Pi^{\perp u}$ be the projection onto the subspace orthogonal to u . Denote ℓ as the max-length of the intersection of S with any affine line in direction u , and denote A as the volume of the projection of S onto the subspace orthogonal to u , $A = \text{vol}(\Pi^{\perp u}(S))$. Then $\frac{A \cdot \ell}{d} \leq \text{vol}(S) \leq A \cdot \ell$.

The fact follows from reshaping S so that it is contained in the “cylinder” whose base is A and height is ℓ , and contains a “pyramid” with base of A and height of ℓ .

The unit-sphere \mathbb{S}^{d-1} is the set of vectors in \mathbb{R}^d of length 1. The *diameter* of the convex body P is defined as $\text{diam}(P) = \max_{p,q \in P} \|p - q\|$, and it is simple to see that $\text{diam}(P) = \max_{u \in \mathbb{S}^{d-1}} \max_{p,q \in P} \langle p - q, u \rangle$. The *width* of a convex body P is analogously defined as $\text{width}(P) = \min_{u \in \mathbb{S}^{d-1}} \max_{p,q \in P} \langle p - q, u \rangle$. A ζ -*angle cover* of the unit sphere is a set of vectors V_ζ such that for any $v \in \mathbb{S}^{d-1}$ there exist u such that $\angle(u, v) \leq \zeta$. It is known that each vector in the sphere can be characterized by $d - 1$ angles $\varphi_1, \varphi_2, \dots, \varphi_{d-1}$ where $\varphi_i \in [0, 2\pi]$ and for any other j , $\varphi_j \in [0, \pi]$. Therefore by discretizing the interval $[0, \pi]$ we can create a ζ -angle cover of size $2\lceil \pi/\zeta \rceil^{d-1}$.

► **Proposition 4 (Proof omitted.)**, Let $\zeta < \frac{1}{2}$. Let V_ζ be a ζ -angle cover of \mathbb{S}^{d-1} . Then $\forall u \in \mathbb{S}^{d-1}$ the closest $v \in V_\zeta$ satisfies $\|u - v\| \leq \sqrt{2}\zeta$.

Tukey Depth

Given a finite set of points $P \subset \mathbb{R}^d$, the Tukey depth [26] of a point $x \in \mathbb{R}^d$ w.r.t P is defined as $\text{TD}(x, P) = \min_{u \in \mathbb{S}^{d-1}} |\{p \in P : \langle p, u \rangle \leq \langle x, u \rangle\}|$. Given P and a depth parameter $\kappa \geq 0$ we denote the κ -*Tukey region* as $D_P(\kappa) = \{x \in \mathbb{R}^d : \text{TD}(x, P) \geq \kappa\}$. It is known that for any set of points P it holds that $\kappa^* = \max_x \text{TD}(x, P) \in [\frac{|P|}{d+1}, \frac{|P|}{2}]$ (see [14]). It is also known that for all κ , the (non-empty) set $D_P(\kappa)$ is a convex polytope which is the intersection of all closed halfspaces that contain at least $n - \kappa + 1$ points out of P [24], this yields a simple algorithm to compute the κ -Tukey region in time $O(n^{\lceil \frac{d}{2} \rceil})$. There is a faster algorithm to compute the κ -Tukey region in time $O(n^d \log n)$ [20], and so to compute *all* of the non-empty Tukey-regions in time $O(n^{d+1} \log n)$.

Differential Privacy

The formal definition of differential privacy [10, 12] is as follows.

► **Definition 5.** Two datasets P and P' are called neighbors if they differ on a single datum, and in this work we assume that this means that $|P \Delta P'| = 1$. A randomized algorithm \mathcal{A} is said to be (ϵ, δ) -differentially private (DP) if for any two neighboring datasets P and P' and for any set of possible outputs S it holds that $\Pr[\mathcal{A}(P) \in S] \leq e^\epsilon \Pr[\mathcal{A}(P') \in S] + \delta$. When $\delta = 0$ we say \mathcal{A} is ϵ -DP or ϵ -pure DP.

Differential privacy composes: if \mathcal{A} is (ϵ, δ) -DP and \mathcal{B} is (ϵ', δ') -DP, then applying \mathcal{A} and then applying \mathcal{B} sequentially on P is a $(\epsilon + \epsilon', \delta + \delta')$ -DP algorithm. It is also worth noting the advanced-composition theorem [13], where the sequential application of k (ϵ, δ) -DP algorithms yields in total an algorithm which is $(O(\epsilon \sqrt{k \ln(1/k\delta)}), 2k\delta)$ -DP (provided $\epsilon < 1$). Since we deal with a constant dimension d , then whenever we compose $\text{poly}(d)$ -many mechanisms, we rely on the basic composition; and whenever we compose $\text{exp}(d)$ -many mechanisms, we rely on the advanced composition.

The Laplace additive noise is a ϵ -DP algorithm that works as follows. Given a function f that maps inputs to real numbers, we first find its global sensitivity $\text{GS}(f) = \max_{P, P' \text{ neighbors}} |f(P) - f(P')|$, then output $f(P) + \text{Lap}(\text{GS}(f)/\epsilon)$. It is also worth noting

the *Sparse Vector Technique* which is an ε -DP algorithm that allows us to assess t queries q_1, q_2, \dots, q_t , each with $GS(q_i) = 1$, and halt on the very first query that exceeds a certain (noisy) threshold. Our algorithms repeatedly rely on the SVT.

Private Approximations of Quasi-Concave Functions

In our work we use as “building blocks” several known results in differential privacy regarding approximating quasi-concave functions. A function $q : \mathbb{R} \rightarrow \mathbb{R}$ is a *quasi-concave function* if for any $x \leq y \leq z$ it holds that $q(y) \geq \min\{q(x), q(z)\}$. Quasi-concave functions that obtain a maximum (namely, there exists some $x \in \mathbb{R}$ such that $\forall y, q(x) \geq q(y)$) have the property that the maximum is obtained on a single closed interval $I = [x, y]$ (we allow the case $x = y$, or $I = \{x\}$). Moreover, it follows that on the interval $(-\infty, x)$ the function q is monotone non-decreasing and on the interval (y, ∞) the function q is monotone non-increasing. The following is known about DP-approximations of quasi-concave functions.

► **Theorem 6.** *Let q be any function $q : \mathbb{R} \rightarrow \mathbb{R}$ satisfying (i) q is quasi-concave, (ii) q has global-sensitivity 1 and (iii) for every closed interval I one can efficiently compute $\max_{x \in I} q(x)$. Let $\mathcal{G} \subset \mathbb{R}$ be a grid of granularity $\Upsilon = 2^{-\nu}$, and denote $q^* = \max_{x \in \mathcal{G}} q(x)$. Then, for any $0 < \beta < 1/2$ there exist differentially private algorithms that w.p. $\geq 1 - \beta$ return some $x \in \mathcal{G}$ such that $q(x) \geq q^* - \alpha^{\text{qc}}(\varepsilon, \delta, \beta)$ where*

$$\alpha^{\text{qc}}(\varepsilon, \delta, \beta) = \begin{cases} O\left(\frac{\nu + \log(1/\beta)}{\varepsilon}\right), & \text{using } \varepsilon\text{-DP binary-search} \\ \tilde{O}\left(\frac{\log(\nu/\beta\varepsilon\delta)}{\varepsilon}\right), & \text{using the “Between Thresholds” Algorithm [7]} \\ O\left(\frac{8^{\log^*(\nu)} \log^*(\nu)}{\varepsilon} \cdot \log\left(\frac{\log^*(\nu)}{\beta\delta}\right)\right) & \text{using the “RecConcave” algorithm [4]} \end{cases}$$

The first bound is given by standard ε -DP binary search algorithm (folklore). The second bound is given by the rather intuitive “Between Threshold” algorithm of Bun et al. [7] where instead of the standard counting function $f(z) = |\{x : x \leq z\}|$ we use the function $f(z) = \max_{x \in (-\infty, z]} q(x) - \max_{x \in [z, \infty)} q(x)$ and set thresholds close to 0 (indicating a maximization point of q). The third is the *RecConcave* algorithm by [4] and is rather involved. (It is unknown⁶ whether the recent work [17] is applicable to general quasi-concave functions.)

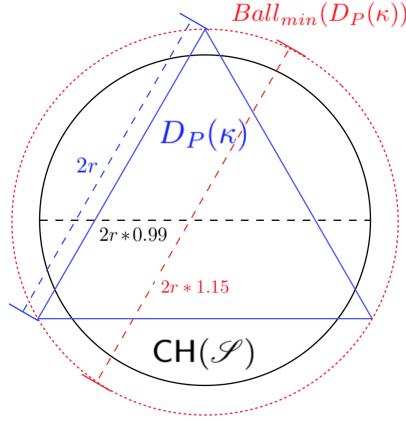
3 Notions of Kernels and Fatness Suitable for Private Approximation

Prior to presenting our algorithm(s) for finding a kernel of a Tukey-region, we first discuss *our goal* – what it is we wish to output, and *our premise* – the kinds of datasets on which we are guaranteed to release such outputs. Recall, our goal is to give a differentially private algorithm that outputs a collection of points \mathcal{S} which is a (α, Δ) -kernel of $D_P(\kappa)$. Namely, this \mathcal{S} satisfies that (after shifting) $(1 - \alpha)D_P(\kappa) \subset \text{CH}(\mathcal{S}) \subset (1 + \alpha)D_P(\kappa - \Delta)$. Clearly, for any two convex bodies s.t. $\mathcal{A} \subset \mathcal{B}$ and for any projection Π we have that $\Pi(\mathcal{A}) \subset \Pi(\mathcal{B})$. (In fact, this holds for any affine transformation.) So if \mathcal{S} is a (α, Δ) -kernel of $D_P(\kappa)$ then it also holds that

$$\forall u \in \mathbb{S}^{d-1} \quad (1 - \alpha) \max_{p, q \in D_P(\kappa)} \langle p - q, u \rangle \leq \max_{p, q \in \text{CH}(\mathcal{S})} \langle p - q, u \rangle \leq (1 + \alpha) \max_{p, q \in D_P(\kappa - \Delta)} \langle p - q, u \rangle \quad (1)$$

In the standard / non-private setting, the definition of kernel [1] is equivalent to $(\alpha, 0)$ -kernel (i.e., setting $\Delta = 0$). Moreover, as defined in [1], a $(\alpha, 0)$ -kernel must satisfy both the property

⁶ Uri Stemmer, private correspondence.



■ **Figure 2** An example showing that the property of Equation (1) doesn't imply that $(1-\alpha)D_P(\kappa) \subset \text{CH}(\mathcal{S})$. Suppose $D_P(\kappa)$ is an equilateral triangle of edge-length $2r$ and \mathcal{S} happens to be a ball of diameter $2 \cdot 0.99 \cdot r$ (and $D_P(\kappa - \Delta)$ is a much larger region). Note that \mathcal{S} does satisfy Equation (1) for $\alpha = 0.01$ yet $0.99D_P(\kappa) \not\subset \text{CH}(\mathcal{S})$.

in (1) and the property that $\mathcal{S} \subset D_P(\kappa)$; and it is straight-forward to show that together, the two properties yield the containment $(1 - O(\alpha))D_P(\kappa) \subset \text{CH}(\mathcal{S}) \subset D_P(\kappa)$. It turns out that in the private setting, with $\Delta > 0$, since it doesn't necessarily hold that $\mathcal{S} \subset D_P(\kappa)$, then property (1) alone does *not* guarantee that we output an (α, Δ) -kernel. Figure 2 illustrates such a setting. So instead, we give algorithms whose resp. outputs satisfy *variations* of the projection property in (1). Below we state two claims showing that the different variations do yield a kernel. The (far from trivial) proofs of the two claims are deferred to the full version of this work.

▷ **Claim 7.** Let \mathcal{S} be a set that satisfies the following property in regards to $D_P(\kappa)$ and $D_P(\kappa - \Delta)$:

$$\forall u \in \mathbb{S}^{d-1}, \quad \max_{p \in D_P(\kappa)} \langle p, u \rangle - \alpha \cdot \text{width}_\kappa \leq \max_{p \in \text{CH}(\mathcal{S})} \langle p, u \rangle \leq \max_{p \in D_P(\kappa - \Delta)} \langle p, u \rangle + \alpha \cdot \text{width}_{\kappa - \Delta} \quad (2)$$

then, denoting $\alpha' = 2\alpha\sqrt{d + \frac{1}{2}}$, there exists two vectors p_1 and p_2 such that we can shift $D_P(\kappa)$ and $D_P(\kappa - \Delta)$ and have that $(1 - \alpha')(D_P(\kappa) - p_1) \subset \text{CH}(\mathcal{S}) - p_1$ and $\text{CH}(\mathcal{S}) - p_2 \subset (1 + \alpha')(D_P(\kappa - \Delta) - p_2)$.

▷ **Claim 8.** Fix $\alpha < 1/6$ and let $\mathcal{S} \subset D_P(\kappa - \Delta)$ be a set such that there exists a point $c \in D_P(\kappa) \cap \mathcal{S}$ for which

$$\forall u \in \mathbb{S}^{d-1}, \quad (1 - \alpha) \max_{p \in D_P(\kappa)} \langle p - c, u \rangle \leq \max_{p \in \text{CH}(\mathcal{S})} \langle p - c, u \rangle + \alpha \cdot \text{width}_\kappa \quad (3)$$

then, denoting $\alpha' = \frac{\alpha}{1-\alpha}(1 + 4\sqrt{d + \frac{1}{2}})$, there exists a vector b such that we can shift $D_P(\kappa)$ and $\text{CH}(\mathcal{S})$ by b and have that $D_P(\kappa) - b \subset (1 + \alpha')(\text{CH}(\mathcal{S}) - b)$.

Definition of Fatness

The algorithms we provide are such that their respective outputs satisfy the premises of Claims 7 and 8. Unfortunately, these algorithms do not return useful sets \mathcal{S} for *any* $D_P(\kappa)$. Much like in the non-private setting [1], in order for each algorithm to output a kernel of $D_P(\kappa)$ we must require that $D_P(\kappa)$ satisfies a certain ‘‘fatness’’ property. In the standard,

non-private setting, a convex polytope $D_P(\kappa)$ is called c_d -fat [1] if there exists a constant $c_d \geq 1$ (depending solely on the dimension d) where $\text{diam}(D_P(\kappa)) \leq c_d \text{width}(D_P(\kappa))$. Alas, our differentially private algorithms require something stronger. Formally, we define the follow various notions of fatness, using the shorthand notation $\text{width}_\kappa \stackrel{\text{def}}{=} \text{width}(D_P(\kappa))$.

► **Definition 9.** We say $D_P(\kappa)$ is (c_d, Δ) -fat if it holds that $\text{width}_\kappa \geq \frac{\text{diam}_{\kappa-\Delta}}{c_d}$. We say $D_P(\kappa)$ is c_d -absolutely fat if $\text{width}_\kappa \geq \frac{1}{c_d}$.

It is clear that the fatness properties can be violated by the addition or removal of a single datapoint to/from P . Therefore, no differentially private algorithm can always assert w.h.p. whether $D_P(\kappa)$ is fat or not, nor estimate its fatness parameter c_d . We comment that in the non-private version [16] the obtained constant is $d^{5/2}2^d(d!)$, whereas our fatness constant is fairly similar: $4d^{5/2}5^d(d!)$.

A Simple Private Kernel Approximation Under “Absolute Fatness”

Under the premise that $D_P(\kappa)$ is c_d -absolutely fat, that is, that $\text{width}_\kappa \geq 1/c_d$ (when c_d is known to the algorithm), we are able to give a pretty simple (α, Δ) -kernel approximation algorithm. The algorithm partitions the $[0, 1]^d$ -cube into subcubes of side length $\frac{2\alpha}{c_d\sqrt{d}}$, and for each subcube C checks whether $\max_{x \in C} \text{TD}(x, P)$ perturbed by Laplace noise is greater than $\kappa' = \kappa - \frac{\Delta}{2}$, and if so – adds C 's center to \mathcal{S} . Here Δ is set using the union-bound on all Laplace random-variables so that w.h.p. any C where $C \cap D_P(\kappa) \neq \emptyset$ adds its center to \mathcal{S} . The full details of the algorithm are deferred to the full version.

► **Theorem 10.** There exists an efficient, (ε, δ) -DP algorithm that returns w.p. $\geq 1 - \beta$ a set \mathcal{S} that satisfies that $\forall u \in \mathbb{S}^{d-1}, \max_{p \in D_P(\kappa)} \langle p, u \rangle - \alpha \cdot \text{width}_\kappa \leq \max_{p \in \mathcal{S}} \langle p, u \rangle \leq \max_{p \in D_P(\kappa - \Delta^{\text{kernel}})} \langle p, u \rangle - \alpha \cdot \text{width}_{\kappa - \Delta^{\text{kernel}}}$, where $\Delta^{\text{kernel}} = O(d(\frac{c_d\sqrt{d}}{\alpha})^{d/2} \sqrt{\log(1/\delta)} \log(\frac{c_d d}{\alpha\beta})/\varepsilon)$. So by Claim 7 \mathcal{S} is a kernel for $D_P(\kappa)$.

Applications

Agarwal et al. [1] define a function μ of a dataset as a *faithful measure* if (i) μ is non-negative, (ii) for every $P \subset \mathbb{R}^d$ we have $\mu(P) = \mu(\text{CH}(P))$, (iii) μ is monotone w.r.t containment of convex bodies, and most importantly, that (iv) for some $c \in (0, 1)$ a $(1 - c\alpha)$ -kernel of P yields a $(1 - \alpha)$ -approximation of $\mu(P) = \mu(\text{CH}(P))$. Obviously, any faithful measure μ can be approximated by a (α, Δ) -kernel \mathcal{S} where $(1 - \frac{\alpha}{c})\mu(D_P(\kappa)) \leq \mu(\text{CH}(\mathcal{S})) \leq (1 + \frac{\alpha}{c})\mu(D_P(\kappa - \Delta))$. Thus, a (α, Δ) -kernel gives suitable approximations for problems such as min/max enclosing ball, min bounding box, John’s Ellipsoid, surface-area etc. (all are faithful measures).

3.1 Remainder of this Extended Abstract

In the full version of this work, we also present another (and more complex) algorithm, that works under the premise that $D_P(\kappa)$ is (c_d, Δ) -fat. Similarly, our proposed heuristic for finding whether the data is (c_d, Δ) -fat is also deferred to the full version of this work. This alternative algorithm and heuristics require additional “building blocks” such as privately finding a point inside $D_P(\kappa)$ and privately estimating the diameter, width and various projections. These building blocks are described in Sections 4 and 5 resp. Note that these additional algorithm and heuristics enable us to return a private kernel of $D_P(\kappa)$ for a user-specified value of κ provided the heuristics return “Yes.” Yet, should the heuristics return “No,” what we do is to find a different value of κ for which a kernel of $D_P(\kappa)$ we can approximate.

In Section 6 we show how to (privately) approximate the bounding box of $D_P(\kappa)$, outputting a box whose volume is comparable to $\text{vol}(D_P(\kappa - \Delta))$. (This algorithm may be of independent interest.) Furthermore, we argue that if it is indeed the case that the volumes of $D_P(\kappa)$ and $D_P(\kappa - \Delta)$ are similar up to a multiplicative factor of 2, then such a bounding box approximation yields a transformation that turns $D_P(\kappa)$ into a $(4d^{\frac{5}{2}}5^d(d!), \Delta)$ -fat Tukey region. Thus, in Section 7 we detail an algorithm that returns a value of κ for which $\frac{\text{vol}(D_P(\kappa - \Delta))}{\text{vol}(D_P(\kappa))} \leq 2$. So the algorithm from Section 7 returns a value of κ , for which the bounding box approximation algorithm of Section 6 does give a transformation that turns $D_P(\kappa)$ fat; implying that the above-mentioned kernel-approximation algorithm can be successfully applied. The reader should be advised that Sections 4-7 are very succinctly described, where we tried to highlight the main ideas of each algorithm and the (often quite subtle) novelties in each algorithm's design.

4 Tools, Part 1: The Tukey-Depth Completion Function

In this section we discuss the implementation of the following *Tukey Depth Completion* function. This function takes as a parameter an i -long tuple of coordinates, where $0 \leq i < d$, and scores each $x \in \mathbb{R}$ with a value κ if the $i + 1$ prefix $\bar{y} \circ x$ can be completed to a point with Tukey-depth of κ .

► **Definition 11** ([3]). *Fix $d \in \mathbb{N}$ and let P be a collection of points in \mathbb{R}^d . For any i -tuple of coordinates $\bar{y} = (y_1, y_2, \dots, y_i)$ where $0 \leq i \leq d - 1$ we define the function $\text{TDC}_{\bar{y}} : \mathbb{R} \rightarrow \mathbb{R}$ by*

$$\text{TDC}_{\bar{y}}^P(x) = \max_{(z_1, z_2, \dots, z_{d-1-i}) \in \mathbb{R}^{d-i-1}} \text{TD}((y_1, \dots, y_i, x, z_1, \dots, z_{d-1-i}), P) \quad (4)$$

For any closed interval $I = [a, b] \subset \mathbb{R}$ we overload the definition of TDC to denote $\text{TDC}_{\bar{y}}^P(I) = \max_{x \in [a, b]} \text{TDC}_{\bar{y}}^P(x)$. Lastly, for any such \bar{y} and any $\ell \in \mathbb{R}$ we denote $\ell\text{-TDC}_{\bar{y}}^P(x) = \min\{\text{TDC}_{\bar{y}}^P(x), \text{TDC}_{\bar{y}}^P(x + \ell)\}$, and similarly, $\ell\text{-TDC}_{\bar{y}}^P(I) = \max_{x \in I} \ell\text{-TDC}_{\bar{y}}^P(x)$. We omit the superscript P whenever the dataset is clear.

In the full version of this work we prove that both the TDC-function and the ℓ -TDC-function are quasi-concave. So it follows that on the real line the values of the TDC-function ascend from 0 to the max-value ($\leq n/2$), then descend back to 0. In particular, for any κ (ranging from 0 to the max-value of the $\text{TDC}_{\bar{y}}$ -function), there exists an interval $[a_\kappa, b_\kappa]$ such that $x \in [a_\kappa, b_\kappa]$ if and only if $\text{TDC}_{\bar{y}}(x) \geq \kappa$. And so, we give a simple, LP-based, algorithm that finds these set of nested intervals $\{[a_\kappa, b_\kappa]\}_{\kappa > 0}$, and then – through binary search – finds the maximum κ whose interval intersect the given point x or interval I . (Note that this binary search is over $\leq n$ elements so it runs in time $O(\log(n))$.)

Extension

One of the key uses to the TDC-function we rely on is when we rotate directions so that the first axis aligns with a given direction v . In such a case, this is equivalent to rotating the set P , so we use the notation $\text{TDC}_{\bar{y}}^{R_v(P)}$ and on occasion just $\text{TDC}_{\bar{y}}^{R_v}$.

A Technical Point: Grid Refinement

We established that for any $0 \leq i \leq d - 1$ and any prefix \bar{y} there exists an efficient algorithm that computes $\text{TDC}_{\bar{y}}(x)$ and $\ell\text{-TDC}_{\bar{y}}(x)$. But as by Beimel et al. [3] noted, it is not a-priori clear that the coordinates of the completion lie on the same grid \mathcal{G}^d we start with. Throughout

most of this paper we ignore this subtlety,⁷ but we do formally show in the full version how to refine \mathcal{G} into a grid \mathcal{G}' with granularity of at least $(\Upsilon/d)^{O(d^4)}$ where the output has all of its coordinates in \mathcal{G}' . The crux of this result is that all coordinates of all vertices of $D_P(\kappa)$ have granularity $\geq (\Upsilon/d)^{O(d^2)}$ (see [19]), and that finding the above-mentioned a_κ, b_κ requires inverting a $(i+1) \times (i+1)$ -matrix whose entries are coordinates of vertices of $D_P(\kappa)$. So we end up requiring a refinement of only $(\Upsilon/d)^{O(i \cdot d^2)}$ per $i \in \{1, 2, \dots, d\}$, so overall our refinement has granularity $(\Upsilon/d)^{O(d^4)}$.

Summary

Now that we refined the grid from \mathcal{G} to \mathcal{G}' with granularity $\Upsilon^{4d^4} = 2^{-v(4d^4)}$, we can apply any DP-algorithm that w.p. $\geq 1 - \beta$ returns a point on \mathcal{G}' with roughly the same value of the maximal value. This gives a DP-algorithm that returns w.p. $\geq 1 - \beta$ a point $x \in \mathcal{G}'$ with either $\text{TDC}_{\bar{y}}$ -value or ℓ - $\text{TDC}_{\bar{y}}$ -value which is $\alpha^{\text{qc}}(\cdot, \cdot, \cdot)$ -close to the max-possible value on the grid. Altogether, we have the following corollary.

► **Corollary 12.** *Fix $\varepsilon > 0$, $\delta \geq 0$, $\beta \in (0, 1/2)$. There exists an efficient (ε, δ) -DP-algorithm, denoted $\text{DPPointInTukeyRegion}$, that takes as input a dataset P and a parameter κ where $D_P(\kappa) \neq \emptyset$ and w.p. $\geq 1 - \beta$ returns a point $\bar{x} \in (\mathcal{G}')^d$ whose Tukey-depth is at least $\kappa - d\alpha^{\text{qc}}(\frac{\varepsilon}{d}, \frac{\delta}{d}, \frac{\beta}{d}) \geq \frac{n}{d+1} - d\alpha^{\text{qc}}(\frac{\varepsilon}{d}, \frac{\delta}{d}, \frac{\beta}{d})$. In particular, for any $\kappa \geq 0$ we return a point of Tukey-depth $\geq \kappa$ provided $n = \Omega(d\kappa + d^2\alpha^{\text{qc}}(\frac{\varepsilon}{d}, \frac{\delta}{d}, \frac{\beta}{d}))$*

$$= \begin{cases} \Omega(d\kappa + d^3 \frac{d^4 v + \log(d/\beta)}{\varepsilon}), & \text{Using the } \varepsilon\text{-DP binary-search} \\ \tilde{\Omega}(d\kappa + d^3 \frac{\log(dv/\beta\varepsilon\delta)}{\varepsilon}), & \text{Using the "Between Thresholds" algorithm} \\ \Omega(d\kappa + d^3 \frac{\delta^{\log^*(dv)} \log^*(dv) \cdot \log(d \log^*(v)/\delta\beta)}{\varepsilon}), & \text{Using the "RecConcave" algorithm} \end{cases} \quad (5)$$

We comment that quantitatively, the results are just as those obtained by [3] (with a minor exception of their granularity level set to Υ^{2^d}), and as such are better than the utility guarantee of [19] when $\delta > 0$. The key improvement of our work is the runtime, decreased to $\text{poly}(v)$.

5 Tools, Part 2: Approximating the Diameter and Width of a Tukey-Region

The Diameter

In this section our goal is to approximate the diameter of $D_P(\kappa)$, denoted $\text{diam}_\kappa = \max_{a, b \in D_P(\kappa)} \|b - a\|$. Our algorithm returns a (α, Δ) -approximation of diam_κ , namely a value ℓ satisfying $(1 - \alpha)\text{diam}_\kappa \leq \ell \leq \text{diam}_{\kappa - \Delta}$. In order to find such an approximation, we leverage on the idea of discretizing all possible directions, which is feasible in constant-dimension Euclidean space. Denoting V_ζ as a ζ -angle cover of the unit sphere it is straight-forward to show that $(1 - \zeta^2)\text{diam}(P) \leq \max_{v \in V_\zeta} \max_{a, b \in P} \langle b - a, v \rangle \leq \text{diam}(P)$. Based on V_ζ , our approximation merely uses the Sparse-Vector Technique (SVT). For each ℓ we pose the query $q_P(\ell) = \max_{v \in V_\zeta} \max_{x \in \mathbb{R}} \ell\text{-TDC}^{R_v(P)}(x)$ where R_v is a rotation that sets v as the first vector basis. The details of the algorithm and its proof of correctness are deferred to the full version of this work.

⁷ So instead of formally stating “we find a point p inside the convex body” we should say “we find a point p within distance $\sqrt{d}\Upsilon$ from a point inside the convex body.” After all, our work already deals with approximations, so under the (rather benign) premise that the diameter of the convex body is sufficiently larger than Υ , this little additive factor changes very little in the overall scheme.

► **Theorem 13.** *There exists an algorithm `DPTukeyDiam` which is an efficient ε -DP algorithm that w.p. $\geq 1 - \beta$ returns a value ℓ which is (α, Δ) -approximation of diam_κ for $\Delta^{\text{diam}}(\varepsilon, \beta) = O\left(\frac{\log((v+\log(d))/\alpha\beta)}{\varepsilon}\right)$.*

The Width

We now turn our attention to the width estimation of the Tukey region $D_P(\kappa)$. Informally, the width of a set is the smallest “sandwich” of parallel hyperplanes that can hold the entire set. Formally, $\text{width}_\kappa = \min_{v \in \mathbb{S}^{d-1}} \max_{a, b \in D_P(\kappa)} |\langle b, v \rangle - \langle a, v \rangle|$. Our private approximation gives a (α, Δ) -approximation of the width – a value w where $(1 - \alpha)\text{width}_\kappa \leq w \leq (1 + \alpha)\text{width}_{\kappa - \Delta}$. It is tempting to think that, much like the approach for diameter approximation, a similar discretization/cover of all directions ought to produce a $(1 + \alpha)$ -approximation of the width. Alas, this approach fails when the width is very small, smaller than the discretization level. But when the discretization is up-to-scale, then we can easily argue the correctness of the discretization approach. The following is proven in the full version of this work.

► **Proposition 14.** *Fix any $\alpha > 0$. Given a set $P \subset \mathbb{R}^d$ with diameter D and width w , if we set $\zeta \leq \min\{\frac{\alpha w}{\sqrt{2}D}, \frac{1}{2}\}$ and take V_ζ as a ζ -angle cover of the unit-sphere, then we have that $w \leq \min_{v \in V_\zeta} \max_{a, b \in P} \langle b - a, v \rangle \leq (1 + \alpha)w$.*

Following Proposition 14 we present our private approximation of width_κ . This approximation also leverages on the query ℓ -TDC for a decreasing sequence of lengths $\ell_1 > \ell_2 > \dots$, however, as opposed to diameter approximation, with each smaller ℓ we also use a different discretization of the unit sphere. For each ℓ_i we set $\zeta_i = \frac{\alpha \ell_i}{4D}$ and use the query $q_P(\ell_i) = \min_{v \in V_{\zeta_i}} \max_{x \in \mathbb{R}} \ell_i\text{-TDC}^{R_v(P)}(x)$. We prove that (i) if $\text{width}(D_P(\kappa)) \geq \ell_i$ then $q_P(\ell_i) \geq \kappa$; and (ii) if $\text{width}(D_P(\kappa)) \leq (1 - \alpha)\ell_i$ and $\zeta_i \leq \frac{\alpha \ell_i}{4D}$ then $q_P(\ell_i) < \kappa$. Thus our algorithm is merely an application of the SVT with these queries. Algorithm’s details and proofs appear in the full version of this work.

► **Theorem 15.** *There exists an algorithm `DPApproxWidth` which is a ε -DP algorithm that w.p. $\geq 1 - \beta$ returns a value ℓ which is (α, Δ) -approximation of width_κ for $\Delta^{\text{width}}(\varepsilon, \beta) = O\left(\frac{\log((v+\log(d))/\alpha\beta)}{\varepsilon}\right)$.*

Note that our width-approximation algorithm requires we refine the angle-cover V_ζ with each iteration. Without any a-priori lower bound on the width, the refinement can be as small as Υ , which renders our algorithm inefficient. That is why in our work we rely on having a particular lower bound, of $1/(4d^{\frac{5}{2}} \cdot 5^d \cdot (d!))$ (which is our fatness bound). In addition, our full version also describes here two additional algorithms (also SVT-based) for subroutines we will require later: estimating the max-projection from a point and finding a direction on which some specific scalar has large TDC-value.

6 Private Approximation of the Bounding Box of $D_P(\kappa)$

In this section we give a differentially private algorithm that returns a transformation that turns $D_P(\kappa)$ into a fat Tukey-region. The transformation is based on (privately) finding an approximated bounding-box for $D_P(\kappa)$, and once such a box is found, then the transformation T is merely an affine transformation, composed of rotation and scaling, that maps the bounding box \mathbf{B} to the hypercube $[0, 1]^d$. We thus focus in this section on a private algorithm that gives a (c_d, Δ) -approximation of the bounding box of $D_P(\kappa)$, so our algorithm’s guarantee relates to both the volume of $D_P(\kappa)$ and the volume of $D_P(\kappa - \Delta)$. Formally, we return (w.h.p) a box \mathbf{B} which is a bounding box that holds $D_P(\kappa)$ and where $\text{vol}(\mathbf{B}) \leq 5^d \cdot (d!) \cdot \text{vol}(D_P(\kappa - \Delta))$.

The algorithm mimics the non-private bounding-box algorithm in [16] (Ch.18). It is a recursive algorithm, where in each level of the recursion we find a segment \bar{st} and an interval I on the line extending this segment where the following three properties must hold: (i) both $s, t \in D_P(\kappa - \Delta)$, (ii) the length of I is $\leq 5\|s - t\|$ and (iii) $\forall x \in D_P(\kappa)$, the projection of x onto this line lies inside I . We then project onto the space orthogonal to \bar{st} and recurse. Property (iii) asserts that $D_P(\kappa)$ is contained inside the box we return; property (i) combined with Fact 3 allows us to infer that $\text{vol}(D_P(\kappa - \Delta)) \geq \|s - t\| \cdot \text{vol}(\Pi^{\perp st}(D_P(\kappa - \Delta)))/d$ where $\Pi^{\perp st}$ is the projection onto the subspace orthogonal to the line connecting s and t ; and property (ii) asserts $\|s - t\| \geq |I|/5$ so that recursively we get a $5^d \cdot d!$ approximation of $\text{vol}(D_P(\kappa - \Delta))$. Thus, asserting that these properties hold w.h.p. becomes the goal of our algorithm, which is far from trivial. Details appear in the full version, along with the proof of the algorithm's correctness.

► **Theorem 16.** *Let $P \subset \mathcal{G}^d$ be a set of points whose Tukey-region $\kappa + d\alpha^{\text{qc}}(\frac{\varepsilon}{d^2+2d-1}, \frac{\delta}{d^2+2d-1}, \frac{\beta}{d^2+2d-1})$ is non-empty. Then there exists an efficient (ε, δ) -DP algorithm that w.p. $\geq 1 - \beta$ returns a box \mathbf{B} where $D_P(\kappa) \subset \mathbf{B}$ and $\text{vol}(D_P(\kappa)) \leq \text{vol}(\mathbf{B}) \leq 5^d(d!)\text{vol}(D_P(\kappa - \Delta^{\text{BB}}))$ for*

$$\Delta^{\text{BB}}(\varepsilon, \delta, \beta) = \begin{cases} O\left(\frac{d^3(v+\log(d/\beta))}{\varepsilon}\right), & \text{Using } \varepsilon\text{-DP binary search} \\ \tilde{O}\left(\frac{d^3 \log(dv/\varepsilon\delta\beta)}{\varepsilon}\right), & \text{Using the "Between Threshold" Alg} \\ O\left(\frac{d^3 \log(dv/\beta)}{\varepsilon} + \frac{d^3 8^{\log^*(v)} \log^*(v) \log(d \log^*(v)/\delta\beta)}{\varepsilon}\right), & \text{Using the "RecConcave" algorithm} \end{cases}$$

From a Bounding Box to a "Fat" Input

In classic, non-private, computational geometry, the bounding-box approximation algorithm can be used to design an affine transformation T that turns the input dataset into a fat input, using a rotation and a separate rescaling of each axis so that \mathbf{B} is mapped to $[0, 1]^d$. Then, finding a kernel for the fat dataset and applying T^{-1} gives a kernel for the original set of points. Unfortunately, we cannot make a similar claim in our setting. Granted, our bounding box is (c_d, Δ) -approximation for any P ; but the resulting affine transformation does not, always, guarantee that applying it turns $D_P(\kappa)$ to be (c'_d, Δ) -fat or c'_d -absolutely fat. This should be obvious, since when $D_P(\kappa - \Delta^{\text{BB}})$ is drastically bigger than $D_P(\kappa)$ and \mathbf{B} is proportional to $D_P(\kappa - \Delta^{\text{BB}})$, mapping \mathbf{B} to $[0, 1]^d$ doesn't "stretch" $D_P(\kappa)$ enough to make it fat. Luckily, we show that non-comparable volumes is the only reason this transformation fails to produce a fat Tukey-region.

► **Lemma 17.** *Fix $\varepsilon > 0$, $\delta \geq 0$ and $\beta > 0$, and define Δ^{BB} as in Theorem 16. Suppose $P \subset \mathcal{G}^d$ is such that for some two parameters $\kappa \geq \kappa'$, where $\kappa - \kappa' \geq \Delta^{\text{BB}}$, we have that $\text{vol}(D_P(\kappa)) \geq \frac{1}{2}\text{vol}(D_P(\kappa'))$. Then there exists a (ε, δ) -differentially private algorithm that w.p. $\geq 1 - \beta$ computes (i) an affine transformation M that turns $M(D_P(\kappa))$ into a convex polytope which is $(c_d, \kappa - \kappa')$ -fat, for $c_d = 4d^{\frac{3}{2}}5^d \cdot (d!)$, and (ii) a transformation \tilde{M} making $\tilde{M}(D_P(\kappa))$ $2d \cdot 5^d \cdot (d!)$ -absolutely fat.*

7 Finding a "Good" κ Privately

Our discussion in Section 6 leaves us with the question of finding a "good" κ and $\kappa' = \kappa - \Delta^{\text{kernel}}$ – where $\text{vol}(D_P(\kappa)) \geq \text{vol}(D_P(\kappa - \Delta^{\text{kernel}}))/2$. First, we establish that there are many such good pairs. [19] proved that if the volume of a Tukey region is non-zero, then it is at least $(d/\Upsilon)^{-d^3}$. Thus, we set $t = \lceil d^3v + d^3 \log_2(d) \rceil$ and so it must hold for any series $\kappa_1 < \kappa_2 < \dots < \kappa_t$ of length t that at least one pair of adjacent κ_i, κ_{i+1} is good,

for otherwise the $\text{vol}(D_P(\kappa_t))$ is below the lower bound of [19]. Consider the specific series where $\kappa_i = i \cdot (4\Delta^{\text{kernel}})$ and denote $m = \kappa_t$. Here, a good pair κ_i, κ_{i+1} are $4\Delta^{\text{kernel}}$ apart, therefore many κ s in some interval $[\kappa_i, \kappa_{i+1}]$ are good, a fact we rely on in the design of our private algorithm.

To that end, we define the query $q_P(\kappa) \stackrel{\text{def}}{=} \max \left\{ 0 \leq i \leq \min\{\kappa - 1, m - \kappa\} : \frac{\text{vol}(D_P(\kappa+i))}{\text{vol}(D_P(\kappa-i))} \geq \frac{1}{2} \right\}$. Our goal is to retrieve a κ where $q_P(\kappa) \geq \Delta^{\text{kernel}}$ since then $(\kappa, \kappa - \Delta^{\text{kernel}})$ is a good pair. It is obvious that $\forall \kappa, q_P(\kappa) \geq 0$ and that $q_P(1) = q_P(m) = 0$, but we also prove in the full version that for any neighboring P and $P' = P \cup \{x\}$ it holds that $|q_P(\kappa) - q_{P'}(\kappa + 1)| \leq 1$. And so our ϵ -DP algorithm first picks a value of κ w.p. $\propto \exp(\frac{\epsilon}{8} q_P(\kappa))$ and then adds Laplace noise (rounded to an integer) to it. Based on all of the above mentioned properties we prove that this ‘‘Shifted Exponential Mechanism’’ is indeed ϵ -DP. We then argue about its utility, which is far more straight-forward, and obtain the following conclusion.

► **Corollary 18.** Fix $\epsilon > 0, \delta \geq 0, \beta > 0$ and set Δ^{kernel} as in Theorem 10 and $m = 4\lceil d^3 v + d^3 \log_2(d) \rceil \Delta^{\text{kernel}}$. Let $P \subset \mathcal{G}^d$ be a set of points such that $D_P(m)$ is non-empty and non-degenerate. Then w.p. $\geq 1 - \beta$, our ‘‘Shifted Exponential Mechanism’’ returns a value κ such that $\text{vol}(D_P(\kappa))/\text{vol}(D_P(\kappa - \Delta^{\text{kernel}})) \geq 1/2$.

References

- 1 Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004.
- 2 Raef Bassily, Kobbi Nissim, Adam D. Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman. Algorithmic stability for adaptive data analysis. In Daniel Wichs and Yishay Mansour, editors, *Symposium on Theory of Computing, STOC*, pages 1046–1059. ACM, 2016.
- 3 Amos Beimel, Shay Moran, Kobbi Nissim, and Uri Stemmer. Private center points and learning of halfspaces. In *COLT*, pages 269–282, 2019.
- 4 Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, volume 8096 of *Lecture Notes in Computer Science*, pages 363–378. Springer, 2013.
- 5 Victor-Emmanuel Brunel. Concentration of the empirical level sets of tukey’s halfspace depth. *Probability Theory and Related Fields*, 173(3-4):1165–1196, 2019.
- 6 Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In *FOCS*, pages 634–649, 2015.
- 7 Mark Bun, Thomas Steinke, and Jonathan Ullman. Make up your mind: The price of online queries in differential privacy. In *Symposium on Discrete Algorithms, SODA*, pages 1306–1325. SIAM, 2017.
- 8 Michael A. Burr and Robert J. Fabrizio. Uniform convergence rates for halfspace depth. *Statistics & Probability Letters*, 124(C):33–40, 2017.
- 9 Timothy M. Chan. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. *Int. J. Comput. Geom. Appl.*, 12(1-2):67–85, 2002.
- 10 C. Dwork, F. Mcsherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.
- 11 Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science (New York)*,

- N. Y.*), 349(6248):636–638, August 2015. URL: <http://www.sciencemag.org/content/349/6248/636>.
- 12 Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, 2006.
 - 13 Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60. IEEE Computer Society, 2010.
 - 14 Herbert Edelsbrunner. *Algorithms in combinatorial geometry*. Monographs in Theoretical Computer Science (10). Springer-Verlag, 1 edition, 1987.
 - 15 Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. Differentially private clustering: Tight approximation ratios. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *NeurIPS*, 2020.
 - 16 Sariel Har-peled. *Geometric Approximation Algorithms*. American Mathematical Society, USA, 2011.
 - 17 Haim Kaplan, Katrina Ligett, Yishay Mansour, Moni Naor, and Uri Stemmer. Privately learning thresholds: Closing the exponential gap. In *Conference on Learning Theory, COLT*, volume 100 of *Proceedings of Machine Learning Research*. PMLR, 2020.
 - 18 Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Differentially private learning of geometric concepts. In *International Conference on Machine Learning, ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 3233–3241. PMLR, 2019.
 - 19 Haim Kaplan, Micha Sharir, and Uri Stemmer. How to find a point in the convex hull privately. In *International Symposium on Computational Geometry (SoCG)*, volume 164 of *LIPICs*, pages 52:1–52:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
 - 20 Xiaohui Liu, Karl Mosler, and Pavlo Mozharovskyi. Fast computation of tukey trimmed regions and median in dimension $p > 2$. *Journal of Computational and Graphical Statistics*, 28(3):682–697, 2019.
 - 21 K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84. ACM, 2007. Full version in: <http://www.cse.psu.edu/~asmith/pubs/NRS07>.
 - 22 Kobbi Nissim and Uri Stemmer. Clustering algorithms for the centralized and local models. In *ALT*, pages 619–653, 2018.
 - 23 Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Locating a small cluster privately. In *PODS*, pages 413–427, 2016.
 - 24 Peter J Rousseeuw and Ida Ruts. Constructing the bivariate tukey median. *Statistica Sinica*, 8(3):827–839, 1998.
 - 25 Haruyuki Sanuki, Rui Fukui, Tsukasa Inajima, and Shin’ichi Warisawa. Cuff-less calibration-free blood pressure estimation under ambulatory environment using pulse wave velocity and photoplethysmogram signals. In *BIOSIGNALS*, 2017.
 - 26 J. W. Tukey. Mathematics and the picturing of data. *Proceedings of the International Congress of Mathematicians*, 2:523–531, 1975.
 - 27 Gary M. Weiss, Kenichi Yoneda, and Thaier Hayajneh. Smartphone and smartwatch-based biometrics using activities of daily living. *IEEE Access*, 7:133190–133202, 2019.
 - 28 Yijun Zuo and Robert Serfling. Structural properties and convergence results for contours of sample statistical depth functions. *Ann. Statist.*, 28(2):483–499, April 2000.

Differentially Oblivious Database Joins: Overcoming the Worst-Case Curse of Fully Oblivious Algorithms

Shumo Chu¹ ✉

University of California, Santa Barbara, CA, USA

Danyang Zhuo ✉

Duke University, Durham, NC, USA

Elaine Shi ✉

Carnegie Mellon University, Pittsburgh, PA, USA

T-H. Hubert Chan ✉

The University of Hong Kong, Hong Kong

Abstract

Numerous high-profile works have shown that access patterns to even encrypted databases can leak secret information and sometimes even lead to reconstruction of the entire database. To thwart access pattern leakage, the literature has focused on *oblivious* algorithms, where obliviousness requires that the access patterns leak nothing about the input data.

In this paper, we consider the `Join` operator, an important database primitive that has been extensively studied and optimized. Unfortunately, any *fully oblivious Join* algorithm would require *always* padding the result to the *worst-case* length which is *quadratic* in the data size N . In comparison, an insecure baseline incurs only $O(R + N)$ cost where R is the true result length, and in the common case in practice, R is relatively short. As a typical example, when $R = O(N)$, any fully oblivious algorithm must inherently incur a prohibitive, N -fold slowdown relative to the insecure baseline. Indeed, the (non-private) database and algorithms literature invariably focuses on studying the *instance-specific* rather than *worst-case* performance of database algorithms. Unfortunately, the stringent notion of full obliviousness precludes the design of efficient algorithms with non-trivial instance-specific performance.

To overcome this worst-case performance barrier of full obliviousness and enable algorithms with good instance-specific performance, we consider a relaxed notion of access pattern privacy called (ϵ, δ) -differential obliviousness (DO), originally proposed in the seminal work of Chan et al. (SODA'19). Rather than insisting that the access patterns leak no information whatsoever, the relaxed DO notion requires that the access patterns satisfy (ϵ, δ) -differential privacy. We show that by adopting the relaxed DO notion, we can obtain efficient database `Join` mechanisms whose instance-specific performance *approximately matches* the insecure baseline, while still offering a meaningful notion of privacy to individual users. Complementing our upper bound results, we also prove new lower bounds regarding the performance of any DO `Join` algorithm.

Differential obliviousness (DO) is a new notion and is a relatively unexplored territory. Following the pioneering investigations by Chan et al. and others, our work is among the very first to formally explore how DO can help overcome the worst-case performance curse of full obliviousness; moreover, we motivate our work with database applications. Our work shows new evidence why DO might be a promising notion, and opens up several exciting future directions.

2012 ACM Subject Classification Theory of computation; Security and privacy → Cryptography; Information systems → Join algorithms; Theory of computation → Design and analysis of algorithms; Security and privacy → Mathematical foundations of cryptography

Keywords and phrases differentially oblivious, database join, instance-specific performance

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.19

¹ (author ordering is randomized)



Related Version *Full Version*: <https://eprint.iacr.org/2021/593>

Funding T-H. Hubert Chan is partially supported by the Hong Kong RGC under the grants 17200418 and 17201220. Elaine Shi is partially funded by NSF under award numbers 1601879 and 2128519, a faculty award from JP Morgan, an ONR YIP award, and a Packard Fellowship.

Acknowledgements We gratefully acknowledge helpful discussions and insightful feedback with Zhao Song and Lianke Qin.

1 Introduction

We consider a scenario in which a trusted client (e.g., an Intel SGX enclave or a trusted client laptop) outsources an encrypted database to an untrusted storage provider (e.g., untrusted memory or a cloud server). The client would like to make queries to the database without endangering the privacy of users in the database. Since all data contents are encrypted, the key challenge is to ensure that *access patterns* to the database do not accidentally harm individual users' privacy. Notably, plenty of recent attacks [24, 51, 52, 56, 59, 63, 70] against encrypted database systems (e.g., CryptDB [74], Cipherbase [6], and TrustedDB [14]) showed that when left unprotected, access patterns can leak highly sensitive information, and in some cases, even lead to reconstruction of the entire database. Given the importance of this problem, several high-profile works implemented *oblivious* database systems, including Opaque [90], OblIDB [41], Obladi [35], as well as the work by Arasu and Kaushik [7], where *obliviousness* requires that access patterns leak nothing about the underlying data.

In this paper, we focus on an important database operation, the **Join** operation, which has been studied extensively in the database literature [1, 13, 16, 33, 44, 55, 61, 71, 78, 89]. Given two tables and a specified attribute (henceforth also called the *join key* or *key* for short)², the **Join** operation computes, for each possible join key value k , the Cartesian product of the rows in each table with the join key k . For example, if the join key k appears twice in the first table and three times in the second table, then in the join result the join key k will have six occurrences. Some works focus on the special case of foreign-key join where it is promised that in one of the input tables, each key appears only once. In this paper, however, we consider the more general case where a key may have multiple occurrences in both tables.

Unfortunately, existing oblivious database systems [7, 41, 90] do not provide a satisfactory solution for the **Join** operation. A fundamental problem is that any *fully oblivious* algorithm for **Join** must always incur the *worst-case cost* even when the actual join result may be short. To see this, recall that an algorithm is said to be *oblivious* iff its memory access patterns (and runtime³) are indistinguishable for any two inputs of the same length [47, 48, 81]. For the case of **Join**, the result size for the worst-case input is $\Theta(N_1 \cdot N_2)$ where N_1 and N_2 denote the sizes of the two input tables, respectively. Thus any fully oblivious algorithm must incur at least $\Omega(N_1 \cdot N_2)$ cost on *any* input instance, even when the input instance has a short join result⁴. This is very expensive in real-world databases, since the join result is typically much smaller than quadratic in the common case.

In this paper, we ask the following natural question:

Can we design join algorithms that provide a meaningful and mathematically rigorous notion of privacy, and moreover, avoid having to pay the worst-case quadratic penalty on every input?

² A join key can also be a set of attributes, without loss of generality, we assume two tables are joined on single attribute in this paper.

³ Note that the length of the physical accesses is the same as the program's runtime.

⁴ The naïve solution of simulating an insecure join algorithm with Oblivious RAM does not provide full obliviousness unless the runtime is padded to the worst case.

Parametrized algorithm design and instance-specific performance. We follow the well-established paradigm in the algorithms and database literature, and focus on *instance-specific* complexity measures. *Parametrized analysis and instance-specific performance have been widely adopted in the database, algorithms, as well as cryptography literature*, and its importance has been explained in numerous prior works. For example, a line of work in the classical (non-private) database literature focused on optimizing the instance-specific performance for database joins [18, 55, 61, 71, 89] – in this context the output length is often used as an additional parameter to characterize the algorithms’ performance. The study of instance-specific performance is also closely related to the prominent line of work on parametrized algorithms design and analysis [31, 37, 42, 69, 77, 79] (see Roughgarden’s textbook for an excellent overview [77].) Last but not the least, notable works in the cryptography literature also consider how to achieve good instance-specific performance (sometimes called “input-specific runtime” in the cryptography literature) in the Turing Machine or the Random Access Machine (RAM) models: for example, the seminal work by Goldwasser et al. [49] is motivated by “overcoming the worst-case curse” in cryptographic constructions; and a similar notion is adopted in subsequent works [5, 58].

Prior approaches introduce arbitrary leakages to achieve good instance-specific performance. As mentioned, full obliviousness *precludes* the design of algorithms with non-trivial instance-specific performance. However, prior oblivious database systems [7, 41, 90] do care about instance-specific performance bounds. To achieve good instance-specific performance, they give up on full obliviousness (even though this line of work is commonly referred to as “oblivious databases”), and introduce arbitrary leakages, e.g., by leaking the multiplicity of keys, the lengths of intermediate arrays, the final output length, and/or the exact runtime of the (non-private) program. The ramifications of such leakages are poorly understood, and can lead to unforeseen privacy breaches. Since numerous prior encrypted database systems allowing arbitrary leakages have been broken [24, 51, 52, 56, 59, 63, 70], our philosophy is to advocate for an approach that provides rigorous mathematical guarantees on the leakage.

Although our work focuses on a privately outsourced database scenario, it is interesting to note that in the cryptography literature, a line of work has focused on general RAM computations on encrypted data [20, 23, 32, 45, 46, 50]. These works also care about plugging access pattern leakage. Thus, to achieve full security, essentially full obliviousness is necessary in these constructions (and indeed these constructions rely on Oblivious RAM as a building block). Typically this line of work either pads the RAM computation to the runtime on the worst-case input, or they allow leakage of the exact runtime and thus violate full obliviousness – the ramification of such leakage is unclear and can lead to severe privacy breaches in some applications.

Overcoming the worst-case curse with differential obliviousness. Since the worst-case performance curse is inherent for full obliviousness, we would need a relaxed (but nonetheless meaningful and rigorous) privacy notion to achieve good instance-specific performance. We therefore turn our attention to the notion of *differential obliviousness* (DO) recently defined by Chan et al. [28]. Simply put, differential obliviousness requires that the access patterns revealed during a program’s execution must satisfy (ϵ, δ) -differential privacy [39]. So far, a couple of prior works [17, 28] have shown theoretical separations between DO and full obliviousness, thus providing initial theoretical evidence why DO is worth studying. Besides the few pioneering investigations, the landscape of DO remains much unexplored. Designing DO algorithms, especially for practically motivated applications, is a relatively new territory.

1.1 Our Contributions and Results

We show novel DO database join algorithms that can *approximately match the instance-specific performance of the insecure baseline*, whereas any fully oblivious join algorithm must inherently incur, on common instances with $O(N)$ -sized outputs, at least a linear (in database size) blowup relative to the insecure baseline. Our work is among the very first to formally explore how DO can help overcome the worst-case curse of full obliviousness.

Specifically, we present *two main upper bound results*: 1) a DO join algorithm in the standard word-RAM model where the primary performance metrics are the algorithm’s runtime and output length, and 2) a DO join algorithm in the *external-memory* model where the primary performance metrics are the algorithm’s cache complexity and output length. Both algorithms approximately match the performance of the insecure baselines in the corresponding setting. Both models are important to consider: the standard word-RAM model is the prevalent model in which algorithms are studied; and the external-memory model is the best fit when we rely on secure processors such as Intel’s SGX to privately outsource the sensitive database to an untrusted server (as we explain more later).

We also prove *lower bound* results regarding the performance of any DO join algorithm. The lower bounds show that some small slowdown relative to the insecure baseline is necessary. Moreover, our upper bound matches the lower bound when the result size is at least quasi-linear. For other parameter regimes, e.g., when the result size is linear or shorter, there remains a small gap between our upper bounds and lower bounds – and bridging this gap is an interesting direction for future work.

We now present the result statements more formally.

Results for the word-RAM model. Recall the application scenario mentioned at the beginning of the paper: a *trusted* client stores an *encrypted* database on an *untrusted* storage. Data can only be decrypted within the trusted client which also runs the database engine. Anything fetched or written to the storage is encrypted such that the adversary can only observe the access patterns.

In the standard word-RAM model, we assume that the trusted client is a CPU with $O(1)$ private registers, and the cost is measured in terms of the number of memory words transmitted between the CPU and memory (which equates to the runtime of the algorithm). Table 1 summarizes our results for the standard RAM model. For simplicity, the results are stated for the typical parameters⁵ $\epsilon = \Theta(1)$ and $\delta = 1/N^c$ for some constant $c \geq 1$ and a more generalize version will be provided in Theorem 1.

As shown in Table 1, our algorithm achieves $O(R + N \log N)$ runtime and $R + O((\mu_{\max} + \log N) \cdot \log N)$ result size where N denotes the total input length, R denotes the true result size (when the insecure algorithm is run), and μ_{\max} denotes the multiplicity of the most frequent join key in the database. Note that even an insecure join algorithm must incur at least $R + N$ runtime since it has to at least read the input and write down the output. In the common case in practice, the true output size R is small, e.g., $R = O(N)$. In this case, our DO algorithm achieves almost a factor of N performance improvement relative to any fully oblivious solution whose cost is inherently quadratic.

We also compare our algorithm with a naïve DO algorithm that basically simulates the insecure algorithm (described in Section 3.5) using the state-of-the-art *statistically* secure Oblivious RAM [29, 88]⁶ and then appends an appropriate noise to the result as well as

⁵ In the cryptography literature, sometimes we want δ to be a negligible function in N . In this case, the $\log N$ factor in the performance bound is replaced with any super-logarithmic function.

⁶ Like Chan et al. [28], we adopt a statistical notion of differential obliviousness that defends against even computationally unbounded adversaries.

■ **Table 1** Our results: stated for the typical parameters when $\epsilon = \Theta(1)$ and $\delta = \frac{1}{N^c}$ for some constant $c \geq 1$. N_1 and N_2 denote the lengths of the two input tables, $N := N_1 + N_2$, R denotes the length of the true join result, and μ_{\max} denotes the maximum multiplicity of any join key in the two input tables. $\Theta(\cdot)$ means that it is both an upper- and lower-bound.

	Runtime	Result size
Insecure	$\Theta(R + N)$	R
Fully oblivious	$\Theta(N_1 \cdot N_2)$	$\Theta(N_1 \cdot N_2)$
Our differentially oblivious algorithms		
Naïve: Theorem 15	$O((R + N \log N) \log^2 N)$	$R + O(N \log N)$
Main scheme 1: Thm 1	$O(R + N \log N)$	$R + O((\mu_{\max} + \log N) \cdot \log N)$
LB: Thm 2	$\Omega(R + N \log \log N + \mu_{\max} \log N)$	$R + \Omega(\mu_{\max} \log N)$

to the program’s runtime (see Section 3.5 for a more detailed description). In comparison, our algorithm is a $\log^2 N$ factor faster than the naïve DO algorithm. Other standard DO techniques such as the work by Komargodski and Shi [62] fail to work in our context – see Section 2.5 for more discussions. In light of our lower bound for any DO join algorithm, our upper bound achieves optimality in terms of result length as long as $\mu_{\max} \geq \log N$, and is optimal in terms of runtime when $\mu_{\max} = \Theta(N)$.

Our main theorems⁷ are also informally described below for a broader range of choices for ϵ and δ than Table 1.

► **Theorem 1** (Our DO join algorithm). *Let R be the length of the true join result (i.e., without fillers), let μ_{\max} denote the multiplicity of the most frequent join key in either input array, and let N denote the total input length. There is an (ϵ, δ) -differentially oblivious join algorithm that runs in time $O(R + N(\log N + \frac{1}{\epsilon} \log \frac{1}{\delta}))$ and produces a result whose length is at most $R + O(\frac{1}{\epsilon} \cdot (\mu_{\max} + \frac{1}{\epsilon} \log \frac{1}{\delta}) \cdot \log \frac{1}{\delta})$.*

Table 1 also shows our lower bound which states that any DO join algorithm must incur at least $\Omega(R + N \log \log N + \mu_{\max} \log N)$ runtime and must have a result size of at least $R + \Omega(\mu_{\max} \log N)$ with high probability (assuming the same typical choices of ϵ and δ). A formal statement with more general parameters is given below.

► **Theorem 2** (Limits of any DO join algorithm (informal)). *Let N be the total input length, then, for most reasonable choices⁸ of ϵ and δ ,*

1. *any (ϵ, δ) -differentially oblivious join algorithm must produce a result of at least $R + \Omega(\mu_{\max} \cdot \frac{1}{\epsilon} \cdot \log \frac{\epsilon}{\delta})$ with at least δ/ϵ probability.*
2. *any “natural” (ϵ, δ) -differentially oblivious join algorithm must have some input of total length N and whose true join result size is R , such that the algorithm incurs at least $\Omega(R + N \log \log \frac{1}{\delta} + \mu_{\max} \cdot \frac{1}{\epsilon} \cdot \log \frac{\epsilon}{\delta})$ runtime with at least δ/ϵ probability.*

In the above, the lower bound for runtime holds for a broad class of *natural* algorithms that do not perform encoding or computation on the elements’ payloads – indeed, most known join algorithms fall into this class. We refer the reader to the online full version [34] for more details.

⁷ In Table 1, we assume that $\delta = 1/N^c$ like the standard differentially privacy literature suggests. In some cryptographic application settings where one may desire δ to be a negligible function in N , there will be an extra (arbitrarily small) super-constant factor added to the bounds in Table 1. See also Theorem 1 for the statement with general parameters.

⁸ See the formal theorem in our online full version [34] for a more precise characterization of the parameter regime in which the lower bound holds.

■ **Table 2** Our results: cache-agnostic cache complexity. See the caption of Table 1 for the meaning of the notations N_1 , N_2 , N , and R . Our results need to assume the standard “tall cache” and “wide block” assumptions, i.e., $M \geq B^2$, and $B \geq \log^{0.55} N$ where M is the cache size and B is the block size.

Cache-oblivious cache complexity	
Insecure	$O(\frac{R}{B} + \frac{N}{B} \cdot \log_{\frac{M}{B}} \frac{N}{B})$
Fully oblivious	$\Theta(N_1 \cdot N_2/B)$
Our differentially oblivious algorithms	
Naïve: Theorem 15	$O((R + N \log N) \log N \cdot \log_B N)$
Main scheme 2: Cor 3	$O(\frac{R}{B} + \frac{N}{B} \cdot \log N)$

Results for in the cache-agnostic, external-memory model. The external-memory model [3, 43, 86] and cache complexity are important for scenarios where we want to employ secure processors to enable encrypted, differentially oblivious databases. Imagine that the server has a secure processor such as Intel SGX. In this case, the database is stored in an encrypted format on the server, and only the secure processor can decrypt the data and perform computation. In other words, we can think of the *trusted client* as the secure processor, and the rest of the server’s software stack is untrusted. Moreover, a remote client can communicate with the secure processor using a secure channel to ask queries and receive answers back.

Interestingly, it turns out that when Intel SGX is used to outsource both the computation and storage to an untrusted server, the major performance metric is the number of pages the SGX enclave needs to fetch. Each enclave page swap is a heavy-weight operation that involves communication with the untrusted operating system, and moreover, the enclave must decrypt (or encrypt) the memory page being swapped in (or out). In this scenario, the trusted enclave memory can be viewed as a *cache* whose size is henceforth denoted M , and each page is a *block* (i.e., the atomic unit being swapped in and out) whose size is henceforth denoted B . Further, the rest of the storage outside the trusted enclave memory is the *external memory*. An algorithm’s *cache complexity* is defined as the number of blocks transmitted between the cache and the external memory during the algorithm’s execution. In the online full version [34], we provide additional background on the external-memory model which is a well-accepted model in the algorithms literature – it is very interesting to observe that the line of work on external-memory algorithms [3, 43, 86] is a perfect fit for studying the performance of algorithms running on commodity secure processors.

We propose a variant of our algorithm optimized for cache complexity. Our algorithm is *cache agnostic*, i.e., the algorithm is unaware of the cache’s parameters, namely, M and B . Cache-agnostic was also commonly referred to as “cache-oblivious” in the algorithms literature [38, 43]. In our paper, we use the term “cache-agnostic” instead to disambiguate from our usage of the term “obliviousness”. The importance and advantages of cache-agnostic algorithms have been extensively discussed in the algorithms literature [38, 43]. First, a cache-agnostic algorithm is “universal” and the performance bounds hold no matter what the system parameters (including M and B) are. Not only so, when deployed on a multi-level memory hierarchy, an optimal cache-agnostic algorithm would give optimal IO performance between any two adjacent levels of the hierarchy [38, 43].

Table 2 summarizes our cache complexity results. The insecure baseline and the naïve DO algorithm in the table are described in Section 3.5. As shown in the table, our cache complexity is quite close to that of the insecure baseline, and outperforms the naïve DO algorithm by a $(B/\log B) \cdot \log^2 N$ factor. In a typical scenario, $B = \text{poly log } N$; in this case, our improvement over the naïve DO algorithm is polylogarithmic.

Last but not the least, our cache efficient instantiation has relatively small constants in the big-O notation, and therefore an interesting future direction is to implement our algorithm and measure its concrete efficiency. We summarize our cache-complexity results in the following corollary:

► **Corollary 3** (Our DO join algorithm: cache complexity). *There is an (ϵ, δ) -DO database join algorithm that incurs cache complexity upper bounded by $\frac{1}{B} \cdot O\left(N\left(\log_{\frac{M}{B}} \frac{N}{B} + \frac{1}{\epsilon} \log \frac{1}{\delta}\right) + R + \left(\frac{1}{\epsilon} \log \frac{1}{\delta}\right)^2\right)$, assuming the standard tall cache assumption $M = \Omega(B^2)$ and the wide block assumption $B = \Omega(\log^{0.55} N)$.*

Both the tall cache assumption and the wide block assumption are standard assumptions adopted commonly in the external-memory algorithms line of work [3, 9, 38, 43, 86].

Technical highlight. Inspired by the original work of Chan et al. [28], we adopt the following design paradigm for devising DO algorithms. At a very high level, we decompose the task of designing a DO algorithm into the following: 1) identify a set of intermediate ideal functionalities with *differentially private leakage*; and 2) leverage oblivious algorithms building blocks to obviously realize these ideal functionalities, such that the access patterns leak only the stated differentially private leakage, and nothing else.

Although the design paradigm is simple to state, the non-trivial challenge is to identify appropriate intermediate functionalities that not only lend to solving our problem, but also being cognizant that the computational tasks they embody must have efficient oblivious realizations. We defer the algorithmic details to subsequent formal sections, and we hope that our algorithmic techniques can inspire the design of DO algorithms for new applications. We believe that our work provides further evidence on top of the early-stage explorations of Chan et al. [28] and Beimel et al. [17] that differential obliviousness is a useful notion that deserves attention.

2 Technical Roadmap

For convenience, henceforth we call the two input tables *arrays*, denoted \mathbf{I}_1 and \mathbf{I}_2 respectively. Each element in \mathbf{I}_1 and \mathbf{I}_2 is either a real element of the form (k, v) or a filler element of the form (\perp, \perp) . For a real element, k is called the join key (or key for short) and v is called the payload. The database join operation wants to compute, for each unique join key k , the Cartesian product of the elements contained in both arrays with join key k . All results are concatenated and output, and moreover, the output is allowed to contain an arbitrary number of filler elements that may be needed for privacy.

► **Remark 4** (Simplifying assumption for the roadmap). Throughout our informal technical roadmap, we will assume the typical parameters $\epsilon = \Theta(1)$ and $\delta = 1/N^c$ for an arbitrary constant $c \geq 1$. In this case, $\frac{1}{\epsilon} \log \frac{1}{\delta} = \Theta(\log N)$, and we thus use two expressions interchangeably – but our formal sections later will differentiate the two to be more general. Specifically, jumping ahead, we often need to add noises of magnitude roughly $\frac{1}{\epsilon} \log \frac{1}{\delta} = \Theta(\log N)$.

Our differential oblivious join algorithm will make use of standard oblivious algorithm building blocks including oblivious sorting [4, 10, 76], and oblivious compaction [11, 67]. We review these building blocks in more detail in Section 3.6.

2.1 Warmup Algorithm

We first present a warmup that achieves $O(R + N \log^2 N)$ runtime – the warmup algorithm does not achieve the bounds stated earlier; but it is conceptually simpler and helps our understanding. Later in Section 2.3, we describe additional techniques to improve the algorithm’s asymptotical performance, and achieve the bounds stated earlier.

A strawman idea. To understand our algorithm, let us first consider a flawed strawman – we sketch the high-level idea, and for the time being, omit the details on how to oblivious sorts to implement the relevant steps.

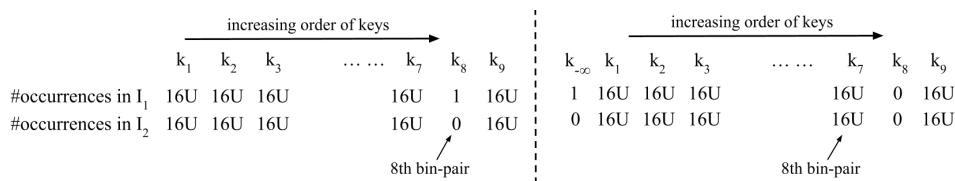
1. *Compute the bin load array L .* First, using a constant number of oblivious sorts, write down a list L of length $N := |\mathbf{I}_1| + |\mathbf{I}_2|$. Each element in L is of the form $(k, \widehat{n}_k^{(1)}, \widehat{n}_k^{(2)})$ where $\widehat{n}_k^{(b)}$ denotes the noisy count of the join key k in table $b \in \{1, 2\}$. The noisy count is obtained by adding an appropriate, independently sampled noise to the actual multiplicity of join key k in the corresponding array. To maintain correctness, the noise must be non-negative. So rather than adding a Laplacian noise with standard deviation $1/\epsilon$, we shift the Laplacian to the right to be centered at $U/2 = \Theta(\frac{1}{\epsilon} \cdot \log \frac{1}{\delta}) = \Theta(\log N)$. In this way, the noise lies within the range $[0, U]$ except with δ probability⁹.

The list L should contain all join keys that appear in at least one input array, padded with fillers of the form $(\star_1, \widehat{n}_{\star_1}^{(1)}, \widehat{n}_{\star_1}^{(2)})$, $(\star_2, \widehat{n}_{\star_2}^{(1)}, \widehat{n}_{\star_2}^{(2)})$, \dots , to a length of N . The filler join keys \star_1, \star_2, \dots have an actual multiplicity of 0 in both input arrays, and thus their noisy counts are the shifted Laplacian noise in the range $[0, U]$. The list L is *sorted by the join key k , and all filler join keys appear at the end.*

2. *Binning.* Now, we have $2N$ bins each indexed by a pair (b, i) where $b \in \{1, 2\}$ and $i \in [N]$. Let k_i denote the i -th smallest join key. Then, the bin indexed (b, i) has capacity $\widehat{n}_{k_i}^{(b)}$, and all elements in \mathbf{I}_b with the join key k_i are destined for this bin. Using a constant number of oblivious sorts, route all elements in either array to their respective destined bins, and pad each bin with fillers to its intended capacity. Note that the bins corresponding to the filler join keys \star_1, \star_2, \dots have no real elements in them and are full of fillers.
3. *Bin-wise Cartesian product.* Now, take every pair of bins $(1, i)$ and $(2, i)$ for $i \in [N]$, and compute the Cartesian product of elements in the two bins – if the two elements being joined have the same real join key, add the joined tuple to the output; otherwise, add a filler element to the output.

A flaw that violates differential obliviousness. The above algorithm is natural and conceptually simple; it almost works, except for a critical flaw that violates differential obliviousness, which is illustrated in Figure 1 and explained in detail below. Observe that the array L is sorted by the join key during Step 1. Now, consider an input $\mathcal{I} := (\mathbf{I}_1, \mathbf{I}_2)$ in which 8th smallest join key k_8 appears only 1 time in \mathbf{I}_1 and does not appear in \mathbf{I}_2 , and all other join

⁹ In our formal sections later, we actually use a shifted geometric distribution which is the discrete counterpart of the real-valued Laplacian, and moreover we simply truncate the δ -probability mass outside the range $[0, U]$ which allows us to get deterministic bounds on the algorithm’s runtime.



■ **Figure 1** Strawman scheme: a flaw that violates differential obliviousness. See the main body for more explanation.

keys that appear in \mathcal{I} appear more than $16U$ times in both arrays. Consider a 2-neighboring input where the only occurrence of k_8 is replaced with a join key $k_{-\infty}$ that 1) does not exist in \mathcal{I} , and 2) is smaller than all other join keys in \mathcal{I} . In this case, an adversary observing the access patterns of the program can easily tell which input is used. Recall that the bin pairs are sorted in the order of the join keys: in the case of \mathcal{I} , the 8-th bin pair has small capacities and the first bin pair has large capacities (where small means between at most $U + 1$ and large means at least $16U$). In the case of \mathcal{I}' , however, the first bin pair, now corresponding to the join key $k_{-\infty}$, has small capacities.

A remedy. It turns out that a simple fix can address the above flaw: instead of ordering the array L using the join key k , we can order it lexicographically based on the $(\hat{n}_k^{(1)}, \hat{n}_k^{(2)})$ fields. Intuitively, this can avoid accidental information leakage through the ordering. More specifically, in the above steps 2 and 3, the capacities of all bins are leaked to the adversary. Therefore, if we use the fields $(\hat{n}_k^{(1)}, \hat{n}_k^{(2)})$ to order the array L and the corresponding bins, then the only information leaked is the *multiset* of $(\hat{n}_k^{(1)}, \hat{n}_k^{(2)})$ pairs. Given the *multiset* of the $(\hat{n}_k^{(1)}, \hat{n}_k^{(2)})$ pairs, the access patterns of the above algorithm are fully determined.

Therefore, it suffices to prove that the leakage, i.e., the *multiset* of the $(\hat{n}_k^{(1)}, \hat{n}_k^{(2)})$ pairs, satisfies $(O(\epsilon), O(\delta))$ -differential privacy. In our formal technical sections later, we shall prove that this is indeed the case as long as the noises are chosen from the shifted and truncated geometric distribution defined in Section 3.4.

Final step: compaction of the join result. The above modification fixes the security flaw, but at this moment, the result output by our algorithm may have length $O(R + N \log^2 N)$ – see Section 2.2 for a more detailed analysis. Specifically, when the true result length R is small, the additive term $N \log^2 N$ is dominant.

We would like our algorithm to output a result that is as short as possible specific to the instance. Clearly, for the result to be correct, it cannot be shorter than R . In the online full version [34], we prove that all (ϵ, δ) -DO algorithms must have result length at least $R + \Omega(\mu_{\max} \cdot \frac{1}{\epsilon} \log \frac{1}{\delta})$ where μ_{\max} is the maximum multiplicity of any join key in either array.

Our idea is to obviously compact the result output by the above algorithm to length $R + \text{noise}$ where *noise* is sampled from an appropriate distribution. The most natural idea is to sample a shifted Laplacian noise proportional to $\Delta_{\text{global}}/\epsilon$ where Δ_{global} is the global sensitivity of the exact result length (i.e., how much the length of the exact result would change in the *worst case* when we change one position in the input). However, Δ_{global} can be as large as $\Theta(N)$. Instead, we would like to achieve an *instance-optimal* bound on the result length (for almost all parameter regimes). To do so, we add noise proportional to the local sensitivity (or instance-specific sensitivity) which is equal to μ_{\max} . To make the idea work, however, we have to first obtain a noisy version $\hat{\mu}_{\max}$ to the local sensitivity μ_{\max} , and then add noise proportional to $\hat{\mu}_{\max}$ to the result length. In this way, our algorithm achieves result

length $R + (\mu_{\max} + \log N) \log N$ which is instance-optimal in light of the $R + \Omega(\mu_{\max} \log N)$ lower bound in almost all parameter regimes. We defer a detailed description of the scheme to the formal technical sections.

2.2 Performance of the Warmup Algorithm

As mentioned, the warmup algorithm, obtained by changing the way the array L is ordered in the strawman solution, achieves runtime $O(R + N \log^2 N)$. To understand the techniques in Section 2.3 that improves the performance to $O(R + N \log N)$, let us first understand the performance breakdown.

1. The first step, which computes the bin load array L , performs a constant number of oblivious sorts on arrays of length at most $N := |\mathbf{I}_1| + |\mathbf{I}_2|$, and thus takes $N \log N$ time.
2. The second step, which places the elements into bins, performs a constant number of oblivious sorts, and the length of the arrays sorted is upper bounded by the sum of the bin capacities. In the worst case, there can be $\Theta(N)$ sparsely loaded bins each with only $O(1)$ number of real elements. The noise added to each bin's capacity is roughly of magnitude $\frac{1}{\epsilon} \log \frac{1}{\delta}$ which is $O(\log N)$ under typical parameters (see Remark 4). Therefore, the length of the array sorted is at most $O(N \log N)$, and the second step takes time $O((N \log N) \log(N \log N)) = N \log^2 N$.
3. The third step computes the Cartesian product of pairs of bins: the runtime of this step is the actual result length R when there is no noise, plus the number of fillers. The number of fillers is maximized when there are $\Theta(N)$ bins each with $O(1)$ number of real elements and $O(\log N)$ fillers. In this case, the total number of fillers after the Cartesian product is $O(N \log^2 N)$. Therefore, the third step takes time $O(R + N \log^2 N)$.

Summarizing the above, our warmup algorithm achieves $O(R + N \log^2 N)$ runtime.

2.3 Final Algorithm

Section 2.2 reveals that the $N \log^2 N$ additive term in the performance bound is incurred because in the worst-case scenario, there can be $\Theta(N)$ sparsely loaded bin-pairs each with $O(1)$ real elements, and padded with $\Theta(\log N)$ fillers¹⁰. This introduces the $N \log^2 N$ additive term in two ways: 1) the binning step requires sorting arrays of length $O(N \log N)$ which takes $O(N \log^2 N)$ time; and 2) the bin-wise Cartesian product introduces $O(N \log^2 N)$ fillers.

Imprecisely speaking, having many sparsely loaded bin-pairs cause a small “signal to noise” ratio, i.e., the ratio of fillers is high. To improve the performance bound to $O(R + N \log N)$, our idea is to reduce the number of bin-pairs to $O(N/\log N)$, thereby improving the “signal to noise” ratio. We say that a join key k is *sparse* iff its noise counts $\hat{n}_k^{(1)}$ and $\hat{n}_k^{(2)}$ are both upper bounded by $2U$, where recall that $U = \Theta(\frac{1}{\epsilon} \log \frac{1}{\delta})$. A join key that is not sparse is said to be *dense*. Recall that $N := |\mathbf{I}_1| + |\mathbf{I}_2|$.

As mentioned, our noise distribution is upper bounded by U except with probability δ . This means that if a bin-pair contains a dense join key, then at least one of the bins in the pair has at least U real elements in it except with δ probability. Thus there can be

¹⁰For example, this can happen if there are many elements with $O(1)$ occurrences in both input arrays, or with $O(1)$ occurrences in one array but not appearing in the other. In the latter case, essentially there are many elements in the symmetric difference of the two input arrays that do not contribute to the true joined result.

no more than $O(N/\log N)$ bin-pairs for dense-keys. Our focus therefore is to consolidate multiple sparse join keys into the same bin-pair such that each bin-pair contains at least $\Theta(U)$ elements (including elements from both input arrays). To achieve this, we perform the following.

Compute key-to-bin mapping. Recall that earlier we computed the bin load array L which contains tuples of the form $(k, \hat{n}_k^{(1)}, \hat{n}_k^{(2)})$ sorted according to lexicographical ordering on $(\hat{n}_k^{(1)}, \hat{n}_k^{(2)})$. It is not too hard to extend the algorithm for computing L such that the bin load array L also stores the actual counts, i.e., L now contains entries of the form $(k, n_k^{(1)}, \hat{n}_k^{(1)}, n_k^{(2)}, \hat{n}_k^{(2)})$. where $n_k^{(1)}$ and $n_k^{(2)}$ denote the actual multiplicity of the join key k in \mathbf{I}_1 and \mathbf{I}_2 , respectively.

Now, we can classify L into a part L_s corresponding to sparse keys, i.e., $L_s := \{(k, n_k^{(1)}, \hat{n}_k^{(1)}, n_k^{(2)}, \hat{n}_k^{(2)}) \in L : \hat{n}_k^{(1)} \leq 2U, \hat{n}_k^{(2)} \leq 2U\}$; and a part $L_d := L \setminus L_s$ corresponding to dense join keys. All of L , L_s , and L_d are sorted according to lexicographical ordering on $(\hat{n}_k^{(1)}, \hat{n}_k^{(2)})$; and L_s and L_d can be constructed in $O(N)$ time if we allow the access patterns to reveal the noisy counts contained in L .

Our goal now is to construct an array called **BinMap** that maps join keys to bin pairs (note by constructing **BinMap**, we have not moved the elements into their bins yet – the actual moving will be done in the subsequent binning step). Each entry of **BinMap** is of the form (k, i) , meaning that the join key k should be mapped to the i -th bin-pair. To construct **BinMap**, we first scan through L_d : for each $i \in \{1, 2, \dots, |L_d|\}$, if the i -th entry in L_d has the join key k , add the tuple (k, i) to the array **BinMap**. At this moment, **BinMap** stores the mapping from each dense join key to its destined bin-pair index. The capacities of these bin-pairs (for dense join keys) are determined by the noisy counts in L_d .

Next, we will add to **BinMap** the mapping from sparse join keys to their bins. Specifically, sparse join keys are mapped to additional bin-pairs numbered $\{(1, j), (2, j) : j = |L_d| + 1, |L_d| + 2, \dots, |L_d| + O(N/U)\}$ where j is also called the bin-pair index. All of these bins (for sparse join keys) will have capacity *exactly* $4U$, and here we allow multiple join keys to be mapped to the same bin pair. The invariant we want is that for each bin-pair $(1, j), (2, j)$ where $j \in \{|L_d| + 1, |L_d| + 2, \dots, |L_d| + O(N/U)\}$, a total of at least $2U$ elements will be mapped to the bin pair (summing across both input arrays). In this way, all the sparse join keys altogether will not consume more than $N/2U$ bins.

To achieve this, we can scan linearly through L_s . For each entry $(k, n_k^{(1)}, \hat{n}_k^{(1)}, n_k^{(2)}, \hat{n}_k^{(2)}) \in L_s$ encountered during the scan, we append to **BinMap** a tuple (k, j) that indicates that join key k is mapped to the j -th bin-pair. Here j is the current bin-pair counter whose starting value is $|L_d| + 1$, and j is incremented whenever one of the current bins is about to exceed its capacity. To achieve this, the algorithm additionally maintains two counters that remember how many cumulative elements have been mapped to the current bins $(1, j)$ and $(2, j)$ so far. Whenever one of the bins $(1, j)$ or $(2, j)$ is about to exceed its capacity $4U$, we increment the bin-pair counter j and reset both counters to be 0 again, i.e., we start mapping join keys to the next bin-pair. We stress that the access pattern of this step is fixed and depends only on $|L_s|$ because we append a single entry to **BinMap** whenever we visit an entry of L_s .

Binning. Next, we perform a binning step and move elements into their desired bins – henceforth a bin designated for a single dense join key is called a D-bin, and a bin designated for possibly multiple sparse join keys is called an S-bin. To perform the binning, we need **BinMap** which provides the mapping between join keys and their bin indices. D-bins have capacities determined by the corresponding noisy counts contained in L_d and there are $|L_d|$

19:12 Differentially Oblivious Database Joins

D-bins. S-bins have capacities exactly $4U$, and the number of S-bins is an a-priori fixed upper bound CN/U for a sufficiently large constant C . We can now use a constant number of oblivious sorts to move elements into their desired bins, padding each bin with fillers to its intended capacity as defined above. Note that it is possible that some S-bins do not receive any element.

Remainder of the algorithm. The remainder would be similar to the warmup algorithm. We perform bin-wise Cartesian product; and finally we obliviously compact the result adding an appropriate noise to the final result length. Since now, multiple join keys can share the same bin-pair, during the Cartesian product step, whenever we try to pairwise-join two elements with different join keys, we append a filler to the output array.

2.4 Lower Bound Results

As mentioned, we prove new lower bounds on the result length and runtime of any DO join algorithm. The result length lower bound is proven using the definition of differential obliviousness. Our lower bound on runtime is obtained by taking the maximum of two lower bounds: 1) the aforementioned lower bound on the result length, and 2) a lower bound that stems from a privacy-preserving and complexity-preserving reduction from sorting to database join. Such a reduction shows that any DO join algorithm must suffer from the same lower bound for DO sorting proven recently by Chan et al. [28]. We refer the reader to the online full version [34] for the detailed statements and proofs.

2.5 Additional Related Work

In this paper, we adopt the differential obliviousness notion defined by Chan et al. [28]. Besides Chan et al. [28], several other works also considered related but somewhat incomparable notions [60, 68, 87].

Besides the aforementioned works on *oblivious* databases [7, 35, 41, 90], a related but incomparable line of work [2, 14, 25, 26, 36, 57, 73, 74, 80, 82, 83] focuses on encrypted databases or searchable encryption systems. Typically, this line of works either give up on hiding access patterns [14, 25, 26, 36, 74, 80, 82, 83], or hide the access pattern by performing a linear scan through the entire database upon every update or query [2]. The cryptographic techniques for computation on encrypted data developed in this line of work is somewhat orthogonal and complementary to our techniques for obfuscating the access patterns.

Following the classical differential privacy (DP) literature, another line of work that focuses on differentially private database queries [30, 53, 54, 65, 66]. These works assume that the database curator is trusted and only aims to guarantee that the result is DP – they are not concerned about information leakage through the runtime behavior of the database engine. Notably, the techniques we use to release the noisy counts for the multiplicity of join keys may be remotely reminiscent of differentially private histogram mechanisms [2, 15, 19, 22, 64, 84]. We stress, however, that our work and techniques are of a different nature from classical DP mechanisms, including classical DP algorithms for releasing histograms. While prior DP mechanisms introduce noise to the statistics released, in our context, we introduce noise to the algorithm’s access patterns (and not its output) – importantly, we need to do so without affecting the algorithm’s correctness. It would be very interesting, however, to apply our DO techniques to classical DP algorithms – in this way, both the runtime behavior of the database as well as the released statistics would guarantee DP, i.e., we get “end-to-end” privacy.

Mazloom and Gordon [68] proposed new techniques that guarantee differential obliviousness for tasks that can be performed in a graph-parallel framework. They rely on shuffling to guarantee that the access patterns of these algorithms reveal only differentially private histograms. While seemingly related to our techniques, we do not know any straightforward way to apply their techniques to the join problem and get our asymptotical bounds: partly, our techniques are non-trivial because *we avoid suffering too much overhead for join keys that are in the symmetric difference of the two input arrays* – these join keys do not contribute to the true joined result. Imprecisely speaking, doing such “pruning” *privately* introduces non-trivial algorithmic challenges.

Komargodski and Shi [62] suggest how to compile any Turing Machine (TM) to a differentially oblivious TM. A strawman idea is to apply their compiler to the (insecure) TM that computes the database join problem. Unfortunately, this completely fails because their work defines neighboring on the *operational sequences* of two TMs; whereas we define neighboring on the *inputs*. For two inputs that are neighboring (i.e., Hamming distance 1), applying the insecure TM that computes database join over these inputs may result in operational sequences that are far apart. This is also partly why our problem is challenging.

2.6 Open Problems

Our work is among the first to explore DO algorithms motivated by practical database systems. Our work reveals that this is a promising direction with many intriguing open questions. For example, can we bridge the gap between our upper- and lower-bounds for a broader range of parameters?

Another exciting direction is to explore DO algorithms for other common database queries. In this paper, we considered two-way joins. In the classical, non-private database literature, however, *multi-way join* [1, 13, 16, 33, 44, 55, 61, 71, 78, 89] received significantly more attention because instance-optimal two-way join is long known to be a solved problem. Our paper reveals that with the extra privacy requirements, even two-way join raises non-trivial algorithmic challenges. Of course, it also makes sense to ask whether one can design efficient DO algorithms for multi-way joins as well, especially, whether we can (approximately) match the performance of the best known insecure algorithms. Recent works in the database literature also considered join algorithms for the special case when the input tables are already sorted based on the join key (or more generally, preprocessed in some way). In this case, it may not be necessary to read the entire input, and sublinear (non-private) algorithms are known [61, 71]. Therefore, an open question is whether we can achieve sublinear DO algorithms for sorted inputs. Besides joins, more general class of database queries such as conjunctive queries [1] are also interesting to consider.

Last but not the least, as mentioned, the cache-efficient variants of our algorithms are potentially implementable and suitable for SGX-type scenarios. Implementing DO algorithms in practical database systems and evaluating their concrete performance is another exciting future direction.

3 Preliminaries

We assume that the algorithm is executed in a standard Random Access Machine (RAM) model. The adversary can observe the access patterns of the program, i.e., in each step, which memory location is accessed and whether each access is a read or write operation. The adversary, however, cannot observe the data contents – for example, in a secure processor setting, the data contents protected by encryption. We will be concerned about two metrics:

1) the program’s runtime; and 2) the program’s cache complexity [3]. For the runtime statements, we assume a standard word-RAM and the CPU has only $O(1)$ number of private registers. For the cache complexity metric, we assume a standard external-memory RAM model [3] where the CPU has M bits of private cache, and every time it needs to transmit data to and from memory, an atomic unit (called a “block”) of B bits is transferred. The cache complexity metric measures how many blocks are transmitted between the CPU and memory. All of our algorithms are cache agnostic [38, 43], i.e., the algorithm need not know the cache’s parameters M and B .

3.1 Database Join

Database join is the following problem. Let \mathcal{K} denote the space of join key and \mathcal{V} denote the space of payloads. Let \mathbf{I}_1 and \mathbf{I}_2 be two input arrays each containing pairs of the form (k, v) , where $k \in \mathcal{K} \cup \{\perp\}$ is called a *join key* and $v \in \mathcal{V} \cup \{\perp\}$ is called the *payload*. If $k \in \mathcal{K}$ and $v \in \mathcal{V}$, the element (k, v) is said to be a *real* element. Without loss of generality, we may assume that if an element’s join key k is \perp , its payload v must be \perp too: such elements, of the form (\perp, \perp) , are said to be *filler* elements.

Our goal is to output an array \mathbf{O} such that for each non-filler join key $k \in \mathcal{K}$ that appears in both \mathbf{I}_1 and \mathbf{I}_2 : let $\{(k, v_1), (k, v_2), \dots, (k, v_m)\}$ be the multi-set of elements having join key k in \mathbf{I}_1 , and let $\{(k, w_1), (k, w_2), \dots, (k, w_{m'})\}$ be the multi-set of elements having join key k in \mathbf{I}_2 ; then the multi-set $\{k, v_i, w_j\}_{i \in [m], j \in [m']} \subseteq \mathbf{O}$. We use R to denote the size of this multi-set. Moreover, besides the multi-set $\{k, v_i, w_j\}_{i \in [m], j \in [m']}$, \mathbf{O} should contain no other element with the join key k . Additionally, the output array \mathbf{O} may contain any number of filler elements of the form (\perp, \perp, \perp) .

In other words, the output array \mathbf{O} contains, for each join key $k \in \mathcal{K}$, the Cartesian product of the elements in both input arrays under join key k ; and additionally, \mathbf{O} may contain some filler elements. The filler elements in the output array \mathbf{O} may be needed for privacy reasons as will become clear later.

► **Remark 5** (Motivation for allowing fillers in the input array). In our formulation, we allow the input arrays \mathbf{I}_1 and \mathbf{I}_2 to contain filler elements, because the length of the input arrays may already be noisy to mask the true number of elements contained in it. For example, if the input array comes from a differentially oblivious database such as in the work by Chan et al. [28], then the input arrays would already contain a random number of filler elements.

3.2 Full Obliviousness

In this paper, we consider execution of algorithms on the Random Access Machine (RAM) model. Let Alg denote a possibly randomized algorithm and let \mathbf{I} denote an input to the algorithm. We use the notation $\text{Accesses}^{\text{Alg}}(\mathbf{I})$, a random variable denoting the sequence of memory addresses accessed and whether each access is a read or write, generated by a random execution of the algorithm Alg on input \mathbf{I} . Therefore, $\text{Accesses}^{\text{Alg}}(\mathbf{I})$ is also called the “access patterns” of Alg on input \mathbf{I} .

► **Definition 6** (δ -obliviousness). *We say that an algorithm Alg satisfies δ -obliviousness w.r.t. the leakage function $\text{Leak}(\cdot)$, iff there exists a simulator Sim , such that $\text{Accesses}^{\text{Alg}}(\mathbf{I})$ has statistical distance at most δ from the simulated access patterns $\text{Sim}(\text{Leak}(\mathbf{I}))$.*

In other words, the access patterns are simulatable by a simulator Sim which knows only the leakage function but nothing more about the input \mathbf{I} . Note also that δ is allowed to be a function in $N = |\mathbf{I}|$.

A typical leakage function is leaking only the length of the input and nothing else, i.e., $\text{Leak}(\mathbf{I}) := |\mathbf{I}|$.

► **Definition 7** (Full obliviousness). *Henceforth, whenever we say Alg is (fully) oblivious (i.e., omitting the leakage function and δ), the leakage function would be the default one $\text{Leak}(\mathbf{I}) := |\mathbf{I}|$, and δ is assumed to be a negligible function in N .*

Throughout the paper, we say that a function $\nu(N)$ is a negligible function, iff for any $c \in \mathbb{N}$, there exists a sufficiently large N_0 such that for all $N \geq N_0$, $\nu(N) \leq 1/N^c$. In other words, ν drops faster than any inverse-polynomial function.

3.3 Differential Obliviousness

Neighboring inputs. Two inputs $(\mathbf{I}_1, \mathbf{I}_2)$ and $(\mathbf{J}_1, \mathbf{J}_2)$ are said to be neighboring, iff $|\mathbf{I}_1| = |\mathbf{J}_1|$ and $|\mathbf{I}_2| = |\mathbf{J}_2|$, and moreover, the following holds:

- either $\mathbf{I}_1 = \mathbf{J}_1$, and moreover, \mathbf{I}_2 and \mathbf{J}_2 differ in exactly one position;
- or $\mathbf{I}_2 = \mathbf{J}_2$, and moreover, \mathbf{I}_1 and \mathbf{J}_1 differ in exactly one position.

Differential obliviousness. Imagine that a database join algorithm Alg is executed on a Random Access Machine (RAM). The two input arrays $(\mathbf{I}_1, \mathbf{I}_2)$ reside in memory, and at the end of the algorithm, the output array \mathbf{O} is written to a designated position in memory.

► **Definition 8** ((ϵ, δ) -differential obliviousness). *We say that a database join algorithm Alg satisfies (ϵ, δ) -differential obliviousness or (ϵ, δ) -DO for short, iff for any neighboring inputs $(\mathbf{I}_1, \mathbf{I}_2)$ and $(\mathbf{J}_1, \mathbf{J}_2)$, for any set S ,*

$$\Pr \left[\text{Accesses}^{\text{Alg}}(\mathbf{I}_1, \mathbf{I}_2) \in S \right] \leq e^\epsilon \cdot \Pr \left[\text{Accesses}^{\text{Alg}}(\mathbf{J}_1, \mathbf{J}_2) \in S \right] + \delta$$

where $\text{Accesses}^{\text{Alg}}(\mathbf{I}_1, \mathbf{I}_2)$ is a random variable denoting the sequence of memory addresses (also called access patterns) generated by a random execution of the algorithm Alg on input $(\mathbf{I}_1, \mathbf{I}_2)$.

Specifically, in the standard RAM model, in each time step, the machine visits one memory location, reading it and then updating it with either the old value or a new value. Therefore, $\text{Accesses}^{\text{Alg}}(\mathbf{I}_1, \mathbf{I}_2)$ is just the ordered list of all memory addresses accessed in all time steps. Moreover, the length of $\text{Accesses}^{\text{Alg}}(\mathbf{I}_1, \mathbf{I}_2)$ is also the (randomized) runtime of the algorithm. Like the standard notion of differential privacy, our notion secures against unbounded adversaries.

Typical choices of ϵ and δ . Typically, we would like $\epsilon = \Theta(1)$. The standard differential privacy literature [85] recommends that δ be set to $1/N^c$ for some constant $c > 1$. In the cryptography literature, sometimes we would like δ to be negligibly small in N .

3.4 Mathematical Building Blocks

► **Definition 9** (Symmetric geometric distribution). *Let $\alpha > 1$. The symmetric geometric distribution $\text{Geom}(\alpha)$ takes integer values such that the probability mass function at k is $\frac{\alpha-1}{\alpha+1} \cdot \alpha^{-|k|}$.*

As we shall see, our algorithm will hide the true cardinality of a set by padding it with a random number of filler elements. Below we define a useful distribution from which we shall sample the noises.

19:16 Differentially Oblivious Database Joins

► **Definition 10** (Shifted and truncated geometric distribution). *Let $\epsilon > 0$ and $\delta \in (0, 1)$ and $\Delta \geq 1$. Let k_0 be the smallest positive integer such that $\Pr[|\text{Geom}(e^{\frac{\epsilon}{\Delta}})| \geq k_0] \leq \delta$, where $k_0 = \frac{\Delta}{\epsilon} \ln \frac{2}{\delta} + O(1)$. The shifted and truncated geometric distribution $\mathcal{G}(\epsilon, \delta, \Delta)$ has support in $[0, 2(k_0 + \Delta - 1)]$, and is defined as:*

$$\min\{\max\{0, k_0 + \Delta - 1 + \text{Geom}(e^\epsilon)\}, 2(k_0 + \Delta - 1)\}$$

For the special case $\Delta = 1$, we write $\mathcal{G}(\epsilon, \delta) := \mathcal{G}(\epsilon, \delta, 1)$.

In the main body of the paper, for simplicity, we shall first assume that we can sample from the shifted and truncated geometric distribution in $O(1)$ time. In the online full version [34], we discuss how to remove this assumption without blowing up the runtime.

Notation. Given two random variables X and Y , we use $X \sim_{(\epsilon, \delta)} Y$ to denote that X and Y satisfy the standard (ϵ, δ) -differentially private inequality, i.e., for all subsets S , $\Pr[X \in S] \leq e^\epsilon \cdot \Pr[Y \in S] + \delta$, and $\Pr[Y \in S] \leq e^\epsilon \cdot \Pr[X \in S] + \delta$,

► **Fact 11** (Differential privacy through adding truncated and shifted geometric noise [15, 28]). *Let $\epsilon > 0$ and $\delta \in (0, 1)$. Suppose u and v are two non-negative integers such that $|u - v| \leq \Delta$. Then,*

$$u + \mathcal{G}(\epsilon, \delta, \Delta) \sim_{(\epsilon, \delta)} v + \mathcal{G}(\epsilon, \delta, \Delta).$$

► **Fact 12** (Post-processing). *Let $X \in \mathcal{X}$ and $X' \in \mathcal{X}$ be random variables and let $F : \mathcal{X} \rightarrow \mathcal{Y}$ be a possibly randomized function. Suppose that $X \sim_{(\epsilon, \delta)} X'$. Then, we have that*

$$F(X) \sim_{(\epsilon, \delta)} F(X').$$

► **Fact 13** (Composition of differentially private mechanisms (Theorem B.1 of [40])). *Suppose that for any neighboring \mathbf{I} and \mathbf{I}' , $T_1(\mathbf{I}) \sim_{(\epsilon_1, \delta_1)} T_1(\mathbf{I}')$. Henceforth let $\text{supp}(T_1(\mathbf{I}))$ denote the support of applying the function T_1 to the data \mathbf{I} . Suppose that for any neighboring \mathbf{I} and \mathbf{I}' , for any $s_1 \in \text{supp}(T_1(\mathbf{I})) \cup \text{supp}(T_1(\mathbf{I}'))$, $T_2(\mathbf{I}, s_1) \sim_{(\epsilon_2, \delta_2)} T_2(\mathbf{I}', s_1)$. Then, for any neighboring \mathbf{I}, \mathbf{I}' , we have that*

$$(T_2, T_1)(\mathbf{I}) \sim_{(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)} (T_2, T_1)(\mathbf{I}')$$

The following operational lemma will guide our algorithm design and analysis.

► **Lemma 14** (Operational lemma for differential obliviousness [28]). *If an algorithm is δ_0 -oblivious w.r.t. a leakage function $\text{Leak}(\cdot)$, and moreover, Leak is (ϵ, δ) -differentially private, then the algorithm satisfies $(\epsilon, \delta_0 + \delta)$ -differential obliviousness.*

3.5 Naïve Solutions

Insecure algorithm. If privacy is not needed, there is an algorithm that computes the join result in (expected) $O(N + R)$ time, where R is the actual result size and $N := |\mathbf{I}_1| + |\mathbf{I}_2|$. Basically, hash elements in each input array into a separate Cuckoo hash table [8, 72] and within each entry of the hash table, use a linked list to store all elements with the same join key. Now, we can pairwise-join elements with the same join key from the two input arrays by querying the Cuckoo hash table. The entire algorithm completes in $O(N + R)$ expected time (where the randomness comes from hashing).

The Cuckoo hashing based insecure solution, however, does not have great cache complexity. For cache complexity, we use a different insecure baseline, that is, first sort the two input arrays by join key, and then use the most straightforward approach to compute pairwise-join elements with the same join key from the two arrays. This algorithm achieves $O(\frac{R}{B} + \frac{N}{B} \cdot \log_{\frac{M}{B}} \frac{N}{B})$.

Observe also that any insecure algorithm must at least read the entire input to guarantee correctness. Thus an always correct insecure algorithm must incur at least $\Omega(N)$ runtime.

Fully oblivious algorithm. The following naïve algorithm, which tries every potential pair of elements from the two input arrays, can achieve fully oblivious database join with $O(|\mathbf{I}_1| \cdot |\mathbf{I}_2|)$ runtime:

For each element $(k, v) \in \mathbf{I}_1$, for each element $(k', v') \in \mathbf{I}_2$: if $k = k'$, add (k, v, v') to the output; else add (\perp, \perp, \perp) to the output.

As argued in the online full version [34], $O(|\mathbf{I}_1| \cdot |\mathbf{I}_2|)$ is also the best we can hope for if full obliviousness is desired.

Naïve differentially oblivious algorithm. A naïve approach to achieve differentially oblivious join is to rely on a *statistically* secure Oblivious RAM (ORAM) that defends against unbounded adversaries, such as Circuit ORAM [29, 88], to simulate the aforementioned insecure algorithm with $O(\log^2 N)$ blowup in runtime. Suppose that the result size is R and let N be the total input length. We know that the insecure algorithm must complete in $T \leq C \cdot (R + N)$ steps for some constant C , and it produces an output of length R . Now, given the $O(\log^2 N)$ ORAM simulation overhead, simulating the insecure algorithm with ORAM requires at most $C'(R + N) \log^2 N$ steps for some sufficiently large $C' > C$. However, if we just stopped here, then the algorithm would leak information through the result length R and its running time (which is the same as the total length of the physical memory accesses in the RAM model).

To plug this leakage, the algorithm proceeds to add noise to the result length and its own runtime. To achieve this, we continue to use the ORAM to simulate the following steps:

1. Append $\xi := \mathcal{G}(\epsilon, \delta, N) = O(\frac{N}{\epsilon} \log \frac{1}{\delta})$ number of filler elements to the joined result – this requires the ORAM to simulate $O(\xi)$ additional steps. Henceforth let $\widehat{R} := R + \xi$.
2. Finally, pad the running time of the simulated algorithm to $C'(\widehat{R} + N) \log^2 N$.

Note that in the above, we add noise $\mathcal{G}(\epsilon, \delta, N)$ noise to R because the global sensitivity of R is upper bounded by N , that is, changing one element in the input can change R by at most N .

To obtain better cache complexity, we can place the ORAM schemes' binary tree data structures in an Emde Boas layout; specifically, each access in the original program will incur a cache complexity of $O(\log N \cdot \log_B N)$ in the ORAM simulation.

► **Theorem 15** (Naïve DO algorithm). *The above naïve algorithm satisfies $(\epsilon, \delta + \text{negl}(N))$ -differential obliviousness where $\text{negl}(\cdot)$ denotes a suitable negligible function.*

Further, suppose that $\epsilon = \Theta(1)$ and $\delta = \frac{1}{\text{poly}(N)}$, let $U = O(\frac{1}{\epsilon} \log \frac{1}{\delta}) = O(\log N)$; then, the above naïve algorithm achieves $O((R + NU) \log^2 N)$ runtime and $O((R + NU) \log N \cdot \log_B N)$ cache complexity, and outputs a result of $O(R + NU)$ length.

Proof. The runtime, cache complexity and output length follows from the above description. Observe that because of ORAM (which can fail with $\text{negl}(N)$ probability), the access pattern are simulatable given N and \widehat{R} . By Lemma 14, it suffices to prove that the leakage \widehat{R} satisfies (ϵ, δ) -differential privacy.

19:18 Differentially Oblivious Database Joins

Consider changing one element in the input $(\mathbf{I}_1, \mathbf{I}_2)$ to form $(\mathbf{I}'_1, \mathbf{I}'_2)$. We have that $|R(\mathbf{I}_1, \mathbf{I}_2) - R(\mathbf{I}'_1, \mathbf{I}'_2)| \leq N$ where $R(\mathbf{I}_1, \mathbf{I}_2)$ denotes the length of the exact result on the input $(\mathbf{I}_1, \mathbf{I}_2)$. By Fact 11, we have that

$$\widehat{R}(\mathbf{I}_1, \mathbf{I}_2) \sim_{(\epsilon, \delta)} \widehat{R}(\mathbf{I}'_1, \mathbf{I}'_2).$$

where $\widehat{R}(\mathbf{I}_1, \mathbf{I}_2)$ denotes the length of the result output by the naïve DO algorithm upon input $(\mathbf{I}_1, \mathbf{I}_2)$. Hence, the algorithm is $(\epsilon, \delta + \text{negl}(N))$ -differentially oblivious, where the extra $\text{negl}(N)$ comes from the failure probability of ORAM. \blacktriangleleft

3.6 Oblivious Algorithm Building Blocks

We describe several oblivious algorithm building blocks. Unless otherwise noted, obliviousness is defined w.r.t. the input-length leakage (i.e., informally, speaking, only the input length is leaked).

Oblivious compaction. Given an input array where some elements are marked as distinguished, output an array where all distinguished elements are moved to the front, and all non-distinguished elements are moved to the end. The very recent works by Asharov et al. [11, 12] constructed a $O(n)$ -time oblivious compaction algorithm that can compact any input array of length n . Their linear-time compaction algorithm is not *stable*, i.e., among the distinguished (or non-distinguished) elements, the output does not preserve the relative order the elements appeared in the input, and this non-stability is inherent [67].

To get our cache complexity result, we will adopt the randomized, cache-agnostic, oblivious compaction algorithm by Lin, Shi, and Xie [67]: their algorithm achieves optimal $O(n/B)$ cache complexity and $O(n \log \log n)$ runtime assuming that $M = \Omega(B^2)$ and $B \geq \log^{0.55} n$.

Oblivious sort. Ajtai, Komlós, and Szemerédi [4] showed that there is a sorting circuit with $O(n \log n)$ comparators that can correctly sort any input array containing n elements. Such a sorting circuit can be executed on a Random Access Machine (RAM) in $O(n \log n)$ time assuming that each element can be represented using $O(1)$ words.

The recent work by Ramachandran and Shi [75] constructed a *randomized*, cache-agnostic, oblivious sort algorithm that achieves $O(n \log n)$ runtime and $O((n/B) \log_{M/B}(n/B))$ cache complexity, assuming the tall cache assumption that $M = \Omega(B^2)$ and further $M = \Omega(\log^{1+\epsilon} n)$ for an arbitrarily small constant $\epsilon \in (0, 1)$.

Given oblivious sorting, we can realize a couple intermediate abstractions including oblivious send-receive and oblivious bin placement which we define below. Both primitives can be realized by invoking oblivious sorting constant number of times.

Oblivious send-receive. The send-receive primitive¹¹ solves the following problem. In the input, there is a source array and a destination array. The source array represents n senders, each of whom holds a key and a value; it is promised that all join keys are distinct. The destination array represents n' receivers each holding a join key. Now, have each receiver learn the value corresponding to the join key it is requesting from one of the sources. If the

¹¹The send-receive abstraction is often referred to as oblivious routing in the data-oblivious algorithms literature [21, 27, 29]. We avoid the name “routing” because of its other connotations in the algorithms literature.

join key is not found, the receiver should receive \perp . Note that although each receiver wants only one value, a sender can send its values to multiple receivers.

Prior works [21, 27, 29] have shown that oblivious send-receive can be accomplished through a constant number of oblivious sorts on arrays of length $O(n + n')$. The algorithm is oblivious w.r.t. the leakage n and n' , (i.e., informally, only the lengths of the input arrays are leaked).

Oblivious bin placement. A bin placement algorithm solves the following problem. Suppose we have m bins each of capacity s_1, s_2, \dots, s_m , respectively. We are given an input array denoted \mathbf{I} , where each element is either a *filler* denoted \perp or a *real* element that is tagged with a bin identifier $\beta \in [m]$ denoting which bin it wants to go to. It is promised that every bin will receive no more elements than its capacity. Now, move each real element in \mathbf{I} to its desired bin. If any bin is not full after the real elements have been placed, pad it with filler elements at the end to its desired capacity. Finally, output the concatenation of the resulting bins.

Chan and Shi [29] describes an oblivious bin placement algorithm that solves the special case of the problem when all the bin sizes are equal. Their algorithm relies on a constant number of oblivious sorts. It is not difficult to extend their algorithm to the case when the bin sizes are not equal. For completeness, we describe the modified algorithm in the online full version [34]. Specifically, *the algorithm satisfies obliviousness w.r.t. to the leakage that contains the input size, as well as the sizes of all bins.* Let n denote the size of the input array, and let $S := \sum_{\beta \in [m]} s_\beta$ be the sum of the sizes of all bins. The runtime of the algorithm is upper bounded by $T_{\text{sort}}(n + S)$ and the cache-agnostic, cache complexity of the algorithm is upper bounded by $Q_{\text{sort}}(n + S)$, where $T_{\text{sort}}(n')$ and $Q_{\text{sort}}(n')$ denote the runtime and (cache-agnostic) cache complexity of oblivious sort over an input array of size n' .

Deferred Contents

Due to space constraints, we defer the full algorithmic details and proofs to the online full version [34].

References

- 1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- 2 Archita Agarwal, Maurice Herlihy, Seny Kamara, and Tarik Moataz. Encrypted databases for differential privacy. *Proc. Priv. Enhancing Technol.*, 2019(3):170–190, 2019. doi:10.2478/popets-2019-0042.
- 3 Alok Aggarwal and S. Vitter, Jeffrey. The Input/Output Complexity of Sorting and Related Problems. *Commun. ACM*, 31(9):1116–1127, September 1988. doi:10.1145/48529.48535.
- 4 M. Ajtai, J. Komlós, and E. Szemerédi. An $O(n \log n)$ sorting network. In *STOC*, 1983.
- 5 Prabhanjan Ananth, Xiong Fan, and Elaine Shi. Towards attribute-based encryption for RAMs from LWE: sub-linear decryption, and more. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 112–141. Springer, 2019.
- 6 Arvind Arasu, Spyros Blanas, Ken Eguro, Raghav Kaushik, Donald Kossmann, Ravishankar Ramamurthy, and Ramarathnam Venkatesan. Orthogonal security with cipherbase. In *CIDR*, 2013.

- 7 Arvind Arasu and Raghav Kaushik. Oblivious query processing. In *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014*, pages 26–37. OpenProceedings.org, 2014.
- 8 Yuriy Arbitman, Moni Naor, and Gil Segev. De-amortized cuckoo hashing: Provable worst-case performance and experimental results. In *ICALP*, 2009.
- 9 Lars Arge, Michael A. Bender, Erik D. Demaine, Bryan Holland-Minkley, and J. Ian Munro. An optimal cache-oblivious priority queue and its application to graph algorithms. *SIAM Journal on Computing*, 36(6):1672–1695, 2007. doi:10.1137/S0097539703428324.
- 10 Gilad Asharov, T.-H. Hubert Chan, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Bucket oblivious sort: An extremely simple oblivious sort. In Martin Farach-Colton and Inge Li Gørtz, editors, *3rd Symposium on Simplicity in Algorithms, SOSA@SODA*, pages 8–14. SIAM, 2020.
- 11 Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, Enoch Peserico, and Elaine Shi. OptORAMa: Optimal Oblivious RAM. In *Advances in Cryptology - EUROCRYPT 2020*, 2020. To appear. See also: <https://eprint.iacr.org/2018/892>.
- 12 Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Enoch Peserico, and Elaine Shi. Oblivious parallel tight compaction. In *Information-Theoretic Cryptography (ITC)*, 2020.
- 13 Albert Atserias, Martin Grohe, and Dániel Marx. Size bounds and query plans for relational joins. In *FOCS*, page 739–748, USA, 2008. IEEE Computer Society.
- 14 Sumeet Bajaj and Radu Sion. Trusteddb: A trusted hardware-based database with privacy and data confidentiality. *IEEE Trans. on Knowl. and Data Eng.*, 26(3):752–765, 2014.
- 15 Victor Balcer and Salil P. Vadhan. Differential privacy on finite computers. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 94, pages 43:1–43:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
- 16 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. *J. ACM*, 64(6):40:1–40:58, 2017.
- 17 Amos Beimel, Kobbi Nissim, and Mohammad Zaheri. Exploring differential obliviousness. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, volume 145 of *LIPICs*, pages 65:1–65:20, 2019.
- 18 Andreas Björklund, Rasmus Pagh, Virginia Vassilevska Williams, and Uri Zwick. Listing Triangles. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 8572 of *Lecture Notes in Computer Science*, pages 223–234. Springer International Publishing, 2014.
- 19 Jeremiah Blocki, Anupam Datta, and Joseph Bonneau. Differentially private password frequency lists. In *NDSS*, 2016.
- 20 Elette Boyle, Kai-Min Chung, and Rafael Pass. Large-scale secure computation: Multi-party computation for (parallel) RAM programs. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 742–762. Springer, 2015.
- 21 Elette Boyle, Kai-Min Chung, and Rafael Pass. Oblivious parallel ram. In *Theory of Cryptography Conference (TCC)*, 2015.
- 22 Mark Bun, Kobbi Nissim, and Uri Stemmer. Simultaneous private learning of multiple concepts. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 369–380, 2016.
- 23 Ran Canetti, Yilei Chen, Justin Holmgren, and Mariana Raykova. Adaptive succinct garbled RAM or: How to delegate your database. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 61–90, 2016.
- 24 David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *ACM CCS*, page 668–679, 2015.

- 25 David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014.
- 26 David Cash, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *CRYPTO*, 2013.
- 27 Hubert Chan, Kai-Min Chung, and Elaine Shi. On the depth of oblivious parallel oram. manuscript, 2017.
- 28 T-H. Hubert Chan, Kai-Min Chung, Bruce M. Maggs, and Elaine Shi. Foundations of differentially oblivious algorithms. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 2448–2467, USA, 2019. Society for Industrial and Applied Mathematics.
- 29 T.-H. Hubert Chan and Elaine Shi. Circuit OPRAM: unifying statistically and computationally secure ORAMs and OPRAMs. In *Theory of Cryptography Conference, (TCC)*, 2017.
- 30 T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *TISSEC*, 14(3):26, 2011.
- 31 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved parameterized upper bounds for vertex cover. In *In MFCS*, 2006.
- 32 Yu-Chi Chen, Sherman S. M. Chow, Kai-Min Chung, Russell W. F. Lai, Wei-Kai Lin, and Hong-Sheng Zhou. Cryptography for parallel RAM from indistinguishability obfuscation. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 179–190. ACM, 2016.
- 33 Shumo Chu, Magdalena Balazinska, and Dan Suciu. From theory to practice: Efficient join query evaluation in a parallel database system. In *SIGMOD Conference*, pages 63–78. ACM, 2015.
- 34 Shumo Chu, Danyang Zhuo, Elaine Shi, and T-H. Hubert Chan (randomized author ordering). Differentially oblivious database joins: Overcoming the worst-case curse of fully oblivious algorithms. Online full version of this paper. Cryptology ePrint Archive, Report 2021/593, 2021. URL: <https://eprint.iacr.org/2021/593>.
- 35 Natacha Crooks, Matthew Burke, Ethan Cecchetti, Sitar Harel, Rachit Agarwal, and Lorenzo Alvisi. Obladi: Oblivious serializable transactions in the cloud. In *OSDI*, page 727–743, USA, 2018. USENIX Association.
- 36 Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *ACM Conference on Computer and Communications Security*, pages 79–88, 2006.
- 37 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.
- 38 Erik D. Demaine. Cache-oblivious algorithms and data structures. In *Lecture Notes from the EEF Summer School on Massive Data Sets*. BRICS, BRICS, University of Aarhus, Denmark, June 27–July 1 2002.
- 39 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.
- 40 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014. URL: <http://dblp.uni-trier.de/db/journals/fttcs/fttcs9.html#DworkR14>.
- 41 Saba Eskandarian and Matei Zaharia. Oblidb: Oblivious query processing for secure databases. *Proc. VLDB Endow.*, 13(2):169–183, 2019.
- 42 J. Flum and M. Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag, Berlin, Heidelberg, 2006.



- 43 Matteo Frigo, Charles E Leiserson, Harald Prokop, and Sridhar Ramachandran. Cache-oblivious algorithms. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 285–297. IEEE, 1999.
- 44 Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database systems - the complete book (2. ed.)*. Pearson Education, 2009.
- 45 Craig Gentry, Shai Halevi, Steve Lu, Rafail Ostrovsky, Mariana Raykova, and Daniel Wichs. Garbled ram revisited. In *EUROCRYPT*, pages 405–422, 2014.
- 46 Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing private ram computation. In *STOC*, 2014.
- 47 O. Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In *STOC*, 1987.
- 48 Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 1996.
- 49 Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nikolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 536–553. Springer, 2013.
- 50 S. Dov Gordon, Jonathan Katz, Vladimir Kolesnikov, Fernando Krell, Tal Malkin, Mariana Raykova, and Yevgeniy Vahlis. Secure two-party computation in sublinear (amortized) time. In *CCS*, 2012.
- 51 Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. Learning to reconstruct: Statistical learning theory and encrypted database attacks. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1067–1083. IEEE, 2019.
- 52 Paul Grubbs, Richard McPherson, Muhammad Naveed, Thomas Ristenpart, and Vitaly Shmatikov. Breaking web applications built on top of encrypted data. In *ACM CCS*, page 1353–1364, 2016.
- 53 Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In *TCC*, volume 7194, pages 339–356, 2012.
- 54 Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, pages 61–70, 2010.
- 55 Xiao Hu and Ke Yi. Instance and output optimal parallel algorithms for acyclic joins. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*, page 450–463, 2019.
- 56 Mohammad Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Network and Distributed System Security Symposium (NDSS)*, 2012.
- 57 Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel C. Rosu, and Michael Steiner. Outsourced symmetric private information retrieval. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.
- 58 Yael Tauman Kalai and Omer Paneth. Delegating RAM computations. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 91–118, 2016.
- 59 Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O’Neill. Generic attacks on secure outsourced databases. In *ACM CCS*, page 1329–1340, 2016.
- 60 Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O’Neill. Accessing data while preserving privacy. *CoRR*, abs/1706.01552, 2017.
- 61 Mahmoud Abo Khamis, Hung Q. Ngo, Christopher Ré, and Atri Rudra. Joins via geometric resolutions: Worst-case and beyond. In *PODS*, pages 213–228. ACM, 2015.
- 62 Ilan Komargodski and Elaine Shi. Differentially oblivious turing machines. In *ITCS*, 2021.

- 63 Evgenios M. Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 1223–1240. IEEE, 2020.
- 64 Aleksandra Korolova, Krishnaram Kenthapadi, Nina Mishra, and Alexandros Ntoulas. Releasing search queries and clicks privately. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 171–180, 2009.
- 65 Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Privatesql: A differentially private sql query engine. *Proc. VLDB Endow.*, 12(11):1371–1384, 2019.
- 66 Ios Kotsogiannis, Yuchao Tao, Ashwin Machanavajjhala, Gerome Miklau, and Michael Hay. Architecting a differentially private SQL engine. In *CIDR*, 2019.
- 67 Wei-Kai Lin, Elaine Shi, and Tiancheng Xie. Can we overcome the $n \log n$ barrier for oblivious sorting? In *SODA*, 2019.
- 68 Sahar Mazloom and S. Dov Gordon. Secure computation with differentially private access patterns. In *CCS*, 2018.
- 69 Shay Moran and Amir Yehudayoff. A note on average-case sorting. *Order*, 33:23–28, 2015.
- 70 Muhammad Naveed, Seny Kamara, and Charles V. Wright. Inference attacks on property-preserving encrypted databases. In *ACM CCS*, page 644–655, 2015.
- 71 Hung Q. Ngo, Dung T. Nguyen, Christopher Ré, and Atri Rudra. Beyond worst-case analysis for joins with minesweeper. In *PODS*, pages 234–245. ACM, 2014.
- 72 Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *J. Algorithms*, 51(2):122–144, 2004.
- 73 Vasilis Pappas, Mariana Raykova, Binh Vo, Steven M. Bellovin, and Tal Malkin. Private search in the real world. In *Annual Computer Security Applications Conference (ACSAC)*, 2011.
- 74 Raluca Ada Popa, Catherine M. S. Redfield, Nikolai Zeldovich, and Hari Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *SOSP*, page 85–100, New York, NY, USA, 2011. Association for Computing Machinery.
- 75 Vijaya Ramachandran and Elaine Shi. Data oblivious algorithms for multicores. <https://eprint.iacr.org/2020/947.pdf>, 2020.
- 76 Vijaya Ramachandran and Elaine Shi. Data oblivious algorithms for multicores. In *SPAA*, 2021.
- 77 Tim Roughgarden. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020.
- 78 Leonard D. Shapiro. Join processing in database systems with large main memories. *ACM Trans. Database Syst.*, 11(3):239–264, 1986.
- 79 Micha Sharir and Mark H. Overmars. A simple output-sensitive algorithm for hidden surface removal. *ACM Trans. Graph.*, 11(1):1–11, 1992.
- 80 Elaine Shi, John Bethencourt, T-H. Hubert Chan, Dawn Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, pages 350–364, 2007.
- 81 Elaine Shi, T.-H. Hubert Chan, Emil Stefanov, and Mingfei Li. Oblivious RAM with $O((\log N)^3)$ worst-case cost. In *ASIACRYPT*, 2011.
- 82 Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, Washington, DC, USA, 2000. IEEE Computer Society.
- 83 Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. Practical dynamic searchable symmetric encryption with small leakage. In *Network and Distributed System Security Symposium (NDSS)*, 2014.

19:24 Differentially Oblivious Database Joins

- 84 Ananda Theertha Suresh. Differentially private anonymized histograms. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *NeurIPS*, pages 7969–7979, 2019.
- 85 Salil Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pages 347–450, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-57048-8_7.
- 86 Jeffrey Scott Vitter. External Memory Algorithms and Data Structures: Dealing with Massive Data. *ACM Comput. Surv.*, 33(2):209–271, June 2001. doi:10.1145/384192.384193.
- 87 Sameer Wagh, Paul Cuff, and Prateek Mittal. Differentially private oblivious RAM. *PoPETs*, 2018(4):64–84, 2018.
- 88 Xiao Shaun Wang, T-H. Hubert Chan, and Elaine Shi. Circuit ORAM: On Tightness of the Goldreich-Ostrovsky Lower Bound. In *CCS*, 2015.
- 89 Mihalis Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, page 82–94, 1981.
- 90 Wenting Zheng, Ankur Dave, Jethro G. Beekman, Raluca Ada Popa, Joseph E. Gonzalez, and Ion Stoica. Opaque: An oblivious and encrypted distributed analytics platform. In *NSDI*, 2017.

Communication Complexity of Private Simultaneous Quantum Messages Protocols

Akinori Kawachi   

Graduate School of Engineering, Mie University, Tsu, Japan

Harumichi Nishimura  

Graduate School of Informatics, Nagoya University, Japan

Institute for Advanced Study, Nagoya University, Japan

Abstract

The private simultaneous messages (PSM) model is a non-interactive version of the multiparty secure computation (MPC), which has been intensively studied to examine the communication cost of the secure computation. We consider its quantum counterpart, the *private simultaneous quantum messages (PSQM)* model, and examine the advantages of quantum communication and prior entanglement of this model.

In the PSQM model, k parties P_1, \dots, P_k initially share a common random string (or entangled states in a stronger setting), and they have private classical inputs x_1, \dots, x_k . Every P_i generates a quantum message from the private input x_i and the shared random string (entangled states), and then sends it to the referee R . Receiving the messages from the k parties, R computes $F(x_1, \dots, x_k)$ from the messages. Then, R learns nothing except for $F(x_1, \dots, x_k)$ as the privacy condition.

We obtain the following results for this PSQM model. (i) We demonstrate that the privacy condition inevitably increases the communication cost in the two-party PSQM model as well as in the classical case presented by Applebaum, Holenstein, Mishra, and Shayevitz [*Journal of Cryptology* 33(3), 916–953 (2020)]. In particular, we prove a lower bound $(3 - o(1))n$ of the communication complexity in PSQM protocols with a shared random string for random Boolean functions of $2n$ -bit input, which is larger than the trivial upper bound $2n$ of the communication complexity without the privacy condition. (ii) We demonstrate a factor two gap between the communication complexity of PSQM protocols with shared entangled states and with shared random strings by designing a multiparty PSQM protocol with shared entangled states for a total function that extends the two-party equality function. (iii) We demonstrate an exponential gap between the communication complexity of PSQM protocols with shared entangled states and with shared random strings for a two-party *partial* function.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Computational complexity and cryptography

Keywords and phrases Communication complexity, private simultaneous messages, quantum protocols, secure multi-party computation

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.20

Funding *Akinori Kawachi*: JSPS Grant-in-Aid for Scientific Research (A) Nos. 16H01705, 21H04879, (B) No. 17H01695, JSPS Grant-in-Aid for Young Scientists (B) No. 17K12640, and MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) Grant Number JPMXS0120319794.

Harumichi Nishimura: JSPS Grant-in-Aid for Scientific Research (A) Nos. 16H01705, 21H04879, (B) No. 19H04066, Grant-in-Aid for Transformative Research Areas No. 20H05966 and MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) Grant Number JPMXS0120319794.

Acknowledgements We thank the anonymous reviewers of ITC 2021 for helpful comments.

1 Introduction

Background. Communication complexity has been an important research area in theoretical computer science for more than four decades, aiming to understand the communication cost of computing functions in a distributed manner [31, 25]. Since the advent of quantum information science, quantum communication complexity has also been studied intensively to



© Akinori Kawachi and Harumichi Nishimura;
licensed under Creative Commons License CC-BY 4.0
2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 20; pp. 20:1–20:19



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

determine the advantage of quantum information processing over its classical counterparts. A number of studies have succeeded in demonstrating the quantum advantages from the early days of quantum complexity theory [9, 28, 8].

Recently, much attention has been given to studying the amount of communication overhead required to preserve privacy in the field of cryptography, particularly, multi-party secure computation (MPC), to explore the optimal communication cost for privacy from the viewpoint of communication complexity [12, 11]. MPC is commonly based on a general network model that has complex communication patterns (e.g., in which each of many parties can freely interact with the other parties bidirectionally) unlike standard models in communication complexity (e.g., in which two parties can exchange messages only with each other). Therefore, many studies have focused on a special class of MPC that has simpler communication patterns, such as *private simultaneous messages* (PSM) protocols [14, 21, 4, 5, 1].

The two-party version of the PSM model was first proposed by Feige, Kilian, and Naor [14], and was later extended by Ishai and Kushilevitz [21]. In the general setting of the PSM model, we consider k parties P_1, \dots, P_k and a unique referee R . The party P_i has its private input x_i , and all parties share a common random string r . Each P_i generates message m_i from x_i and r , and then, sends m_i to the referee R only once. Note that each party is not allowed to interact with other parties. The referee R receives the messages m_1, \dots, m_k , and computes an output value of a predetermined function F . The protocol generally has two properties *correctness* and *privacy*. Correctness signifies that the referee can compute $F(x_1, \dots, x_k)$ correctly from the messages m_1, \dots, m_k , while privacy signifies that the referee R learns nothing except for $F(x_1, \dots, x_k)$ from the received messages m_1, \dots, m_k in the information-theoretical sense.

In fact, the communication model of PSM protocols coincides with simultaneous message passing (SMP) protocols, which are known as traditional communication models in communication complexity [31, 25]. In the (number-in-hand) SMP model, k parties P_1, \dots, P_k that share a common random string r (and sometimes entangled states), send their messages m_1, \dots, m_k computed from individual inputs x_1, \dots, x_k and the referee computes $F(x_1, \dots, x_k)$ from m_1, \dots, m_k , as performed in the PSM model. Note that the SMP model does not require the privacy condition unlike the PSM model. The communication complexity of SMP protocols has been widely studied from the viewpoint of classical/quantum information to demonstrate the power of quantum communication [8, 18, 16].

Two-party quantum SMP models were first studied by Buhrman, Cleve, Watrous, and de Wolf [8] in the setting that the two parties do not share any randomness or entanglement. In this model, they demonstrated that the quantum communication complexity of the equality function is exponentially smaller than in the classical case. This result has been strengthened in the literature [3, 15], and Gavinsky [16] demonstrated that there is a relational problem whose quantum communication complexity is exponentially smaller than that of the two-way classical communication model. However, the power of shared entanglement in the SMP model is unclear. In one of the few related studies, Gavinsky, Kempe, Regev, and de Wolf [18] demonstrated that there is a relational problem that has an exponential gap between quantum SMP models with shared entanglement and without shared entanglement. However, the known maximum gap between them for *total functions* is only a constant multiplicative factor of 2 [20, 22].

Although various studies have examined quantum versions of MPC so far (e.g., [10, 29, 13]), to the best of the authors' knowledge, there has been no attempt to analyze quantum communication complexity under the privacy condition in a cryptographic setting, and such analysis is important to understand the advantages of quantum communication in a cryptographic setting.

Contributions. In this paper, we examine the power of quantum communication and shared entanglement under the information-theoretical privacy condition based on a standard communication model, namely, the PSM (or, equivalently, SMP) model. In particular, we propose a quantum counterpart of the classical PSM model called *private simultaneous quantum messages (PSQM)* model. In the PSQM model, parties P_1, \dots, P_k which have classical private inputs x_1, \dots, x_k share a common random string or entangled states in advance, and can send quantum messages to a quantum referee, R . Then, R computes a classical output value $F(x_1, \dots, x_k)$ for a given function F .

In the PSM (and its related) model, there are few results on lower bounds of communication complexity [1, 12, 2]. As one of such results, Applebaum, Holenstein, Mishra, and Shayevitz [1] proved a lower bound $(3 - o(1))n$ of the communication complexity for random functions $F_n : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ in the PSM model. In contrast, every function has the trivial upper bound $2n$ in the SMP model (i.e., the PSM model without the privacy condition). This result implies that the privacy condition creates communication overhead in the PSM model for some functions. Our first result demonstrates that this communication overhead is inevitable even if the parties can send quantum messages as in the PSQM model.

► **Theorem 1.** *For a $(1 - o(1))$ fraction of functions $F_n : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$, the communication complexity of two-party PSQM protocols with shared randomness for F_n is at least $3n - 2 \log n - O(1)$.*

We also present a multiparty PSQM protocol for a total function that reduces the amount of quantum communication by half under the condition that the parties share entanglement compared to the case in which they do not share entanglement.

► **Theorem 2.** *For any even n and k , there is a total function $F_n : (\{0, 1\}^n)^k \rightarrow \{0, 1\}$ such that the communication complexity of the k -party PSQM protocol with shared entanglement is at most $kn/2$, while that without shared entanglement is kn .*

Actually, this function matches the equality function for the two-party case. It is known that for the equality function, the two-party quantum SMP model with shared entanglement reduces the amount of quantum communication by half compared to the corresponding model without shared entanglement (e.g. [20]). Our result implies that this reduction still holds even if the privacy condition is required.

Moreover, we present a two-party PSQM protocol with shared entanglement for a *partial* function that reduces the amount of quantum communication exponentially compared to the case in which the parties do not share entanglement.

► **Theorem 3.** *There is a partial function $F_n : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ such that the communication complexity of the PSQM protocol with shared entanglement is $O(\log n)$ while that without entanglement is $\Omega(n)$.*

Related Work. There have been several studies on quantum communication complexity with privacy conditions (e.g., [23, 17]), although they differed from a cryptographic setting. For example, Gavinsky and Ito [17] considered the SMP model with privacy; however it considered the information leakage of quantum messages when the input was randomly chosen, while in the cryptographic setting, privacy should be retained for any input.

In a study related to the PSQM model, Brakerski and Yuen [6] constructed a quantum version of decomposable randomized encoding schemes. In fact, decomposable randomized encoding is equivalent to the PSM model from a communication-complexity perspective. They demonstrated how to garble a general quantum circuit on quantum inputs in a

decomposable manner via a constant-depth quantum circuit. In contrast, our study focuses on the communication complexity of computing several classical functions on classical inputs in the communication model.

More recently, Morimae [26] investigated relationships between quantum randomized encoding and other quantum protocols including quantum computing verification and blind quantum computing. For example, he proved that a randomized encoding scheme of the BB84 state generation implies a two-round verification scheme of quantum computing with a classical verifier that additionally performs the encoding operation, and that a quantum randomized encoding scheme with a classical encoding operation implies violation of the no-cloning theorem. His target of quantum randomized encoding schemes is similar to that of [6], that is, encoding for quantum circuits on quantum inputs rather than classical functions on classical inputs.

2 Preliminaries

Let $[n] := \{1, 2, \dots, n\}$. For any two m -bit strings $x = x_1 \cdots x_m$ and $y = y_1 \cdots y_m$, the product $x \cdot y$ denotes $\sum_{i \in [m]} x_i y_i \pmod{2}$, and $x \oplus y$ denotes the m -bit string whose i th bit is the XOR of x_i and y_i .

For any m -bit string $x = x_1 x_2 \cdots x_m$, let

$$\mathfrak{p}(x) = x_1 + x_2 \alpha + \cdots + x_m \alpha^{m-1} \pmod{\mathfrak{q}_m} \quad (1)$$

be the corresponding polynomial over \mathbb{F}_2 , where \mathfrak{q}_m is some irreducible polynomial of degree m over \mathbb{F}_2 . Note that $\mathfrak{p}(x)$ is regarded as an element in \mathbb{F}_{2^m} , and \mathfrak{p} is a one-to-one correspondence between $\{0, 1\}^m \setminus \{0^m\}$ and the multiplicative group $\mathbb{F}_{2^m}^*$.

We assume the reader is familiar with the basics of quantum information and computation such as quantum states and quantum operations (see, e.g., [19, 27]). According to the standard notations, Pauli gates X , Z , and the Hadamard gate H denote

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

respectively.

2.1 Private simultaneous quantum messages protocols

Private simultaneous quantum messages (PSQM) protocols are formally defined as follows.

► **Definition 4** (private simultaneous quantum messages (PSQM) protocols). *For positive integers $n, k > 0$, let n be the size parameter and k be the number of parties. Let $F_n : \prod_{i=1}^k \mathcal{X}_{n,i} \rightarrow \mathcal{Y}_n$. We say that a $(k+1)$ -tuple $\Pi = (P_1, \dots, P_k, R)$ of quantum algorithms is an ε -error k -party private simultaneous quantum messages (PSQM) protocol if the following holds: given an individual input $x_i \in \mathcal{X}_{n,i}$ and shared random string r among P_1, \dots, P_k , the i th party P_i prepares a quantum message, represented as $\rho_i = P_i(x_i, r)$, in a Hilbert space $\mathcal{M}_{n,i}$ called a quantum register, and sends $\mathcal{M}_{n,i}$ (or equivalently ρ_i) to the party R , which is called the referee. Then, the following two properties hold:*

1. **(Correctness)** *The referee R outputs the classical value $F_n(x_1, \dots, x_k) \in \mathcal{Y}_n$ using the received joint quantum register $\mathcal{M}_n := \bigotimes_{i=1}^k \mathcal{M}_{n,i}$ with a probability of at least $1 - \varepsilon$.*
2. **((Perfect) Privacy)** *There exists a quantum algorithm S_n , which is called the simulator, such that the output quantum state $S_n(F_n(x_1, \dots, x_k))$ is identical to the quantum state in \mathcal{M}_n (before R), namely, $\bigotimes_{i=1}^k \rho_i$.*

We say that the protocol is exact when $\varepsilon = 0$.

If the shared random string r is replaced by a predetermined multipartite entangled quantum state $|\Phi\rangle$ among the k parties, we say that Π is a PSQM protocol with a shared entangled state $|\Phi\rangle$, where the algorithms and the properties are similarly defined except that P_i prepares the message using its own part of $|\Phi\rangle$ (instead of r), and that the quantum state in \mathcal{M}_n is not a product state of the k local states in $\mathcal{M}_{n,1}, \dots, \mathcal{M}_{n,k}$ any more.

The communication complexity of Π is defined by the total length $\log \dim(\mathcal{M}_n)$ of the messages.

Let $C_\varepsilon^{psm}(F_n)$ (resp. $Q_\varepsilon^{psm}(F_n)$) be the ε -error classical (quantum) communication complexity of the problem F_n in the PSM (PSQM) model with a shared random string. Let $C_\varepsilon^{psm,*}(F_n)$ (resp. $Q_\varepsilon^{psm,*}(F_n)$) be the ε -error classical (quantum) communication complexity of F_n in the PSM (PSQM) model with shared entangled states (the PSM model with shared entangled states is defined similarly to the PSQM model with shared entangled states except that the messages sent to the referee are restricted to classical strings).

3 Communication Lower Bounds of Two-Party PSQM Protocols

In this section, we present the communication lower bounds of random functions for two-party PSQM protocols (Theorem 1).

The proof strategy is based on that of the classical case presented by Applebaum et al. [1]. The proof for the classical case considers two independent executions of a PSM protocol. It then evaluated the upper bounds of the collision probability, that is, the probability that the message in the first execution coincides with the one in the second execution, between two independent random messages. Because the collision probability is lower-bounded by the inverse of the size of the message domain, we can obtain the communication lower bound from the upper bound of the collision probability. Note that this argument is not available for quantum messages since they vary infinitely even over a finite number of qubits.

In order to extend the above argument to the case of quantum messages, we use the *purity*, $\text{tr}\rho^2$, of a quantum message ρ in a PSQM protocol instead of the collision probability. In accordance with its name, the purity is originally a measure of how pure a quantum state is. (For example, any pure state has a purity of 1, and the d -dimensional maximally mixed state has $1/d$.) It is easy to see that the purity of a quantum state ρ is lower-bounded by $1/\dim(\rho)$, and thus, we can obtain the communication lower bounds for a PSQM protocol by evaluating the upper bound of the purity of the quantum messages, similarly to the collision probability for a PSM protocol.

However, the purity of quantum messages is different from the collision probability between classical messages; thus, we must further adapt the proof technique in [1] to the purity. For example, while the collision probability is analyzed by combinatorial techniques in the proof of [1], we need to analyze the trace $\text{tr}\rho^2$ combinatorially by extending the original proof (Claim 7). Also, the proof technique in [1] uses a unique collision (which is obtained from the property called non-degeneracy that random functions have with high probability) between two messages in two independent executions with any fixed shared random string. Instead of the unique collision, we consider weighted collisions defined from the inner product of two quantum messages and extend the original argument for the weighted collisions (Lemma 9).

Before discussing the details of the proof, we provide several technical definitions and notation required for the proof of the lower bounds. In this section, we denote $\mathcal{X}_{n,i}$ by \mathcal{X}_i . We use $\rho(x_1, x_2; r) = \rho_1(x_1; r) \otimes \rho_2(x_2; r)$ to the entire quantum message sent from P_1 and P_2 on individual inputs $x_1 \in \mathcal{X}_1$ and $x_2 \in \mathcal{X}_2$ with a shared random string r to R , where $\rho_1(x_1; r)$ denotes P_1 's message and $\rho_2(x_2; r)$ denotes P_2 's message.

20:6 Communication Complexity of PSQM Protocols

Let μ be a distribution over $\mathcal{X}_1 \times \mathcal{X}_2$ with marginal distributions μ_1 and μ_2 . We define $\text{Supp}(\mu)$ for a distribution μ as a set $\{x : \Pr_{X \sim \mu}[X = x] > 0\}$. We say that function F_n is non-degenerate under distribution μ if for every distinct $x_1 \in \text{Supp}(\mu_1)$ and $x'_1 \in \text{Supp}(\mu_1)$, there exists $x_2 \in \text{Supp}(\mu_2)$ such that $F_n(x_1, x_2) \neq F_n(x'_1, x_2)$ and for every distinct $x_2 \in \text{Supp}(\mu_2)$ and $x'_2 \in \text{Supp}(\mu_2)$ there exists x_1 such that $F_n(x_1, x_2) \neq F_n(x_1, x'_2)$. We say that F_n is non-degenerate if the above holds when replacing $\text{Supp}(\mu_1)$ and $\text{Supp}(\mu_2)$ by \mathcal{X}_1 and \mathcal{X}_2 , respectively.

A rectangle \mathcal{R} of size $k \times \ell$ over $\mathcal{X}_1 \times \mathcal{X}_2$ is defined as $((x_{1,1}, \dots, x_{1,k}), (x_{2,1}, \dots, x_{2,\ell}))$, where $x_{1,i} \in \mathcal{X}_1, x_{2,j} \in \mathcal{X}_2$ for every i, j , $x_{1,i} \neq x_{1,i'}$ for every distinct i, i' , and $x_{2,j} \neq x_{2,j'}$ for every distinct j, j' . We say that two rectangles $\mathcal{R} = ((x_{1,1}, \dots, x_{1,k}), (x_{2,1}, \dots, x_{2,\ell}))$ and $\mathcal{R}' = ((x'_{1,1}, \dots, x'_{1,k}), (x'_{2,1}, \dots, x'_{2,\ell}))$ are \mathcal{X}_1 -disjoint (resp. \mathcal{X}_2 -disjoint) if $x_{1,i} \neq x'_{1,i}$ for every $i \in [k]$ (resp. if $x_{2,j} \neq x'_{2,j}$ for every $j \in [\ell]$). In particular, we say that \mathcal{R} and \mathcal{R}' are disjoint if they are either \mathcal{X}_1 -disjoint or \mathcal{X}_2 -disjoint.

For a rectangle $\mathcal{R} = ((x_{1,1}, \dots, x_{1,k}), (x_{2,1}, \dots, x_{2,\ell}))$, let $F_n[\mathcal{R}]$ be a matrix whose (i, j) -entry is $F_n(x_{1,i}, x_{2,j})$, and let $\mu(\mathcal{R}) = \sum_{i \in [k], j \in [\ell]} \mu(x_{1,i}, x_{2,j})$. We say that \mathcal{R} is similar to \mathcal{R}' if $F_n[\mathcal{R}] = F_n[\mathcal{R}']$. We define

$$\alpha(\mu) := \max_{(\mathcal{R}_1, \mathcal{R}_2)} \min\{\mu(\mathcal{R}_1), \mu(\mathcal{R}_2)\},$$

where the maximum ranges over all pairs of similar disjoint rectangles $(\mathcal{R}_1, \mathcal{R}_2)$. In addition,

$$\beta(\mu) := \min_y \Pr_{(X_1, X_2), (X'_1, X'_2) \sim \mu} [(X_1, X_2) \neq (X'_1, X'_2) \mid F_n(X_1, X_2) = F_n(X'_1, X'_2) = y],$$

where (X_1, X_2) and (X'_1, X'_2) are independent.

We can demonstrate the communication lower bound in Theorem 1 from the following main technical lemma combined with an appropriate function F_n , which is provided in the study by Applebaum et al. [1].

► **Lemma 5.** *For every non-degenerate function $F_n : \mathcal{X}_1 \times \mathcal{X}_2 \rightarrow \{0, 1\}$, we have*

$$Q_0^{psm}(F_n) \geq \max_{\mu} (\log(\alpha(\mu)^{-1}) + H_{\infty}(\mu) - \log(\beta(\mu)^{-1})) - 1,$$

where μ is taken over all distributions over $\mathcal{X}_1 \times \mathcal{X}_2$ under which F_n is non-degenerate, and $H_{\infty}(\mu)$ is the min-entropy of μ .

From the previous study [1], we can obtain the appropriate function by selecting a function at random, as illustrated in the following theorem.

► **Theorem 6** (Applebaum et al. [1]). *For a $(1 - o(1))$ fraction of the functions $F_n : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$, F_n is non-degenerate and $|\mathcal{R}| \leq 2^n \cdot n^2$ holds for every pair $(\mathcal{R}, \mathcal{R}')$ of similar disjoint rectangles.*

Considering the uniform distribution U over $\{0, 1\}^n \times \{0, 1\}^n$, the communication lower bound from Lemma 5 of PSQM protocols for $F_n : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ is bounded by $\log(\alpha(U)^{-1}) + H_{\infty}(U) - \log(\beta(U)^{-1}) - 1$. By Theorem 6, we can easily see that this bound is $3n - 2 \log n - O(1)$ for a $(1 - o(1))$ fraction of the functions $\{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$, as given in Theorem 1.

Now, we provide the proof of the main technical lemma.

Proof of Lemma 5. From the correctness, the referee R outputs $F_n(x_1, x_2)$ for the received quantum message $\rho(x_1, x_2; r)$ for every x_1, x_2 and every r . Without loss of generality, we can assume that P_1 and P_2 generate pure states $|\psi_1(x_1; r)\rangle$ and $|\psi_2(x_2; r)\rangle$ with a shared randomness r , respectively.

This assumption is justified as follows. Suppose that P_1 and P_2 generate mixed states $\rho_1(x_1; r)$ and $\rho_2(x_2; r)$ as their messages on inputs x_1, x_2 and a shared random string r . By the spectral decomposition, we have $\rho_1(x_1; r) = \sum_i p_i |\psi_1^{(i)}(x_1; r)\rangle\langle\psi_1^{(i)}(x_1; r)|$ and $\rho_2(x_2; r) = \sum_j q_j |\psi_2^{(j)}(x_2; r)\rangle\langle\psi_2^{(j)}(x_2; r)|$. The joint message state is

$$\rho(x_1, x_2; r) := \sum_{i,j} p_i q_j |\psi_1^{(i)}(x_1; r), \psi_2^{(j)}(x_2; r)\rangle\langle\psi_1^{(i)}(x_1; r), \psi_2^{(j)}(x_2; r)|$$

and its probabilistic mixture over the shared random string r is

$$\rho(x_1, x_2) := \sum_r \pi(r) \rho(x_1, x_2; r) = \sum_{i,j,r} p_i q_j \pi(r) |\psi_1^{(i)}(x_1; r), \psi_2^{(j)}(x_2; r)\rangle\langle\psi_1^{(i)}(x_1; r), \psi_2^{(j)}(x_2; r)|.$$

Rephrasing the probability distribution $\{\pi(r)p_i q_j\}_{i,j,r}$ and the pure message states $|\psi_1^{(i)}(x_1; r)\rangle, |\psi_2^{(j)}(x_2; r)\rangle$ to $\{\pi(r)\}_r$ and $|\psi_1(x_1; r)\rangle, |\psi_2(x_2; r)\rangle$ respectively, we can assume that they generate pure states as their messages from the beginning.

We denote by $|\psi(x_1, x_2; r)\rangle$ their joint state $|\psi_1(x_1; r)\rangle \otimes |\psi_2(x_2; r)\rangle$. Namely, we set

$$\rho(x_1, x_2; r) := |\psi(x_1, x_2; r)\rangle\langle\psi(x_1, x_2; r)| = |\psi_1(x_1; r)\rangle\langle\psi_1(x_1; r)| \otimes |\psi_2(x_2; r)\rangle\langle\psi_2(x_2; r)|. \quad (2)$$

In addition, we set

$$\rho(x_1, x_2) := \sum_r \pi(r) \rho(x_1, x_2; r), \quad (3)$$

where $\pi(r)$ denotes the probability that r is selected as a shared random string under the uniform distribution.

As mentioned above, we use the purity as a collision measure of quantum messages in order to obtain lower bounds of quantum message length from upper bounds of the purity. We set $\rho := \sum_{x_1, x_2} \mu(x_1, x_2) \rho(x_1, x_2)$. We then have

$$\begin{aligned} \frac{1}{\dim(\mathcal{M})} &\leq \text{tr} \rho^2 = \text{tr} \left(\sum_{x_1, x_2} \mu(x_1, x_2) \rho(x_1, x_2) \right)^2 \\ &= \text{tr} \sum_{x_1, x_2, x'_1, x'_2} \mu(x_1, x_2) \rho(x_1, x_2) \mu(x'_1, x'_2) \rho(x'_1, x'_2). \end{aligned} \quad (4)$$

Then, the purity of ρ is upper-bounded as follows.

▷ **Claim 7.**

$$\text{tr} \rho^2 \leq \beta(\mu)^{-1} \text{tr} \sum_{(x_1, x_2) \neq (x'_1, x'_2)} \mu(x_1, x_2) \rho(x_1, x_2) \mu(x'_1, x'_2) \rho(x'_1, x'_2).$$

The proof of this claim is done by a combinatorial analysis of the trace as a quantum counterpart of the analysis of the collision probability in [1]. The detailed proof will be given in Appendix A.

Next, we consider an upper bound of

$$\text{tr} \sum_{(x_1, x_2) \neq (x'_1, x'_2)} \mu(x_1, x_2) \rho(x_1, x_2) \mu(x'_1, x'_2) \rho(x'_1, x'_2). \quad (5)$$

Actually, we show that for every r and r' ,

$$\text{tr} \sum_{(x_1, x_2) \neq (x'_1, x'_2)} \mu(x_1, x_2) \rho(x_1, x_2; r) \mu(x'_1, x'_2) \rho(x'_1, x'_2; r')$$

is at most $2\alpha(\mu)2^{-H_\infty(\mu)}$ as this implies that Eq. (5) is also at most $2\alpha(\mu)2^{-H_\infty(\mu)}$ (by Eq. (3)). This completes the proof of Lemma 5 by Claim 7 and Eq. (4).

Now we fix any r and r' . From Eq. (2) and the union bound, we have

$$\begin{aligned} & \text{tr} \sum_{(x_1, x_2) \neq (x'_1, x'_2)} \mu(x_1, x_2) \rho(x_1, x_2; r) \mu(x'_1, x'_2) \rho(x'_1, x'_2; r') \\ &= \sum_{(x_1, x_2) \neq (x'_1, x'_2)} \mu(x_1, x_2) \mu(x'_1, x'_2) |\langle \psi_1(x_1; r) | \psi_1(x'_1; r') \rangle|^2 \cdot |\langle \psi_2(x_2; r) | \psi_2(x'_2; r') \rangle|^2. \\ &\leq \sum_{x_1 \neq x'_1, x_2, x'_2} \mu(x_1, x_2) \mu(x'_1, x'_2) |\langle \psi_1(x_1; r) | \psi_1(x'_1; r') \rangle|^2 \cdot |\langle \psi_2(x_2; r) | \psi_2(x'_2; r') \rangle|^2 \\ &\quad + \sum_{x_1, x'_1, x_2 \neq x'_2} \mu(x_1, x_2) \mu(x'_1, x'_2) |\langle \psi_1(x_1; r) | \psi_1(x'_1; r') \rangle|^2 \cdot |\langle \psi_2(x_2; r) | \psi_2(x'_2; r') \rangle|^2. \end{aligned} \quad (6)$$

It suffices to show that the first term of the right-hand of Eq. (6) is at most $\alpha(\mu)2^{-H_\infty(\mu)}$ from the symmetry of P_1 and P_2 .

Note that we can regard the referee as a POVM $R = \{R_y\}_{y \in \{0,1\}}$. The following claim demonstrates that the referee is projective in the two-party PSQM setting. (Its proof will be given in Appendix A.)

▷ **Claim 8.** The referee $R = \{R_y\}_{y \in \{0,1\}}$ is a PVM.

In the classical case of [1], Applebaum et al. used the fact that for every x_1 and every r, r' there exists at most one z such that $|\psi_1(x_1; r)\rangle = |\psi_1(z; r')\rangle$ if $|\psi_1(x_1; r)\rangle, |\psi_1(z; r')\rangle$ are classical; that is, either $\langle \psi_1(x_1; r) | \psi_1(z; r') \rangle = 0$ or $\langle \psi_1(x_1; r) | \psi_1(z; r') \rangle = 1$, which can be derived from the non-degeneracy of F_n . However, we cannot demonstrate the same fact for quantum messages. Instead, we can prove the following relaxed version of the fact for quantum messages.

► **Lemma 9.** *If F_n is non-degenerate, we have*

$$\sum_{z \neq x_1} |\langle \psi_1(x_1; r) | \psi_1(z; r') \rangle|^2 \leq 1$$

for every r, r' and every x_1 . Similarly

$$\sum_z |\langle \psi_2(x_2; r) | \psi_2(z; r') \rangle|^2 \leq 1$$

for every r, r' and every x_2 .

The proof of this lemma will be given in Appendix A.

Let $w_1(x_1, x'_1) := |\langle \psi_1(x_1; r) | \psi_1(x'_1; r') \rangle|^2$ and let $w_2(x_2, x'_2) := |\langle \psi_2(x_2; r) | \psi_2(x'_2; r') \rangle|^2$. We say that x_1 collides with x'_1 if $w_1(x_1, x'_1) > 0$. Similarly, we say that x_2 collides with x'_2 if $w_2(x_2, x'_2) > 0$.

Now, our final goal is to upper-bound

$$\sum_{x_1 \neq x'_1, x_2, x'_2} \mu(x_1, x_2) \mu(x'_1, x'_2) w_1(x_1, x'_1) w_2(x_2, x'_2). \quad (7)$$

Let $C(x_1)$ be the set of the elements in \mathcal{X}_1 with which x_1 collides except for x_1 itself. Similarly, let $C(x_2)$ be the set of the elements in \mathcal{X}_2 with which x_2 collides (note that $C(x_2)$ may contain x_2).

Let $\mathbf{x}_1 := (x_1 : C(x_1) \neq \emptyset)$ with an arbitrary (e.g., lexicographical) order in \mathcal{X}_1 . We denote $\mathbf{x}_1 = (u_1, u_2, \dots, u_k)$. Then, we select any element $\mathbf{x}_1' = (u'_1, u'_2, \dots, u'_k)$ from $C(u_1) \times \dots \times C(u_k)$. Note that u_i collides with u'_i and $u_i \neq u'_i$ for every i . Similarly, let $\mathbf{x}_2 := (x_2 : C(x_2) \neq \emptyset) = (v_1, v_2, \dots, v_\ell)$ with an arbitrary order in \mathcal{X}_2 , and we then select any element $\mathbf{x}_2' = (v'_1, v'_2, \dots, v'_\ell)$ from $C(v_1) \times \dots \times C(v_\ell)$.

Then, we can show that for every choice of \mathbf{x}_1' and \mathbf{x}_2' , we have

$$\sum_{i,j} \mu(u_i, v_j) \mu(u'_i, v'_j) \leq \alpha(\mu) 2^{-H_\infty(\mu)}.$$

The reason is as follows. We consider two rectangles $\mathcal{R} := (\mathbf{x}_1, \mathbf{x}_2)$ and $\mathcal{R}' := (\mathbf{x}'_1, \mathbf{x}'_2)$. We observe that $R(|\psi_1(x_1; r)\rangle|\psi_2(x_2; r)\rangle) = R(|\psi_1(x'_1; r')\rangle|\psi_2(x'_2; r')\rangle)$ (where $R(|\varphi\rangle)$ denotes the classical value that R outputs on input $|\varphi\rangle$) if x_1 collides with x'_1 and x_2 collides with x'_2 from the perfect correctness. Therefore, \mathcal{X} -disjoint rectangles \mathcal{R} and \mathcal{R}' are similar; that is, $F_n[\mathcal{R}] = F_n[\mathcal{R}']$. Without loss of generality, we can assume that $\mu(\mathcal{R}) \leq \mu(\mathcal{R}')$. Hence, we have $\mu(\mathcal{R}) \leq \alpha(\mu)$. Thus, we can see that for random variables X_1, X_2, X'_1, X'_2

$$\begin{aligned} \sum_{i,j} \mu(u_i, v_j) \mu(u'_i, v'_j) &= \sum_{i,j} \Pr [(X_1, X_2) = (u_i, v_j) \wedge (X'_1, X'_2) = (u'_i, v'_j)] \\ &\leq \max_{(x_1, x_2)} \mu(x_1, x_2) \sum_{i,j} \Pr [(X_1, X_2) = (u_i, v_j)] \\ &\leq 2^{-H_\infty(\mu)} \alpha(\mu). \end{aligned}$$

Furthermore, it holds for every i, j that

$$\begin{aligned} \sum_{u'_i \in C(u_i), v'_j \in C(v_j)} w_1(u_i, u'_i) w_2(v_j, v'_j) &= \sum_{u'_i \in C(u_i)} w_1(u_i, u'_i) \left(\sum_{v'_j \in C(v_j)} w_2(v_j, v'_j) \right) \\ &\leq \sum_{u'_i \in C(u_i)} w_1(u_i, u'_i) \leq 1 \end{aligned}$$

from Lemma 9. Thus, we have

$$\begin{aligned} &\sum_{x_1 \neq x'_1, x_2, x'_2} \mu(x_1, x_2) \mu(x'_1, x'_2) w_1(x_1, x'_1) w_2(x_2, x'_2) \\ &= \sum_{i,j} \sum_{u'_i \in C(u_i), v'_j \in C(v_j)} \mu(u_i, v_j) \mu(u'_i, v'_j) w_1(u_i, u'_i) w_2(v_j, v'_j) \\ &\leq \sum_{i,j} \mu(u_i, v_j) \mu(\hat{u}_i, \hat{v}_j) \sum_{u'_i \in C(u_i), v'_j \in C(v_j)} w_1(u_i, u'_i) w_2(v_j, v'_j) \\ &\leq \sum_{i,j} \mu(u_i, v_j) \mu(\hat{u}_i, \hat{v}_j) \\ &\leq \alpha(\mu) 2^{-H_\infty(\mu)}, \end{aligned}$$

where $\mu(\hat{u}_i, \hat{v}_j) = \max_{u'_i \in C(u_i), v'_j \in C(v_j)} \mu(u'_i, v'_j)$. Eventually, an upper bound of Eq. (7) is $\alpha(\mu) 2^{-H_\infty(\mu)}$. \blacktriangleleft

4 Power of Shared Entanglement for Total Functions

In this section, we prove Theorem 2, which implies a factor two gap between PSQMs with shared entanglement and without shared entanglement for a total function. The main part of Theorem 2 provides a k -party PSQM protocol for a total function $GEQ_{2l} : (\{0, 1\}^{2l})^k \rightarrow \{0, 1\}$ defined by

$$GEQ_{2l}(x_1, x_2, \dots, x_k) = \begin{cases} 1 & (\sum_{j=1}^k x_j^1 = \sum_{j=1}^k x_j^2 = \dots = \sum_{j=1}^k x_j^{2l-1} = \sum_{j=1}^k x_j^{2l} = 0), \\ 0 & (\text{otherwise}), \end{cases}$$

where each $x_j = x_j^1 x_j^2 \dots x_j^{2l-1} x_j^{2l}$ is an element of $\{0, 1\}^{2l}$, and the summation is taken over \mathbb{F}_2 . To reduce the communication complexity from the trivial $2kl$ qubits to kl qubits, we encode half of the input bits by bit flipping of the shared state $\frac{1}{\sqrt{2}}(|0^k\rangle + |1^k\rangle)$ (called the k -qubit GHZ state) among the k parties, and the other half by phase flipping of the state, by a method similar to superdense coding (e.g., see [27]). More precisely, we exploit an encoding similar to two-party quantum SMP protocols with shared entangled states to compute the equality function [20]. However, this is not sufficient for PSQM protocols. To convert quantum SMP protocols into PSQM protocols, we further use shared randomness among the k parties, and hide the input strings from the referee except for the output of the function GEQ_{2l} . This hiding can be shown to be possible by multiplying a random element in $\mathbb{F}_{2^{2l}}^*$ by the element in $\mathbb{F}_{2^{2l}}^*$ that corresponds to the input x_j .

For the proof of Theorem 2, we first consider a PSQM protocol for a finite function. Let $Sum_2 : (\{0, 1\}^2)^k \rightarrow \{0, 1\}^2$ be

$$Sum_2(x_1, x_2, \dots, x_k) = \left(\sum_{j=1}^k x_j^1, \sum_{j=1}^k x_j^2 \right),$$

where each $x_j = x_j^1 x_j^2$ is an element of $\{0, 1\}^2$, and the summation is taken over \mathbb{F}_2 .

► **Lemma 10.** *For any even (resp. odd) k , $Q_0^{psm,*}(Sum_2) \leq k$ (resp. $\leq k + 1$).*

Proof. First, we consider the case in which k is even. The quantum protocol is as follows.

Protocol \mathcal{P}_{Sum_2} : 0. All the parties share the entangled state

$$\frac{1}{\sqrt{2}} \left(\bigotimes_{j=1}^k |0\rangle_{Q_j} + \bigotimes_{j=1}^k |1\rangle_{Q_j} \right),$$

where the single-qubit register Q_j is owned by party P_j . Moreover, the parties share a k -bit string $r = r_1 r_2 \dots r_k$ such that $\sum_{j=1}^k r_j = 0$.

1. Each party P_j applies Z on Q_j if $x_j^2 = 1$. The resulting state is

$$\frac{1}{\sqrt{2}} \left(\bigotimes_{j=1}^k |0\rangle_{Q_j} + \bigotimes_{j=1}^k (-1)^{\sum_{j=1}^k x_j^2} |1\rangle_{Q_j} \right).$$

2. Each party P_j applies X on Q_j if $x_j^1 \oplus r_j = 1$. The resulting state is

$$\frac{1}{\sqrt{2}} \left(\bigotimes_{j=1}^k |x_j^1 \oplus r_j\rangle_{Q_j} + (-1)^{\sum_{j=1}^k x_j^2} \bigotimes_{j=1}^k |x_j^1 \oplus r_j \oplus 1\rangle_{Q_j} \right). \quad (8)$$

3. Each party P_j sends Q_j to the referee R .
4. R measures quantum registers Q_1, Q_2, \dots, Q_k in the basis

$$\left\{ |\Phi(y_1, y_2, \dots, y_{k-1}, z)\rangle := \frac{1}{\sqrt{2}} (|y_1, y_2, \dots, y_{k-1}, 0\rangle + (-1)^z |y_1 \oplus 1, y_2 \oplus 1, \dots, y_{k-1} \oplus 1, 1\rangle) \right\},$$

and let $y_1 y_2 \cdots y_{k-1} z$ be the measurement result.

5. R outputs the two bits $\sum_{j=1}^{k-1} y_j$ and z .

Correctness: The second bit of the output of R is $z = \sum_{j=1}^k x_j^2$, as desired. For the first bit, we consider two cases: (i) $x_k^1 \oplus r_k = 0$ and (ii) $x_k^1 \oplus r_k = 1$. We first consider case (i). Then, $y_j = x_j^1 \oplus r_j$ for $j = 1, 2, \dots, k-1$, and we thus obtain the desired output

$$\sum_{j=1}^{k-1} y_j = \sum_{j=1}^{k-1} x_j^1 \oplus r_j = \sum_{j=1}^k x_j^1 \oplus r_j = \sum_{j=1}^k x_j^1,$$

where the second inequality originates from $x_k^1 \oplus r_k = 0$, and the last equality originates from $\sum_{j=1}^k r_k = 0$. For case (ii), $y_j = x_j^1 \oplus r_j \oplus 1$ for $j = 1, 2, \dots, k-1$, and we thus obtain the desired output

$$\sum_{j=1}^{k-1} y_j = \sum_{j=1}^{k-1} x_j^1 \oplus r_j \oplus 1 = \left(\sum_{j=1}^{k-1} x_j^1 \oplus r_j \right) \oplus 1 = \sum_{j=1}^k x_j^1 \oplus r_j = \sum_{j=1}^k x_j^1,$$

where the second equality originates from the fact that k is even, the third equality originates from $x_k^1 \oplus r_k = 1$, and the last equality originates from $\sum_{j=1}^k r_k = 0$.

Privacy: Let the output of Sum_2 be (b_1, b_2) , where $b_1 = \sum_{j=1}^k x_j^1$ and $b_2 = \sum_{j=1}^k x_j^2$. As R has no knowledge about r_1, \dots, r_k except that the sum is 0, we can observe that the quantum state that R receives (represented by Eq. (8)) is taken from the set of 2^{k-2} orthogonal pure states

$$\left\{ |\Phi(y_1, \dots, y_{k-1}, b_2)\rangle : \sum_{j=1}^{k-1} y_j = b_1 \right\}$$

(up to the total phase) uniformly at random. Thus, the simulator can simulate the distribution of the message given the output of Sum_2 .

In the case in which k is odd, the last party P_k prepares an extra two-bit string $x_{k+1} = 00$, and \mathcal{P}_{Sum_2} is run for the $(k+1)$ -party case, where P_k also plays the role of the party P_{k+1} . ◀

Next, we present the main lemma for Theorem 2.

► **Lemma 11.** *For any even (resp. odd) k , $Q_0^{psm,*}(GEQ_{2l}) \leq kl$ (resp. $\leq (k+1)l$).*

Proof. We only demonstrate the case in which k is even (as the odd case is considered similarly to the proof of Lemma 10). The quantum protocol is as follows.

20:12 Communication Complexity of PSQM Protocols

Protocol $\mathcal{P}_{GEQ_{2l}}$: 0. All parties share the entangled state

$$\bigotimes_{i=1}^l \left(\frac{1}{\sqrt{2}} \left(\bigotimes_{j=1}^k |0\rangle_{Q_j^i} + \bigotimes_{j=1}^k |1\rangle_{Q_j^i} \right) \right),$$

where the single-qubit registers Q_j^1, \dots, Q_j^l are owned by party P_j . Moreover, they share l k -bit strings $r^i := r_1^i r_2^i \dots r_k^i$ such that $\sum_{j=1}^k r_j^i = 0$ ($i = 1, 2, \dots, l$), and a non-zero $2l$ -bit string $r' = r'_1 r'_2 \dots r'_{2l-1} r'_{2l}$.

1. Each party P_j computes the $2l$ -bit string $a_j = a_j^1 a_j^2 \dots a_j^{2l}$ defined as $\mathbf{p}(a_j) = \mathbf{p}(r') \mathbf{p}(x_j)$.
2. Each party P_j applies Z on Q_j^i if $a_j^{2i} = 1$. The resulting state is

$$\bigotimes_{i=1}^l \left(\frac{1}{\sqrt{2}} \left(\bigotimes_{j=1}^k |0\rangle_{Q_j^i} + \bigotimes_{j=1}^k (-1)^{\sum_{j=1}^k a_j^{2i}} |1\rangle_{Q_j^i} \right) \right).$$

3. Each party P_j applies X on Q_j^i if $a_j^{2i-1} \oplus r_j^i = 1$. The resulting state is

$$\bigotimes_{i=1}^l \left(\frac{1}{\sqrt{2}} \left(\bigotimes_{j=1}^k |a_j^{2i-1} \oplus r_j^i\rangle_{Q_j^i} + (-1)^{\sum_{j=1}^k a_j^{2i}} \bigotimes_{j=1}^k |a_j^{2i-1} \oplus r_j^i \oplus 1\rangle_{Q_j^i} \right) \right). \quad (9)$$

4. Each party P_j sends l quantum registers Q_j^1, \dots, Q_j^l to R .
5. R measures kl quantum registers $Q_1^1, \dots, Q_k^1, \dots, Q_1^l, \dots, Q_k^l$ in the basis

$$B := \left\{ \bigotimes_{j=1}^l |\Phi(y_1^i, \dots, y_{k-1}^i, z^i)\rangle : y_1^i \dots y_{k-1}^i z^i \in \{0, 1\}^k \text{ for every } i \in [l] \right\},$$

and let $y_1^i y_2^i \dots y_{k-1}^i z^i$ ($i = 1, \dots, l$) be the measurement results.

6. R accepts if $(\sum_{j=1}^{k-1} y_j^i) = z^i = 0$ for all $i = 1, \dots, l$ and rejects otherwise.

Correctness: Note that $(\sum_{j=1}^k x_j^1, \dots, \sum_{j=1}^k x_j^{2l}) = (0, \dots, 0)$ if and only if $(\sum_{j=1}^k a_j^1, \dots, \sum_{j=1}^k a_j^{2l}) = (0, \dots, 0)$, since

$$\mathbf{p} \left(\left(\sum_{j=1}^k a_j^1, \dots, \sum_{j=1}^k a_j^{2l} \right) \right) = \mathbf{p}(r') \mathbf{p} \left(\left(\sum_{j=1}^k x_j^1, \dots, \sum_{j=1}^k x_j^{2l} \right) \right). \quad (10)$$

Now, the correctness of \mathcal{P}_{Sum_2} in Lemma 10 also guarantees the correctness of $\mathcal{P}_{GEQ_{2l}}$.

Privacy: First, we consider the case in which $GEQ_{2l}(x_1, x_2, \dots, x_k) = 0$: that is, $(\sum_{j=1}^k x_j^1, \dots, \sum_{j=1}^k x_j^{2l}) = (0, \dots, 0)$. Then, as demonstrated in Eq. (10), (a_1, a_2, \dots, a_k) satisfies $(\sum_{j=1}^k a_j^1, \dots, \sum_{j=1}^k a_j^{2l}) = (0, \dots, 0)$. Moreover, $(a_1^{2i-1}, a_2^{2i-1}, \dots, a_k^{2i-1})$ is uniformly randomized by r^i in Step 3 under the restriction that the sum is 0. Thus, the quantum state that R receives (represented by Eq. (9)) is taken from the set of $2^{(k-2)l}$ orthogonal pure states

$$\left\{ \bigotimes_{i=1}^l |\Phi(y_1^i, \dots, y_{k-1}^i, 0)\rangle : \sum_{j=1}^{k-1} y_j^i = 0 \text{ for every } i \in [l] \right\}$$

(up to the total phase) uniformly at random. Second, we consider the case in which $GEQ_{2l}(x_1, x_2, \dots, x_k) = 1$, i.e., $(\sum_{j=1}^k x_j^1, \dots, \sum_{j=1}^k x_j^{2l})$ is in the set

$$S := \{(b_1, \dots, b_{2l}) : b_1 \cdots b_{2l} \in \{0, 1\}^{2l} \setminus \{(0, \dots, 0)\}\}.$$

Then, by Eq. (10), (a_1, a_2, \dots, a_k) is taken so that $(\sum_{j=1}^k a_j^1, \dots, \sum_{j=1}^k a_j^{2l})$ can be distributed from S uniformly at random. Moreover, $(a_1^{2i-1}, a_2^{2i-1}, \dots, a_k^{2i-1})$ is uniformly randomized by r^i in Step 3 under the restriction that the sum remains the same (since $\sum_{j=1}^k r_j^i = 0$). Thus, the quantum state that R receives (represented by Eq. (9)) is taken from the set of $(2^{2l} - 1)2^{(k-2)l}$ orthogonal pure states

$$B \setminus \left\{ \bigotimes_{i=1}^l |\Phi(y_1^i, \dots, y_{k-1}^i, 0)\rangle : \sum_{j=1}^{k-1} y_j^i = 0 \text{ for every } i \in [l] \right\}$$

(up to the total phase) uniformly at random. ◀

Now Lemma 11 provides the upper bound $kn/2$ of Theorem 2. The lower bound kn of Theorem 2 originates from the lower bound n of the exact (two-party) one-way quantum communication complexity with no shared entanglement of the n -bit equality function (see, e.g., [24, Theorem 5.11]) as it implies that for any $j \in [k]$, the j th party must send n qubits (considering the one-way communication setting from the j th party with input $x \in \{0, 1\}^n$ to the group of the referee and the other $k - 1$ parties in which one party has input $y \in \{0, 1\}^n$ and the $k - 2$ remaining parties have input 0^n , the length of the message of the j th party must be n). This completes the proof of Theorem 2.

Actually, the upper bound kl of Lemma 11 for GEQ_{2l} is tight when k is even. The matching lower bound kl originates from the lower bound l of the exact one-way quantum communication complexity with shared entanglement of the $2l$ -bit equality function shown by Klauck [24, Theorem 5.12] as it implies that each party must send l qubits.

5 Power of Shared Entanglement for Partial Functions

In this section, we prove Theorem 3. We consider the so-called distributed Deutsch-Jozsa problem DJ_n introduced by Brassard, Cleve, and Tapp [7]. First we show that $C_0^{psm,*}(DJ_n) = O(\log n)$. Our PSM protocol is based on the protocol provided in [7], which we modify so that the privacy condition can be satisfied. Second we show $Q_0^{psm}(DJ_n) = \Omega(n)$ by observing that the fact that the exact classical and quantum SMP communication complexities are the same for total functions can be extended to the case of partial functions.

Let n be any power of 2. The distributed Deutsch-Jozsa problem $DJ_n : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$, introduced in [7], is defined as

$$DJ_n(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \Delta(x, y) = n/2, \end{cases}$$

where $\Delta(x, y)$ denotes the Hamming distance between $x = x_0x_1 \cdots x_{n-1}$ and $y = y_0y_1 \cdots y_{n-1}$.

► **Lemma 12.** *There is a PSM protocol with shared entanglement that solves DJ_n with probability 1, and the classical communication complexity is $2 \log n$.*

Proof. The PSM protocol is as follows.

20:14 Communication Complexity of PSQM Protocols

Protocol \mathcal{P}_{DJ} : Let $n = 2^m$.

0. P_1 and P_2 share the entangled state

$$\frac{1}{\sqrt{n}} \sum_{i \in \{0,1\}^m} |i\rangle_A |i\rangle_B$$

and the two prearranged random m -bit strings $r \in \{0,1\}^m \setminus \{0^m\}$ and $r' \in \{0,1\}^m$.

1. P_1 (resp. P_2) adds phase $(-1)^{x_i}$ ($(-1)^{y_i}$) to the m -qubit register A (B) if the content of A (B) is i (where $i \in \{0,1\}^m$ is identified as the corresponding non-negative integer). The resulting state is

$$\frac{1}{\sqrt{n}} \sum_{i \in \{0,1\}^m} (-1)^{x_i} |i\rangle_A (-1)^{y_i} |i\rangle_B.$$

2. P_1 and P_2 apply the Hadamard gate H for each qubit of their registers A and B , respectively. The resulting state is

$$\frac{1}{n\sqrt{n}} \sum_{i \in \{0,1\}^m} \left((-1)^{x_i} \sum_{k \in \{0,1\}^m} (-1)^{i \cdot k} |k\rangle_A \right) \left((-1)^{y_i} \sum_{l \in \{0,1\}^m} (-1)^{i \cdot l} |l\rangle_B \right). \quad (11)$$

3. P_1 and P_2 measure A and B in the computational basis, respectively, and let K and L be the resulting bit strings in $\{0,1\}^m$.
4. P_1 and P_2 send classical messages m_A and m_B defined as $\mathbf{p}(m_A) = \mathbf{p}(r)\mathbf{p}(K) + \mathbf{p}(r')$ and $\mathbf{p}(m_B) = \mathbf{p}(r)\mathbf{p}(L) + \mathbf{p}(r')$ to R , respectively.
5. R accepts if $m_A = m_B$ and rejects otherwise.

Correctness: Note that the amplitude of $|k\rangle_A |l\rangle_B$ in Eq. (11) is

$$\frac{1}{n^{3/2}} \sum_{i \in \{0,1\}^m} (-1)^{x_i \oplus y_i} (-1)^{i \cdot (k \oplus l)}. \quad (12)$$

When $x = y$, Eq. (12) is 0 if $K \neq L$. Thus, the event $K = L$ occurs with probability 1; therefore, R always accepts. When $\Delta(x, y) = n/2$, Eq. (12) is 0 if $K = L$. Thus, the event $K \neq L$ occurs with probability 1; therefore, R always rejects.

Privacy: Again, by Eq. (12), if $x = y$, then $K = L = k$ is obtained with $1/n$ for each $k \in \{0,1\}^m$. Thus, the simulator can simulate the messages by generating the same m -bit string chosen uniformly at random as P_1 's and P_2 's messages. If $\Delta(x, y) = n/2$, the element $\mathbf{p}(K) - \mathbf{p}(L)$ in \mathbb{F}_{2^m} is nonzero; thus, the difference between $\mathbf{p}(r)\mathbf{p}(K) + \mathbf{p}(r')$ and $\mathbf{p}(r)\mathbf{p}(L) + \mathbf{p}(r')$ is distributed uniformly at random in $\mathbb{F}_{2^m}^*$. Moreover, $\mathbf{p}(K)$ (and $\mathbf{p}(L)$) is distributed uniformly at random in \mathbb{F}_{2^m} by multiplying $\mathbf{p}(r)$ and adding $\mathbf{p}(r')$. Thus, the simulator can simulate the messages by choosing two different m -bit non-zero strings uniformly at random as P_1 's and P_2 's messages. ◀

Using the result in [9] that DJ_n has the exact classical communication complexity $\Omega(n)$ (even in the two-way communication model), we can show that DJ_n provides the following exponential separation between exact PSMs with shared entanglement and exact PSQMs without shared entanglement. (Note that Theorem 13 implies Theorem 3, namely, an exponential gap between $Q_0^{psm,*}(DJ_n)$ and $Q_0^{psm}(DJ_n)$, as well as between $C_0^{psm,*}(DJ_n)$ and $C_0^{psm}(DJ_n)$.)

► **Theorem 13.** $C_0^{psm,*}(DJ_n) = O(\log n)$ and $Q_0^{psm}(DJ_n) = \Omega(n)$.

Proof. The upper bound $C_0^{psm,*}(DJ_n) = O(\log n)$ is shown by Lemma 12.

The lower bound comes from the fact that both the exact quantum and classical SMP communication complexities of a total function f over $X \times Y$ are equal to the sum of the number of the different row vectors of the communication matrix of f , M_f , and the number of the different column vectors of M_f (this fact can be found in [30, p.142]). The proof idea is that any two (classical or quantum) messages m_x and $m_{x'}$ corresponding to different row vectors indexed with input x and x' must be perfectly distinguished since there is some column input y such that $M_f(x, y) \neq M_f(x', y)$ (and a similar argument holds for different column vectors), and choosing different messages for such different row vectors or column vectors is sufficient for the referee to compute f exactly. This proof idea also holds for a partial function by replacing the number of the different row (resp. column) vectors by the size of the maximum clique of the graph $G_1(M_f) = (X, E_{1,f})$ (resp. $G_2(M_f) = (Y, E_{2,f})$) defined as follows: two rows x and x' (resp. columns y and y') have an edge in $E_{1,f}$ (resp. $E_{2,f}$) if and only if there is a column y (resp. row x) such that (x, y) and (x', y) (resp. (x, y) and (x, y')) are in the domain of the partial function f and $M_f(x, y) \neq M_f(x', y)$ (resp. $M_f(x, y) \neq M_f(x, y')$).

The above observation implies that the exact quantum and classical SMP communication complexities of the partial function DJ_n are the same. By the result in [9], DJ_n has the exact classical communication complexity $\Omega(n)$. This concludes the desired lower bound $Q_0^{psm}(DJ_n) = \Omega(n)$. ◀

Theorem 13 provides an exponential gap between PSMs with shared entanglement and PSQMs without shared entanglement for partial functions; however, it is obtained only in the exact setting, and we do not know whether this exponential gap can be obtained in the bounded-error setting for partial or total functions. However, we can observe that there is a relational problem that has an exponential gap between PSMs with shared entanglement and PSQMs without shared entanglement in the bounded-error setting from the result by Gavinsky et al. [18]. They demonstrated that the problem has an exponentially smaller communication complexity of a classical SMP protocol with shared entanglement than the communication complexity of quantum SMP protocols only with shared randomness, while it is easy to see that their protocol is in fact a PSQM.

6 Conclusion

This paper introduced a quantum analogue of the well-studied PSM model which was called the PSQM model, and provided several initial results in the exact setting. Here we list a number of open problems.

- Can the lower bound of Theorem 1 be extended to the bounded-error setting or to the shared entanglement case?
- Can any non-trivial communication complexity gap between the PSM model and the PSQM model for some function in the bounded-error setting? How about any non-trivial communication complexity gap between the PSQM model with shared entanglement and the PSQM model without it in the bounded-error setting?
- The PSQM model in this paper was defined only for perfect privacy. What results are obtained for imperfect privacy? To extend the lower bound of Theorem 1 to imperfect privacy, a quantumly tailored modification of [1, Section 5] might be worth considering, while it seems much more complicated than the proof of Theorem 1.

References

- 1 Benny Applebaum, Thomas Holenstein, Manoj Mishra, and Ofer Shayelevitz. The communication complexity of private simultaneous messages, revisited. *Journal of Cryptology*, 33(3):916–953, 2020. doi:10.1007/s00145-019-09334-y.
- 2 Marshall Ball, Justin Holmgren, Yuval Ishai, Tianren Liu, and Tal Malkin. On the complexity of decomposable randomized encodings, or: how friendly can a garbling-friendly PRF be? In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, pages 86:1–86:22, 2020. doi:10.4230/LIPIcs.ITCS.2020.86.
- 3 Ziv Bar-Yossef, T. S. Jayram, and Iordanis Kerenidis. Exponential separation of quantum and classical one-way communication complexity. *SIAM Journal on Computing*, 38(1):366–384, 2008. doi:10.1137/060651835.
- 4 Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In *Proceedings of the 11th IACR Theory of Cryptography Conference (TCC 2014)*, pages 317–342, 2014. doi:10.1007/978-3-642-54242-8_14.
- 5 Amos Beimel, Eyal Kushilevitz, and Pnina Nissim. The complexity of multiparty PSM protocols and related models. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part II*, pages 287–318, 2018. doi:10.1007/978-3-319-78375-8_10.
- 6 Zvika Brakerski and Henry Yuen. Quantum garbled circuits. arXiv:2006.01085, 2020. URL: <https://arxiv.org/abs/2006.01085>.
- 7 Gilles Brassard, Richard Cleve, and Alain Tapp. Cost of exactly simulating quantum entanglement with classical communication. *Physical Review Letters*, 83:1874–1877, 1999. doi:10.1103/PhysRevLett.83.1874.
- 8 Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. Quantum fingerprinting. *Physical Review Letters*, 87:167902 (4pages), 2001. doi:10.1103/PhysRevLett.87.167902.
- 9 Harry Buhrman, Richard Cleve, and Avi Wigderson. Quantum vs. classical communication and computation. In *Proceedings of the 30th annual ACM Symposium on Theory of Computing (STOC 1998)*, pages 63–68, 1998. doi:10.1145/276698.276713.
- 10 Claude Crépeau, Daniel Gottesman, and Adam Smith. Secure multi-party quantum computation. In *Proceedings of the 34th Annual Symposium on Theory of Computing (STOC 2002)*, pages 643–652, 2002. doi:10.1145/509907.510000.
- 11 Ivan Damgård, Kasper Green Larsen, and Jesper Buus Nielsen. Communication lower bounds for statistically secure MPC, with or without preprocessing. In *Proceedings of the 39th Annual International Cryptology Conference (CRYPTO 2019) Part II*, pages 61–84, 2019. doi:10.1007/978-3-030-26951-7_3.
- 12 Deepesh Data, Manoj M. Prabhakaran, and Vinod M. Prabhakaran. Communication and randomness lower bounds for secure computation. *IEEE Transactions on Information Theory*, 62(7):3901–3929, 2016. doi:10.1109/TIT.2016.2568207.
- 13 Yfke Dulek, Alex B. Grilo, Stacey Jeffery, Christian Majenz, and Christian Schaffner. Secure multi-party quantum computation with a dishonest majority. In *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2020) Part III*, pages 729–758, 2020. doi:10.1007/978-3-030-45727-3_25.
- 14 Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC 1994)*, pages 554–563, 1994. doi:10.1145/195058.195408.
- 15 Dmitry Gavinsky. Quantum versus classical simultaneity in communication complexity. *IEEE Transactions on Information Theory*, 65(10):6466–6483, 2019. doi:10.1109/TIT.2019.2918453.
- 16 Dmitry Gavinsky. Bare quantum simultaneity versus classical interactivity in communication complexity. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC 2020)*, pages 401–411, 2020. doi:10.1145/3357713.3384243.

- 17 Dmitry Gavinsky and Tsuyoshi Ito. Quantum fingerprints that keep secrets. *Quantum Information and Computation*, 13(7-8):583–606, 2013. doi:10.26421/QIC13.7-8-3.
- 18 Dmitry Gavinsky, Julia Kempe, Oded Regev, and Ronald de Wolf. Bounded-error quantum state identification and exponential separations in communication complexity. *SIAM Journal on Computing*, 39(1):1–24, 2009. doi:10.1137/060665798.
- 19 Masahito Hayashi, Satoshi Ishizuka, Akinori Kawachi, Gen Kimura, and Tomohiro Ogawa. *Introduction to Quantum Information Science*. Springer, 2015. doi:10.1007/978-3-662-43502-1.
- 20 Rolf T. Horn, A. J. Scott, Jonathan Walgate, Richard Cleve, A. I. Lvovsky, and Barry C. Sanders. Classical and quantum fingerprinting with shared randomness and one-sided error. *Quantum Information and Computation*, 5(3):258–271, 2005. doi:10.26421/QIC5.3-6.
- 21 Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocol with applications. In *Proceedings of the 5th Israel Symposium on Theory of Computing and Systems (ISTCS 1997)*, pages 174–183, 1997. doi:10.1109/ISTCS.1997.595170.
- 22 Roy Kasher and Julia Kempe. Two-source extractors secure against quantum adversaries. *Theory of Computing*, 8:461–486, 2012. doi:10.4086/toc.2012.v008a021.
- 23 Hartmut Klauck. Quantum and approximate privacy. *Theory of Computing Systems*, 37(1):221–246, 2004. doi:10.1007/s00224-003-1113-7.
- 24 Hartmut Klauck. One-way communication complexity and the Nečiporuk lower bound on formula size. *SIAM Journal on Computing*, 37(2):552–583, 2007. doi:10.1137/S009753970140004X.
- 25 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997. doi:10.1017/CB09780511574948.
- 26 Tomoyuki Morimae. Quantum randomized encoding, verification of quantum computing, no-cloning, and blind quantum computing. arXiv:2011.03141, 2020. URL: <https://arxiv.org/abs/2011.03141>.
- 27 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- 28 Ran Raz. Exponential separation of quantum and classical communication complexity. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC 1999)*, pages 358–367, 1999. doi:10.1145/301250.301343.
- 29 Dominique Unruh. Universally composable quantum multi-party computation. In *Proceedings of 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2010)*, pages 486–505, 2010. doi:10.1007/978-3-642-13190-5_25.
- 30 Ronald de Wolf. *Quantum Computing and Communication Complexity*. PhD thesis, University of Amsterdam, 2001. URL: <https://dare.uva.nl/search?identifier=480e76ad-11b7-4226-9c54-6b39c51e6f37>.
- 31 Andrew C.-C. Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC 1979)*, pages 209–213, 1979. doi:10.1145/800135.804414.

A Proofs of Claim 7, Claim 8, and Lemma 9

In this appendix, we give the detailed proofs of the technical claims and lemma used in the proof of Lemma 5.

Proof of Claim 7. For any two quantum states ρ, σ , the condition $\rho\sigma = 0$ is necessary (and sufficient) for perfect discrimination of the two states (see, e.g., Proposition 5.13 in [19]). Since the referee R perfectly discriminates $\rho(x_1, x_2)$ and $\rho(x'_1, x'_2)$ for which $F_n(x_1, x_2) \neq F_n(x'_1, x'_2)$

from the correctness, it must hold that $\rho(x_1, x_2)\rho(x'_1, x'_2) = 0$. Then, we have

$$\begin{aligned} & \text{tr} \sum_{(x_1, x_2) \neq (x'_1, x'_2)} \mu(x_1, x_2)\rho(x_1, x_2)\mu(x'_1, x'_2)\rho(x'_1, x'_2) \\ &= \sum_y \text{tr} \sum_{\substack{(x_1, x_2) \neq (x'_1, x'_2) \\ F_n(x_1, x_2) = F_n(x'_1, x'_2) = y}} \mu(x_1, x_2)\rho(x_1, x_2)\mu(x'_1, x'_2)\rho(x'_1, x'_2). \end{aligned}$$

From the privacy, there exists a quantum state ρ_y for each y such that $\rho_y = \rho(x_1, x_2)$ for every (x_1, x_2) for which $F_n(x_1, x_2) = y$. Therefore,

$$\begin{aligned} & \sum_y \text{tr} \sum_{\substack{(x_1, x_2) \neq (x'_1, x'_2) \\ F_n(x_1, x_2) = F_n(x'_1, x'_2) = y}} \mu(x_1, x_2)\rho(x_1, x_2)\mu(x'_1, x'_2)\rho(x'_1, x'_2) \\ &= \sum_y \text{tr} \rho_y^2 \sum_{\substack{(x_1, x_2) \neq (x'_1, x'_2) \\ F_n(x_1, x_2) = F_n(x'_1, x'_2) = y}} \mu(x_1, x_2)\mu(x'_1, x'_2) \\ &= \sum_y \frac{\sum_{\substack{(x_1, x_2) \neq (x'_1, x'_2) \\ F_n(x_1, x_2) = F_n(x'_1, x'_2) = y}} \mu(x_1, x_2)\mu(x'_1, x'_2)}{\sum_{F_n(x_1, x_2) = F_n(x'_1, x'_2) = y} \mu(x_1, x_2)\mu(x'_1, x'_2)} \text{tr} \rho_y^2 \sum_{F_n(x_1, x_2) = F_n(x'_1, x'_2) = y} \mu(x_1, x_2)\mu(x'_1, x'_2) \\ &= \sum_y \Pr[(X_1, X_2) \neq (X'_1, X'_2) \mid F_n(X_1, X_2) = F_n(X'_1, X'_2) = y] \text{tr} \rho_y^2 \\ & \quad \times \sum_{F_n(x_1, x_2) = F_n(x'_1, x'_2) = y} \mu(x_1, x_2)\mu(x'_1, x'_2) \\ &\geq \beta(\mu) \sum_y \text{tr} \rho_y^2 \sum_{F_n(x_1, x_2) = F_n(x'_1, x'_2) = y} \mu(x_1, x_2)\mu(x'_1, x'_2) \\ &= \beta(\mu) \text{tr} \sum_{x_1, x_2, x'_1, x'_2} \mu(x_1, x_2)\rho(x_1, x_2)\mu(x'_1, x'_2)\rho(x'_1, x'_2). \quad \triangleleft \end{aligned}$$

Proof of Claim 8. From the privacy, there exists ρ_y such that $\rho_y = \rho(x_1, x_2)$ for every $y \in \{0, 1\}$ and every (x_1, x_2) for which $F_n(x_1, x_2) = y$. From the correctness and the necessary condition of the perfect quantum state discrimination, $\rho_0\rho_1 = 0$ must hold. From the spectral decomposition we have $\rho_y = \sum_i \lambda_{y,i} |\phi_{y,i}\rangle\langle\phi_{y,i}|$ for some orthonormal basis $\{|\phi_{y,i}\rangle\}_i$ ($\lambda_{y,i} > 0$). Then, we have $\langle\phi_{0,i}|\phi_{1,j}\rangle = 0$ for every i, j since $\rho_0\rho_1 = 0$. Therefore, we can assume $R_0 = \sum_i |\phi_{0,i}\rangle\langle\phi_{0,i}|$ and $R_1 = I - R_0$ without loss of generality. This $R = \{R_y\}_{y \in \{0,1\}}$ is a PVM. \triangleleft

Proof of Lemma 9. We demonstrate that $|\langle\psi_1(z; r')|\psi_1(z'; r')\rangle|^2 = 0$ for every distinct z, z' and every r' . Assuming this, we can decompose

$$|\psi_1(x_1; r)\rangle = \sum_{z'} \alpha_{z'} |\psi_1(z'; r')\rangle + \alpha^\perp |\psi_1^\perp\rangle$$

with orthonormal vectors $\{|\psi_1(z'; r')\rangle\}_{z'} \cup \{|\psi_1^\perp\rangle\}$ for some coefficients $\alpha_{z'}$ and α^\perp . Then, it holds that

$$\begin{aligned} \sum_{z \neq x_1} |\langle\psi_1(x_1; r)|\psi_1(z; r')\rangle|^2 &= \sum_{z \neq x_1} \left| \sum_{z'} \alpha_{z'}^* \langle\psi_1(z'; r')|\psi_1(z; r')\rangle + \alpha^{\perp*} \langle\psi_1^\perp|\psi_1(z; r')\rangle \right|^2 \\ &= \sum_{z \neq x_1} |\alpha_z|^2 \leq 1. \end{aligned}$$

Therefore, it suffices to demonstrate $|\langle \psi_1(z; r') | \psi_1(z'; r') \rangle|^2 = 0$ for every distinct z, z' and every r' . For contradiction, assume that $|\langle \psi_1(z; r') | \psi_1(z'; r') \rangle|^2 > 0$ for some z, z' and some r' .

From this assumption, we have

$$|\psi_1(z'; r')\rangle = \beta |\psi_1(z; r')\rangle + \beta^\perp |\psi_1^\perp(z; r')\rangle,$$

where $\beta \neq 0$ and $|\psi_1^\perp(z; r')\rangle$ is orthogonal to $|\psi_1(z; r')\rangle$.

We fix x_2 arbitrarily. Then, we have

$$|\psi_1(z'; r')\rangle |\psi_2(x_2; r')\rangle = \beta |\psi_1(z; r')\rangle |\psi_2(x_2; r')\rangle + \beta^\perp |\psi_1^\perp(z; r')\rangle |\psi_2(x_2; r')\rangle.$$

Since R is a PVM, we have

$$R_{F_n(z, x_2)} |\psi_1(z; r')\rangle |\psi_2(x_2; r')\rangle = |\psi_1(z; r')\rangle |\psi_2(x_2; r')\rangle$$

from the correctness. We also have

$$|\psi_1^\perp(z; r')\rangle = \gamma |\phi(z; r')\rangle + \gamma^\perp |\phi^\perp(z; r')\rangle,$$

where $|\psi_1(z; r')\rangle, |\phi(z; r')\rangle, |\phi^\perp(z; r')\rangle$ are orthogonal to each other, and $R_{F_n(z, x_2)} |\phi(z; r')\rangle |\psi_2(x_2; r')\rangle = |\phi(z; r')\rangle |\psi_2(x_2; r')\rangle, R_{F_n(z, x_2)} |\phi^\perp(z; r')\rangle |\psi_2(x_2; r')\rangle = 0$.

Therefore, it holds that

$$|\langle \psi_1(z; r') | \langle \psi_2(x_2; r') | R_{F_n(z, x_2)} |\psi_1(z'; r')\rangle |\psi_2(x_2; r')\rangle |^2 = |\beta|^2 + |\gamma|^2 > 0.$$

The value $|\beta|^2 + |\gamma|^2 > 0$ must be 1 from the correctness; therefore, $R(|\psi_1(z'; r')\rangle |\psi_2(x_2; r')\rangle) = F_n(z', x_2)$ for every x_2 and every r' . This implies that $F_n(z, x_2) = F_n(z', x_2)$ holds for every x_2 , which contradicts the non-degeneracy of F_n . The same argument also works for x_2 . ◀

Quantum-Access Security of the Winternitz One-Time Signature Scheme

Christian Majenz ✉ 

Centrum Wiskunde & Informatica and QuSoft, Amsterdam, The Netherlands

Chanelle Matadah Manfouo ✉ 

African Institute for Mathematical Science & Quantum Leap Africa, Kigali, Rwanda

Maris Ozols ✉ 

Institute for Logic, Language, and Computation, Korteweg-de Vries Institute for Mathematics, and Institute for Theoretical Physics, University of Amsterdam and QuSoft, Amsterdam, The Netherlands

Abstract

Quantum-access security, where an attacker is granted superposition access to secret-keyed functionalities, is a fundamental security model and its study has inspired results in post-quantum security. We revisit, and fill a gap in, the quantum-access security analysis of the Lamport one-time signature scheme (OTS) in the quantum random oracle model (QROM) by Alagic et al. (Eurocrypt 2020). We then go on to generalize the technique to the Winternitz OTS. Along the way, we develop a tool for the analysis of hash chains in the QROM based on the superposition oracle technique by Zhandry (Crypto 2019) which might be of independent interest.

2012 ACM Subject Classification Theory of computation → Quantum information theory; Security and privacy → Digital signatures; Security and privacy → Information-theoretic techniques

Keywords and phrases quantum cryptography, one-time signature schemes, quantum random oracle model, post-quantum cryptography, quantum world, hash-based signatures, information-theoretic security

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.21

Related Version *Full Version*: <https://arxiv.org/abs/2103.12448> [23]

Funding *Christian Majenz*: NWO Veni grant (Project No. VI.Veni.192.159).

Chanelle Matadah Manfouo: NWO Gravitation grant QSC (Project No. 024.003.037) and AIMS-NEI.

Maris Ozols: NWO Vidi grant (Project No. VI.Vidi.192.109).

Acknowledgements The authors thank anonymous reviewers for insightful comments and Stacey Jeffery for helpful discussions. CMM deeply thanks the African Institute for Mathematical Sciences, Quantum Leap Africa Rwanda, and QuSoft Amsterdam for their support.

1 Overview

1.1 Introduction

Recently, research and development efforts towards building a universal quantum computer have intensified. As quantum computers will break currently deployed public-key cryptosystems [27], finding adequate replacement schemes (called *post-quantum* secure) has been increasingly a priority, too, as reflected by the ongoing NIST standardization effort for post-quantum secure digital signature schemes and key encapsulation mechanisms [1].

Quantum-access security. While post-quantum security is the most important attack model involving quantum computers, the stronger *quantum-access* or *quantum world* attack model [7, 13], where attackers are granted quantum access to secret-keyed functionalities,



© Christian Majenz, Chanelle Matadah Manfouo, and Maris Ozols;
licensed under Creative Commons License CC-BY 4.0

2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 21; pp. 21:1–21:22



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

has received considerable attention, too. There are a number of reasons why this stronger attack model is important. On the one hand, it is of theoretical importance because it captures the strongest-known achievable security notions for standard classical cryptographic primitives. On the other hand, there are a number of conceivable scenarios where they become relevant, e.g. for composability with obfuscation or when constructing quantum-cryptographic schemes, or to prevent implementation-level vulnerabilities in a future hybrid quantum-classical computing infrastructure. Finally, results in the quantum access model can inform post-quantum cryptographic research, as exemplified by the offline Simon’s algorithm attack [8].

Blind unforgeability. In this work, we study the security of signature schemes under quantum-access attacks, in the quantum random oracle model (QROM) [6]. Here, generalizing the standard notion of existential unforgeability under chosen message attacks, the attacker is granted quantum query access to the signing algorithm. In the end, the adversary should output a forgery that they did not obtain from a query. Formalizing such a security notion is complicated due to the so-called *quantum no-cloning principle* according to which quantum states cannot be copied. We use the notion of blind unforgeability introduced in [2] (see [7, 15] for previous and complementary notions). We remark that the choice of the blind unforgeability definition is due to the fact that it implies the previous notions, which are the Boneh and Zhandry definition [7] and the one-time unforgeability [15], as established in [2]. Informally, blind unforgeability credits an adversary with a successful break of, e.g., a digital signature scheme, if it outputs a valid message-signature pair given a modified signing oracle that is “blinded” on a random subset of all messages, in the sense that it outputs a dummy symbol instead of a signature, and if the output message is among these blinded messages (see Section 2 for details).

Hash-based signature schemes. Hash-based signature schemes are prominent candidates for the replacement of digital signature schemes based on quantum-broken number-theoretic hardness assumptions. In particular, the stateful hash-based signature scheme XMSS [10] has been standardized as RFC8391 [19], and the stateless hash-based signature scheme SPHINCS+ [4] is an alternate candidate in the ongoing NIST standardization process for post-quantum cryptographic schemes [1]. The security of hash-based signature schemes can be based on weak computational assumptions, like e.g. the one-wayness of the underlying hash function. Common hash-based signature schemes, including the mentioned examples, are constructed using one-time¹ signature (OTS) schemes in combination with a hash-based authentication graph (e.g. a Merkle tree). The most well-known OTSs are the Lamport [21] and Winternitz [24] OTS. Variations of the latter are used in both XMSS and SPHINCS+.

Previous work. In [2], the Lamport OTS is studied in the context of blind-unforgeability. More precisely, a proof of one-time blind-unforgeability in the QROM is provided. That proof, however, has a gap in the analysis of the adversarial success. In particular, an auxiliary measurement is used to “collapse” an invariant property that holds *in superposition* into holding *classically*, but the effect of the dependence of this auxiliary measurement on the forgery message is not analyzed.

¹ And sometimes few-time signature schemes, e.g. in SPHINCS+.

Related work. Quantum-access security for encryption is an active research area, and generalizing chosen-ciphertext security notions to the quantum-access setting has posed, and poses, similar challenges as the ones encountered in the authenticity setting [7, 13, 14]. On the negative side, key recovery attacks in the quantum-access model against a number of symmetric-key primitives that are secure in the respective standard attack models have been discovered [26, 20], and have lead to the discovery of quantum attacks that can be performed without quantum access to secret-keyed functionalities [8].

There are a number of works that prove query lower bounds using variants of the superposition oracle technique [22, 17, 11, 5]. The last two papers prove query complexity lower bounds for creating hash chains, which are not directly useful for the analysis of hash-based signatures.

1.2 Summary of results

The Lamport OTS is blind-unforgeable. We revisit the analysis of the Lamport OTS in the QROM presented in [2] and give a complete proof of blind unforgeability as stated in the following theorem.

► **Theorem 1** (Blind unforgeability of the Lamport OTS, informal). *The Lamport OTS is blind-unforgeable if the underlying hash function h is modeled as a quantum-accessible random oracle. More precisely, the success probability of any blind unforgeability adversary \mathcal{A} against the Lamport OTS that makes $q > 0$ quantum queries to the random oracle is bounded as*

$$\Pr[\mathcal{A} \text{ succeeds}] \leq C_L q^2 l^3 \cdot 2^{-n},$$

where C_L is a constant, n is the security parameter of the Lamport OTS and l is the message length.

Compared to [2], our security proof features the following improvements:

- We streamline the usage of the superposition oracle technique of Zhandry [28]. In particular, our analysis only uses (a variant of) the superposition oracle technique to sample the secret key. We reprogram, *in superposition*, the standard random oracle at inputs contained in the secret key. This technique represents a general tool to analyze hash chains in the QROM and might be of independent interest.
- We give a full analysis of the success probability using an auxiliary measurement idea from [2]. To tackle the problem mentioned above, we introduce a novel technique of tracking an invariant property *in superposition* using projectors and commutators.

The Winternitz OTS is blind-unforgeable. With the full blind unforgeability analysis of the Lamport OTS in hand, we generalize the approach to the Winternitz OTS.

► **Theorem 2** (Blind unforgeability of the Winternitz OTS, informal). *The Winternitz OTS is blind-unforgeable if the underlying hash function h is modeled as a quantum-accessible random oracle. More precisely, the success probability of any blind unforgeability adversary \mathcal{A} against the Winternitz OTS that makes $q > 0$ quantum queries to the random oracle is bounded as*

$$\Pr[\mathcal{A} \text{ succeeds}] \leq C_W q^2 a^3 \frac{w^4}{\log^3 w} \cdot 2^{-n},$$

where C_W is a constant, n is the security parameter of the Winternitz OTS, a is the message length and $w \geq 2$ is the Winternitz parameter used to trade off signature size versus signing and verification time.

While the simplified analysis of hash chains in the QROM described above was advantageous in proving the blind unforgeability security of the Lamport OTS, it is indispensable in the analysis of the Winternitz scheme. Here, long hash chains are considered and the technique of using the superposition oracle to detect which hash chain elements are known to the adversary relies on the oracle register (or rather here: the hash chain register) being in a product state.

1.3 Technical overview

In this technical overview, we give a high-level description of our techniques for analyzing the blind unforgeability security of the Lamport and Winternitz OTSs in the QROM.

The superposition oracle technique and hash chains. As in many contexts that concern message authenticity and integrity, the main roadblock we have to overcome in our analysis is the so-called *recording barrier*: quantum oracle queries can, in general, not be recorded for later use. In particular, after a single quantum signing query, it is not possible to reason about the unused parts of the secret key. This is because, in general, all secret key strings have been used in some part of the superposition.

In [2], Zhandry’s superposition oracle technique is used in a novel way to recover the ability to reason about which secret key strings are (un)known to the adversary. There, the secret key of the Lamport scheme, which is a $2 \times l$ array of independent uniformly random n -bit strings, is essentially regarded as a random function from $\{0, 1\} \times \{1, \dots, l\}$. This function, as well as the hash function the Lamport OTS is constructed from, is then modelled using the superposition oracle technique.

We improve this technique as follows. We use the fact that sampling two correlated random variables X and Y can be done by first sampling X , and then Y according to the conditional distribution, or vice versa. In the context of *hash chains* in the (Q)ROM, i.e. sequences of strings $x_0, x_1 = H(x_0), x_2 = H(x_1), \dots$ for a random oracle H , this means that instead of sampling x_0 and H , and then computing the remaining hash chain elements, we can as well sample x_0, x_1, \dots from their joint distribution, sample H , and *reprogram H to be consistent with the x_i* . This allows us to i) change the distribution of the x_i to a simpler one that is close in total variational distance, and ii) refrain from using the full superposition oracle technique for H . In particular, we use i) to replace the hash chains that are generated by the key generation algorithms of the Lamport and Winternitz schemes by tuples of independent random strings. This incurs only a small error, as the uniform distribution and the distribution of a hash chain in the (Q)ROM with random starting value x_0 are equal conditioned on all x_i being distinct. But collisions between different hash chain elements are unlikely.

Now that the hash chain elements are independent strings, we can use the full power of the superposition oracle technique. In particular, the one-to-one correspondence between the adversary’s ignorance of a hash chain element and the corresponding superposition oracle register being in uniform superposition, is restored.

Throughout the paper, and in the rest of this technical overview, we perform the analysis in a world where hash chains are formed using a superposition oracle modeling independent uniformly random strings, and the random oracle is reprogrammed accordingly. We call this the **Quantum independent world**. To conclude our analysis, we make use of the approximate indistinguishability of the Real and the Quantum independent world.

Blind unforgeability and classical invariants in superposition. With the tools for analyzing hash chains in the QROM in hand, the next challenge consists of generalizing the classical security arguments for the Blind Unforgeability (BU) of the Lamport and Winternitz OTSs to the quantum-access setting. The core of these security arguments is, at a high level, that for each unqueried message, any valid signature contains a string that is unknown to the adversary.² As mentioned above, this kind of reasoning does not generalize to the quantum-access setting, as here an adversary can query all messages in superposition.

In the security game for the notion of BU instead of the full signing oracle the adversary is provided with a modified oracle that is “blinded” on a random subset of messages, in the sense that for these messages it outputs a dummy symbol \perp instead of a signature. These “blinded messages” can now replace the unqueried messages in security arguments, as by definition the adversary is prevented from obtaining a valid signature for them from the blinded signing oracle.

For obtaining a quantum generalization, we need to reformulate this argument. The statement that for each unqueried message any valid signature contains a string unknown to the adversary, is equivalent to saying that, for each fixed message m^* and all $m \neq m^*$, some information related to the secret key and not revealed by the signature of m is necessary to compute the signature for m^* . For BU, it suffices to consider blinded m^* and unblinded m . In the superposition oracle framework, the statement “there exists an unblinded message such that the registers corresponding to all parts of the secret key not revealed by that message are in the uniform superposition state” defines a subspace I . By definition, the global state after a BU-adversary makes a single query to the blinded signing oracle, and no queries to the random oracle, is in that subspace.

The crucial step in our analysis is to show that the joint adversary-oracle state approximately remains in the subspace I , even if the adversary performs a moderate number of quantum queries to the random oracle. This means the subspace I can serve as an *invariant*.

Random oracle queries and commutators. To analyze the “leakage” from the invariant subspace I , we use bounds on the norm of matrix commutators: to prove that the final oracle-adversary state is approximately in the invariant subspace I , we can equivalently show that applying the corresponding projector Π_I does not change the state by a lot. We know, however, that the projector does not change the state at all before any random oracle queries have been made. Therefore it suffices to bound the operator norm of the commutator between the projector Π_I and the unitary operator that facilitates random oracle queries in the Quantum independent world. We derive such a norm bound (see e.g. Lemma 15 for the Lamport case), and the proof follows the classical intuition about the one-wayness of the random oracle.

2 Preliminaries

We introduce some notation and conventions that will be used throughout the paper. Registers of quantum systems will be denoted by capital letters. We say that $\epsilon = \epsilon(n)$ is negligible if, for all polynomials $p(n)$, $\epsilon(n) < 1/p(n)$ for large enough $n \in \mathbb{N}$. We use the notation $x \xleftarrow{\$} D$ to say that x is chosen uniformly at random from a set D . We write S^c to denote the

² When basing the security on one-wayness, “unknown” is to be taken in a computational sense, but as this paper is about security in the (Q)ROM, it is sufficient to interpret “unknown to” as “independent of the state of”.

complement of set S (in a superset that is clear from the context). We write $s \parallel t$ to denote the *concatenation* of strings s and t , and $[A, B] = AB - BA$ to denote the *commutator* of operators A and B . Throughout this paper, quantum adversaries refer to quantum polynomial-time algorithms and are denoted by \mathcal{A} .

Quantum computing. We use standard quantum computing notation, see e.g. [25]. A d -level quantum system is associated with a d -dimensional complex Euclidean space $\mathcal{H} = \mathbb{C}^d$ with inner product $\langle \cdot | \cdot \rangle$. We refer to the standard basis of \mathbb{C}^d as the *computational basis*. The *state* of a system is described by a unit vector in $|\psi\rangle \in \mathcal{H}$, and $\langle \psi|$ denotes its dual vector. Given two quantum systems A and B , the composite system AB has state space equal to the tensor product $\mathcal{H}_A \otimes \mathcal{H}_B$. We will often refer to the subsystems A and B as *registers*. We denote the n -bit uniform superposition and the corresponding projector by

$$|\Phi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle, \quad \Phi = |\Phi\rangle\langle\Phi|. \quad (1)$$

Quantum computation proceeds by applying *unitary transformations*, i.e., complex $d \times d$ matrices U such that $UU^\dagger = I$, where $U^\dagger = \bar{U}^\top$ denotes the conjugate transpose of U . We omit tensor products with identity matrices, indicating which registers an operator acts on by subscripts, e.g. $U_A|\psi\rangle_{AB} = (U_A \otimes I_B)|\psi\rangle_{AB}$.

We can extract information from a quantum state $|\psi\rangle$ by performing a *measurement*. A (projective) measurement is described by a set $\{P_1, \dots, P_k\}$ of orthogonal projectors ($P_i^\dagger = P_i$ and $P_i^2 = P_i$) such that $\sum_{i=1}^k P_i = I$. When performing a measurement on a quantum state $|\psi\rangle$, the probability of getting outcome i is $p(i) = \langle \psi | P_i | \psi \rangle$. Upon getting outcome i , the state $|\psi\rangle$ collapses to $P_i|\psi\rangle / \sqrt{p(i)}$.

The standard way of modelling quantum black-box access to a function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ is by providing an oracle for the unitary operation O_f that acts on $n + m$ qubits as $O_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$ for all $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$. Without loss of generality, an algorithm \mathcal{A} that makes q queries to such an oracle has the form $U_q O_f \cdots U_1 O_f U_0 |\Psi_0\rangle = V_{\mathcal{A}}^{O_f} |\Psi_0\rangle = |\Psi\rangle$, possibly followed by a measurement. Here, $|\Psi_0\rangle$ is an initial state and U_i are arbitrary unitary operations that do not depend on f .

We will deal with algorithms that have two oracles, O_1 and O_2 , but may only query O_2 at most once (O_1 will be a random oracle and O_2 a signing oracle for a one-time signature scheme). We can regard such an algorithm $\mathcal{A}^{O_1, O_2} = (\mathcal{A}_0^{O_1}, \mathcal{A}_1^{O_1})$ as a two-stage process: $\mathcal{A}_0^{O_1}$ prepares the input for O_2 and an internal register, $\mathcal{A}_1^{O_1}$ receives the internal state and the output of O_2 , and produces the final output of \mathcal{A} , $|\Psi\rangle = V_{\mathcal{A}_1}^{O_1} O_2 V_{\mathcal{A}_0}^{O_1} |\Psi_0\rangle$.

The most well-known situation in cryptography that features a quantum oracle is the so-called *quantum random oracle model* (QROM) [6]. In the QROM, just as in the classical random oracle model (ROM) [3], a hash function is modeled as a uniformly random function h that all agents have oracle access to.

Tools from linear algebra. In this section, we state a couple of simple lemmas used in security proofs in Sections 4 and 5. For the first lemma, we use the formulation from [7] (Lemma 2.1), and the proof is also provided in the same reference.

► **Lemma 3** (Special case of the pinching lemma [18]). *Let \mathcal{A} be a quantum algorithm and x any output value of \mathcal{A} . Let \mathcal{A}_0 be another quantum algorithm obtained from \mathcal{A} by pausing \mathcal{A} in an arbitrary stage of execution, performing a projective measurement that obtains one of k outcomes, and then resuming \mathcal{A} . Then, $\Pr[\mathcal{A}_0(1^n) = x] \geq \Pr[\mathcal{A}(1^n) = x]/k$.*

► **Lemma 4.** Let A and $\{B_i\}_{i=1}^n$ be operators, acting on the same space, with $\|A\|_\infty, \|B_i\|_\infty \leq 1$. Then $\|[A, \prod_{i=1}^n B_i]\|_\infty \leq \sum_{i=1}^n \| [A, B_i] \|_\infty$.

► **Lemma 5.** Let X and Y be two n -qubit quantum systems and let $P_{XY}^- = \sum_{x \in \{0,1\}^n} |x\rangle\langle x|_X \otimes |x\rangle\langle x|_Y$ be the projector onto the subspace spanned by those computational basis vectors where the two registers are equal. Let $\Phi = |\Phi\rangle\langle\Phi|$ denote the projector onto the uniform superposition, see Equation (1). Then $\|P_{XY}^- \Phi_Y\|_\infty = 2^{-n/2}$.

By applying the triangle inequality, Lemma 5 implies $\|[P_{XY}^-, \Phi_Y]\|_\infty \leq 2 \cdot 2^{-n/2}$.

Hash-based one-time signature schemes. Hash-based signature schemes [21, 24] are digital signature schemes whose security relies on cryptographic hash functions. In this paper, we study hash-based one-time signatures (OTSs), i.e. schemes that use a pair of keys for a single message. Below we introduce the Lamport and Winternitz OTSs.

The **Lamport OTS** is the simplest hash-based OTS. It uses a hash function $h : \{0, 1\}^n \rightarrow \{0, 1\}^n$ for key generation and verification and is defined as follows:

1. *Parameters:* Security parameter $n \in \mathbb{N}$ and message length $l \in \mathbb{N}$.
2. *Key generation algorithm (KeyGen):* On receiving the security parameter n in unary, KeyGen outputs a secret signing key $\text{sk} = (s_i^j)_{i=1, \dots, l}^{j=0, 1}$ with $s_i^j \xleftarrow{\$} \{0, 1\}^n$ and a public verification key $\text{pk} = (p_i^j)_{i=1, \dots, l}^{j=0, 1}$ where $p_i^j = h(s_i^j) \in \{0, 1\}^n$.
3. *Signature algorithm (Sign_{sk}):* On input message $m = m_1 \dots m_l \in \{0, 1\}^l$ of length l , Sign_{sk} outputs $\text{Sign}_{\text{sk}}(m) = \sigma = \sigma_1 \dots \sigma_l$ where $\sigma_i = s_i^{m_i} \in \{0, 1\}^n$.
4. *Verification procedure (Ver_{pk}):* Upon receiving a message m and a signature $\sigma = \sigma_1 \dots \sigma_l$, Ver_{pk} outputs **acc** if $h(\sigma_i) = p_i^{m_i}$ for all $i \in \{1, \dots, l\}$, and **rej** otherwise.

The **Winternitz OTS** was introduced by Merkle [24]. In this work, we study a variant that uses a hash function $h : \{0, 1\}^n \rightarrow \{0, 1\}^n$ and is defined as follows:

1. *Parameters:* Security parameter n , binary message length a , and the Winternitz parameter $w \geq 2$. Based on parameters a and w we define

$$l_1 = \lceil a / \log(w) \rceil, \quad l_2 = \lfloor \log(l_1(w-1)) / \log(w) \rfloor + 1, \quad l = l_1 + l_2. \quad (2)$$

2. *Key generation algorithm (KeyGen):* On receiving the security parameter n , choose uniformly at random l values that form the signing key $\text{sk} = (s_1, \dots, s_l) \xleftarrow{\$} (\{0, 1\}^n)^l$. Then, compute the public verification key $\text{pk} = (p_1, \dots, p_l) = (h^{w-1}(s_1), \dots, h^{w-1}(s_l))$.
3. *Signature algorithm (Sign_{sk}):* For a given input message $x \in \{0, 1\}^a$ and secret key sk , convert x to base w : $m = (b_1, \dots, b_{l_1})$ where $b_i \in \{0, \dots, w-1\}$. Next, compute the checksum $C(m) = \sum_{i=1}^{l_1} (w-1-b_i)$ and convert it to base w : $C(m) = (b_{l_1+1}, \dots, b_l)$. The reader may refer to [12] for more details on the checksum. Then set $b(m) = (b_1, \dots, b_l) = m \parallel C(m)$. The signature is then computed as $\sigma = (\sigma_1, \dots, \sigma_l) = (h^{b_1}(s_1), \dots, h^{b_l}(s_l))$.
4. *Verification algorithm (Ver_{pk}):* Given input message m , signature σ and public verification key pk , compute (b_1, \dots, b_l) as described above and output **acc** if $h^{w-1-b_i}(\sigma_i) = p_i$ for all $i \in \{1, \dots, l\}$, and **rej** otherwise.

Blind unforgeability. *Blind unforgeability* (BU) [2] is a quantum-access replacement for EU-CMA introduced in [16]. It uses the concept of *blinding*. Let $f : X \rightarrow Y$ be a function and $B \subset X$ a subset of X . The *blinded* function Bf with respect to the *blinding set* B is defined as $Bf(x) = \perp$ if $x \in B$ and $Bf(x) = f(x)$ otherwise, where \perp is a special blinding symbol. One concrete way to instantiate this is by means of an extra bit: given a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$, we define $Bf : \{0, 1\}^n \rightarrow \{0, 1\}^{m+1}$ by setting $\perp = 0^n \parallel 1$ and replacing $f(x)$ by $f(x) \parallel 0$. We refer to B^c as the set of *unblinded* messages.

Let $S = (\text{KeyGen}, \text{Sign}, \text{Ver})$ be a digital signature scheme with a security parameter n and message space M . Let \mathcal{A} be an adversary and let $\epsilon : \mathbb{N} \rightarrow \mathbb{R}_+$ be a negligible function. We define the *blind forgery* experiment $\text{BlindForge}_{S, \mathcal{A}}(n, \epsilon)$ as follows:

- *Key generation*: $(\text{sk}, \text{pk}) \leftarrow \text{KeyGen}(1^n)$.
- *Generation of blinding set*: Select $B \subseteq M$ by choosing each $m \in M$ independently at random with probability $\epsilon(n)$ provided by the adversary \mathcal{A} .
- *Forgery*: $(m, \sigma) \leftarrow \mathcal{A}^{B, \text{Sign}_{\text{sk}}}(1^n)$.
- *Outcome*: Win if $\text{Ver}_{\text{pk}}(m, \sigma) = \text{acc}$ and $m \in B$, and lose otherwise.

► **Definition 6** (Blind unforgeability (BU)). *A digital signature scheme S is q-BU secure if for any adversary \mathcal{A} making at most q queries to $B, \text{Sign}_{\text{sk}}$ and for all ϵ , the success probability of winning the blind forgery experiment is negligible in the security parameter n .*

3 Hash chains in the QROM

3.1 Quantum hash chain sampling

In this section, we introduce hash chains and describe a technical tool consisting of modeling hash chains as independent uniform superposition states akin to Zhandry’s compressed oracle technique [28]. This technique will enable us to prove BU security for the Lamport and Winternitz OTSs. *Hash chain* is a sequences of strings obtained by iteratively applying a hash function. They provide key pairs for the Lamport and Winternitz OTS’.

In the (Q)ROM, to generate a hash chain based on a hash function h , we first sample an initial string s_0 uniformly at random and then compute $s_i = h(s_{i-1})$ for $i = 1, \dots, w-1$ to obtain a hash chain of length w . For key generation in the Lamport and Winternitz OTSs, the secret key sk is, respectively, a tuple of $2l$ and l initial strings sampled uniformly at random in the domain $\{0, 1\}^n$. Then a tuple of hash chains $\gamma = (\gamma_i^j)_{i=1, \dots, l}^{j=0, \dots, w-1}$ is obtained by querying the hash function h on each string of the secret key $w-1$ times:

$$\gamma_i^0 = s_i, \quad \gamma_i^j = h^j(\gamma_i^0), \quad p_i = \gamma_i^{w-1} = h^{w-1}(\gamma_i^0), \quad j = 0, \dots, w-1, \quad i = 1, \dots, l,$$

where w is the length of the hash chain ($w = 2$ for Lamport) and l is the number of hash chains. The final entry of each chain is used as a public key.

In the BlindForge game, the secret key is only used by the blinded signing oracle. When analyzing this game, we can thus modify the key generation, signing and random oracle algorithms in an arbitrary way, as long as the modified triple is indistinguishable from the real one to an adversary.

In the proofs in Sections 4 and 5 we make use of the following modified triple, which we will refer to as defining the **Quantum independent world**. We construct the secret key and the intermediate hash chain elements initially in uniform superposition. That is, we prepare each hash chain register $(\Gamma_i^j)_{i=1, \dots, l}^{j=0, \dots, w-2}$ in the uniform superposition state $|\Phi\rangle$, with the intention of measuring them to sample the strings γ_i^j in mind. Then, we sample the final hash chain at random. The random oracle is then “reprogrammed in superposition” to be approximately consistent with the hash chains.

We proceed to show that the way of implementing the hash chain and the random oracle in the **Real world** and in the **Quantum independent world** are indistinguishable. For that purpose, we first formally define both worlds and some intermediate worlds between them. Each world is specified by two oracles, H and Sign , replacing the random oracle h and the signing oracle in the **Real world** (in each world, the KeyGen algorithm is implicitly replaced by the setup described below that generates the initial state and the public key). The oracles of the **Quantum independent world** are described below as well.

Real world. In the Real world, the first element γ_i^0 of each hash chain γ_i is generated at random and the hash function is evaluated to generate the rest of the hash chain, i.e., $\gamma_i^j = h^j(\gamma_i^0)$. Here, the random oracle is implemented at random, i.e. $H = h$, and the Sign oracle uses the secret key sk consisting of the γ_i^0 .

Intermediate world 1. Here, the first hash chain element is generated at random and the following elements are successively sampled uniformly except for the collision tuples. That is $s_i = \gamma_i^0 \xleftarrow{\$} \{0, 1\}^n$ and γ_i^1 is uniform except for the cases where $\gamma_i^1 = \gamma_{i'}^1$ if $\gamma_i^0 = \gamma_{i'}^0$; γ_i^2 is uniform except for the cases where $\gamma_i^2 = \gamma_{i'}^2$ if $\gamma_i^1 = \gamma_{i'}^1$; $\gamma_{i'}^2 = \gamma_i^2$ if $\gamma_i^1 = \gamma_{i'}^1$, etc. This world is very similar to the Real world, the only difference is that here we first sample the secret and public key (hash chain), then we reprogram the random oracle according to the secret and public key that we sampled, i.e. whenever the input to the random oracle is equal to a hash chain element γ_i^j with $j \leq w - 2$, we return γ_i^{j+1} , otherwise answer with the actual random oracle. The Sign oracle is the same as in the Real world.

Intermediate world 2. In this world, the hash chain elements γ_i^j are first sampled uniformly at random with possible collision tuples. It means that the γ_i^j are uniformly independent strings. Afterwards, the random oracle is reprogrammed to be consistent with the secret and public keys. When queried, it compares the input with the hash chain. If the input is not equal to any of the hash chain elements, the oracle answers with a random function \hat{h} . Otherwise, for each hash chain element the input is equal to, it XORs the next hash chain element into the output register. If there are two hash chain elements that are the same, the random oracle XORs both following hash chain elements into the output register. In this case, the Sign oracle uses the full list of hash chains $(\gamma_i^j)_{i=1, \dots, l}^{j=0, \dots, w-1}$ to answer the query with all the hash chain elements consistent with the input.

Quantum independent world. In this world, the hash chain registers $(\Gamma_i^j)_{i=1, \dots, l}^{j=0, \dots, w-2}$ are initially prepared in the uniform superposition $|\Phi\rangle$, and the last hash chain elements $(\gamma_i^{w-1})_{i=1, \dots, l}$ are sampled uniformly at random. The random oracle is constructed in such a way that it is compatible with the hash chain. When queried with register X and Y , the random oracle compares the X and Γ registers, then answers the query in the Y register. Abstractly speaking, H is implemented as in the Intermediate world 2, except that the comparison and XOR operations involving γ_i^j are replaced by controlled unitary operations with Γ as the control register. It can be expanded as³

$$(U_h)_{XY\Gamma} = \left(\prod_{i=1}^l \prod_{j=0}^{w-2} (U_i^j)_{XY\Gamma_i^j\Gamma_i^{j+1}} \right) U_{XY\Gamma}^\neq, \quad (3)$$

where the unitaries U_{ij} apply CNOT from register Γ_i^{j+1} into Y , controlled on registers X and Γ_i^j being equal, and U^\neq uses the actual random oracle in case X is not equal to any of the registers Γ_i^j . The signing oracle on the other hand just uses the superposition hash chain elements by means of a controlled unitary with control register Γ . For a detailed mathematical description, see the full version [23].

³ Note that the ordering of the product is unimportant because the operators U_i^j commute.

3.2 Indistinguishability

The following lemma allows us to conclude the indistinguishability of the Real world and the Quantum independent world.

► **Lemma 7.** *Let p and q be output distributions over n -bit strings of an algorithm \mathcal{A} interacting with the Real and the Quantum independent world, respectively. Then $\|p - q\|_1 \leq 3(wl)^2/2^n$.*

This lemma follows from the following three results (see the full version [23] for proofs).

► **Lemma 8.** *The Real world and the Intermediate world 1 are indistinguishable.*

► **Lemma 9.** *The distribution p and q of hash chains in the Intermediate worlds 1 and 2 are close: $\|p - q\|_1 \leq 3(wl)^2/2^n$.*

► **Lemma 10.** *The way the random oracle is implemented in the Intermediate world 2 and in the Quantum independent world are indistinguishable.*

4 One-time BU security of the Lamport OTS

In the BlindForge experiment, the adversary has quantum access to both a blinded signing oracle and a random oracle. For one-time signature schemes, the adversary is allowed only to query the signing oracle at most once. So, to produce a forged message-signature pair, the adversary can make a desired number of quantum queries to the random oracle, then query the signing oracle once, and then query again the random oracle as many times as desired. Our goal is to prove that the probability that an adversary outputs a correct forged signature on a valid forged message is negligible.

In the Lamport OTS, the signature algorithm uses only half of the secret key to produce the signature. Classically, the property that enables security is that the adversary does not have any information about the other half, the *invariant*, of the secret key. Quantumly, since in the BlindForge experiment the forged message must be outside the queried region, for any queried message there exists at least one bit in which the forged and queried messages differ. Thus, the secret key corresponding to that bit should still be in its initial state. To show blind-unforgeability, we separately analyze three cases: *hash queries before Sign query*, *Sign query*, and *hash queries after Sign query*. We describe below our proof strategy in these cases on a high level.

For *hash queries before Sign query*, we know that before any query the entire secret key is in uniform superposition. We therefore define a projector of the secret key register being in uniform superposition, and show that this projector approximately commutes with the random oracle unitary. This means that after a moderate number of queries, the secret key registers will still be in uniform superposition, indicating that the adversary learns almost no information about the secret key.

In the *Sign query* case, the first step is to track the unused part of the secret key. This part can be easily determined in the classical setting since the adversary queries only one message in each query. In contrast, in the quantum setting we consider quantum queries and hence have to track the invariant in superposition over the different queried messages. This is difficult because the invariant is different within each term of the superposition, so we cannot simply describe the invariant for the whole state. We address this problem as follows. We define an *invariant projector* that tracks the invariance of the unused superposition-secret-keys under queries and show that this projector is orthogonal to the projector corresponding

to the outcome where *none of the secret key registers relevant to the forged signature belong to the invariant*. Then, we show that if there is only **Sign** query, this new projector does not change the adversary state immediately after the signature. We also establish that if the adversary state after forgery is in the range of this new projector, then the adversary has negligible probability to win the **BlindForge** game. Besides, we prove that the new projector approximately commutes with the random oracle unitary.

Finally, for the case of *hash queries after Sign query*, we use the latter argument of the commutator to prove that after hash queries the final adversary state remains roughly in the image of the invariant projector of the secret key.

The arguments from these three cases together constitute a proof of the following theorem.

► **Theorem 11.** *The Lamport OTS is 1-BU secure if the hash function h is modeled as a quantum-accessible random oracle. More precisely, let \mathcal{A} be an adversary that plays the **BlindForge** game for the Lamport OTS, making a total of q queries to the random oracle. Then \mathcal{A} succeeds with a probability bounded as*

$$\Pr[\mathcal{A} \text{ wins BlindForge}] \leq l^2 \cdot 2^{-n} (3137q^2(l+1) + 12) \leq 6286q^2l^3 \cdot 2^{-n}, \quad (4)$$

where n is the security parameter of the Lamport OTS, l is the message length, and the simplified bound holds for $q > 0$.

We present a proof of this result in subsequent sections. In particular, we prove it in the Quantum independent world first, and then conclude the statement in the Real world via an application of Lemma 7. In the remainder of the article, we use a subscript QI to indicate that a probability statement holds in the Quantum independent world.

4.1 Q measurement for Lamport OTS

We begin by presenting some concepts and tools which will be used in the proof. Subsequently, we prove the steps outlined above as separate lemmas. Our proof will make use of a projective measurement to track an invariant on the Quantum independent world secret key register for the verification of the forged message in the case of no hash queries. Let (m^*, σ^*) be a forged message-signature pair with $\sigma^* = s_1^{m^*} \cdots s_l^{m^*}$, where $(s_i^j)_{i=1, \dots, l}^{j=0,1}$ is the secret key and l is the message length.

For any message $m^* \in \{0, 1\}^l$ we define an $(l+1)$ -outcome projective measurement that finds the smallest index $i^* \in \{1, \dots, l\}$ for which the register $S_{i^*}^{m^*}$ is in uniform superposition, or determines that *none of the relevant secret key registers are in uniform superposition* (this corresponds to the outcome $l+1$). We define projectors $Q_{i^*}^{m^*}$ with $i^* \in \{1, \dots, l\}$ in terms of projectors $\Phi = |\Phi\rangle\langle\Phi|$ and $\Phi^\perp = I - |\Phi\rangle\langle\Phi|$ placed onto different registers depending on the message m^* (they act as I on all other registers S_i^j that are not specified):

$$Q_{i^*}^{m^*} = \Phi_{S_1^{m^*}}^\perp \otimes \cdots \otimes \Phi_{S_{i^*-1}^{m^*}}^\perp \otimes \Phi_{S_{i^*}^{m^*}}, \quad Q_{l+1}^{m^*} = \bigotimes_{i=1}^l \Phi_{S_i^{m^*}}^\perp. \quad (5)$$

4.2 Invariant projector

In this section we define a projector P_S that will be useful for our analysis, and state some of its properties as lemmas.

Let $\alpha = (\alpha_i^j)_{i=1, \dots, l}^{j=0,1}$ be a $2l$ -bit string whose each bit $\alpha_i^j \in \{0, 1\}$ indicates that the projector $\Phi(\alpha_i^j)$ is applied on the corresponding secret key register S_i^j where $\Phi(0) = \Phi$ and $\Phi(1) = \Phi^\perp$. For each string α , we define the associated projector $\Phi(\alpha)$ on the whole secret key register S as $\Phi(\alpha)_S = \bigotimes_{i=1}^l \bigotimes_{j=0}^1 \Phi(\alpha_i^j)_{S_i^j}$. Note that $\sum_{\alpha \in \{0,1\}^{2l}} \Phi(\alpha)_S = I_S$.

21:12 Quantum-Access Security of the Winternitz One-Time Signature Scheme

Since we are interested in the unused part of the secret key register S , we need to filter those α 's for which S_i^j is in state $|\Phi\rangle$. Recall from our discussion of blind unforgeability in Section 2 that B denotes the set of blinded messages. Since the blinded signing oracle has signed (at most) a single, un-blinded message, the state after the oracle call can be written as a superposition of states where, for some un-blinded message $m \in B^c$, the secret key register of the complementary value \bar{m}_i is still in the uniform superposition $|\Phi\rangle$, for all i . We collect all strings α that are consistent with no blinded messages having been signed in $\widehat{B^c} = \bigcup_{m \in B^c} \{\alpha \in \{0,1\}^{2l} \mid \alpha_i^{\bar{m}_i} = 0 \text{ for all } i = 1, \dots, l\}$. These strings indicate which secret key registers were not used during hash queries and **Sign** query. Finally, we define $P_S = \sum_{\alpha \in \widehat{B^c}} \Phi(\alpha)_S$ as the projector onto the subspace compatible with $\widehat{B^c}$. Note that P_S is indeed a projector since it is a sum of mutually orthogonal projectors.

We proceed to state several lemmas used to prove our main results both for the Lamport and Winternitz OTSs. Proofs of these lemmas are provided in the full version [23].

The first lemma says that hash queries do not affect the secret key registers significantly as long as they are in their initial state Φ .

► **Lemma 12.** *Let U_h be the random oracle unitary for any given function h (see Section 3) and let $\Phi = |\Phi\rangle\langle\Phi|$ denote the projector onto the uniform superposition. Then, for any $i \in \{1, \dots, l\}$ and $j \in \{0, 1\}$, $\| [U_h, \Phi_{S_i^j}] \|_\infty \leq 6/2^{n/2} = \epsilon_L(n)$ is negligible in n .*

The quantum analogue of the following property holds: signing a message m^* requires at least one secret key string that was not used to sign $m \neq m'$.

► **Lemma 13.** *For all $m^* \in B$, the projectors $Q_{l+1}^{m^*}$ defined in Equation (5) and P_S are orthogonal.*

The projector P_S defined above is an invariant of the secret key registers after a signing query but no hash queries.

► **Lemma 14.** *Let $B \text{Sign}_{\text{sk}}$ be the blinded signing oracle for the Lamport OTS and let $|\psi_0\rangle$ be the adversary's state before the **Sign** query. If there are no hash queries, $P_S B \text{Sign}_{\text{sk}} |\psi_0\rangle = B \text{Sign}_{\text{sk}} |\psi_0\rangle$.*

The invariant specified by P_S approximately holds also after hash queries.

► **Lemma 15.** *The invariant projector P_S defined above and the random oracle unitary U_h defined in the Quantum independent world approximately commute, i.e., $\| [U_h, P_S] \|_\infty \leq \delta_L(n)$, where $\delta_L(n) = 32l/2^{n/2}$ is negligible in n .*

In the following sections, we use the above lemmas to analyze the situation where the adversary makes q_0 hash queries before the **Sign** query and q_1 hash queries after. Maximizing the resulting bound under the condition $q_0 + q_1 = q$ gives Theorem 11.

4.3 Hash queries before Sign query

In this section, we study the impact of hash queries before **Sign** query on the secret key register S . Our main goal is to show that, for a moderate number of queries to the random oracle, no adversary can learn a significant amount of information about the secret key. Therefore, she cannot produce a valid forgery except with a small probability.

Let $|\psi\rangle_{XYM\Sigma E}$ be adversary's initial state before any queries (see Table 1 for a summary of registers and their roles). Before any query is performed, the whole secret key register S is in the uniform superposition state $|\Phi\rangle^{\otimes 2l}$. Assume the adversary \mathcal{A}_0 queries the random

■ **Table 1** Registers used in the analysis.

Register	Meaning
X	adversary's input
Y	adversary's output
M	Sign query input
Σ	Sign query output
E	adversary's internal workspace
S	secret key

oracle q_0 times before querying the signing oracle. If V_{XYE}^i denotes the unitary she performs after the i -th query, the final adversary state after q_0 hash queries is

$$|\psi_0\rangle_{XYM\Sigma ES} = V_{XYE}^{q_0}(U_h)_{XYS} V_{XYE}^{q_0-1} \cdots V_{XYE}^2(U_h)_{XYS} V_{XYE}^1(U_h)_{XYS} |\psi\rangle_{XYM\Sigma E} |\Phi\rangle_S^{\otimes 2l} \quad (6)$$

where U_h is the random oracle unitary that answers hash queries. The following lemma shows that secret key registers of this state are still close to the uniform superposition.

► **Lemma 16.** *In the Quantum independent world, without querying the B Sign oracle, hash queries leave the state of the secret key registers approximately unchanged:*

$$\|\Phi_S^{\otimes 2l} |\psi_0\rangle_{XYM\Sigma ES} - |\psi_0\rangle_{XYM\Sigma ES}\|_2 \leq 2lq_0\epsilon_L(n).$$

Proof. We want to show that after q_0 hash queries, the state of the secret key register S is still approximately in the uniform superposition state $|\Phi\rangle_S^{\otimes 2l}$. Let us abbreviate the overall unitary in Equation (6) by W_{XYES} . Since the only operations in W_{XYES} that act on the S register are the hash queries U_h , and they are in fact controlled by the S register, we have $W_{XYES}\Phi_S^{\otimes 2l} = W_{XYES}$. Using this, we get

$$\begin{aligned} & \|\Phi_S^{\otimes 2l} |\psi_0\rangle_{XYM\Sigma ES} - |\psi_0\rangle_{XYM\Sigma ES}\|_2 \\ &= \|\Phi_S^{\otimes 2l} W_{XYES} |\psi\rangle_{XYM\Sigma E} |\Phi\rangle_S^{\otimes 2l} - W_{XYES} \Phi_S^{\otimes 2l} |\psi\rangle_{XYM\Sigma E} |\Phi\rangle_S^{\otimes 2l}\|_2 \end{aligned} \quad (7)$$

$$= \|\Phi_S^{\otimes 2l}, W_{XYES}\| |\psi\rangle_{XYM\Sigma E} |\Phi\rangle_S^{\otimes 2l}\|_2 \quad (8)$$

$$\leq \|\Phi_S^{\otimes 2l}, W_{XYES}\|_\infty \underbrace{\|\psi\rangle_{XYM\Sigma E} |\Phi\rangle_S^{\otimes 2l}\|_2}_{=1} \quad (9)$$

$$= \|\Phi_S^{\otimes 2l}, V_{XYE}^{q_0}(U_h)_{XYS} V_{XYE}^{q_0-1} \cdots V_{XYE}^2(U_h)_{XYS} V_{XYE}^1(U_h)_{XYS}\|_\infty \quad (10)$$

$$\leq q_0 \|\Phi_S^{\otimes 2l}, (U_h)_{XYS}\|_\infty + \sum_{i=1}^{q_0} \|\Phi_S^{\otimes 2l}, V_{XYE}^i\|_\infty, \quad (11)$$

where Equation (9) follows from the definition of the operator norm and the last inequality follows from Lemma 4.

The first term in Equation (11) can be bounded as follows:

$$\|\Phi_S^{\otimes 2l}, (U_h)_{XYS}\|_\infty \leq \sum_{\substack{i \in \{1, \dots, l\} \\ j \in \{0, 1\}}} \|\Phi_{S^i}^{\otimes 2l}, (U_h)_{XYS}\|_\infty \leq 2l\epsilon_L(n),$$

which follows by first applying Lemma 4 and then Lemma 12. Since $\Phi_S^{\otimes 2l}$ and V_{XYE}^i act on different registers, they commute and the second term in Equation (11) vanishes. Hence

$$\|\Phi_S^{\otimes 2l} |\psi_0\rangle_{XYM\Sigma ES} - |\psi_0\rangle_{XYM\Sigma ES}\|_2 \leq 2lq_0\epsilon_L(n). \quad \blacktriangleleft$$

4.4 Query to the signing oracle

Now that we have control over the advantage an adversary can gain from making hash queries before the sign query, we need to analyze the possible advantage from hash queries after the sign query and bound the overall success probability using Lemma 16.

A crucial property of the Lamport OTS when analyzing classical security is that for all messages m that have not been queried, there exists an index j such that $s_j^{m_j}$ is hidden from the adversary by the one-wayness of the used hash function. In blind-unforgeability (for classical adversaries), this property holds for all *blinded messages*. In the setting of quantum queries, we have to track this property in superposition while the adversary is making hash queries after the sign query. As this is complicated by the “for all”-quantifier, we begin by analyzing the case where the adversary makes no hash queries after the sign query to ease the reader into our proof technique.

The discussion in this section does not concern the random oracle, so we absorb the random oracle query registers XY into E for the purpose of this section. In the 1-BlindForge game, an adversary \mathcal{A} is allowed to query the Sign-oracle at most once to produce a valid forged message-signature pair (m^*, σ^*) . To analyze the interaction between \mathcal{A} and the signing oracle, we will break it into the following steps:

$$|\psi_0\rangle_{M\Sigma BES} \xrightarrow{B \text{ Sign}_{\text{sk}}} |\psi_1\rangle_{M\Sigma BES} \xrightarrow{U_{M\Sigma E}} |\psi_2\rangle_{M\Sigma BES} \xrightarrow{\langle m^* |_M} |\psi_3(m^*)\rangle_{\Sigma BES} \xrightarrow{\langle \sigma^* |_\Sigma} |\psi_4(m^*, \sigma^*)\rangle_{BES}.$$

They correspond to applying the Sign-oracle and an arbitrary unitary $U_{M\Sigma E}$, followed by measuring the message and signature registers M and Σ . Let us now analyze these steps in more detail and write down the corresponding quantum states.

First, \mathcal{A} prepares her input state as an arbitrary superposition of messages:

$$|\psi_0\rangle_{M\Sigma BES} = \left(\sum_{m \in \{0,1\}^l} \sum_{\sigma \in (\{0,1\}^n)^l} \sum_{b \in \{0,1\}} \kappa_{m\sigma b} |m\rangle_M |\sigma\rangle_\Sigma |b\rangle_B |\alpha_{m\sigma b}\rangle_E \right) \otimes (|\Phi\rangle^{\otimes 2l})_S \quad (12)$$

where the B register indicates whether the message is blinded or not ($|1\rangle_B$ for blinded and $|0\rangle_B$ for un-blinded). The adversary then supplies this to the Sign oracle which produces the following signed state:

$$|\psi_1\rangle_{M\Sigma BES} = B \text{ Sign}_{\text{sk}} |\psi_0\rangle_{M\Sigma BES} = |\psi_1^1\rangle_{M\Sigma BES} + |\psi_1^0\rangle_{M\Sigma BES} \quad (13)$$

where superscripts 1 and 0 refer to blinded (B) and un-blinded (B^c) messages, respectively:

$$\begin{aligned} |\psi_1^1\rangle_{M\Sigma BES} &= \sum_{m \in B} \sum_{\sigma \in (\{0,1\}^n)^l} \kappa_{m\sigma 1} |m\rangle_M |\sigma\rangle_\Sigma |1\rangle_B |\alpha_{m\sigma 1}\rangle_E |\Phi\rangle_S^{\otimes 2l}, \\ |\psi_1^0\rangle_{M\Sigma BES} &= \sum_{m \in B^c} \sum_{\sigma \in (\{0,1\}^n)^l} \frac{1}{2^{nl/2}} \sum_{s \in (\{0,1\}^n)^l} \kappa_{m\sigma 0} |m\rangle_M |\sigma \oplus s\rangle_\Sigma |0\rangle_B |\alpha_{m\sigma 0}\rangle_E |\Omega(s, m)\rangle_S, \end{aligned}$$

where $m = m_1 \dots m_l$, $\sigma = \sigma_1 \dots \sigma_l$, and

$$|\Omega(s, m)\rangle_S = |s_1^{m_1}\rangle_{S_1^{m_1}} \dots |s_l^{m_l}\rangle_{S_l^{m_l}} |\Phi\rangle_{S_1^{\bar{m}_1}} \dots |\Phi\rangle_{S_l^{\bar{m}_l}}. \quad (14)$$

Once the adversary \mathcal{A} gets the signed state $|\psi_1\rangle_{M\Sigma BES}$, she performs some operations with the intention of producing a forgery message m^* . Intuitively, those operations can be considered as applying an arbitrary unitary $U_{M\Sigma E}$ to $|\psi_1\rangle_{M\Sigma BES}$. Let us denote the resulting state by $|\psi_2\rangle_{M\Sigma BES} = U_{M\Sigma E} |\psi_1\rangle_{M\Sigma BES}$.

Next, \mathcal{A} measures the registers M and Σ to produce a forgery candidate (m^*, σ^*) , collapsing the state to $|\psi_4(m^*, \sigma^*)\rangle_{BES} = \langle \sigma^* |_{\Sigma} |\psi_3(m^*)\rangle_{\Sigma BES} = \langle m^* |_M \langle \sigma^* |_{\Sigma} |\psi_2\rangle_{M\Sigma BES}$. Similar to Equation (13), we can split the final (unnormalized) post-measurement state as

$$|\psi_4(m^*, \sigma^*)\rangle_{BES} = |\psi_4^1(m^*, \sigma^*)\rangle_{BES} + |\psi_4^0(m^*, \sigma^*)\rangle_{BES}$$

where $|\psi_4^i(m^*, \sigma^*)\rangle_{BES} = \langle m^* |_M \langle \sigma^* |_{\Sigma} U_{M\Sigma E} |\psi_1^i\rangle_{M\Sigma BES}$. We can rewrite $|\psi_4^0\rangle_{BES}$ as

$$|\psi_4^0(m^*, \sigma^*)\rangle_{BES} = \sum_{m \in B^c} \frac{1}{2^{nl/2}} \sum_{s \in (\{0,1\}^n)^l} |\eta(m, s)\rangle_{BE} |\Omega(s, m)\rangle_S \quad (15)$$

where only $|\eta(m, s)\rangle_{BE}$ depends on m^* and σ^* :

$$|\eta(m, s)\rangle_{BE} = \sum_{\sigma \in (\{0,1\}^n)^l} \kappa_{m\sigma 0} \langle m^* |_M \langle \sigma^* |_{\Sigma} U_{M\Sigma E} |m\rangle_M |\sigma \oplus s\rangle_{\Sigma} |0\rangle_B |\alpha_{m\sigma 0}\rangle_E.$$

Finally, the adversary \mathcal{A} outputs the measurement outcome (m^*, σ^*) as a forged message-signature pair. The probability of producing this pair is $\| |\psi_4^0(m^*, \sigma^*)\rangle_{BES} \|^2$.

The next step is to analyse the probability that \mathcal{A} 's forgery candidate (m^*, σ^*) is correct. For that purpose, we consider two cases. The first case, namely when $m^* \notin B$, is trivial since then \mathcal{A} has lost the **BlindForge** experiment because m^* must be blinded by definition. The rest of this section is devoted to analyzing the second case.

If $m^* \in B$, the forged message m^* has not been signed since the blinded signing oracle signs only un-blinded messages. Hence, for any message $m \notin B$, there exists at least one index $i \in \{1, \dots, l\}$ such that $m_i \neq m_i^*$. This implies that for some index $i^* \in \{1, \dots, l\}$ the register $S_{i^*}^{m_i^*}$ has not been used for the signature of the adversary's queried message and is therefore still in the uniform superposition state $|\Phi\rangle$. Note that this holds only in superposition over m . Indeed, i^* depends on m and is in general different for each term of the superposition.

We break that superposition by analyzing a modified **BlindForge** experiment, where an additional measurement, the Q -measurement defined in Equation (5), is performed on the secret key register after the adversary has output their forgery, but before the secret key register is measured to actually sample the secret key as required in the **Quantum independent world**. Since the measurement has few outcomes, its effect on the adversary's winning probability is limited and can be bounded by the pinching lemma (Lemma 3).

If the Q -measurement yields outcome $i^* \in \{1, \dots, l\}$, then the secret key sub-register $S_{i^*}^{m_i^*}$ is in uniform superposition, and the adversary is bound to fail as σ^* is independent of the secret key string $s_{i^*}^{m_i^*}$ (the result of measuring $S_{i^*}^{m_i^*}$). Hence, it remains to analyze the outcome $l+1$ that corresponds to the projector $Q_{l+1}^m = (\Phi^\perp)^{\otimes l}$, see Equation (5), where $\Phi^\perp = I - |\Phi\rangle\langle\Phi|$ projects onto the orthogonal complement of $|\Phi\rangle$.

For the rest of our analysis, we fix the message m^* and focus on the un-blinded term $|\psi_4^0(m^*, \sigma^*)\rangle_{BES}$ whose expression is given by Equation (15). Given that for each $m \notin B$ there is at least one index $i \in \{1, \dots, l\}$ such that $m_i \neq m_i^*$, we define $i(m) = \min\{j \in \{1, \dots, l\} \mid m_j \neq m_j^*\}$ as the smallest index for which $m \neq m^*$. Intuitively, it is the first sub-register of S that still remains in uniform superposition. In the following, let $S(m) := S_1^{m_1} \cdots S_l^{m_l}$. We want to split the first sum in Equation (15) into l parts, one for each value of $i(m)$, so that we can easily evaluate $(\Phi^\perp)_{S(\bar{m})}^{\otimes l} |\psi_4^0(m^*, \sigma^*)\rangle_{BES}$. For that

21:16 Quantum-Access Security of the Winternitz One-Time Signature Scheme

purpose, we define $B_j^c = \{m \in B^c \mid i(m) = j\}$ and note that $\bigcup_{j=1}^l B_j^c = B^c$. We can now rewrite $|\psi_4^0(m^*, \sigma^*)\rangle_{BES}$ as

$$\begin{aligned} |\psi_4^0(m^*, \sigma^*)\rangle_{BES} &= \sum_{j=1}^l \sum_{m \in B_j^c} \frac{1}{2^{nl/2}} \sum_{s \in (\{0,1\}^n)^l} |\eta(m, s)\rangle_{BE} |s^m\rangle_{S(m)} |\Phi\rangle_{S(\bar{m})}^{\otimes l} \\ &= \sum_{j=1}^l |\hat{\eta}(m^*, \sigma^*, j)\rangle_{BES_{\{(j, m_j^*)\}^c}} |\Phi\rangle_{S_j^{m_j^*}}, \end{aligned} \quad (16)$$

where we absorbed all registers except for $S_j^{m_j^*}$ into the first system. The remaining register $S_j^{m_j^*}$ is still in the uniform superposition $|\Phi\rangle$ since $j = i(m)$ is the smallest index such that $m_j \neq m_j^*$. Applying Q_{l+1} hence clearly maps the state to zero, and so the situation where none of the secret key sub-registers relevant for the verification of the forged signature σ^* is in state $|\Phi\rangle$ can ever occur.

Now, we execute the last part of the **BlindForge** experiment which consists of checking the correctness of the forged signature σ^* . For this purpose, we perform a computational basis measurement on the entire secret key register S to sample the strings s_i^j . As mentioned above, the probability of (m^*, σ^*) being valid is at most 2^{-n} as $s_i^{m_i}$ is independent of σ_i^* , where i is the outcome of the Q -measurement. Applying the pinching lemma (Lemma 3) to relate the success probabilities with and without Q -measurement, and Lemma 7 for $w = 2$ to relate the success probabilities in the Real world and the Quantum independent world (see details in the full version [23]), we arrive at

$$\Pr[\mathcal{A} \text{ wins BlindForge}] \leq \frac{l+1}{2^n} + 12l^2 \cdot 2^{-n}. \quad (17)$$

Hence, the success probability of the adversary \mathcal{A} in winning the **BlindForge** experiment game is at most $(l+1)/2^n$, which is negligible since l is polynomial in n , and n is large enough. We conclude that a **Sign** query does not help the adversary to get significant information about the secret key.

4.5 Hash queries after Sign query

To complete the proof of Theorem 11 and bound the success probability an adversary can achieve in the **BlindForge** game with a given number of queries, it remains to analyse *hash queries after Sign query*. In this case, it is not obvious how to track the secret key invariant (the fact that there is at least one unused part of the secret key that is relevant for the forged signature). Therefore we use a special projector P_S that projects onto the subspace of the secret key register that is consistent with a single blinded sign query and no hash queries. If the final adversary state after producing the forgery candidate is in the image of P_S , then according to Lemma 13 the outcome $l+1$ corresponding to the situation when *none of the secret key sub-registers useful for the forged signature is in state $|\Phi\rangle$* can never occur. We thus want to show that adversary's final state is approximately in the range of P_S .

If there are no hash queries before the **Sign** query, then from Lemma 14 the adversary state after the **Sign** query remains completely in the range of P_S , which means that the outcome $l+1$ cannot occur. That is, $P_S|\psi_1\rangle = P_S B \text{Sign}_{\text{sk}} |\psi_0\rangle = B \text{Sign}_{\text{sk}} |\psi_0\rangle = |\psi_1\rangle$ where $|\psi_0\rangle$ and $|\psi_1\rangle$ are adversary's states immediately before and after the **Sign** query.

Now, assuming there are hash queries before the **Sign** query, since the projector P_S and the random oracle unitary U_h approximately commute by Lemma 15, it follows that hash queries before **Sign** query give no significant information to the adversary about the invariant of the secret key register.

Suppose there are hash queries after the Sign query and let us examine in detail what happen in this case. From the previous case, we know that the adversary's state directly after the Sign query is $|\psi_1\rangle_{M\Sigma XYES}$. Just like for hash queries before the Sign query, suppose that the adversary makes q_1 hash queries after querying the signing oracle. Let $(W_{XYE}^i)_{i=1,\dots,q_1}$ be the unitaries applied between the hash queries. Then, let

$$|\psi'_1\rangle_{M\Sigma XYES} = (U_h)_{XYS} W_{XYE}^{q_1} (U_h)_{XYS} W_{XYE}^{q_1-1} \cdots W_{XYE}^2 (U_h)_{XYS} W_{XYE}^1 |\psi_1\rangle_{M\Sigma XYES}$$

be the adversary's state after q_1 hash queries and before performing some unitary operation $U_{M\Sigma E}$ on the post-hash-queried state, or any measurement leading to the forgery candidate.

► **Lemma 17.** *In the Quantum independent world, the state $|\psi'_1\rangle_{M\Sigma XYES}$ right before the adversary's measurement determining the forgery is applied is approximately in the range of P_S :*

$$\|P_S |\psi'_1\rangle_{M\Sigma XYES} - |\psi'_1\rangle_{M\Sigma XYES}\|_2 \leq q_1 \delta_L(n) + 4l q_1 \epsilon_L(n) = q_1 (\delta_L(n) + 4l \epsilon_L(n)). \quad (18)$$

The proof uses commutator arguments via Lemma 15 akin to the ones used in the proof of Lemma 16, and can be found in the full version [23].

Recall that, just like in Section 4.4, we want to analyze the modified BlindForge experiment where the Q -measurement is applied after the adversary has output a forgery, but before the secret key register is measured to sample the secret key and verify the forgery. It thus remains to show that due to the fact that $|\psi'_1\rangle$ is approximately in the range of P_S , the outcome $l+1$ only occurs with small probability.

To that end, we define a new measurement given by projectors \tilde{Q}_i that performs the Q -measurement controlled on the content of the M -register, i.e., $\tilde{Q}_i = \sum_m |m\rangle\langle m|_M \otimes Q_i^m$. Now, observe that applying the Q -measurement after the adversary has output a forgery is equivalent to applying the \tilde{Q} -measurement right before the adversary's measurement that produces the forgery. If $m^* \in B$, the outcome $l+1$ occurs only with small probability in the modified BlindForge experiment and it suffices to prove the following lemma.

► **Lemma 18.** *In the Quantum independent world, for blinded messages, the outcome $l+1$ occurs with small probability: $\|\tilde{Q}_{l+1} \Pi_M^B |\psi'_1\rangle_{M\Sigma XYES}\|_2 \leq q_1 (\delta_L(n) + 4l \epsilon_L(n))$, $\Pi^B = \sum_{m \in B} |m\rangle\langle m|$.*

The proof is a simple application of Lemma 13 and can be found in the full version [23]. We are now ready to combine our lemmas and prove Theorem 11.

Proof of Theorem 11. We begin by bounding the success probability of the adversary in the modified BlindForge experiment, in the Quantum independent world. Abbreviating the modified BlindForge experiment as MBF and writing "outcome i " to denote the event that the Q -measurement yields outcome i ,

$$\begin{aligned} \Pr_{QI,MBF} [\mathcal{A} \text{ succeeds}] &= \sum_{i=1}^{l+1} \Pr_{QI,MBF} [\mathcal{A} \text{ succeeds} \wedge \text{outcome } i] \\ &= \sum_{i=1}^l \Pr_{QI,MBF} [\mathcal{A} \text{ succeeds} \wedge \text{outcome } i] + \Pr_{QI,MBF} [\mathcal{A} \text{ succeeds} \wedge \text{outcome } l+1] \\ &\leq \sum_{i=1}^l \Pr_{QI,MBF} [\text{outcome } i] \times 2^{-n} + \Pr_{QI,MBF} [\text{outcome } l+1] \leq 2^{-n} + q^2 (\delta_L(n) + 4l \epsilon_L(n))^2, \end{aligned}$$

21:18 Quantum-Access Security of the Winternitz One-Time Signature Scheme

where the first inequality uses the fact that σ^* and $s_i^{m_i^*}$ are independent conditioned on outcome i , and the last inequality uses the square of the inequality from Lemma 18.

Exactly as in the simplified case in Section 4.4, we can bound the success probability in the actual `BlindForge` experiment using the pinching lemma (Lemma 3):

$$\Pr_{QI, \text{BlindForge}}[\mathcal{A} \text{ succeeds}] \leq (l+1) \left(2^{-n} + q^2 (\delta_L(n) + 4l\epsilon_L(n))^2 \right).$$

Finally, plugging in the functions $\epsilon_L(n)$ and $\delta_L(n)$ from Lemmas 12 and 15, and applying Lemma 7 for $w = 2$, we obtain

$$\begin{aligned} \Pr_{\text{BlindForge}}[\mathcal{A} \text{ succeeds}] &\leq (l+1) \left(2^{-n} + q^2 \left(\frac{32l}{2^{n/2}} + 4l \frac{6}{2^{n/2}} \right)^2 \right) + 12l^2 2^{-n} \\ &\leq l^2 \cdot 2^{-n} (3137q^2(l+1) + 12). \end{aligned} \quad \blacktriangleleft$$

5 One-time BU security of the Winternitz OTS

The Lamport OTS that we analyzed in the last section is, in some sense, a special case of the Winternitz OTS. Indeed, the Winternitz scheme for $w = 2$ is fairly similar to the Lamport OTS, except that the public key is used to sign the bits that are equal to 1, which is compensated for by the checksum encoding. As a result, the analysis of the Winternitz OTS in the QROM is, in a similar sense, a generalization of the one of the Lamport OTS.

Before getting started, we give an overview of our strategy. In this section, we use the same register labels as in Table 1, except that the secret key register S is now replaced by the hash chain register Γ . The security proof follows a similar outline as in the Lamport case. Some differences are as follows. After a Winternitz signing query, the adversary does not have any information about the part of the hash chain below the queried position, and this represents the invariant of the hash chain. Quantumly, this invariant has to be tracked in superposition like for the Lamport scheme, requiring the definition of a new and slightly more involved invariant projector (see details in the full version [23]).

► **Theorem 19.** *The Winternitz OTS is 1-BU secure if the function chain \mathcal{C} is modeled as a quantum-accessible random oracle. More precisely, let \mathcal{A} be an adversary that plays the `BlindForge` game for the Winternitz OTS, making a total of q queries to the random oracle. Then \mathcal{A} succeeds with a probability bounded as*

$$\Pr[\mathcal{A} \text{ wins } \text{BlindForge}] \leq 2^{-n} \left[(1 + q^2 l^2 (w-1)^2 (20w-4)^2) (l+1) + 3w^2 l^2 \right] \quad (19)$$

$$\leq 800w^4 q^2 l^3 \cdot 2^{-n}. \quad (20)$$

Here, l is the length of the encoded message in w -ary, see Equation (2), $w \geq 2$ is the Winternitz parameter, and the simplified bound in the last line holds for $q > 0$.

The main difference between the analyses of the Lamport and Winternitz OTS is as follows. For the Lamport OTS, the public key is obtained from the private key by applying a hash function once. For the Winternitz OTS, on the other hand, the secret and public keys consist of the start and end points of length- w hash chains, respectively. Thus, while following the same proof strategy, the Q projectors as well as the invariant projector P needs to be defined differently. Thus, we start our analysis by describing the Q projectors and the invariant projector for the Winternitz OTS.

The Q -measurement for the Winternitz OTS. The Winternitz signature of a message consists of l hash chain elements. In complete analogy to Equation (5) in Section 4.1, we define a measurement whose projectors correspond, respectively, to the events that *the i -th hash chain element relevant for the forged signature is in state $|\Phi\rangle$ and none of them is in state $|\Phi\rangle$* :

$$Q_{i^*}^{b^*} = \Phi_{\Gamma_1^{b_1^*}}^\perp \otimes \cdots \otimes \Phi_{\Gamma_{i^*-1}^{b_{i^*-1}^*}}^\perp \otimes \Phi_{\Gamma_{i^*}^{b_{i^*}^*}}, \quad Q_{l+1}^{b^*} = \bigotimes_{i=1}^l \Phi_{\Gamma_i^{b_i^*}}^\perp, \quad (21)$$

where $i^* \in \{1, \dots, l\}$, $b_i^* = b_i(m^*)$ and l is the number of blocks of the message and the checksum, see Equation (2). These operators act as I on all other registers Γ_i^j not specified.

The Invariant projector for the Winternitz OTS. In this section, we define the invariant projector P_Γ , the analogue of P_S for the Winternitz OTS. We also state several of its properties. Just like in Section 4.2, for any string $\alpha = (\alpha_i^j)_{i=1, \dots, l}^{j=0, \dots, w-2}$ we define an associated projector $\Phi(\alpha)$ on the whole hash chain (except for the last) register Γ . This is a complete set of projectors: $\sum_{\alpha \in \{0,1\}^{l(w-1)}} \Phi(\alpha)_\Gamma = I_\Gamma$.

Since we are interested in the unused part of the hash chain register, we need to filter those α 's for which Γ_i^j is in state $|\Phi\rangle$. By construction of the checksum, if a block b of a message m is computed, then in the block b' of any other message m' there exists at least one position i at which $b'_i < b_i$, $1 \leq i \leq l$. Therefore, since the blinded signing oracle signs at most a single un-blinded message $m \in B^c$, the state after the signing oracle call can be written as a superposition of states where, for some un-blinded message $m' \in B^c$, $b'_i < b_i$ for all i . The latter implies that the hash chain registers corresponding to those b'_i are still in the uniform superposition $|\Phi\rangle$, for all i . Thus, we collect all strings α that are consistent with no blinded messages having been signed in the set

$$\widehat{B^c} = \bigcup_{m \in B^c} \left\{ \alpha \in \{0,1\}^{l(w-1)} \mid \alpha_i^j = 0 \text{ for all } i = 1, \dots, l \text{ and } j < b_i(m) \right\}. \quad (22)$$

Finally, we define the invariant projector as $P_\Gamma = \sum_{\alpha \in \widehat{B^c}} \Phi(\alpha)_\Gamma$.

Using these definitions of the Q_i and P_Γ , a set of lemmas similar to Lemmas 12–15 forms the basis of the BU security proof for the Winternitz OTS. In fact, Lemma 12 is a special case of Lemma 20 where the register Γ is replaced by S and we set $w = 2$ (see Appendix A.1 of [23] for proof). Lemma 13 holds for the new projectors Q_{l+1} and P_Γ by construction. Finally, Lemmas 14 and 15 need to be changed slightly for the Winternitz OTS and are stated below. Lemmas 21–23 are proved in the full version [23].

► **Lemma 20.** *Let U_h be the random oracle unitary for any given function h (see Section 3) and let $\Phi = |\Phi\rangle\langle\Phi|$ denote the projector onto the uniform superposition $|\Phi\rangle$. Furthermore, let $\Gamma_i^{\leq j} = \Gamma_i^0 \dots \Gamma_i^j$ and $\Phi_{\Gamma_i^{\leq j}} = (\Phi^{\otimes j})_{\Gamma_i^{\leq j}}$. Then, for any $i' \in \{1, \dots, l\}$ and $j' \in \{0, \dots, w-2\}$, $\|[(U_h)_{XY\Gamma}, \Phi_{\Gamma_{i'}^{\leq j'}}]\|_\infty \leq 6(w-1)/2^{n/2} = \epsilon_W(n)$ is negligible in n .*

► **Lemma 21.** *Let $B \text{ Sign}_{\text{sk}}$ be the blinded signing oracle for the Winternitz OTS, and let $|\psi_0\rangle$ be the adversary's state before the **Sign** query. If there are no hash queries, then after making a single **Sign** query the adversary's state $|\psi_1\rangle = B \text{ Sign}_{\text{sk}} |\psi_0\rangle$ is completely in the range of the invariant projector P_Γ defined below Equation (22). That is, $P_\Gamma B \text{ Sign}_{\text{sk}} |\psi_0\rangle = B \text{ Sign}_{\text{sk}} |\psi_0\rangle$.*

► **Lemma 22.** *Let $m^* \in B$ and $b^* = b(m^*)$ the concatenation of m^* and its checksum in w -ary. Then the projectors $Q_{l+1}^{b^*}$ defined in Equation (21) and P_Γ are orthogonal, that is $Q_{l+1}^{b^*} P_\Gamma = 0$.*

► **Lemma 23.** *Let P_Γ and U_h be, respectively, the invariant projector for the Winternitz OTS and the random oracle unitary defined with respect to the Quantum independent world. If there are hash queries after the Sign query, then $\| [U_h, P_\Gamma] \|_\infty \leq \delta_W(n)$ where $\delta_W(n) = 8l(w+1)(w-1)/2^{n/2}$.*

The proof of Theorem 19 is based on the preceding lemmas and follows the same outline as the proof for the Lamport OTS. It can be found in the full version [23].

6 Tightness

The notion of blind-unforgeability does not have as close of a relation to the intuitive security property it strives to model as EU-CMA.⁴ The concrete security bounds, however, arguably nevertheless provide an indication of concrete security levels. It is hence an interesting question whether the bounds proven in Section 4 above and in Section 5 of the full version [23] are tight. In the following, we present an attack against the BU security of the Lamport scheme in the QROM and analyze its success probability to show that the bound in Theorem 11 is tight up to a factor l in the number of queries. The attack generalizes to the Winternitz scheme in a straight-forward manner.

We begin by describing a straightforward classical attack based on search. To attack the BU security of the Lamport scheme, choose a blinding probability of $1/2$. Now make q distinct queries to the random oracle to search for a preimage of one of the $2l$ public key strings. This succeeds with probability

$$p_{\text{search}}(q) = 1 - (1 - 2l \cdot 2^{-n})^q \geq 2ql \cdot 2^{-n}. \quad (23)$$

Suppose this search succeeded, finding a preimage y^* of $p_{i^*}^{j^*}$. Then chose $m \in \{0, 1\}^l$ such that $m_{i^*} = j^*$ and query the oracle to obtain a signature for m . This succeeds with probability $1/2$. Now output m' obtained from m by flipping the i^* th bit, and σ' obtained from σ by replacing σ_{i^*} with y^* . Note that m' is blinded with probability $1/2$, and y^* is equal to the correct secret key string $s_{i^*}^{j^*}$ with constant probability. In summary, the entire attack succeeds with constant probability if $q = \Omega(2^n \cdot l^{-1})$.

This search step can now be replaced by a Grover search in the QROM. Using the analysis of Grover's algorithm for multiple targets from [9], together with a basic analysis of the number of targets (which follows a binomial distribution), a constant success probability can be achieved if $q = \Omega(2^{n/2} \cdot l^{-1/2})$. To compare this result with Theorem 11, note that the inequality in Equation (4), implies that to achieve a constant success probability, at least $q \geq C \cdot 2^{n/2} \cdot l^{-3/2}$ are necessary for some constant C , i.e. the upper and lower bounds on the number of queries the optimal attack requires indeed differ by a factor of l up to constant factors. For the Winternitz scheme, the bounds differ by a factor of w^2l .

References

- 1 Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, et al. Status report on the second round of the NIST post-quantum cryptography standardization process. *US Department of Commerce, NIST*, 2020. doi:10.6028/NIST.IR.8309.

⁴ Indeed, it is a nice exercise to show that an adversary against (say, q -time) EU-CMA with success probability ϵ can be used to construct a BU-adversary with success probability $\Theta(\epsilon/q)$, and this reduction is tight for efficient adversaries if one-way functions exists.


- 2 Gorjan Alagic, Christian Majenz, Alexander Russell, and Fang Song. Quantum-access-secure message authentication via blind-unforgeability. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 788–817. Springer, 2020. doi:10.1007/978-3-030-45727-3_27.
- 3 Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM, 1993. doi:10.1145/168588.168596.
- 4 Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS+ signature framework. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS'19*, pages 2129–2146, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3319535.3363229.
- 5 Jeremiah Blocki, Seunghoon Lee, and Samson Zhou. On the security of proofs of sequential work in a post-quantum world, 2020. arXiv:2006.10972.
- 6 Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 41–69. Springer, 2011. doi:10.1007/978-3-642-25385-0_3.
- 7 Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 361–379, Berlin, Heidelberg, 2013. Springer. doi:10.1007/978-3-642-40084-1_21.
- 8 Xavier Bonnetain, Akinori Hosoyamada, María Naya-Plasencia, Yu Sasaki, and André Schrottenloher. Quantum attacks without superposition queries: The offline Simon’s algorithm. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019*, pages 552–583, Cham, 2019. Springer. doi:10.1007/978-3-030-34578-5_20.
- 9 Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In Cláudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN'98: Theoretical Informatics*, pages 163–169, Berlin, Heidelberg, 1998. Springer. doi:10.1007/BFb0054319.
- 10 Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - a practical forward secure signature scheme based on minimal security assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 117–129, Berlin, Heidelberg, 2011. Springer. doi:10.1007/978-3-642-25405-5_8.
- 11 Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. On the compressed-oracle technique, and post-quantum security of proofs of sequential work, 2020. arXiv:2010.11658.
- 12 Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, 1996. doi:10.1007/0-387-34805-0_24.
- 13 Tommaso Gagliardoni, Andreas Hülsing, and Christian Schaffner. Semantic security and indistinguishability in the quantum world. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 60–89, Berlin, Heidelberg, 2016. Springer. doi:10.1007/978-3-662-53015-3_3.
- 14 Tommaso Gagliardoni, Juliane Krämer, and Patrick Struck. Quantum indistinguishability for public key encryption, 2020. arXiv:2003.00578.
- 15 Sumegha Garg, Henry Yuen, and Mark Zhandry. New security notions and feasibility results for authentication of quantum data. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 342–371, Cham, 2017. Springer. doi:10.1007/978-3-319-63715-0_12.
- 16 Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on computing*, 17(2):281–308, 1988. doi:10.1137/0217017.
- 17 Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Tight adaptive reprogramming in the QROM, 2020. arXiv:2010.15103.

- 18 Masahito Hayashi. Optimal sequence of quantum measurements in the sense of Stein's lemma in quantum hypothesis testing. *Journal of Physics A: Mathematical and General*, 35(50):10759, 2002. doi:10.1088/0305-4470/35/50/307.
- 19 Andreas Hülsing, Denise Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. XMSS: Extended hash-based signatures. RFC 8391, 2018. doi:10.17487/RFC8391.
- 20 Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking symmetric cryptosystems using quantum period finding. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 207–237, Berlin, Heidelberg, 2016. Springer. doi:10.1007/978-3-662-53008-5_8.
- 21 Leslie Lamport. Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979. URL: <http://lamport.azurewebsites.net/pubs/dig-sig.pdf>.
- 22 Qipeng Liu and Mark Zhandry. On finding quantum multi-collisions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 189–218, Cham, 2019. Springer. doi:10.1007/978-3-030-17659-4_7.
- 23 Christian Majenz, Channele Matadah Manfouo, and Maris Ozols. Quantum-access security of the Winternitz one-time signature scheme, 2021. arXiv:2103.12448.
- 24 Ralph C. Merkle. A certified digital signature. In *Conference on the Theory and Application of Cryptology*, pages 218–238. Springer, 1989. doi:10.1007/0-387-34805-0_21.
- 25 Michael A. Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002. doi:10.1023/A:1012603118140.
- 26 Thomas Santoli and Christian Schaffner. Using Simon's algorithm to attack symmetric-key cryptographic primitives. *Quantum Info. Comput.*, 17(1–2):65–78, 2017. doi:10.26421/QIC17.1-2-4.
- 27 Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. IEEE, 1994. doi:10.1109/SFCS.1994.365700.
- 28 Mark Zhandry. How to record quantum queries, and applications to quantum indifferenciability. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 239–268, Cham, 2019. Springer. doi:10.1007/978-3-030-26951-7_9.

On the Security of Proofs of Sequential Work in a Post-Quantum World

Jeremiah Blocki   

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Seunghoon Lee   

Department of Computer Science, Purdue University, West Lafayette, IN, USA

Samson Zhou   

School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

A Proof of Sequential Work (PoSW) allows a prover to convince a resource-bounded verifier that the prover invested a substantial amount of sequential time to perform some underlying computation. PoSWs have many applications including time-stamping, blockchain design, and universally verifiable CPU benchmarks. Mahmoody, Moran, and Vadhan (ITCS 2013) gave the first construction of a PoSW in the random oracle model though the construction relied on expensive depth-robust graphs. In a recent breakthrough, Cohen and Pietrzak (EUROCRYPT 2018) gave an efficient PoSW construction that does not require expensive depth-robust graphs.

In the classical parallel random oracle model, it is straightforward to argue that any successful PoSW attacker must produce a long \mathcal{H} -sequence and that any malicious party running in sequential time $T - 1$ will fail to produce an \mathcal{H} -sequence of length T except with negligible probability. In this paper, we prove that any quantum attacker running in sequential time $T - 1$ will fail to produce an \mathcal{H} -sequence except with negligible probability – even if the attacker submits a large batch of quantum queries in each round. The proof is substantially more challenging and highlights the power of Zhandry’s recent compressed oracle technique (CRYPTO 2019). We further extend this result to establish post-quantum security of a non-interactive PoSW obtained by applying the Fiat-Shamir transform to Cohen and Pietrzak’s efficient construction (EUROCRYPT 2018).

2012 ACM Subject Classification Security and privacy → Hash functions and message authentication codes; Security and privacy → Information-theoretic techniques

Keywords and phrases Proof of Sequential Work, Parallel Quantum Random Oracle Model, Lower Bounds

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.22

Related Version *Full Version:* <https://arxiv.org/abs/2006.10972>

Funding *Jeremiah Blocki:* Research supported in part by NSF CNS-1755708 and NSF CNS-1931443. *Seunghoon Lee:* Research supported in part by NSF Award CNS-1755708 and by the Center for Science of Information at Purdue University (NSF CCF-0939370).

Acknowledgements The authors wish to thank Fang Song (shepherd) and other anonymous reviewers for comments which improved the presentation of this paper.

1 Introduction

As we make progress towards the development of quantum computers, it is imperative to understand which cryptographic primitives can be securely and efficiently instantiated in a post-quantum world. In this work, we consider the security of proofs of sequential work against quantum adversaries.

A proof of sequential work (PoSW) [37, 23, 3, 26] is a protocol for proving that one spent significant sequential computation work to validate some statement χ . One motivation for a proof of sequential work is in time-stamping, e.g., if Bob can produce a valid proof π_χ that



© Jeremiah Blocki, Seunghoon Lee, and Samson Zhou;
licensed under Creative Commons License CC-BY 4.0
2nd Conference on Information-Theoretic Cryptography (ITC 2021).
Editor: Stefano Tessaro; Article No. 22; pp. 22:1–22:27



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

N sequential steps were spent to validate χ , then Bob can prove that he must have known about χ at least time $\Omega(N)$ seconds in the past. A verifier should be able to validate the proof π_χ quickly, i.e., in time $\text{polylog}(N)$.

Mahmoody et al. [37] gave the first construction of a proof of sequential work in the random oracle model. Their construction was based on labeling a depth-robust graph, i.e., given a random oracle \mathcal{H} and a directed acyclic graph $G = (V = [N], E)$ with N nodes and an initial input x , we can compute labels ℓ_1, \dots, ℓ_N , where the label of the source node is $\ell_1 = \mathcal{H}(\chi, 1, x)$ and an internal node v with parents v_1, \dots, v_δ has label $\ell_v = \mathcal{H}(\chi, v, \ell_{v_1}, \dots, \ell_{v_\delta})$.

The prover commits to labels ℓ'_1, \dots, ℓ'_N (a cheating prover might commit to the wrong labels) and then the verifier selects a random subset $S \subseteq [N]$ of $|S| = c$ challenge nodes. For each challenge node $v \in S$ with parents v_1, \dots, v_δ , the prover reveals ℓ'_v along with $\ell'_{v_1}, \dots, \ell'_{v_\delta}$ and the verifier checks that v is locally consistent, i.e., $\ell'_v = \mathcal{H}(\chi, v, \ell'_{v_1}, \dots, \ell'_{v_\delta})$. If we let R denote the subset of locally inconsistent nodes, then the verifier will accept with probability at most $(1 - |R|/N)^c$.

Mahmoody et al. [37] selected G such that G was ϵ -extremely depth-robust¹, meaning that for any set $R \subseteq [N]$ of locally inconsistent nodes, there is a directed path of length $T + 1 = (1 - \epsilon)N - R$. This path $P = v_0, \dots, v_T$ corresponds to an \mathcal{H} -sequence of length T where an \mathcal{H} -sequence is any sequence of strings x_0, \dots, x_T with the property that $\mathcal{H}(x_i)$ is a substring of x_{i+1} for each $i < T$. Note that the labels $\ell'_{v_0}, \dots, \ell'_{v_T}$ have this property. In the classical parallel random oracle model (pROM), it is relatively straightforward to prove that any algorithm running in $T - 1$ rounds and making at most q queries in total fails to produce an \mathcal{H} -sequence except with probability $\tilde{\Omega}(q^2 2^{-\lambda})$ when $\mathcal{H} : \{0, 1\}^{\delta\lambda} \rightarrow \{0, 1\}^\lambda$ outputs binary strings of length λ [23].

The ϵ -extreme depth-robust graphs used in the construction of Mahmoody et al. [37] were quite expensive, having indegree $\delta = \tilde{\Omega}(\log N)$. Alwen et al. [7] showed how to construct ϵ -extreme depth-robust graphs with indegree just $\mathcal{O}(\log N)$ though the hidden constants were quite large. Cohen and Pietrzak [23] gave an efficient (practical) construction that avoids depth-robust graphs entirely by cleverly modifying the Merkle tree structure to obtain a graph G on $N = 2^{n+1} - 1$ nodes², for any integer $n \geq 1$.

Both proofs of sequential work can (optionally) be converted into a non-interactive proof by applying the Fiat-Shamir paradigm [28], i.e., given a commitment c' to labels ℓ'_1, \dots, ℓ'_N we can use public randomness $r = \mathcal{H}(\chi, N + 1, c')$ to sample our set of challenge nodes S . The non-interactive version could be useful in cases where a prover wants to silently timestamp a statement χ without even signaling that s/he might have a statement important enough to timestamp, e.g., a researcher who believes they might resolved a famous open problem may wish to timestamp the discovery without signaling the community until s/he carefully double checks the proof.

In all of the above constructions, security relies on the hardness of computing \mathcal{H} -sequences of length T in sequential time $T - 1$. While this can be readily established in the classical parallel random oracle model, proving that this task is in fact hard for a quantum attacker is a much more daunting challenge. As Boneh et al. [16] pointed out, many of the convenient

¹ A DAG G is said to be ϵ -extremely depth-robust if it is (e, d) -depth robust for any $e, d > 0$ such that $e + d \leq (1 - \epsilon)N$ where N is the number of nodes in G . Recall that a DAG $G = (V, E)$ is (e, d) -depth robust if for any subset $S \subseteq V$ with $|S| \leq e$ there exists a path of length d in $G - S$.

² The graph G is “weighted” depth robust. In particular, there is a weighting function $w : V \rightarrow \mathbb{R}_{\geq 0}$ with the property that $\sum_v w(v) \in \mathcal{O}(N \log N)$ and for any subset $S \subseteq V$ with sufficiently small weight $\sum_{v \in S} w(v) \leq cN$ the DAG $G - S$ contains a path of length $\Omega(N)$.

properties (e.g., extractability, programmability, efficient simulation, rewinding, etc.) that are used in classical random oracle security proofs no longer apply in the quantum random oracle model (qROM). An attacker in the (parallel) quantum random oracle model is able to submit entangled queries, giving the attacker much more power. For example, given y a quantum attacker can find a preimage x' such that $\mathcal{H}(x') = y$ with just $\mathcal{O}(2^{\lambda/2})$ quantum random oracle queries using Grover's algorithm. By contrast, a classical attacker would need at least $\Omega(2^\lambda)$ queries to a classical random oracle. Similarly, a quantum attacker can find hash collisions with at most $\mathcal{O}(2^{\lambda/3})$ queries, while a classical attacker requires $\Omega(2^{\lambda/2})$ queries. In this paper, we explore the post-quantum security of proofs of sequential work in the parallel quantum random oracle model. We aim to answer the following questions:

Can a quantum attacker running in $T - 1$ sequential rounds produce an \mathcal{H} -sequence of length T ?

Can a quantum attacker running in time $T = (1 - \alpha)N$ produce a valid non-interactive proof of sequential work with non-negligible probability?

1.1 Our Contributions

We answer these questions in the negative, thus confirming the security of proof of sequential work schemes in a post-quantum world. We first prove that any quantum attacker making $N - 1$ rounds of queries cannot produce an \mathcal{H} -sequence of length N , except with negligible probability.

► **Definition 1** (\mathcal{H} -Sequence). *An \mathcal{H} -sequence $x_0, x_1, \dots, x_s \in \{0, 1\}^*$ satisfies the property that for each $1 \leq i \leq s$, there exist $a, b \in \{0, 1\}^*$ such that $x_i = a \parallel \mathcal{H}(x_{i-1}) \parallel b$. For indexing reasons, we say such an \mathcal{H} -sequence has length s (even though there are $s + 1$ variables x_i).*

In the classical random oracle model (ROM), it is straightforward to argue that any PoSW prover must find a long \mathcal{H} -sequence to pass the audit phase with non-negligible probability. Thus, this result already provides compelling evidence that proofs of sequential work are post-quantum secure in the parallel random oracle model.

Next we consider a non-interactive proof of sequential work applying the Fiat-Shamir transform to the efficient construction of Cohen and Pietrzak [23], and we prove that this construction is secure in the quantum parallel random oracle model. In particular, we show that any attacker running in sequential time $T = (1 - \alpha)N$ will fail to produce a valid proof π_χ for any statement $\chi \in \{0, 1\}^\lambda$.

While Cohen and Pietrzak [23] proved analogous results in the classical random oracle model, we stress that from a technical standpoint, proving security in the quantum random oracle model is significantly more challenging. In general, there is a clear need to develop new techniques to reason about the security of cryptographic protocols in the quantum random oracle model. Most of the techniques that are used in classical random oracle model do not carry over to the (parallel) quantum random oracle model [16]. For example, if we are simulating a classical attacker, then we can see (extract) all of the random oracle queries that the attacker makes, while we cannot observe a quantum query without measuring it, which would collapse the attacker's quantum state might significantly alter the final output.

Warm-Up Problem: Iterative Hashing

As a warm-up, we first prove an easier result in Theorem 2 that an attacker cannot compute $\mathcal{H}^N(x)$ in sequential time less than $N - 1$ in the parallel quantum random oracle model, where a similar result was previously proved by Unruh [39] in the (non-parallel) quantum

random oracle model. Along the way we highlight some of the key challenges that make it difficult to extend the proof to arbitrary \mathcal{H} -sequences.

► **Theorem 2.** *Given a hash function $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ and a random input x , any quantum attacker that makes up to q queries in each of $N - 1$ sequential steps can only compute $\mathcal{H}^N(x)$ with probability at most $\frac{N^2}{2^\lambda} + \frac{1}{2^{\lambda-N}} + \sqrt{\frac{48\lambda N^4 q^2 T}{2^{\lambda/2}}}$ in the quantum parallel random oracle model.*

The proof of Theorem 2 is straightforward and we defer it to the full version. Intuitively, iteratively computing $\mathcal{H}^N(x)$ induces an \mathcal{H} -sequence x_0, x_1, \dots, x_n with $x_0 = x$, $x_N = \mathcal{H}^N(x)$ and $x_{i+1} = \mathcal{H}(x_i)$. One can easily define a sequence of indistinguishable hybrids where in the last hybrid the final output $x_N = \mathcal{H}^N(x) = \mathcal{H}(x_{N-1})$ is information theoretically hidden from the attacker. In general, in hybrid i , for each $j \leq i$, the value $x_j = \mathcal{H}^j(x) = \mathcal{H}(x_{j-1})$ remains information theoretically hidden until round j . In particular, we replace the random oracle \mathcal{H} with a new stateful oracle $\mathcal{H}'_i(\cdot)$ that is almost identical to $\mathcal{H}(\cdot)$, except that for any $j \leq i$ if the query $\mathcal{H}(x_j)$ is submitted to $\mathcal{H}'(\cdot)$ before round j then the response will be a random unrelated λ -bit string instead of $\mathcal{H}(x_j)$.

We can argue indistinguishability of hybrids i using a result of [10] because if $j > i$, then x_j is information theoretically hidden up until round i and the total query magnitude of $x_j = \mathcal{H}^j(x)$ during round i is negligible. Here, the total query magnitude of a string x_j during round i is defined as the sum of squared amplitudes on states where the attacker is querying string x_j . It then follows that except with negligible probability a quantum attacker cannot compute $\mathcal{H}^N(x)$. The argument does rely on the assumption that the running time T of the attacker is bounded, e.g., $T \leq 2^{c\lambda}$ for some constant $c > 0$.

Our main results are summarized in Theorem 3 and Theorem 4. We show that quantum attackers running in at most $N - 1$ sequential steps cannot find an \mathcal{H} -sequence of length N with high probability. We also show that for any quantum attackers making at most q quantum queries to the random oracle \mathcal{H} over at most $(1 - \alpha)N$ rounds will only be able to produce a valid PoSW with negligible probability.

Technical Challenges: Iterative Hashing vs \mathcal{H} -Sequences

Proving that an attacker cannot find an \mathcal{H} -sequence of length N in $N - 1$ rounds of parallel queries is significantly more challenging. One key difference is that there are exponentially many distinct \mathcal{H} -sequences of length N that are consistent with the initial string x_0 . By contrast, when we analyze a hash chain, each value on the chain $\mathcal{H}^j(x_0)$ can be viewed as fixed a priori. For \mathcal{H} -sequences, it is not clear how one would even define a hybrid where all candidate values of x_i are information-theoretically hidden because these values are not known a priori and there might be exponentially many such candidates. In fact, for any $2 \leq i \leq N$ and any string y , it is *likely* that there exists an \mathcal{H} -sequence x_0, \dots, x_N such that $y = x_i$.

Instead, we use a recent idea introduced by [42] that views the random oracle as a superposition of *databases* rather than queries. This view facilitates intuitive simulation of quantum random oracles in a manner similar to classical models, which provides intuitive simulation for queries and circumvents the need to “record all possible queries”, which would give an exponential number of possible \mathcal{H} -sequences in our case. We give significantly more intuition in Section 4, after formalizing the relevant definitions.

► **Theorem 3.** *Let $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ be a random hash function and let $\delta \geq 1$ be a parameter. Let p be the probability that a quantum adversary making at most q queries over $N - 1$ rounds outputs $(x_0, y_0), \dots, (x_{N-1}, y_{N-1})$ and x_N s.t. $|x_i| \leq \delta\lambda$, $y_i = \mathcal{H}(x_i)$ and $\text{Substring}(y_{i-1}, x_i) = 1$ for each i , i.e., x_0, \dots, x_N is an \mathcal{H} -sequence. Then*

$$p \leq \frac{64q^3\delta\lambda}{2^\lambda} + \frac{2N}{2^\lambda}.$$

Here, $\text{Substring}(y_{i-1}, x_i) = 1$ means that y_{i-1} is a substring of x_i , i.e., there exist $a, b \in \{0, 1\}^*$ such that $x_i = a\|y_{i-1}\|b$.

From \mathcal{H} -Sequences to Proof of Sequential Work

Theorem 4 focuses on a non-interactive proof of sequential work obtained by applying the Fiat-Shamir transform to the efficient construction of Cohen and Pietrzak [23]. This construction is based on a DAG G with $N = 2^{n+1} - 1$ nodes and maximum indegree n . Given a random oracle $\mathcal{H} : \{0, 1\}^{\lambda(n+2)} \rightarrow \{0, 1\}^\lambda$, an honest prover can generate a proof for any statement $\chi \in \{0, 1\}^\lambda$ in sequential time $\mathcal{O}(N)$. We prove that for any constant $\alpha > 0$, an attacker making q queries over $s = N(1 - \alpha)$ rounds will fail to produce a valid proof of sequential work for any statement except with negligible probability.

► **Theorem 4.** *Suppose \mathcal{A} makes at most q quantum queries to our random oracle \mathcal{H} over at most $s = N(1 - \alpha)$ rounds and let p denote the probability that \mathcal{A} outputs a valid (non-interactive) proof of sequential work. Then*

$$p \leq 32q^2(1 - \alpha)^{\lfloor \lambda/n \rfloor} + \frac{2q^3}{2^\lambda} + \frac{64q^3(n+2)\lambda}{2^\lambda} + \frac{2\lfloor \lambda/n \rfloor(n+2)}{2^\lambda}.$$

The main intuition for the proof Theorem 4 works as follows. Given a quantum database $\mathcal{D} = \{(x_i, y_i) : i \geq 1\}$ where y_i encodes the output on input x_i with λ bits, we define a set LUCKY_s of databases \mathcal{D} based on the graph coloring (see the full version), in which \mathcal{D} does not contain any collision or \mathcal{H} -sequence of length s , yet still contains a “lucky” Merkle tree that has a green path from the challenged node to the root that can be used to extract a proof of sequential work. We show that any attacker making (possibly parallel) q queries can only succeed in measuring a lucky database \mathcal{D} with negligible probability. Finally, we show that any attacker who produces a valid PoSW must measure a database \mathcal{D} that either (1) contains an \mathcal{H} -sequence of length s , (2) contains a collision, or (3) is a lucky database. Since each of these events has negligible probability, then it follows that with high probability, the attacker cannot produce a valid PoSW.

1.2 Related Work

Functions that are inherently sequential to compute are a cryptographic primitive used in many applications, such as proof of sequential work [37], verifiable delay functions [15], and time-lock puzzles [36]. The original construction [37] used depth-robust graphs, which have found applications in many areas of cryptography including memory-hard functions (e.g., [8, 5, 6, 14, 13, 7, 12]), proofs of replication [29, 20], and proofs of space [27, 38]. Recently, Cohen and Pietrzak [23] show that \mathcal{H} -sequences are difficult for a classical adversary to compute in the classical parallel random oracle model.

The Quantum Random Oracle Model (qROM) was introduced by Boneh et al. [16], who pointed out that for any real world instantiation for the hash function \mathcal{H} (e.g., SHA3), one can build a quantum circuit implementing \mathcal{H} . Boneh et al. [16] also provided an example

of a protocol that is secure in the classical ROM, but not in the qROM. Quantum attacks and constructions under the quantum random oracle model have been studied in a number of previous settings, such as unclonable public-key quantum money [1, 2], quantum Merkle puzzles [19, 18], signature schemes [17] and construction of random functions [41].

Security reductions in the classical ROM often exploit properties such as programability and extractability of queries – properties that are lost in the qROM. Zhandry introduced compressed oracles [42] as a way to record quantum queries so that they can be viewed after computation has completed. The new technique has proven to be a useful tool to extend many classical security proofs to the quantum random oracle model, e.g., [11, 21, 34, 4, 31]. Don et al. [25] recently showed how queries can be extracted on-the-fly in certain settings, e.g., once the algorithm outputs a classical commitment t (e.g., $t = \mathcal{H}(x)$ or $t = \text{Enc}_{pk}(\mathcal{H}(x))$) that is tightly related to the input x .

The non-interactive PoSW we consider in this work is obtained by applying the Fiat-Shamir transform to the interactive PoSW construction of Cohen and Pietrzak [23]. While there is a recent line of work analyzing the security of the Fiat-Shamir transform [28] in the quantum random oracle model [33, 24, 35], applying these results would require us to first establish the security of the interactive PoSW in the (parallel) qROM. We find it easier to directly show that the non-interactive PoSW construction is secure in the (parallel) qROM.

There have been a number of work on parallelizing quantum algorithms or considering parallel queries in the quantum random oracle model. Zalka [40] showed that the parallel version of Grover’s algorithm is optimal, e.g., in the ideal cipher model, any parallel key-recovery attacker making at most $q = \mathcal{O}(\sqrt{k2^\lambda})$ quantum queries to the ideal cipher must run in sequential time $\Omega(\sqrt{2^\lambda/k})$. Grover and Radhakrishnan [30] generalized Zalka’s result in the setting of multiple items to search. Jeffery et al. [32] studied the parallel quantum query complexity for the element distinctness and the k -sum problem. Ambainis et al. [9] provided an improved one-way to hiding (O2H) theorem in the parallel quantum random oracle model.

In independent work, Chung et al. [22] also studied the problem of finding an \mathcal{H} -sequence and non-interactive proofs of sequential work in the parallel quantum random oracle model. They gave comparable bounds also using Zhandry’s compressed oracle technique [42], while leveraging an abstract view of Fourier transforms for arbitrary finite Abelian groups. By comparison, our proofs avoid the need for an understanding of abstract algebra, instead using quantum information theory to bound the quantum query complexity through a reduction to classical query complexity. Thus we believe our techniques to be of independent interest, perhaps appealing to a more general audience while also providing the necessary framework to analyze the security of other classical protocols in a post-quantum world.

2 Preliminaries

Let \mathbb{N} denote the set $\{0, 1, \dots\}$, $[n]$ denote the set $\{1, 2, \dots, n\}$, and $[a, b] = \{a, a + 1, \dots, b\}$ where $a, b \in \mathbb{N}$ with $a \leq b$. For a function $f(x)$, we recursively define $f^N(x) = f \circ f^{N-1}(x)$ where \circ is a function/operator composition. We say that a non-negative function $\mu(x)$ is *negligible*, if for all polynomials $p(x)$, it holds that $0 \leq \mu(x) < \frac{1}{p(x)}$ for all sufficiently large x .

Let $\{0, 1\}^n$ be the set of all bitstrings of length n . Then we define $\{0, 1\}^{\leq n} = \cup_{i=0}^n \{0, 1\}^i$ to be the set of all bitstrings of length at most n including an empty string ε . We denote $\|$ as the concatenation of bitstrings. For a bitstring $x \in \{0, 1\}^*$, $x[i]$ denotes its i^{th} bit, and $x[i \dots j] = x[i] \| \dots \| x[j]$.

Given quantum states $|\phi\rangle = \sum \alpha_x |x\rangle$ and $|\psi\rangle = \sum \beta_x |x\rangle$, we define the Euclidean distance between the two states to be the quantity $\sqrt{\sum |\alpha_x - \beta_x|^2}$. The *magnitude* of $|x\rangle$ in $|\phi\rangle = \sum \alpha_x |x\rangle$ is α_x and the query probability is $|\alpha_x|^2$ – when we measure the state $|\phi\rangle_x$ we will observe $|x\rangle$ with probability $|\alpha_x|^2$.

2.1 Quantum Random Oracle Model

In the (sequential) quantum random oracle model (qROM), an adversary is given oracle access to a random hash function $\mathcal{H} : \{0, 1\}^m \rightarrow \{0, 1\}^\lambda$. The adversary can submit quantum states as queries to the oracle, so that \mathcal{H} takes as input superposition $|\phi_1\rangle, |\phi_2\rangle, \dots$. Each $|\phi_i\rangle$ can be expanded as $|\phi_i\rangle = \sum \alpha_{i,x,y} |x, y\rangle$ so that the output is $\sum \alpha_{i,x,y} |x, y \oplus \mathcal{H}(x)\rangle$. Note that when the initial state is of the form $|\phi\rangle = \sum \alpha_x |x, 0^w\rangle$, then the output state will be of the form $\sum \alpha_x |x, \mathcal{H}(x)\rangle$.

Compressed Oracle Technique in the Sequential qROM

Here we introduce the compressed oracle representation introduced by Zhandry [42], which is equivalent to the standard oracle in function. However, the difference between the compressed oracle and the regular oracle is in the encodings of the oracle and query registers as queries are made to the oracles. We will extend the ideas of this technique to the parallel qROM later on.

First, we formally define a database \mathcal{D} . A database \mathcal{D} is defined by $\mathcal{D} = \{(x_i, y_i) : i \geq 1\}$ where we write $\mathcal{D}(x_i) = y_i$ to denote that y_i encodes the output on input x_i with λ bits. When $\mathcal{D} = \{\}$ is empty, it is equivalent to viewing the random oracle as being in superposition of all possible random oracles. After q queries, the state can be viewed as $\sum_{x,y,z,\mathcal{D}} \alpha_{x,y,z} |x, y, z\rangle \otimes |\mathcal{D}\rangle$, where \mathcal{D} is a compressed dataset of at most q input/output pairs, x, y are the query registers, and z is the adversary's private storage.

Formally, the compressed oracle technique for the sequential qROM works as follows. Let $\mathcal{H} : \{0, 1\}^m \rightarrow \{0, 1\}^\lambda$ be a random hash function and suppose an adversary is given an oracle access to \mathcal{H} . Then we have the following observations:

- It is equivalent to view the usual random oracle mapping $|x, y\rangle \mapsto |x, y \oplus \mathcal{H}(x)\rangle$ (denote as StO) as the *phase* oracle PhsO that maps $|x, y\rangle$ to $(-1)^{y \cdot \mathcal{H}(x)} |x, y\rangle$ by applying Hadamard transforms before and after the oracle query.³
- It is also equivalent to view the oracle \mathcal{H} as being in (initially uniform) superposition $\sum_{\mathcal{H}} |\mathcal{H}\rangle$ where we can encode \mathcal{H} as a binary vector of length $2^m \times \lambda$ encoding the λ -bit output for each m -bit input string. Under this view the oracle maps the state $|\phi\rangle = \sum_{x,y} \alpha_{x,y} |x, y\rangle \otimes \sum_{\mathcal{H}} |\mathcal{H}\rangle$ to $\sum_{x,y} \alpha_{x,y} |x, y\rangle \otimes \sum_{\mathcal{H}} |\mathcal{H}\rangle (-1)^{y \cdot \mathcal{H}(x)}$.

If the attacker makes at most q queries, then we can compress the oracle \mathcal{H} and write $|\phi\rangle = \sum_{x,y} \alpha_{x,y} |x, y\rangle \otimes \sum_{\mathcal{D}} |\mathcal{D}\rangle$, where each dataset $\mathcal{D} \in \{0, 1\}^{\lambda \times 2^m}$ is sparse, i.e., $\mathcal{D}(x) \neq \perp$ for at most q entries. Intuitively, when $\mathcal{D}(x) = \perp$, we view the random oracle as being in a uniform superposition over potential outputs. Moreover, we can think of the basis state $|\mathcal{D}\rangle$ as corresponding to the superposition $\sum_{\mathcal{H} \in \mathcal{H}_{\mathcal{D}}} |\mathcal{H}\rangle$ where $\mathcal{H}_{\mathcal{D}} \subseteq \{0, 1\}^{\lambda \times 2^m}$ denote the set of all random oracles that are consistent with \mathcal{D} , i.e., if $\mathcal{H} \in \mathcal{H}_{\mathcal{D}}$ then for all inputs x we either have $\mathcal{D}(x) = \perp$ or $\mathcal{D}(x) = \mathcal{H}(x)$. When viewed in this way, the basis state $|\mathcal{D}\rangle$ encodes prior queries to the random oracle along with the corresponding responses. We can use a compressed phase oracle CPhsO (described below) to model a phase oracle.

³ Notice that both StO and PhsO are unitary matrices and $\text{StO} = (I^m \otimes \mathcal{H}^{\otimes \lambda}) \text{PhsO} (I^m \otimes \mathcal{H}^{\otimes \lambda})$ where I^m is the identity matrix on the first m qubits and $\mathcal{H}^{\otimes \lambda}$ is the Hadamard transform on the λ output qubits.

Compressed Phase Oracle

To properly define a compressed phase oracle CPhsO in the sequential qROM, a unitary local decompression procedure StdDecomp_x that acts on databases was first defined in [42]. Intuitively, StdDecomp_x decompresses the value of the database at position x when the database \mathcal{D} is not specified on x and there is a room to expand \mathcal{D} , and StdDecomp_x does nothing when there is no room for decompression. If \mathcal{D} is already specified on x , then we have two cases: if the corresponding y registers are in a state orthogonal to a uniform superposition, then StdDecomp_x is the identity (no need to decompress). If the y registers are in the state of a uniform superposition, then StdDecomp_x removes x from \mathcal{D} . We refer to the full version for a full description of StdDecomp_x . Now we define StdDecomp , Increase , CPhsO' on the computational basis states as

$$\begin{aligned}\text{StdDecomp}(|x, y\rangle \otimes |\mathcal{D}\rangle) &= |x, y\rangle \otimes \text{StdDecomp}_x|\mathcal{D}\rangle, \\ \text{Increase}(|x, y\rangle \otimes |\mathcal{D}\rangle) &= |x, y\rangle \otimes |\mathcal{D}\rangle(|\perp, 0^\lambda\rangle), \text{ and} \\ \text{CPhsO}'(|x, y\rangle \otimes |\mathcal{D}\rangle) &= (-1)^{y \cdot \mathcal{D}(x)} |x, y\rangle \otimes |\mathcal{D}\rangle,\end{aligned}$$

where the procedure Increase appends a new register $(|\perp, 0^\lambda\rangle)$ at the end of the database. Note that $|\mathcal{D}\rangle(|\perp, 0^\lambda\rangle)$ is a database that computes the same partial function as \mathcal{D} , but the upper bound on the number of points is increased by 1. Here, we remark that we define $\perp \cdot y = 0$ when defining CPhsO' , which implies that CPhsO' does nothing if (x, y) has not yet been added to the database \mathcal{D} . Finally, the compressed phase oracle CPhsO can be defined as follows:

$$\text{CPhsO} = \text{StdDecomp} \circ \text{CPhsO}' \circ \text{StdDecomp} \circ \text{Increase},$$

which means that when we receive a query, we first make enough space by increasing the bound and then decompress at x , apply the query, and then re-compress the database. We remark that CPhsO successfully keeps track of positions that are orthogonal to the uniform superposition only because if (x, y) was already specified in \mathcal{D} and the y registers are in the state of a uniform superposition, then StdDecomp removes x from \mathcal{D} so that CPhsO' does nothing as explained before and the second StdDecomp in CPhsO will revert (x, y) back to the database.

2.2 Useful Lemmas for Compressed Oracles

Next we introduce some useful lemmas given by Zhandry [42] that are helpful for proving our main result. We first introduce the following variant of Lemma 5 from [42], which is still true for CPhsO because StO and PhsO are perfectly indistinguishable by applying a Hadamard transform before and after each query.

► **Lemma 5** ([42]). *Consider a quantum algorithm \mathcal{A} making queries to a random oracle H and outputting tuples $(x_1, \dots, x_k, y_1, \dots, y_k, z)$. Let R be a collection of such tuples. Suppose with probability p , \mathcal{A} outputs a tuple such that (1) the tuple is in R , and (2) $\mathcal{H}(x_i) = y_i$ for all i . Now consider running \mathcal{A} with the oracle CPhsO, and suppose the database \mathcal{D} is measured after \mathcal{A} produces its output. Let p' be the probability that (1) the tuple is in R , and (2) $\mathcal{D}(x_i) = y_i$ for all i (and in particular $\mathcal{D}(x_i) \neq \perp$). Then $\sqrt{p} \leq \sqrt{p'} + \sqrt{k/2^n}$.*

We say that a database \mathcal{D} contains a collision if we have $(x, y), (x', y) \in \mathcal{D}$ for $x \neq x'$. We use the notation COLLIDE to denote the set of all databases that contain a collision. Zhandry upper bounded the probability of finding collisions in a database after making queries to a compressed oracle in the following lemma.

► **Lemma 6** ([42]). *For any adversary making q queries to CPhsO and an arbitrary number of database read queries, if the database \mathcal{D} is measured after the q queries, the resulting database will contain a collision with probability at most $q^3/2^\lambda$.*

3 Parallel Quantum Random Oracle Model

Quantum Query Bounds with Compressed Dataset

As a warm-up, we review how Zhandry [42] used his compressed oracle technique to provide a greatly simplified proof that Grover's algorithm is asymptotically optimal. Theorem 7 proves that the final measured database \mathcal{D} will not contain a pre-image of 0^λ except with probability $\mathcal{O}(q^2/2^\lambda)$. We sketch some of the key ideas below as a warm-up and to highlight some of the additional challenges faced in our setting.

► **Theorem 7** ([42]). *For any adversary making q queries to CPhsO and an arbitrary number of database read queries, if the database \mathcal{D} is measured after the q queries, the probability it contains a pair of the form $(x, 0^\lambda)$ is at most $\mathcal{O}(q^2/2^\lambda)$.*

We can view Theorem 7 as providing a bound for amplitudes of basis states with a database \mathcal{D} in a set BAD that is defined as

$$\text{BAD} = \{\mathcal{D} : \mathcal{D} \text{ contains a pair of the form } (x, 0^\lambda)\}.$$

Given a basis state $|x, y, z\rangle \otimes |\mathcal{D}\rangle$ with $\mathcal{D} \notin \text{BAD}$ and $x \notin \mathcal{D}$, then the random oracle CPhsO maps this basis state to

$$|\psi\rangle = |x, y, z\rangle \otimes \frac{1}{2^{\lambda/2}} \sum_w (-1)^{y \cdot w} |\mathcal{D} \cup (x, w)\rangle,$$

where the amplitude on states with the corresponding database \mathcal{D} in BAD is just $2^{-\lambda/2}$. We use the following notation to generalize this approach for other purposes:

► **Definition 8.** *For a collection of basis states $\tilde{\mathcal{S}}$ and $|\psi\rangle = \sum_X \alpha_X |X\rangle$, we define*

$$L_2(|\psi\rangle, \tilde{\mathcal{S}}) = \sqrt{\sum_{X \in \tilde{\mathcal{S}}} |\alpha_X|^2}$$

to denote the root of the sum of the squared magnitudes of the projection of ψ onto the set of basis states $\tilde{\mathcal{S}}$. If \mathcal{S} is a set of databases and $|\psi\rangle = \sum_{x,y,z,\mathcal{D}} \alpha_{x,y,z,\mathcal{D}} |x, y, z\rangle \otimes |\mathcal{D}\rangle$ is a state we define $L_2(|\psi\rangle, \mathcal{S}) = \sqrt{\sum_{x,y,z} \sum_{\mathcal{D} \in \mathcal{S}} |\alpha_{x,y,z,\mathcal{D}}|^2}$.

An equivalent way to define $L_2(|\psi\rangle, \tilde{\mathcal{S}})$ is $L_2(|\psi\rangle, \tilde{\mathcal{S}}) = \left\| P_{\tilde{\mathcal{S}}}(|\psi\rangle) \right\|_2$ where $P_{\tilde{\mathcal{S}}}$ projects $|\psi\rangle$ onto the space spanned by $\tilde{\mathcal{S}}$ e.g., if $|\psi\rangle = \sum_{x,y,z,\mathcal{D}} \alpha_{x,y,z,\mathcal{D}} |x, y, z\rangle \otimes |\mathcal{D}\rangle$ then

$$P_{\tilde{\mathcal{S}}}(|\psi\rangle) = \sum_{|x,y,z\rangle \otimes |\mathcal{D}\rangle \in \tilde{\mathcal{S}}} \alpha_{x,y,z,\mathcal{D}} |x, y, z\rangle \otimes |\mathcal{D}\rangle.$$

Using this notation, we can view the proof of Theorem 7 as bounding $L_2(\text{CPhsO}|\psi\rangle, \text{BAD}) - L_2(|\psi\rangle, \text{BAD})$, i.e., the increase in root of squared amplitudes on bad states after each random oracle query.

There are a number of challenges to overcome when directly applying this idea to analyze \mathcal{H} -sequences. First, we note that Theorem 7 works in the sequential qROM, which means that the attacker can make only one query in each round. In our setting, the quantum

attacker is allowed to make more than T queries provided that the queries are submitted in parallel batches over $T - 1$ rounds. Without the latter restriction, an attacker that makes T total queries will trivially be able to find an \mathcal{H} -sequence, even if the attacker is not quantum, by computing $\mathcal{H}^T(x)$ over T rounds. We formalize the *parallel quantum random oracle model* pqROM in Section 3.1 to model an attacker who submits batches $(x_1, y_1), \dots$ of random oracle queries in each round.

The second primary challenge is that we can not find a static (a priori fixed) set BAD . A naïve approach would fix BAD to be the set of databases that contain an \mathcal{H} -sequence of length T , but this would not allow us to bound $L_2(\text{CPhsO}|\psi\rangle, \text{BAD}) - L_2(|\psi\rangle, \text{BAD})$. In particular, if our state $|\psi\rangle$ after round r has non-negligible squared amplitudes on datasets \mathcal{D} that contain an \mathcal{H} -sequence of length $r + 1$, then it is likely that the attacker will be able to produce an \mathcal{H} -sequence of length T after round $T - 1$ – in this sense a bad event has already occurred. In our setting, the bad sets must be defined carefully in a round-dependent fashion r . Intuitively, we want to show that $L_2(\text{CPhsO}^k|\psi\rangle, \text{BAD}_{r+1}) - L_2(|\psi\rangle, \text{BAD}_r)$ is small, where BAD_r contains datasets \mathcal{D} that contain an \mathcal{H} -sequence of length $r + 1$ and CPhsO^k denotes a parallel phase oracle that processes $k \geq 1$ queries in each round. However, reasoning about the behavior of CPhsO^k introduces its own set of challenges when $k > 1$. We address these challenges by carefully defining sets $\text{BAD}_{r,j}$, i.e., the bad set of states after the first j (out of k) queries in round j have been processed. See Section 4 for more details.

3.1 Parallel Quantum Random Oracle Model pqROM

Recall that in the sequential quantum random oracle model (qROM), an adversary can submit quantum states as queries to the oracle, so that a random hash function H takes as input superposition $|\phi_1\rangle, |\phi_2\rangle, \dots$, and so on. Each $|\phi_i\rangle$ can be expanded as $|\phi_i\rangle = \sum \alpha_{i,x,y} |x, y\rangle$ so that the output is $\sum \alpha_{i,x,y} |x, y \oplus \mathcal{H}(x)\rangle$. Note that when the initial state is of the form $|\phi\rangle = \sum \alpha_x |x, 0^w\rangle$, then the output state will be of the form $\sum \alpha_x |x, \mathcal{H}(x)\rangle$. In the *parallel quantum random oracle model* (pqROM), the adversary can make a batch of queries q_1, q_2, \dots each round and receives the corresponding output for each of the queries. More precisely, if r_i is the number of queries made in round i , then the input takes the form $|(x_1, y_1), \dots, (x_{r_i}, y_{r_i})\rangle$ and the corresponding output is $|(x_1, y_1 \oplus \mathcal{H}(x_1)), \dots, (x_{r_i}, y_{r_i} \oplus \mathcal{H}(x_{r_i}))\rangle$.

We also remark that the equivalence of the standard and phase oracles that we discussed in Section 2.1 remains true with parallelism (by applying Hadamard transforms before and after the query); therefore, we will only consider extending the compressed oracles only on the CPhsO by convenience.

Extending Compressed Oracle Technique to pqROM

As mentioned before, we need to extend CPhsO to be able to handle parallel queries. To make the analysis simpler, we have an approach that essentially sequentially simulates a batch of parallel queries, which as a result, is equivalent to process parallel queries at once. Consider the following example; given a state $|\mathcal{B}_i\rangle = |(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle$, let $|\psi_1\rangle$ be the state after processing the first query (x_1, y_1) , and let $|\psi_2\rangle$ be the state after processing the second query (x_2, y_2) so that $|\psi_2\rangle = \text{CPhsO}|\psi_1\rangle$. However, recall that CPhsO only acts on the first coordinate. Thus to handle the parallel query sequentially, we would need to switch the order of the coordinates to process the second query properly. Hence, we make a slight modification and redefine $|\psi_2\rangle = \text{Swap}_{1,2} \circ \text{CPhsO} \circ \text{Swap}_{1,2}|\psi_1\rangle$, where

$$\text{Swap}_{1,2} |(x_1, y_1), (x_2, y_2), \dots\rangle = |(x_2, y_2), (x_1, y_1), \dots\rangle,$$

and similarly $\text{Swap}_{i,j}(\cdot) = \text{Swap}_{j,i}(\cdot)$ swaps the positions of queries x_i and x_j . Thus, we now define our *parallel* version of a CPhsO oracle, called as CPhsO^i , which handle i parallel queries, recursively as

$$\text{CPhsO}^i = \text{Swap}_{1,i} \circ \text{CPhsO} \circ \text{Swap}_{1,i} \circ \text{CPhsO}^{i-1},$$

where $\text{CPhsO}^1 = \text{CPhsO}$. For a compact notation, we define

$$\text{SCPhsO}_i := \text{Swap}_{1,i} \circ \text{CPhsO} \circ \text{Swap}_{1,i}.$$

That is, we can interpret CPhsO^i as applying SCPhsO_j (essentially) sequentially for $j = 1, \dots, i$, i.e., $\text{CPhsO}^i = \prod_{j=1}^i \text{SCPhsO}_j = \text{SCPhsO}^i \circ \dots \circ \text{SCPhsO}^1$.

We remark that there are other possible approaches in extending CPhsO to the pqROM. For example, see the full version for further details regarding an additional approach to extending CPhsO to the pqROM.

4 Finding \mathcal{H} -Sequences in the pqROM

In this section, we show that quantum adversaries cannot find \mathcal{H} -sequences of length N using fewer than $N - 1$ steps, thereby showing the security of a construction for proof of sequential work in the *parallel* quantum random oracle model in the next section. Recall that an \mathcal{H} -sequence $x_0, x_1, \dots, x_s \in \{0, 1\}^*$ satisfies the property that for each $0 \leq i \leq s - 1$, there exist $a, b \in \{0, 1\}^*$ such that $x_{i+1} = a \parallel \mathcal{H}(x_i) \parallel b$. Note that the sequence $\mathcal{H}(x), \mathcal{H}^2(x), \dots, \mathcal{H}^N(x)$ is an \mathcal{H} -sequence, so in fact, this proves that quantum adversaries are even more limited than being unable to compute $\mathcal{H}^N(x)$ for a given input x in fewer than $N - 1$ steps. In our analysis, we require that \mathcal{H} outputs a λ -bit string but permit each term x_i in the \mathcal{H} -sequence to have length $\delta\lambda$, for some variable parameter $\delta \geq 1$.

We begin by introducing some helpful notation. Given a database $\mathcal{D} = \{(x_1, y_1), \dots, (x_q, y_q)\}$ in the compressed standard oracle view, we can define a directed graph $G_{\mathcal{D}}$ on q nodes $(v_{x_1}, \dots, v_{x_q})$ so that there is an edge from node v_{x_i} to node v_{x_j} if and only if there exist strings a, b such that $x_j = a \parallel y_i \parallel b$. Thus, the graph $G_{\mathcal{D}}$ essentially encodes possible \mathcal{H} -sequences by forming edges between nodes v_{x_i} and v_{x_j} if and only if y_i is a substring of x_j . More precisely, given a path $P = (v_{x_{i_0}}, v_{x_{i_1}}, \dots, v_{x_{i_k}})$ in $G_{\mathcal{D}}$, we define

- $\text{HSeq}(P) := (x_{i_0}, x_{i_1}, \dots, x_{i_k})$ denotes a corresponding \mathcal{H} -sequence of length k , and
- $\text{LAST}(P) := v_{x_{i_k}}$ denotes the endpoint of the path P in $G_{\mathcal{D}}$ that corresponds to the output of the last label in the corresponding \mathcal{H} -sequence.

We also define a predicate $\text{Substring}(x, y)$ where $\text{Substring}(x, y) = 1$ if and only if x is a substring of y , i.e., there exist $a, b \in \{0, 1\}^*$ such that $y = a \parallel x \parallel b$, and $\text{Substring}(x, y) = 0$ otherwise.

► **Example 9.** Suppose that $\mathcal{D} = \{(x_1, y_1), \dots, (x_8, y_8)\}$ where $(x_1, y_1) = (00000, 000)$, $(x_2, y_2) = (00010, 001)$, $(x_3, y_3) = (00101, 010)$, $(x_4, y_4) = (00110, 011)$, $(x_5, y_5) = (01001, 100)$, $(x_6, y_6) = (01100, 101)$, $(x_7, y_7) = (10010, 110)$, and $(x_8, y_8) = (11001, 111)$. We observe that the graph $G_{\mathcal{D}}$ induced from the database \mathcal{D} should include the edge (v_1, v_2) since $x_2 = 00010 = y_1 \parallel 10$, and so forth. Then we have the following graph $G_{\mathcal{D}}$ (see the full version), which includes an \mathcal{H} -sequence of length $s = 5$. In this example, we can say that for a path $P = (v_1, v_2, v_3, v_5, v_7, v_8)$ of length 5, we have a corresponding \mathcal{H} -sequence $\text{HSeq}(P) = (x_1, x_2, x_3, x_5, x_7, x_8)$ of length 5 since we have $x_2 = y_1 \parallel 10$, $x_3 = y_2 \parallel 01$, and so on. Note that in this case we have $\text{LAST}(P) = v_8$.

22:12 On the Security of Proofs of Sequential Work in a Post-Quantum World

► **Definition 10.** We define PATH_s to be the set of the databases (compressed random oracles) \mathcal{D} such that $G_{\mathcal{D}}$ contains a path of length s .

$$\text{PATH}_s := \{\mathcal{D} : G_{\mathcal{D}} \text{ contains a path of length } s\}.$$

Note that PATH_s intuitively corresponds to an \mathcal{H} -sequence of length s . We also define $\widetilde{\text{PATH}}_s$ to be the set of basis states with \mathcal{D} in PATH_s as follows:

$$\widetilde{\text{PATH}}_s := \{|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle : \mathcal{D} \in \text{PATH}_s\}.$$

Challenges of Quantum Query Bounds on Finding an \mathcal{H} -Sequence

To bound the probability that a single round of queries finds an \mathcal{H} -sequence of length $s + 1$ conditioned on the previous queries not finding an \mathcal{H} -sequence of length s , we consider the set of basis states $\{|B_i\rangle\}_i$ of the form $|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle$, where \mathcal{D} contains at most $q - k$ entries and $\mathcal{D} \notin \text{PATH}_s$. Let $|\psi\rangle = \sum \alpha_i |B_i\rangle$ be an arbitrary state that is a linear combination of $\{|B_i\rangle\}_i$ and let $|\psi'\rangle = \text{CPhsO}^k |\psi\rangle$. Then we would like to bound $L_2(|\psi'\rangle, \text{PATH}_{s+1})$, but there are substantial challenges in computing $L_2(|\psi'\rangle, \text{PATH}_{s+1})$ directly.

For example, given a decomposition of $|\psi\rangle = \sum_B \alpha_B |B\rangle$ into basis states, we might like to compute $\eta_B = \text{CPhsO}^k(|B\rangle)$ for each basis state $|B\rangle$ and then decompose $|\psi'\rangle = \sum_B \alpha_B \eta_B$. However, each term η_B is no longer a basis state making it difficult to describe the state $|\psi'\rangle$ in a helpful way so that we can bound $L_2(|\psi'\rangle, \widetilde{\text{PATH}}_{s+1})$. The challenges are amplified as $|\psi'\rangle$ is the result of k parallel queries.

Overcoming the Challenges

Our approach is to consider an intermediate sequence of states $|\psi_0\rangle = |\psi\rangle, \dots, |\psi_k\rangle = |\psi'\rangle$, where $|\psi_i\rangle$ intuitively encodes the state after the i^{th} query (in the block of parallel queries) is processed. Then from the definition of CPhsO^i , we have $|\psi_{i+1}\rangle = \text{Swap}_{1,i+1} \circ \text{CPhsO} \circ \text{Swap}_{1,i+1} |\psi_i\rangle = \text{SCPhsO}_{i+1} |\psi_i\rangle$ for all $i \in [k]$. This approach presents a new subtle challenge. Consider a basis state $|B\rangle = |(x_1, y_1), \dots, (x_k, y_k)\rangle \otimes |\mathcal{D}\rangle$, where the longest path in $G_{\mathcal{D}}$ (the \mathcal{H} -sequence) has length $s - 1$. We can easily argue that $L_2(\text{SCPhsO}_1 |B\rangle, \widetilde{\text{PATH}}_{s+1})$ is negligible since the initial basis state $|B\rangle \notin \widetilde{\text{PATH}}_s$. Now we would like to argue that $L_2(\text{SCPhsO}_2 \circ \text{SCPhsO}_1 |B\rangle, \widetilde{\text{PATH}}_{s+1})$ is negligibly small, but it is unclear how to prove this since we might have $L_2(\text{SCPhsO}_1 |B\rangle, \widetilde{\text{PATH}}_s) = 1$, e.g., all of the datasets \mathcal{D} found in the support of $\text{SCPhsO}_1 |B\rangle$ have paths of length s .

Overcoming this barrier requires a much more careful definition of our “bad” states. We introduce some new notions to make the explanations clearer. Suppose that a database $\mathcal{D} \notin \text{PATH}_s$. If \mathcal{D} has no \mathcal{H} -sequence of length s , it may not be the case that $|\psi_i\rangle$ has no \mathcal{H} -sequence of length s . However, the intuition is that since the queries x_1, \dots, x_k are made in the same round, then it is acceptable to have an \mathcal{H} -sequence of length s , provided that the last entry in the \mathcal{H} -sequence involves some (x_i, y_i) . Thus we define $\text{PATH}_{s,i}(x_1, \dots, x_k)$ to be a set of the databases with the induced graph $G_{\mathcal{D}}$ having a path of length s that does not end in a term that contains $\mathcal{H}(x_i)$:

$$\text{PATH}_{s,i}(x_1, \dots, x_k) := \{\mathcal{D} : G_{\mathcal{D}} \text{ contains a path } P \text{ of length } s \text{ and} \\ \text{LAST}(P) \notin \{v_{x_1}, \dots, v_{x_i}\}\},$$

where we recall that $\text{LAST}(P)$ denotes the endpoint of the path P in $G_{\mathcal{D}}$, which corresponds to the output of the last label in the corresponding \mathcal{H} -sequence as defined before. We then define

$$\widetilde{\text{PATH}}_{s,i} := \{|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle : \mathcal{D} \in \text{PATH}_{s,i}(x_1, \dots, x_k)\},$$

which denotes the set of the states where the corresponding database \mathcal{D} is in the set $\text{PATH}_{s,i}(x_1, \dots, x_k)$.

Now we define a set $\text{Contain}_{s,i}$, which intuitively represents the set of databases so that the queries correspond to the guesses for preimages:

$$\text{Contain}_{s,i}(x_1, \dots, x_k) := \{\mathcal{D} : \exists j \leq k \text{ s.t. } \text{Substring}(\mathcal{D}(x_i), x_j) = 1 \text{ and } G_{\mathcal{D}} \text{ contains a path of length } s \text{ ending at } x_i\}.$$

We then define

$$\widetilde{\text{Contain}}_{s,i} := \{|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle : \mathcal{D} \in \text{Contain}_{s,i}(x_1, \dots, x_k)\},$$

which denotes the set of the states where the corresponding database \mathcal{D} is in the set $\text{Contain}_{s,i}(x_1, \dots, x_k)$. Therefore, we define $\text{BAD}_{s,i}(x_1, \dots, x_k)$ to be the set of databases that are not in PATH_s but upon the insertion of $(x_1, y_1), \dots, (x_i, y_i)$, is a member of PATH_{s+1} :

$$\text{BAD}_{s,i}(x_1, \dots, x_k) := \text{PATH}_{s,i}(x_1, \dots, x_k) \cup \bigcup_{j=1}^i \text{Contain}_{s,j}(x_1, \dots, x_k)$$

Finally, we define

$$\widetilde{\text{BAD}}_{s,i} := \{|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle : \mathcal{D} \in \text{BAD}_{s,i}(x_1, \dots, x_k)\}$$

to represent the set of the states where the corresponding database \mathcal{D} is in the set $\text{BAD}_{s,i}(x_1, \dots, x_k)$.

We now process each query $(x_1, y_1), \dots, (x_k, y_k)$ sequentially and argue that the mass projected onto PATH_{s+1} by each step CPhsO^i is negligible. To prove this, we argue that $L_2(|\psi_i\rangle, \widetilde{\text{BAD}}_{s,i})$ is negligible for all $i \leq k$. We use the convention that $|\psi_0\rangle$ is the initial state and $|\psi_i\rangle = \text{SCPhsO}_i|\psi_{i-1}\rangle$ for all $i \in [k]$ so that $|\psi_k\rangle$ is the last state, after all the queries have been processed. Similarly, we use the convention that $\widetilde{\text{BAD}}_{s,0} = \widetilde{\text{PATH}}_s$.

We first give the following lemma.

► **Lemma 11.** *Suppose that \mathcal{D}' is a database such that $\mathcal{D}'(x_{i+1}) = \perp$. If $\mathcal{D} = \mathcal{D}' \cup (x_{i+1}, w) \notin \text{BAD}_{s,i}(x_1, \dots, x_k)$, then $\mathcal{D} \notin \text{BAD}_{s,i+1}(x_1, \dots, x_k)$.*

Proof. Since $\mathcal{D} \notin \text{PATH}_{s,i}(x_1, \dots, x_k)$, any path of length s must end at one of the vertices v_{x_1}, \dots, v_{x_i} in the graph $G_{\mathcal{D}}$. Hence, no path of length s ends at $v_{x_{i+1}}$ unless we have a duplicate query $x_j = x_{i+1}$ for some $j < i + 1$. Now we have two cases:

- (1) If x_{i+1} is distinct from x_j for all $j < i + 1$, then by the previous observation we immediately have that $\mathcal{D} \notin \text{PATH}_{s,i+1}(x_1, \dots, x_k)$. Furthermore, $G_{\mathcal{D}}$ contains no path of length s ending at node $v_{x_{i+1}}$ since any path of length s must end at one of v_{x_1}, \dots, v_{x_i} . Hence, $\mathcal{D} \notin \text{Contain}_{s,i+1}(x_1, \dots, x_k)$. Taken together, we have that $\mathcal{D} \notin \text{BAD}_{s,i+1}(x_1, \dots, x_k)$.
- (2) If $x_{i+1} = x_j$ ($j < i + 1$) is a duplicate query, then there might be a path of length s ending at $v_{x_{i+1}} = v_{x_j}$ in \mathcal{D} . However, due to the duplicate we have $\{v_{x_1}, \dots, v_{x_i}\} = \{v_{x_1}, \dots, v_{x_{i+1}}\}$. Therefore, $\mathcal{D} \notin \text{PATH}_{s,i+1}(x_1, \dots, x_k)$. Now we want to argue that

$\mathcal{D} \notin \text{Contain}_{s,i+1}(x_1, \dots, x_k)$. Note that $\mathcal{D}(x_{i+1}) = \mathcal{D}(x_j)$ for some $j < i + 1$, which implies that $\text{Substring}(\mathcal{D}(x_j), x_l) = 1 \Leftrightarrow \text{Substring}(\mathcal{D}(x_{i+1}), x_l) = 1$ for all $l \in [k]$. If $\mathcal{D} \in \text{Contain}_{s,i+1}(x_1, \dots, x_k)$, then there exists a path of length s ending at x_j and $\text{Substring}(\mathcal{D}(x_j), x_l) = 1$ for some $l \leq k$. This implies that $\mathcal{D} \in \text{Contain}_{s,j}(x_1, \dots, x_k)$ and therefore $\mathcal{D} \in \text{BAD}_{s,i}(x_1, \dots, x_k)$, which is a contradiction. Hence, we have that $\mathcal{D} \notin \text{Contain}_{s,i+1}(x_1, \dots, x_k)$ and therefore $\mathcal{D} \notin \text{BAD}_{s,i+1}(x_1, \dots, x_k)$ in this case as well. Taken together, we can conclude that if $\mathcal{D} \notin \text{BAD}_{s,i}(x_1, \dots, x_k)$, then it is also the case that $\mathcal{D} \notin \text{BAD}_{s,i+1}(x_1, \dots, x_k)$. \blacktriangleleft

► Lemma 12. For each $i \in \{0, 1, \dots, k-1\}$,

$$L_2(|\psi_{i+1}\rangle, \widetilde{\text{BAD}}_{s,i+1}) - L_2(|\psi_i\rangle, \widetilde{\text{BAD}}_{s,i}) \leq \frac{4\sqrt{q\delta\lambda + k\delta\lambda}}{2^{\lambda/2}}.$$

Proof. To argue that the projection onto $\widetilde{\text{BAD}}_{s,i}$ increases by a negligible amount for each query, we use a similar argument to [42]. Recall that $\text{SCPhsO}_{i+1} = \text{Swap}_{1,i+1} \circ \text{CPhsO} \circ \text{Swap}_{1,i+1}$. Namely, we consider the projection of $|\psi_{i+1}\rangle = \text{SCPhsO}_{i+1}|\psi_i\rangle$ onto orthogonal spaces as follows:

- We first define P_i (resp. P) to be the projection onto the span of basis states $|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle \in \widetilde{\text{BAD}}_{s,i}$ (resp. basis states in $\widetilde{\text{BAD}}_{s,i+1}$).
- Next we define Q_i to be the projection onto states $|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle \notin \widetilde{\text{BAD}}_{s,i}$ such that $y_{i+1} \neq 0$ and $\mathcal{D}(x_{i+1}) = \perp$. Intuitively, Q_i represents the projection onto states that are not bad where $\text{SCPhsO}_{i+1}|\psi_i\rangle = \sum_w |\mathcal{D} \cup (x_{i+1}, w)\rangle$ will add a new tuple (x_{i+1}, w) to the dataset.
- We then define R_i to be the projection onto states $|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle$ such that $|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle \notin \widetilde{\text{BAD}}_{s,i}$, $y_{i+1} \neq 0$ and $\mathcal{D}(x_{i+1}) \neq \perp$, so that the value of x_{i+1} has been specified in databases corresponding to these states.
- Finally, we define S_i to be the projection onto states $|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle$ such that $|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle \notin \widetilde{\text{BAD}}_{s,i}$ and $y_{i+1} = 0$.

Since P_i, Q_i, R_i, S_i project onto disjoint states that span the entirety of $|\psi_{i+1}\rangle$ then we have $P_i + Q_i + R_i + S_i = \mathbb{I}$, where \mathbb{I} denotes the identity operator. We analyze how P acts on these components separately. For the component $P_i|\psi_i\rangle$, it is easy to verify that $\|P \circ \text{SCPhsO}_{i+1}(P_i|\psi_i)\|_2 \leq \|\text{SCPhsO}_{i+1}(P_i|\psi_i)\|_2 \leq \|P_i|\psi_i\rangle\|_2$. See the full version for the formal proof of Lemma 13.

► Lemma 13. $\|P \circ \text{SCPhsO}_{i+1}(P_i|\psi_i)\|_2 \leq \|P_i|\psi_i\rangle\|_2$.

Next, to analyze how the projection P acts on $\text{SCPhsO}_{i+1}(Q_i|\psi_i)$, we note that $\text{SCPhsO}_{i+1}(|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle) = |(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes \sum_w 2^{-\lambda/2} (-1)^{w \cdot y_{i+1}} |\mathcal{D} \cup (x_{i+1}, w)\rangle$ for any basis state in the support of $Q_i|\psi_i\rangle$. We then use a classical counting argument to upper bound the number of strings w such that $|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes \sum_w |\mathcal{D} \cup (x_{i+1}, w)\rangle \in \widetilde{\text{BAD}}_{s,i+1}$ by decomposing the databases in $\text{BAD}_{s,i+1}(x_1, \dots, x_k)$ into the databases in $\text{PATH}_{s,i+1}(x_1, \dots, x_k)$ and the databases in $\bigcup_{j=1}^{i+1} \text{Contain}_{s,j}(x_1, \dots, x_k)$. Intuitively, since $\mathcal{D} \notin \text{BAD}_{s,i}(x_1, \dots, x_k)$ the only way to have $\mathcal{D} \cup (x_{i+1}, w) \in \text{PATH}_{s,i+1}(x_1, \dots, x_k)$ is if w is a substring of some x_j with $j \leq k$ or w is the substring of some other input x in the database \mathcal{D} . We bound the number of databases in $\text{PATH}_{s,i+1}(x_1, \dots, x_k)$ by noting that any string $x \in \{0, 1\}^{\delta\lambda}$ contains at most $\delta\lambda$ unique contiguous substrings of length λ , so there are at most $\delta\lambda$ values of w such that $\text{Substring}(w, x) = 1$. Since $|\mathcal{D} \cup (x_{i+1}, w)\rangle$ contains at most q entries, then by a union bound, there are at most $q\delta\lambda$ strings w such

that exists $x \in \{0, 1\}^*$ such that $\text{Substring}(w, x) = 1$ and $\mathcal{D}(x) \neq \perp$ or $x = x_{i+1}$. Thus, $|\{w : \mathcal{D} \cup (x_{i+1}, w) \in \text{BAD}_{s,i+1}(x_1, \dots, x_k)\}| \leq q\delta\lambda$. We similarly bound the number of databases in $\bigcup_{j=1}^{i+1} \text{Contain}_{s,j}(x_1, \dots, x_k)$ by noting that if $\mathcal{D} \notin \text{BAD}_{s,i}(x_1, \dots, x_k)$ then the only way for $\mathcal{D} \cup (x_{i+1}, w)$ to be in $\bigcup_{j=1}^{i+1} \text{Contain}_{s,j}(x_1, \dots, x_k)$ is if for some $j \leq k$ the string w is a substring of x_j i.e., $\text{Substring}(w, x_j) = 1$. A similar argument shows that the number of strings w such that $\mathcal{D} \cup (x_{i+1}, w) \in \bigcup_{j=1}^{i+1} \text{Contain}_{s,j}(x_1, \dots, x_k)$ is at most $k\delta\lambda$.

We thus show the following (see the full version for the full proof):

► **Lemma 14.** $\|P \circ \text{SCPhsO}_{i+1}(Q_i|\psi_i)\|_2^2 \leq \frac{q\delta\lambda + k\delta\lambda}{2^\lambda}$.

We next consider how P acts upon the basis states of $\text{SCPhsO}_{i+1}(R_i|\psi_i)$. We first relate this quantity to $\text{SCPhsO}_{i+1}(|x, y, z\rangle \otimes |\mathcal{D}' \cup (x_{i+1}, w)\rangle)$, where \mathcal{D}' is the database \mathcal{D} with x_{i+1} removed. Since $\mathcal{D} = |\mathcal{D}' \cup (x_{i+1}, w)\rangle \notin \text{BAD}_{s,i}(x_1, \dots, x_k)$, then we again use a similar classical counting argument to upper bound the number of strings w' such that $\mathcal{D}' \cup (x_{i+1}, w') \in \text{BAD}_{s,i+1}(x_1, \dots, x_k)$. Lemma 15 then follows from algebraic manipulation similar to [42]. See the full version for the formal proof.

► **Lemma 15.** $\|P \circ \text{SCPhsO}_{i+1}(R_i|\psi_i)\|_2^2 \leq \frac{9(q\delta\lambda + k\delta\lambda)}{2^\lambda}$.

Finally, we bound the projection of P onto the states of $\text{SCPhsO}_{i+1}(S_i|\psi_i)$:

► **Lemma 16.** $\|P \circ \text{SCPhsO}_{i+1}(S_i|\psi_i)\|_2 = 0$.

Proof. We observe that $P \circ \text{SCPhsO}_{i+1}(S_i|\psi_i) = 0$, since for any basis state $|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle$ state in the support of $S_i|\psi_i\rangle$, we have

$$\text{SCPhsO}_{i+1}(|(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle) = |(x_1, y_1), \dots, (x_k, y_k), z\rangle \otimes |\mathcal{D}\rangle,$$

i.e., $\text{SCPhsO}_{i+1}(S_i|\psi_i) = S_i|\psi_i\rangle$. We also note that since $\mathcal{D} \notin \text{BAD}_{s,i}(x_1, \dots, x_k)$ and x_{i+1} is not being inserted into the dataset that $\mathcal{D} \notin \text{BAD}_{s,i+1}(x_1, \dots, x_k)$. Hence, we have that $\|P \circ \text{SCPhsO}_{i+1}(S_i|\psi_i)\|_2 = 0$. ◀

Thus from Lemma 13, Lemma 14, Lemma 15, Lemma 16, and by triangle inequality, we have

$$\begin{aligned} \|P \circ \text{SCPhsO}_{i+1}|\psi_i\rangle\|_2 &\leq \|P \circ \text{SCPhsO}_{i+1}(P_i|\psi_i)\|_2 + \|P \circ \text{SCPhsO}_{i+1}(Q_i|\psi_i)\|_2 \\ &\quad + \|P \circ \text{SCPhsO}_{i+1}(R_i|\psi_i)\|_2 + \|P \circ \text{SCPhsO}_{i+1}(S_i|\psi_i)\|_2 \\ &\leq \|P_i|\psi_i\rangle\|_2 + \frac{3\sqrt{q\delta\lambda + k\delta\lambda}}{2^{\lambda/2}} + \frac{\sqrt{q\delta\lambda + k\delta\lambda}}{2^{\lambda/2}} \\ &\leq \|P_i|\psi_i\rangle\|_2 + \frac{4\sqrt{q\delta\lambda + k\delta\lambda}}{2^{\lambda/2}}. \end{aligned}$$

Since we have $\|P \circ \text{SCPhsO}_i|\psi_i\rangle\|_2 = L_2(|\psi_{i+1}\rangle, \widetilde{\text{BAD}}_{s,i+1})$ and $\|P_i|\psi_i\rangle\|_2 = L_2(|\psi_i\rangle, \widetilde{\text{BAD}}_{s,i})$, we can conclude that $L_2(|\psi_{i+1}\rangle, \widetilde{\text{BAD}}_{s,i+1}) - L_2(|\psi_i\rangle, \widetilde{\text{BAD}}_{s,i}) \leq \frac{4\sqrt{q\delta\lambda + k\delta\lambda}}{2^{\lambda/2}}$. ◀

We now bound the probability that a single round of queries finds an \mathcal{H} -sequence of length $s+1$ conditioned on the previous queries not finding an \mathcal{H} -sequence of length s .

► **Lemma 17.** *Let $|\psi\rangle$ be an initial state and let $|\psi'\rangle = \text{CPhsO}^k|\psi\rangle$. Then we have $L_2(|\psi'\rangle, \widetilde{\text{PATH}}_{s+1}) - L_2(|\psi\rangle, \widetilde{\text{PATH}}_s) \leq \frac{4k\sqrt{q\delta\lambda + k\delta\lambda}}{2^{\lambda/2}}$.*

Proof. We consider the sequence of states $|\psi\rangle = |\psi_0\rangle, \dots, |\psi_k\rangle = |\psi'\rangle$ with $|\psi_i\rangle = \text{CPhsO}^i|\psi\rangle$. By Claim 18 it suffices to bound $L_2(|\psi_k\rangle, \widetilde{\text{BAD}}_{s,k})$ as $L_2(|\psi_k\rangle, \widetilde{\text{PATH}}_{s+1}) \leq L_2(|\psi_k\rangle, \widetilde{\text{BAD}}_{s,k})$. See the full version for the proof of Claim 18.

22:16 On the Security of Proofs of Sequential Work in a Post-Quantum World

▷ Claim 18. $\widetilde{\text{PATH}}_{s+1} \subseteq \widetilde{\text{BAD}}_{s,k}$.

Recall that we use the convention $\widetilde{\text{BAD}}_{s,0} = \widetilde{\text{PATH}}_s$ and $|\psi_0\rangle$ is the initial state so that by Lemma 12,

$$\begin{aligned} L_2(|\psi_k\rangle, \widetilde{\text{BAD}}_{s,k}) &= L_2(|\psi_0\rangle, \widetilde{\text{BAD}}_{s,0}) + \sum_{i=0}^{k-1} [L_2(|\psi_{i+1}\rangle, \widetilde{\text{BAD}}_{s,i+1}) - L_2(|\psi_i\rangle, \widetilde{\text{BAD}}_{s,i})] \\ &\leq L_2(|\psi_0\rangle, \widetilde{\text{BAD}}_{s,0}) + \sum_{i=0}^{k-1} \frac{4\sqrt{q\delta\lambda + k\delta\lambda}}{2^{\lambda/2}} \\ &= L_2(|\psi_0\rangle, \widetilde{\text{PATH}}_s) + \frac{4k\sqrt{q\delta\lambda + k\delta\lambda}}{2^{\lambda/2}}. \end{aligned}$$

Hence, $L_2(|\psi_k\rangle, \widetilde{\text{PATH}}_{s+1}) \leq L_2(|\psi_k\rangle, \widetilde{\text{BAD}}_{s,k}) \leq L_2(|\psi_0\rangle, \widetilde{\text{PATH}}_s) + \frac{4k\sqrt{q\delta\lambda + k\delta\lambda}}{2^{\lambda/2}}$ which implies that $L_2(|\psi'\rangle, \widetilde{\text{PATH}}_{s+1}) - L_2(|\psi'\rangle, \widetilde{\text{PATH}}_s) \leq \frac{4k\sqrt{q\delta\lambda + k\delta\lambda}}{2^{\lambda/2}}$. ◀

We now show that a quantum adversary that makes up to q rounds over $N - 1$ rounds can only find an \mathcal{H} -sequence of length N with negligible probability.

► **Lemma 19.** *Suppose that in each round $i \in [N - 1]$, the adversary \mathcal{A} makes a query to the parallel oracle CPhsO^{k_i} and that the total number of queries is bounded by q , i.e., $\sum_{i=1}^{N-1} k_i \leq q$. Then \mathcal{A} measures a database in PATH_N with probability at most $\frac{32q^3\delta\lambda}{2^\lambda}$.*

Proof. Let $|\psi_0\rangle$ be the initial state and let U_r represent a unitary transform applied by \mathcal{A} in between batches of queries to the quantum oracle. Then we define $|\psi_r\rangle = U_r \circ \text{CPhsO}^{k_r} |\psi_{r-1}\rangle$ for each round $r \in [N - 1]$. Thus, the attacker \mathcal{A} yields a sequence of states $|\psi_0\rangle, \dots, |\psi_{N-1}\rangle$. We remark that U_r may only operate on the state $|x, y, z\rangle$ and cannot impact the compressed oracle \mathcal{D} , e.g., $U_r(|x', y', z'\rangle \otimes |\mathcal{D}\rangle) = \sum_{x,y,z} \alpha_{x,y,z} |x, y, z\rangle \otimes |\mathcal{D}\rangle$. Thus,

$$L_2(U_r \circ \text{CPhsO}^{k_r} |\psi_{r-1}\rangle, \widetilde{\text{PATH}}_{r+1}) = L_2(\text{CPhsO}^{k_r} |\psi_{r-1}\rangle, \widetilde{\text{PATH}}_{r+1}),$$

so we can effectively ignore the intermediate unitary transform U_r in our analysis below. Now we can apply the previous lemma to conclude that

$$L_2(|\psi_i\rangle, \widetilde{\text{PATH}}_{i+1}) \leq \frac{4k_i\sqrt{2q\delta\lambda}}{2^{\lambda/2}} + L_2(|\psi_{i-1}\rangle, \widetilde{\text{PATH}}_i).$$

By the triangle inequality we have

$$L_2(|\psi_{N-1}\rangle, \widetilde{\text{PATH}}_N) \leq \sum_{i=0}^{N-1} \frac{4k_i\sqrt{2q\delta\lambda}}{2^{\lambda/2}} \leq \frac{\sqrt{32q^3\delta\lambda}}{2^{\lambda/2}}.$$

Hence, \mathcal{A} measures a database in PATH_N with probability at most $\left(\frac{\sqrt{32q^3\delta\lambda}}{2^{\lambda/2}}\right)^2 = \frac{32q^3\delta\lambda}{2^\lambda}$. ◀

Thus we have shown that a quantum adversary that makes $N - 1$ rounds of parallel queries should generally not find an \mathcal{H} -sequence of length N within their queries. Then we bound the probability that the quantum adversary outputs an \mathcal{H} -sequence of length N by a standard approach, e.g. [23, 42] of additionally the probability that the quantum adversary makes a “lucky guess”.

► **Theorem 3.** Let $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ be a random hash function and let $\delta \geq 1$ be a parameter. Let p be the probability that a quantum adversary making at most q queries over $N - 1$ rounds outputs $(x_0, y_0), \dots, (x_{N-1}, y_{N-1})$ and x_N s.t. $|x_i| \leq \delta\lambda$, $y_i = \mathcal{H}(x_i)$ and $\text{Substring}(y_{i-1}, x_i) = 1$ for each i , i.e., x_0, \dots, x_N is an \mathcal{H} -sequence. Then

$$p \leq \frac{64q^3\delta\lambda}{2^\lambda} + \frac{2N}{2^\lambda}.$$

Proof. By Lemma 5 the probability p is upper bounded by $2p' + 2N2^{-\lambda}$ where p' denotes the probability that an attacker measures $\mathcal{D} \in \text{PATH}_N$. By Lemma 19 we have $p' \leq \frac{32q^3\delta\lambda}{2^\lambda}$ and the result follows immediately. ◀

5 Security of PoSW in the pqROM

In this section, we show the security of a construction for proofs of sequential work in the *parallel* quantum random oracle model (pqROM). Here, we focus on the non-interactive version of PoSW, which can be obtained by applying a Fiat-Shamir transform to the PoSW defined in [23]. Note that the PoSW defined in both [37] and [23] are interactive, in which a statement χ is randomly sampled from the verifier and the prover constructs a proof based on the input statement χ .

5.1 The Definition of Non-Interactive PoSW

We first formally define the non-interactive PoSW in the random oracle model.

► **Definition 20** (Non-Interactive PoSW). A Non-Interactive Proof of Sequential Work (niPoSW) consists of polynomial-time oracle algorithms $\Pi_{\text{niPoSW}} = (\text{Solve}, \text{Verify})$ that use public parameters, as defined below.

- **Public Parameters.** Security parameter $\lambda \in \mathbb{N}$ and a random oracle $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$.
- **Solve.** Given a time parameter $T \in \mathbb{N}$, the statement χ , \mathcal{P} computes a solution $\pi \leftarrow \text{Solve}^{\mathcal{H}(\cdot)}(1^\lambda, T, \chi)$. The final proof is (χ, π) .
- **Verify.** \mathcal{P} can verify that the proof is genuine by running $\{0, 1\} \leftarrow \text{Verify}^{\mathcal{H}(\cdot)}(1^\lambda, T, \chi, \pi)$.

We require the following properties:

- (1) **Correctness.** For any $\chi \in \{0, 1\}^\lambda$, $\lambda, T \in \mathbb{N}$ we have

$$\text{Verify}^{\mathcal{H}(\cdot)}(1^\lambda, T, \chi, \text{Solve}^{\mathcal{H}(\cdot)}(1^\lambda, T, \chi)) = 1.$$

That is, an honest prover should always produce a valid proof with probability 1, regardless of the choice of the statement $\chi \in \{0, 1\}^*$, running in time parameter T and security parameter λ .

- (2) **Efficiency.** Solve should run in time $T \cdot \text{poly}(\lambda)$ and Verify should run in time $\text{poly}(\lambda, \log T)$. Similarly, the solution π must have size $\text{poly}(\lambda, \log T)$.
- (3) **Security.** We say that a construction Π_{niPoSW} is $(T(\cdot), q(\cdot), \epsilon(\cdot))$ -secure (resp. Π_{niPoSW} is $(T(\cdot), q(\cdot), \epsilon(\cdot))$ -quantum secure) if any algorithm \mathcal{A} running in sequential time at most $T = T(\lambda)$ in the pROM (resp. pqROM) and making at most $q = q(\lambda)$ total queries to the random oracle should fail to produce a valid proof for any statement $\chi \in \{0, 1\}^\lambda$ (selected by the adversary) except with a negligible probability $\epsilon(\lambda)$, i.e., if $(\pi', \chi) \leftarrow \mathcal{A}^{\mathcal{H}(\cdot)}(1^\lambda, T)$ denotes the solution generated from \mathcal{A} , then

$$\Pr_{\mathcal{H}(\cdot)} \left[\text{Verify}^{\mathcal{H}(\cdot)}(1^\lambda, T, \chi, \pi') = 1 \right] \leq \epsilon(\lambda),$$

where the probability is taken over the randomness of the random oracle \mathcal{H} .

5.2 The Underlying DAG G_n^{PoSW} ([23])

We will use the same DAG in our construction of the non-interactive PoSW as the DAG defined in [23]. Here, we briefly recall their construction of the DAG G_n^{PoSW} (the figure is given in the full version). For $n \in \mathbb{N}$, let $N = 2^{n+1} - 1$ and first construct a complete binary tree $B_n = (V, E')$ of depth n , where $|V| = N$ and all the directed edges go from the leaves towards the root. We identify the N nodes $V = \{0, 1\}^{\leq n}$ with the binary strings of length at most n except for the root, and we let the root be the empty string ε . Below the root we add directed edges from nodes 0 and 1 to node ε . Similarly, for each node v we add directed edges from nodes $(v\|0)$ and $(v\|1)$ to v . Here, $\|$ denotes the concatenation of the strings. Now we define the DAG $G_n^{\text{PoSW}} = (V, E)$ by starting with $B_n = (V, E')$ and appending some edges as follows:

- For all leaf nodes $u \in \{0, 1\}^n$, add an edge (v, u) for any v that is a left sibling of a node on the path from u to the root ε .

For example, for $u = 0110$, the path from u to the root is $0110 \rightarrow 011 \rightarrow 01 \rightarrow 0 \rightarrow \varepsilon$ and the left siblings of the nodes on this path are 010 and 00. Hence, we add the edges $(010, 0110)$ and $(00, 0110)$ to E' . We refer to [23] for the full description of G_n^{PoSW} .

5.3 The Non-Interactive Version of [23] Construction

Applying the Fiat-Shamir transform to the interactive PoSW construction [23], we have the following algorithms **Solve** and **Verify** in the non-interactive PoSW construction. For the notational simplicity, let $G_n = G_n^{\text{PoSW}}$ be the underlying DAG from [23].

Solve $^{\mathcal{H}(\cdot)}(1^\lambda, T, \chi, C)$:

- Compute the labels of the graph G_n with $n = 1 + \lceil \log T \rceil$, i.e., compute $\ell_i = \mathcal{H}_\chi(i, \ell_{p_1^i}, \dots, \ell_{p_{d_i}^i})$, $1 \leq i \leq N$, where $p_1^i, \dots, p_{d_i}^i$ denotes the parents⁴ of node i and $\mathcal{H}_\chi(\cdot) := \mathcal{H}(\chi, \cdot)$.
- Compute $R = \mathcal{H}_\chi(N + 1, \ell_\varepsilon)$ and parse R to get $k = \lfloor \lambda/n \rfloor$ strings $c_1, \dots, c_k \in \{0, 1\}^n$. Compute the challenges $C = \{c_1, \dots, c_k\}$ where each n -bit string c_i corresponds to a leaf node in G_n .
- Prove that everything on the path from node c_i to the root is locally consistent. This can be done by a Merkle tree reveal **MT.Reveal**, which reveals the labels of all the siblings on path from node c_i to the root. More precisely, for a node $y \in \{0, 1\}^{\leq n}$, we define

$$\text{MT.Reveal}(y) = \{\ell_{y[0\dots j-1]\|(y[j]\oplus 1)}\}_{j=1}^k,$$

where we recall that $y[0\dots j]$ denotes the substring of y to the j^{th} bit and $y[0\dots 0]$ denotes the empty string. In this way, we can reveal the labels of all the siblings on path from x to the root ε . In particular, a solution π consists of the following:

$$\pi = \{\ell_\varepsilon, c_i, \text{MT.Reveal}(c_i) \text{ for } 1 \leq i \leq k\}.$$

- Output (χ, π) .

Verify $^{\mathcal{H}(\cdot)}(1^\lambda, T, \chi, \pi)$:

- Parse π to extract ℓ'_ε and c'_1, \dots, c'_k . Set $R' = \mathcal{H}_\chi(N + 1, \ell'_\varepsilon)$ and split R' to obtain $k = \lfloor \lambda/n \rfloor$ challenges c''_1, \dots, c''_k each of length n . Output 0 if $c''_i \neq c'_i$ for any $i \leq k$. Otherwise, let $(p_1^{i'}, \dots, p_{d_i}^{i'}) = \text{parents}(c'_i)$ for each i .

⁴ Given a DAG G and a directed edge (u, v) we say that node u is a parent of node v . While this is the standard notion of parent in a DAG it can be counter-intuitive since the “tree” edges in our DAG are directed towards the root i.e., nodes 011 and 010 are both parents of node 01.

- Extract $\ell'_{c'_i}$ and $\ell'_{p_j^{i'}}$ from π for each $i \leq k$ and $j \leq d'_i$.
- Check that each leaf node c'_i is locally consistent. That is, one checks that $\ell'_{c'_i}$ is correctly computed from its parent labels:

$$\ell'_{c'_i} \stackrel{?}{=} \mathcal{H}_\chi(c'_i, \ell'_{p_1^{i'}}, \dots, \ell'_{p_{d'_i}^{i'}}) \text{ where } (p_1^{i'}, \dots, p_{d'_i}^{i'}) = \text{parents}(c'_i).$$

(Note that in G_n all of c'_i 's parents are siblings of nodes on the path from c'_i to the root ϵ).

- Check the Merkle tree openings for consistency. That is, for $i = n - 1, n - 2, \dots, 0$, check that

$$\ell_{c'_i[0\dots i]} := \mathcal{H}_\chi(c'_i[0\dots i], \ell_{c'_i[0\dots i]||0}, \ell_{c'_i[0\dots i]||1}),$$

and verify that the computed root $\ell_{c'_i[0\dots 0]}$ is equal to $\ell_\epsilon \in \pi$ that we received in the proof.

5.4 Security

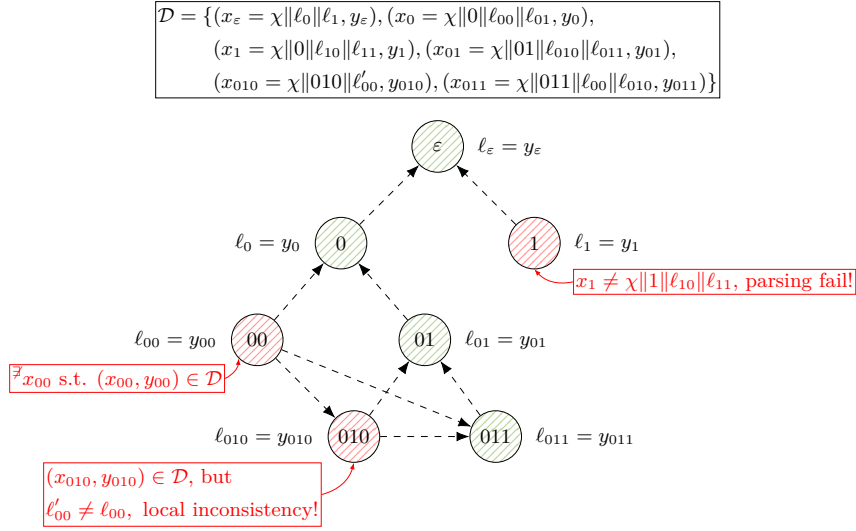
We argue that for any fixed constant $\alpha > 0$ the non-interactive proof of sequential work outline above is $(T = (1 - \alpha)N, q, \epsilon)$ -secure for $\epsilon(\lambda) = 32q^2(1 - \alpha)^{\lfloor \lambda/n \rfloor} + 2q^32^{-\lambda} + 64q^3(n + 2)\lambda 2^{-\lambda} + 2\lfloor \lambda/n \rfloor(n + 2)2^{-\lambda}$. We first define a set LUCKY_s of databases \mathcal{D} in which \mathcal{D} contains no collision or \mathcal{H} -sequence of length s , yet the dataset \mathcal{D} contains a lucky Merkle tree that can be used to extract a proof of sequential work for some statement χ . We then argue that *any* attacker making q queries fails to measure such a lucky dataset \mathcal{D} except with negligible probability. This is true even if the attacker is not restricted to run in sequential time s . Finally, to complete the argument we argue that any cheating attacker who produces a valid proof of sequential work can be converted into an algorithm that measures a dataset \mathcal{D} that either (1) contains an \mathcal{H} -sequence of length s , (2) contains a collision or (3) is in LUCKY_s . Assuming that our attacker is sequentially bounded and makes at most q queries, the probability of any of these three outcomes must be negligible. Thus, the PoSW construction must be secure against any sequentially bounded attacker.

Coloring the Graph G_{PoSW}^n

Given a database \mathcal{D} , a statement χ , and a candidate PoSW solution $y = \ell_\epsilon$ we define an algorithm $\text{ColoredMT}_{\mathcal{D}}(\chi, y)$ which returns a copy of the DAG G_n^{PoSW} in which each node is colored **red** or **green**. Intuitively, green nodes indicate that the corresponding labels are locally consistent with the database \mathcal{D} while red nodes are locally inconsistent. If the PoSW solution ℓ_ϵ is entirely consistent with \mathcal{D} then every node in G_n^{PoSW} will be colored green. On the other hand, if there is no entry of the form $(x, y) \in \mathcal{D}$ then the root node in G_n^{PoSW} would be colored red along with every other node below it. To define $\text{ColoredMT}_{\mathcal{D}}(\chi, y)$ we use a recursive helper function $\text{ColorSubTree}_{\mathcal{D}}$ which outputs a colored subgraph rooted at an intermediate node v . We briefly introduce how these algorithms work below and refer to the full version for the complete descriptions.

- (1) The algorithm $\text{ColorSubTree}_{\mathcal{D}}(\chi, v, x_v, y_v)$ generates a subset of nodes that consists of a Merkle subtree along with the coloring of each node in the set.

⁵ Note that if we have another entry $(x'_1, y_1) \in \mathcal{D}$ satisfying $x'_1 = \chi \| 1 \| \ell_{10} \| \ell_{11}$ then we have a collision in the database and we do not have a parsing fail here. Similar argument holds for another parsing fail case in (3) as well. We will deal with the case that the database has collisions separately and assume that we do not have any collisions so that a unique Merkle subtree is generated as output.



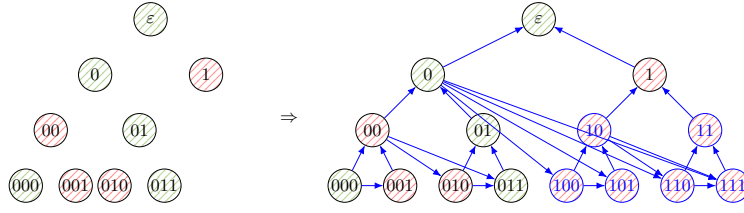
■ **Figure 1** A succinct illustration of $\text{ColorSubTree}_{\mathcal{D}}(\chi, \varepsilon, x_\varepsilon, y_\varepsilon)$ where ε is an empty string and the database \mathcal{D} is defined as above. Note that since $n = 3$, the nodes on the lowest layer are leaf nodes and the dashed edges are shown to help understand how the coloring works (we do not actually draw the edges in the algorithm). As described above, we have the following cases to color the node to red: (1) in node 1, there exists x_1 such that $(x_1, y_1) \in \mathcal{D}$, however, x_1 cannot be parsed properly, i.e., $x_1 \neq \chi \parallel 1 \parallel \ell_{10} \parallel \ell_{11}$ (parsing fail⁵), (2) in node 00, there is no x_{00} such that $(x_{00}, y_{00}) \in \mathcal{D}$ (undefined entry in \mathcal{D}), and (3) in node 010 – which is a leaf node – we have an entry $(x_{010}, y_{010}) \in \mathcal{D}$, but when parsing x_{010} into $\chi \parallel 010 \parallel \ell'_{00}$, we observe that the predefined label ℓ_{00} of node 00 and the value ℓ'_{00} does not match (local inconsistency).

- The algorithm takes as input (χ, v, x_v, y_v) where $\chi \in \{0, 1\}^*$ is a statement, $v \in \{0, 1\}^{\leq n}$ denotes a node in $G_{\mathcal{D}}$ ⁶, and $x_v \in \{0, 1\}^*$, $y_v \in \{0, 1\}^\lambda$ are the bitstrings. Here, y_v is a potential candidate to be the label of node v . It outputs a subset of nodes $V' \subseteq V(G_{\mathcal{D}})$, which consists of a Merkle subtree with root node v and the corresponding coloring set $\text{Color}(V') := \{\text{Color}(v) : v \in V'\}$.
- Recall that a node v is *green* if it is locally consistent; for example, let $(x, y) \in \mathcal{D}$ and for node v with label $\ell_v = y$, if v_0, v_1 with labels y_0, y_1 are the parents of v then v is locally consistent if and only if it satisfies $\mathcal{H}_\chi(v, y_0, y_1) = y$. Since we satisfies $\mathcal{H}(x) = y$ as $(x, y) \in \mathcal{D}$, one would need to satisfy $x = \chi \parallel v \parallel y_0 \parallel y_1$ for v to be locally consistent.
- Hence, we start parsing x_v into $\chi \parallel v' \parallel y_{v \parallel 0} \parallel y_{v \parallel 1}$ and see if it succeeds. That is, check if $v' = v$. If it fails, then we say it as “parsing fail”, which is illustrated in Figure 1 (node 1). In this case, we color the node red and stop generating the subtree.
- If we succeed to parse x_v , then we color v to green and can proceed to its parents and see if there is an entry in the database \mathcal{D} with having its y -coordinate as the label of its parent node. If it fails, then it becomes our second fail and we color the node red and stop generating the subtree. It is illustrated in Figure 1 (node 00).
- When we color the leaf nodes, we follow the same procedure except that we could have more than two parents based on the edge structure in [23], and we have another

⁶ We remark that v corresponds to the identification of a node that is a binary string of length at most n , different from the label of the node ℓ_v .

possibility of “parsing fail” because the labels of its parents should be predefined by construction. If this kind of parsing fail occurs (node 010 in Figure 1), then we color the node to red.

- (2) $\text{ColoredMT}_{\mathcal{D}}(\chi, y)$ generates a complete Merkle tree rooted at a node ε with label $\ell_\varepsilon = y$ and appends the edges as shown in [23]. The algorithm works simple; find an entry $(x, y) \in \mathcal{D}$ and call $\text{ColorSubTree}_{\mathcal{D}}(\chi, \varepsilon, x, y)$. Fill the missing nodes with label \perp and color them all red. Then we add the edges as described in Section 5.2. If there is no such (x, y) in \mathcal{D} then we abort the entire algorithm. We refer to Figure 2 for an example of running the algorithm.



■ **Figure 2** One example of $\text{ColoredMT}_{\mathcal{D}}(\chi, y)$ where $n = 3$ and $(x, y) \in \mathcal{D}$. On the left side is the output of $\text{ColorSubTree}_{\mathcal{D}}(\chi, \varepsilon, x, y)$ (different from Figure 1) and we fill the undefined nodes colored red and add the edges on the right side. Note that newly added nodes and edges are shown in blue.

Notations

Recall that in Definition 10, we defined PATH_s to be the set of the databases (compressed random oracles) \mathcal{D} such that $G_{\mathcal{D}}$ contains a path of length s , which corresponds to the \mathcal{H} -sequence of length s . Now we define the set COLLIDE to be the set of the databases that contains a collision:

$$\text{COLLIDE} := \{\mathcal{D} : \mathcal{D} \text{ contains pairs } (x, y), (x', y) \text{ such that } x \neq x'\}.$$

Given a node (string) $v = (v_1 \| \dots \| v_n) \in \{0, 1\}^n$, we use $v_{\leq i} \in \{0, 1\}^i$ to denote the substring $v_1 \| \dots \| v_i$, $v_{\leq 0} := \varepsilon$, and we use $\text{PTR}(v, \chi) = \{v_{\leq i} : 0 \leq i \leq n\}$ to denote the set of all nodes on the direct path from v to the root of a Merkle tree constructed from $\text{ColoredMT}_{\mathcal{D}}(\chi, \cdot)$. Given a coloring of the Merkle tree, we define the predicate $\text{gPTR}(v, \chi)$, which verifies that every node on the path from v to the root is green.⁷ That is, $\text{gPTR}(v, \chi) = 1$ if and only if $\text{Color}(v') = \text{green}$ for all $v' \in \text{PTR}(v, \chi)$ and 0 otherwise. For example, in Figure 2, we have $\text{gPTR}(011, \chi) = 1$ because the color of nodes in $\text{PTR}(011, \chi) = \{011, 01, 0, \varepsilon\}$ are all green. On the other hand, we observe that $\text{gPTR}(000, \chi) = 0$ despite the node 000 is green as we have an intermediate red node 00 in $\text{PTR}(000, \chi)$.

Now we define $\text{LUCKY}(\mathcal{D}, \chi, y)$ to be the set of λ -bit strings that produce lucky challenges for the Merkle tree rooted at y :

$$\begin{aligned} \text{LUCKY}(\mathcal{D}, \chi, y) := \{w \in \{0, 1\}^\lambda : w = w_1 \| \dots \| w_k \| z \text{ where } k = \lfloor \lambda/n \rfloor, \\ |w_i| = n, \text{ and } \text{gPTR}(w_i, \chi) = 1 \forall 0 \leq i \leq k\}. \end{aligned}$$

⁷ Here, PTR stands for “Path To the Root” and gPTR stands for “green Path To the Root”.

22:22 On the Security of Proofs of Sequential Work in a Post-Quantum World

Then the set LUCKY_s is defined to be the set of databases that contains lucky challenges not in COLLIDE and PATH_s , i.e.,

$$\text{LUCKY}_s := \{\mathcal{D} : \exists(x, y) \in \mathcal{D} \text{ s.t. } x = \chi \| N + 1 \|_{\ell_\epsilon} \wedge y \in \text{LUCKY}(\mathcal{D}, \chi, \ell_\epsilon)\} \\ \setminus (\text{COLLIDE} \cup \text{PATH}_s).$$

We also define $\widetilde{\text{LUCKY}}_s$ to be the set of basis states with \mathcal{D} in LUCKY_s as follows:

$$\widetilde{\text{LUCKY}}_s := \{|x, y, z\rangle \otimes |\mathcal{D}\rangle : \mathcal{D} \in \text{LUCKY}_s\}.$$

We remark that when defining the set LUCKY_s , we need to assume that $\mathcal{D} \notin \text{COLLIDE}$ to ensure that we get a unique Merkle tree rooted at ℓ_ϵ and we additionally need to assume that $\mathcal{D} \notin \text{PATH}_s$ otherwise the set $\{v \in \{0, 1\}^n : \text{gPTR}(v, \chi) = 0\}$ may not be large, i.e., if all nodes are green.

We also define $\text{PRE}(\mathcal{D})$ to be the set of λ -bit strings w that become a preimage of a hash value. It happens when w is a substring of x where $(x, y) \in \mathcal{D}$.

$$\text{PRE}(\mathcal{D}) := \{w \in \{0, 1\}^\lambda : \exists(x, y) \in \mathcal{D} \text{ s.t. } \text{Substring}(w, x) = 1\}.$$

Security of PoSW against Quantum Attackers

Let $G_n = (V, E)$, $\text{Color}(V) \leftarrow \text{ColoredMT}_{\mathcal{D}}(\chi, \cdot)$. We first introduce a helper lemma that is a classical argument (not quantum) and comes immediately from rephrasing the intermediate claim in the proof of [23, Theorem 1].

► **Lemma 21** ([23]). *If $\mathcal{D} \notin \text{COLLIDE}$ and $\mathcal{D} \notin \text{PATH}_T$ with $T = (1 - \alpha)N$ for some constant $0 < \alpha < 1$, then*

$$|\{v \in \{0, 1\}^n : \text{gPTR}(v, \chi) = 0\}| \geq \alpha 2^n,$$

i.e., at least $\alpha 2^n$ out of 2^n challenges (leaf nodes) in G_n must fail to respond correctly.

We immediately have the following corollary which bounds the number of tuples (v_1, \dots, v_k, y) such that all challenges v_i are lucky i.e., $\text{gPTR}(v_i, \chi) = 1 \forall i \leq k$. Here, $y \in \{0, 1\}^{k'}$ is an auxiliary string.

► **Corollary 22.** *If v_1, \dots, v_k are the leaf nodes in G_n (i.e., $v_i \in V, |v_i| = n \forall i \leq k$), then we have that*

$$|\{(v_1, \dots, v_k, y) : \text{gPTR}(v_i, \chi) = 1 \forall i \leq k, y \in \{0, 1\}^{k'}\}| \leq 2^{nk+k'}(1 - \alpha)^k.$$

► **Lemma 23.** *Let α be any constant satisfying $q \leq \frac{2^\lambda(1-\alpha)^{\lfloor \lambda/n \rfloor}}{(n+1)^\lambda}$ then for any state*

$$|\phi\rangle = \sum_{x, y, z, \mathcal{D}: |\mathcal{D}| \leq q} \alpha_{x, y, z, \mathcal{D}} |x, y, z\rangle \otimes |\mathcal{D}\rangle$$

whose database register is a superposition of databases with at most q entries, we have

$$L_2(\text{CPhsO}|\phi\rangle, \widetilde{\text{LUCKY}}_s) - L_2(|\phi\rangle, \widetilde{\text{LUCKY}}_s) \leq 4(1 - \alpha)^{\frac{\lfloor \lambda/n \rfloor}{2}}.$$

Proof. (Sketch) The proof of Lemma 23 is similar to Lemma 12. We provide a complete proof in the full version and sketch the high level details here. We consider the projection of $|\phi'\rangle = \text{CPhsO}|\phi\rangle$ onto orthogonal spaces P, Q, R, S where (1) P projects onto the span of basis states $|x, y, z\rangle \otimes |\mathcal{D}\rangle \in \widetilde{\text{LUCKY}}_s$, (2) Q projects onto states $|x, y, z\rangle \otimes |\mathcal{D}\rangle$ such that $|x, y, z\rangle \otimes |\mathcal{D}\rangle \notin \widetilde{\text{LUCKY}}_s$, $y \neq 0$, and $\mathcal{D}(x) = \perp$, (3) R projects onto states $|x, y, z\rangle \otimes |\mathcal{D}\rangle$ such that $|x, y, z\rangle \otimes |\mathcal{D}\rangle \notin \widetilde{\text{LUCKY}}_s$, $y \neq 0$, and $\mathcal{D}(x) \neq \perp$, and (4) S projects onto states $|x, y, z\rangle \otimes |\mathcal{D}\rangle$ such that $|x, y, z\rangle \otimes |\mathcal{D}\rangle \notin \widetilde{\text{LUCKY}}_s$ and $y = 0$. Then since P, Q, R, S project onto disjoint states that span the entirety of $|\phi'\rangle$ then we have $P + Q + R + S = \mathbb{I}$, where \mathbb{I} denotes the identity operator. We analyze how P acts on these components separately and have that $\|P \circ \text{CPhsO}(P|\phi)\|_2 \leq \|P|\phi\rangle\|_2$, $\|P \circ \text{CPhsO}(Q|\phi)\|_2^2 \leq (1-\alpha)^{\lfloor \lambda/n \rfloor}$, $\|P \circ \text{CPhsO}(R|\phi)\|_2^2 \leq 9(1-\alpha)^{\lfloor \lambda/n \rfloor}$, and $\|P \circ \text{CPhsO}(S|\phi)\|_2 = 0$. Then by triangle inequality, we have that

$$\begin{aligned} \|P \circ \text{CPhsO}|\phi\rangle\|_2 &\leq \|P \circ \text{CPhsO}(P|\phi)\|_2 + \|P \circ \text{CPhsO}(Q|\phi)\|_2 \\ &\quad + \|P \circ \text{CPhsO}(R|\phi)\|_2 + \|P \circ \text{CPhsO}(S|\phi)\|_2 \\ &\leq \|P|\phi\rangle\|_2 + (1-\alpha)^{\frac{\lfloor \lambda/n \rfloor}{2}} + 3(1-\alpha)^{\frac{\lfloor \lambda/n \rfloor}{2}} \\ &\leq L_2(|\phi\rangle, \widetilde{\text{LUCKY}}_s) + 4(1-\alpha)^{\frac{\lfloor \lambda/n \rfloor}{2}}. \end{aligned}$$

Since we have that $\|P \circ \text{CPhsO}|\phi\rangle\|_2 = L_2(\text{CPhsO}|\phi\rangle, \widetilde{\text{LUCKY}}_s)$, we complete the proof. \blacktriangleleft

From Lemma 23, we have the following Lemma. The proof of Lemma 24 can be found in the full version.

► **Lemma 24.** *Suppose that our quantum attacker \mathcal{A} makes at most q queries to CPhsO then the probability p' of measuring a database $\mathcal{D} \in \text{LUCKY}_s$ for $s = N(1-\alpha)$ is at most $16q^2(1-\alpha)^{\lfloor \lambda/n \rfloor}$.*

► **Theorem 4.** *Suppose \mathcal{A} makes at most q quantum queries to our random oracle \mathcal{H} over at most $s = N(1-\alpha)$ rounds and let p denote the probability that \mathcal{A} outputs a valid (non-interactive) proof of sequential work. Then*

$$p \leq 32q^2(1-\alpha)^{\lfloor \lambda/n \rfloor} + \frac{2q^3}{2^\lambda} + \frac{64q^3(n+2)\lambda}{2^\lambda} + \frac{2\lfloor \lambda/n \rfloor(n+2)}{2^\lambda}.$$

Proof. Suppose that \mathcal{A} make queries to a random oracle \mathcal{H} and outputs tuples $((x_1, y_1), \dots, (x_k, y_k), z)$ and let R be a collection of such tuples that contain a valid PoSW for some statement $\chi \in \{0, 1\}^\lambda$. Recall that with probability p , the algorithm \mathcal{A} outputs a tuple such that (1) the tuple is in R (contains a valid PoSW), and (2) $\mathcal{H}(x_i) = y_i$ for all i . Now consider running \mathcal{A} with the oracle CPhsO (applying the Hadamard Transform before/after each query) and measuring the database \mathcal{D} after \mathcal{A} outputs. We first observe that if the final tuple is in R and $\mathcal{D}(x_i) = y_i$ for all i , then we must have $\mathcal{D} \in \text{LUCKY}_{s+1} \cup \text{PATH}_{s+1} \cup \text{COLLIDE}$. In particular, if \mathcal{D} does not contain an \mathcal{H} -sequence of length $s+1$ or a collision, then we must have $\mathcal{D} \in \text{LUCKY}_{s+1}$ since the proof of sequential work is valid.

However, the probability of measuring a dataset $\mathcal{D} \in \text{LUCKY}_{s+1} \cup \text{PATH}_{s+1} \cup \text{COLLIDE}$ can be upper bounded by $16q^2(1-\alpha)^{\lfloor \lambda/n \rfloor} + 32q^3(n+2)\lambda 2^{-\lambda} + q^3 2^{-\lambda}$ by applying Lemma 24 (Lucky Merkle Tree), Lemma 19 (Long \mathcal{H} -sequence), and Lemma 6 to upper bound the probability that $\mathcal{D} \in \text{LUCKY}_{s+1}$, $\mathcal{D} \in \text{PATH}_{s+1}$ and $\mathcal{D} \in \text{COLLIDE}$, respectively.

We also observe that the number of input/output pairs in our PoSW is $k = \lfloor \lambda/n \rfloor(n+2)$ since we have $\lfloor \lambda/n \rfloor$ challenges where each challenge consists of a statement χ , a node itself, and at most n parents. Hence, by applying Lemma 5, we have that

$$\sqrt{p} \leq \sqrt{16q^2(1-\alpha)^{\lfloor \lambda/n \rfloor} + \frac{32q^3(n+2)\lambda}{2^\lambda} + \frac{q^3}{2^\lambda} + \frac{\lfloor \lambda/n \rfloor(n+2)}{2^\lambda}},$$

which implies that

$$p \leq 32q^2(1 - \alpha)^{\lfloor \lambda/n \rfloor} + \frac{2q^3}{2^\lambda} + \frac{64q^3(n+2)\lambda}{2^\lambda} + \frac{2\lfloor \lambda/n \rfloor(n+2)}{2^\lambda},$$

since $\sqrt{a} \leq \sqrt{b} + \sqrt{c}$ implies $a \leq b + c + 2\sqrt{bc} \leq 2(b + c)$ for any $a, b, c > 0$. ◀

6 Conclusion

We have shown that any attacker in the parallel quantum random oracle model making $q \ll 2^{\lambda/3}$ total queries cannot find an \mathcal{H} -sequence of length N in $N - 1$ sequential rounds except with negligible probability. Using this result as a building block, we then prove that the non-interactive proof of sequential work of Cohen and Pietrzak [23] is secure against any attacker making $q \ll 2^{\lambda/n}$ queries and running in sequential time $T = (1 - \alpha)N$. We leave it as an open question whether or not the λ/n term from this lower bound is inherent or whether the construction could be tweaked to establish security when $q \ll 2^{\lambda/n}$. The main technical hurdle is extracting more than λ/n challenges from a single random oracle output or adapting the proof to handle a modified construction where we extract challenges from multiple random oracle outputs. An alternative approach would be to introduce a second random oracle with longer outputs, which could be used to extract $\Omega(\lambda)$ challenges.

Our results also highlight the power of the recent compressed random oracle technique of Zhandry [42] and raises a natural question about whether or not these techniques could be extended to establish the security of important cryptographic primitives such as memory-hard functions or proofs of space in the quantum random oracle model. Alwen and Serbinenko [8] previously gave a classical pebbling reduction in the classical parallel random oracle model showing that the cumulative memory complexity of a data-independent memory hard function is tightly characterized by the pebbling complexity of the underlying graph. Would it be possible to establish the post-quantum security of memory hard functions through a similar reduction in the parallel quantum random oracle model?

References

- 1 Scott Aaronson. Quantum copy-protection and quantum money. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC*, pages 229–242, 2009.
- 2 Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. *Theory of Computing*, 9:349–401, 2013.
- 3 Hamza Abusalah, Chethan Kamath, Karen Klein, Krzysztof Pietrzak, and Michael Walter. Reversible proofs of sequential work. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part II*, pages 277–291, 2019.
- 4 Gorjan Alagic, Christian Majenz, Alexander Russell, and Fang Song. Quantum-access-secure message authentication via blind-unforgeability. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 788–817. Springer, Heidelberg, May 2020. PATHdoi:10.1007/978-3-030-45727-3_27.
- 5 Joël Alwen and Jeremiah Blocki. Efficiently computing data-independent memory-hard functions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 241–271. Springer, Heidelberg, August 2016. PATHdoi:10.1007/978-3-662-53008-5_9.
- 6 Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Depth-robust graphs and their cumulative memory complexity. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 3–32. Springer, Heidelberg, April / May 2017. PATHdoi:10.1007/978-3-319-56617-7_1.

- 7 Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Sustained space complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 99–130. Springer, Heidelberg, April / May 2018. PATHdoi:10.1007/978-3-319-78375-8_4.
- 8 Joël Alwen and Vladimir Serbinenko. High parallel complexity graphs and memory-hard functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 595–603. ACM Press, June 2015. PATHdoi:10.1145/2746539.2746622.
- 9 Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 269–295. Springer, Heidelberg, August 2019. PATHdoi:10.1007/978-3-030-26951-7_10.
- 10 Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh V. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997.
- 11 Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. Tighter proofs of CCA security in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 61–90. Springer, Heidelberg, December 2019. PATHdoi:10.1007/978-3-030-36033-7_3.
- 12 Jeremiah Blocki, Benjamin Harsha, Siteng Kang, Seunghoon Lee, Lu Xing, and Samson Zhou. Data-independent memory hard functions: New attacks and stronger constructions. In *Advances in Cryptology - CRYPTO - 39th Annual International Cryptology Conference, Proceedings, Part II*, pages 573–607, 2019.
- 13 Jeremiah Blocki, Ling Ren, and Samson Zhou. Bandwidth-hard functions: Reductions and lower bounds. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 1820–1836, 2018.
- 14 Jeremiah Blocki and Samson Zhou. On the depth-robustness and cumulative pebbling cost of argon2i. In *Theory of Cryptography - 15th International Conference, TCC Proceedings, Part I*, pages 445–465, 2017.
- 15 Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Proceedings, Part I*, pages 757–788, 2018.
- 16 Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Heidelberg, December 2011. PATHdoi:10.1007/978-3-642-25385-0_3.
- 17 Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference. Proceedings, Part II*, pages 361–379, 2013.
- 18 Gilles Brassard, Peter Hoyer, Kasseem Kalach, Marc Kaplan, Sophie Laplante, and Louis Salvail. Merkle puzzles in a quantum world. In *Advances in Cryptology - CRYPTO 2011. Proceedings*, pages 391–410, 2011.
- 19 Gilles Brassard and Louis Salvail. Quantum merkle puzzles. In *Second International Conference on Quantum, Nano, and Micro Technologies, ICQNM*, pages 76–79, 2008.
- 20 Ethan Cecchetti, Ben Fisch, Ian Miers, and Ari Juels. PIEs: Public incompressible encodings for decentralized storage. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 1351–1367. ACM Press, November 2019. PATHdoi:10.1145/3319535.3354231.
- 21 Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 1–29. Springer, Heidelberg, December 2019. PATHdoi:10.1007/978-3-030-36033-7_1.

- 22 Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. On the compressed-oracle technique, and post-quantum security of proofs of sequential work, 2020. *To appear at EUROCRYPT 2021*. PATHarXiv:2010.11658.
- 23 Bram Cohen and Krzysztof Pietrzak. Simple proofs of sequential work. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 451–467. Springer, Heidelberg, April / May 2018. PATHdoi:10.1007/978-3-319-78375-8_15.
- 24 Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 356–383. Springer, Heidelberg, August 2019. PATHdoi:10.1007/978-3-030-26951-7_13.
- 25 Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Online-extractability in the quantum random-oracle model, 2021. PATHarXiv:2103.03085.
- 26 Nico Döttling, Russell W. F. Lai, and Giulio Malavolta. Incremental proofs of sequential work. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part II*, pages 292–323, 2019.
- 27 Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 585–605. Springer, Heidelberg, August 2015. PATHdoi:10.1007/978-3-662-48000-7_29.
- 28 Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. PATHdoi:10.1007/3-540-47721-7_12.
- 29 Ben Fisch. Tight proofs of space and replication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 324–348. Springer, Heidelberg, May 2019. PATHdoi:10.1007/978-3-030-17656-3_12.
- 30 Lov K. Grover and J. Radhakrishnan. Quantum search for multiple items using parallel queries. *arXiv: Quantum Physics*, 2004.
- 31 Yassine Hamoudi and Frédéric Magniez. Quantum time-space tradeoff for finding multiple collision pairs, 2020. PATHarXiv:2002.08944.
- 32 Stacey Jeffery, Frederic Magniez, and Ronald de Wolf. Optimal parallel quantum query algorithms. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014*, pages 592–604, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
- 33 Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 552–586. Springer, Heidelberg, April / May 2018. PATHdoi:10.1007/978-3-319-78372-7_18.
- 34 Qipeng Liu and Mark Zhandry. On finding quantum multi-collisions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 189–218. Springer, Heidelberg, May 2019. PATHdoi:10.1007/978-3-030-17659-4_7.
- 35 Qipeng Liu and Mark Zhandry. Revisiting post-quantum Fiat-Shamir. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 326–355. Springer, Heidelberg, August 2019. PATHdoi:10.1007/978-3-030-26951-7_12.
- 36 Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Time-lock puzzles in the random oracle model. In *Advances in Cryptology - CRYPTO. Proceedings*, pages 39–50, 2011.
- 37 Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Publicly verifiable proofs of sequential work. In Robert D. Kleinberg, editor, *ITCS 2013*, pages 373–388. ACM, January 2013. PATHdoi:10.1145/2422436.2422479.
- 38 Krzysztof Pietrzak. Proofs of catalytic space. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 59:1–59:25. LIPIcs, January 2019. PATHdoi:10.4230/LIPIcs.ITCS.2019.59.
- 39 Dominique Unruh. Revocable quantum timed-release encryption. *J. ACM*, 62(6), December 2015. PATHdoi:10.1145/2817206.

- 40 Christof Zalka. Grover's quantum searching algorithm is optimal. *Phys. Rev. A*, 60:2746–2751, October 1999. PATHdoi:10.1103/PhysRevA.60.2746.
- 41 Mark Zhandry. How to construct quantum random functions. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 679–687, 2012.
- 42 Mark Zhandry. How to record quantum queries, and applications to quantum indistinguishability. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 239–268. Springer, Heidelberg, August 2019. PATHdoi:10.1007/978-3-030-26951-7_9.

Fooling an Unbounded Adversary with a Short Key, Repeatedly: The Honey Encryption Perspective

Xinze Li ✉

Tsinghua University, Beijing, China

Qiang Tang ✉

The University of Sydney, Australia

Zhenfeng Zhang ✉

Institute of Software, Chinese Academy of Sciences, Beijing, China

Abstract

This article is motivated by the classical results from Shannon that put the simple and elegant one-time pad away from practice: key length has to be as large as message length and the same key could not be used more than once. In particular, we consider encryption algorithm to be defined relative to specific message distributions in order to trade for unconditional security. Such a notion named honey encryption (HE) was originally proposed for achieving best possible security for password based encryption where secret key may have very small amount of entropy.

Exploring message distributions as in HE indeed helps circumvent the classical restrictions on secret keys. We give a new and very simple honey encryption scheme satisfying the unconditional semantic security (for the targeted message distribution) in the standard model (all previous constructions are in the random oracle model, even for message recovery security only). Our new construction can be paired with an extremely simple yet “tighter” analysis, while all previous analyses (even for message recovery security only) were fairly complicated and require stronger assumptions. We also show a concrete instantiation further enables the secret key to be used for encrypting multiple messages.

2012 ACM Subject Classification Security and privacy → Cryptography; Theory of computation → Cryptographic primitives

Keywords and phrases unconditional security, information theoretic encryption, honey encryption

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.23

Funding *Qiang Tang*: Supported by a Google Faculty Award, and a gift from Filecoin.

Zhenfeng Zhang: Supported by National Key R&D Program of China (2017YFB0802000)

Acknowledgements We thank anonymous reviewers for valuable comments, specifically the simplification of the key re-use analysis.

1 Introduction

The celebrated *one-time pad* is extremely simple and elegant, while satisfying perfect secrecy that can be against an even computationally unbounded attacker. It also has two well-known drawbacks that hinder its practical deployment: (1) the key length has to be as large as the message size as shown in Shannon’s classical work [23] that perfect secrecy must incur such a cost; and (2) one key can only be used to encrypt one message. These two main drawbacks of one time pad have motivated cryptographers to introduce the concept of computational security, and invented corresponding tools. In particular, pseudo-random generators were developed to stretch a short key to be longer for the stream cipher, and pseudo-random functions were introduced to design a symmetric key encryption that can use the same short key to encrypt multiple messages.



© Xinze Li, Qiang Tang, and Zhenfeng Zhang;
licensed under Creative Commons License CC-BY 4.0
2nd Conference on Information-Theoretic Cryptography (ITC 2021).
Editor: Stefano Tessaro; Article No. 23; pp. 23:1–23:21



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Despite that encryption schemes based on computational hardness are commonly used in practice (as one of the greatest achievements of modern (computational) cryptography in general), various application scenarios exist where information-theoretic models and arguments still play a dominant or defining role: (i) one wishes to maintain security despite the use of “weak” keys, such as passwords [16], or short keys in resource constraint devices such as IoT nodes – the information-theoretic setting is forced upon by the fact that brute-force attacks on the key space become feasible; (ii) one wishes to maintain long-term or even everlasting security [8] for encrypted storage of highly confidential contents, such as user secret keys, passwords, or genomics data – the information theoretic setting is demanded as the validity of computational assumptions highly depends on existing cryptanalysis techniques and computing powers and infrastructures, (e.g., the emerging threats of quantum computers).

In this article, we ask whether we can have an information theoretic encryption in the plain model (without random oracles or extra common random sources) whose secret key can be used as “conveniently” as that in the computational encryption, i.e., with length at security parameter, and can be used to encrypt arbitrary number of messages (polynomially bounded). We consider how to circumvent the well-known obstacles by *trading generality of the encryption for unconditional security*, in particular, via the lens of honey encryption [15, 14], which works on message distributions that are known to the encryption algorithm.

Fooling an unbounded adversary with a short key. There have been interesting progresses regarding how to achieve information theoretic encryption with a short key. To circumvent the inherent barrier that perfect secrecy requires each message bit to “burn up” a bit of secret key during encryption, a thread of research considers to give encryptor more information about the plaintext (but no more than what is known to the adversary). Or to put it another way, instead of encrypting arbitrary message, *the encryption algorithm only works for specific types of messages to trade for unconditional security*.

It began with Russell and Wang’s 2002 article on Entropic Security [21], and the follow-up work of Dodis and Smith [9]. They showed how to sidestep the obstacle of key length, obtaining (information-theoretic) semantic security guarantees even with small keys, by assuming that the *message comes from a high-entropy distribution*. On the positive side, entropic security circumvents the classical entropy bound on the secret key. This model, however, still requires the key to be long enough and satisfies $\mu > n - \ell$, where n is the message length, and μ, ℓ are the key entropy and message entropy respectively. Such an entropy requirement (though relaxed than one-time pad) still puts a restriction on the key length, which is often much larger than $O(\lambda)$. To achieve best possible security of (weak) password based encryption facing an offline brute-force attack, Juels and Ristenpart proposed an interesting concept of honey encryption [15] that further explores the *message distribution*. The encryption algorithm there is provided with the details of the message distribution. The key insight is putting attacker to face all plausible messages after brute-force decryption. However, they considered only a security notion that prevents the adversary to recover the whole message, which is arguably insufficient in practice.

For this reason, Jaeger, Ristenpart and Tang (JRT) [14] did an in-depth study of honey encryption: they first defined a semantic security like notion¹ that requires the ciphertext to hide all partial information, when message is sampled from the distribution (for example, password, bio-metric data or secret key); and they demonstrated that a simple hash based encryption achieves the security assuming the hash is a random oracle (or ideal cipher).

¹ It is called targeted distribution semantic security, as the encryption algorithm now is designed relative to a specific message distribution.

More importantly, the analysis were done in the information theoretic setting. A particularly interesting message [14] about honey encryption was that designing an encryption algorithm for a particular message distribution may help to approach our goals: it seems possible in this case to have semantic security like notion against even unbounded attackers, but only requiring a secret key that could be as short as that in the computational setting.

Honey encryption with “semantic security” in the standard model. Relying on random oracles to establish security is certainly unsatisfying, especially in the information theoretic setting, as a random oracle can pump out unlimited amount of entropy as an idealized assumption that could not be instantiated in real-life. An immediate natural question to consider is *whether we can construct honey encryption that satisfies a unconditional semantic security in the standard model*, which was left as an open problem in [15, 14]. Thus this becomes the first problem for us to tackle in this article.

Our first standard model HE construction is a natural instantiation of the DTE-then-Encrypt construction from [14, 15]: previously, encrypting algorithm is simply the hash based encryption using hash as a random function (or even an ideal cipher) to generate a session key for the message. Now we would like to instantiate the random oracle. Let us first walk through the high-level intuition of the complicated security analysis from [14] (similar for [15] even though it was only against a weaker message recovery adversary):

Phase (1) of the security analysis in JRT [14] and JR [15] was to transform from the security game defining semantic security (for a targeted message distribution) to a game in which the ciphertext is chosen uniformly, and the secret key is sampled after the adversary has run. In the new game, one can show that the advantage of any adversary is no larger than that of an “optimal” adversary who decrypts the ciphertext using all possible keys, computes the predicate value on the resulting plaintext, and outputs the bit which has the higher cumulative mass of keys that resulted in this bit. Such a cumulative probability can be viewed as the total weight of the balls in a bin at the end of a balls-and-bins game.

Phase (2) is the complicated part that was to analyze the maximum load in the non-uniform bin selection with non-uniform balls. A majorization lemma [4] was applied to upper-bound the quantity that obtained from *uniformly* weighted balls (with the same weights). The latter thus can be simply be derived by bounding the number of balls in the experiment, for which we can use Chernoff inequality.

Now without the random oracle, the balls-into-bins experiment is no longer valid (at least not in the normal sense), since “ball throws” become correlated with each other. Furthermore, the majorization technique also requires independence. To get around the major challenges, we observe that a direct analysis might be possible without using the majorization techniques, if we leverage a generalized Chernoff-bound [22] to deal with correlated (to some extent) random variables that even may not be identically distributed. Such technique may also be applied to simplify the analysis of the JRT result regarding semantic security [14] and also the JR result [15] for even message recovery security. Since a generalized Chernoff bound dealt with q -wise independent random variables, it becomes natural to instantiate the random oracle with a q -wise independent hash. Moreover, after a careful analysis, we can set the parameter with just a small q such as the security parameter.

Our second standard model HE construction starts from an entropically secure encryption (ES). This construction and analysis turn out to be surprisingly simple. There is a seeming dilemma: what we would like to have is an HE scheme with a key of length at most $O(\lambda)$, where λ is the security parameter, as in the computational setting; while entropic security requires the key entropy (length) no smaller than message length minus message entropy. For most of the message distributions (say with entropy half of the length), it already requires the key length to be depending on message length.

But a closer look reveals the power of one important building block for all existing HE constructions, distribution transforming encoder (DTE for short). A distribution transforming encoder takes a message distribution, and encodes it to an almost uniform distribution over another space S . If S is close to $\{0, 1\}^n$, the encoded distribution is already close to uniform. Applying an entropically secure encryption on “encoded message distribution” now offers opportunities to allow the key entropy to be minimal. Interestingly, it is very easy to see the security of the DTE-then-ES construction of HE, though previous concrete constructions of HE were all paired with very complicated analysis.²

It is also worth noting that we can consider the output of DTE to be a uniform distribution in our analysis, which only adds negligible error. Since the ES scheme is applied on this distribution, we can further lessen restrictions on the ES scheme: it only needs to work for uniform input (with full entropy). In this way, we can achieve an even better security bound than those previous ones, which was considered to be asymptotically optimal.

Fooling an unbounded adversary again with the same short key. It’s well-known that in one-time pad (and many related constructions), if the secret key is used to encrypt two messages, some pattern (e.g., whether two messages are equal) is leaked immediately. Such vulnerability was widely exploited in practice, and lead to numerous highly impacting attacks [5, 25]. With the encouragement of HE in circumventing the entropy bound in the information theoretic setting, we are now more ambitious and would like to ask *whether one can further encrypt multiple messages using the same short key*. This becomes our second major question to address in this article.

Insecurity of ES when reusing a key. Neither previous works on entropic security, nor honey encryption discussed the key reuse issue. To see whether entropic security is helpful enabling key reuse, we start with the security notion which generalizes the conventional semantic security definition. Conventionally, the adversary who sees a ciphertext, tries to learn a predicate on the corresponding message. Now, adversary seeing a vector of ciphertext, can infer a predicate bit on the vector of messages (which were sampled independently).

Having this definition in place, (informal) intuition that entropic security does not seem to be promising enabling the key reuse can be seen as follows: One can view the vector of messages as one large piece of message. Plugging in the entropy bound from [21, 9], suppose there are t messages, each with entropy $n/2$ and length n . The entropy bound would require length of the key to be no smaller than $t \cdot n - t \cdot n/2 = tn/2$. Even a secret key with length n could be at most used twice. Such intuition can be easily generalized to the case that message entropy to be $n - 1$ (in which the key could be reused at most μ times, where μ is the entropy of the key)! To formally prove the impossibility, we establish the lower bound on the key length if an ES scheme reuses the key for T times. This generalizes the original lower bound in the single ciphertext setting [21, 9]. An analysis on key length requirements is given in Sec. 4.1 to show that in order to reuse the key for T times, we must have the key length at least to be $(n - t)T$, where the messages are from distribution with entropy t .

² We remark that, as far as we know, such a simple construction has not been shown before. Previously, honey encryption was motivated by password based encryption, thus their main goal is to obtain security with a weak key. Thus comparison focused on the insufficiency of entropic security [15, 14] due to its key length requirement. When we consider honey encryption from the angle of unbounded security, entropic security becomes a natural candidate to leverage, which enables us to simplify the analysis for HE [15, 14] dramatically. We still presented the q -wise independent hash based construction to showcase (already simplified) existing analysis structure, which in turn demonstrates the simplicity of HE from entropic security. Ironically, with existing analysis of the q -wise independent hash based construction, we cannot even choose q to be as small as 2, while entropic security based construction can!

Honey encryption that allows key re-use in the standard model. The existence of entropy bounds in both one-time pad and entropic security hints that security in those two settings needs to burn key entropy for each ciphertext. However, the entropy bounds do not seem to appear in honey encryption when one encrypts only for a particular message distribution. We ask whether HE can remain secure when the secret key is reused.

We first remark that, key re-use does not come for free in honey encryption. The above impossibility in entropic security directly hints this, and also gives an example: the HE construction instantiating with the small-biased set based ES scheme from [21], does not allow key reuse, even when the message is uniform. There, the overall construction is basically in the form of $M \oplus s(K)$ for a function $s()$ defining the small biased set.

Fortunately, we show, the HE construction from the pairwise independent hash (trivially holds for the q -wise independent hash based construction as well) allows one to reuse key for arbitrary number of times t with only a security loss linear to t . We also give a definition of (targeted distribution) semantic security that allows key reuses. Interestingly, the view that leads to the entropy deficiency in entropic security explains the intuition why the second HE construction enables key reuses. As a high level intuition, now facing a vector of independently sampled messages, encoding them individually and then concatenating works as a good DTE; hashing them individually and then concatenating also works as a good pairwise independent hash! Those observations essentially reduce the security of key reuse to the security of HE on another message distribution.

Now we finally have an information theoretically secure encryption that can have and use a key as convenient as in the computational setting, just by giving the encryption algorithm details of the message distribution (which is no more than what the adversary knows).

Discussions. Honey encryption was originally proposed to deal with best possible security for password based encryption, where a very small amount of entropy in password is available. We find the underlying concept exploring the message distribution to “mute” the brute-force attack very inspiring and applicable to broader information theoretic setting, which motivated us to examine information theoretic encryption via the HE lens and vice versa.

As we demonstrate, exploring message distribution enables us to circumvent the major obstacle about secret key in information theoretic encryption; on the other hand, putting HE in the unbounded security domain also leads to a very natural and simple construction of HE from entropic security, which simplifies our understanding of HE itself. Indeed, all previous HE constructions (even in the random oracle model and only against a weaker message recovery adversary) require fairly complicated analysis.

Other related works. There exist several lines of exploration of relaxing information theoretic security or adding extra setup for smaller key length. One attempt is adding constraints to the adversary, e.g. restricting adversary’s access to limited ciphertext bits[19], or constraining adversary’s memory usage (bounded storage model)[6, 3, 8, 11, 18]. These schemes mostly focus on key expansion by leveraging honest guy’s advantage. In bounded storage model, for example, a short secret key is expanded to a one-time pad key using the random sources, and the actual key in use should still satisfy Shannon’s bound. In comparison, we do not give encryption algorithm any *extra* knowledge in honey encryption; what we give to the encryption algorithm (message distribution) is already known to the attacker. Another relaxed notation proposed by Calmon et al. is ε -symbol secrecy[10], in which it is hard for an adversary to recover message bits (but not functions of messages). Limited by underlying encoding scheme, though, the key size cannot be compressed to be

as small as security parameter. A third way of relaxation is to restrict input messages to concrete distribution, for example, maximal correlation secrecy[17] requires input to be uniformly distributed.

We note that, none of the constructions dealt with security when a key is reused, except that might be possible in the bounded storage model; but, essentially, the random source generate fresh randomness for each encryption, which implicitly uses a super large key.

Shikata[24] proposed formalization of several information theoretic security definitions and gave lower bounds of key length, which was later extended to multiple-use model. While our work restricts the input messages to follow certain distribution, in their work, however, a key is required to have $\Omega(m^T)$ length if it is reused T times, where m denotes the message length. Therefore, it is clear they cannot have a short key and enable the key reuse.

HE has a number of real-world applications, such as secure password vaults[7, 12], genomic data[13], and natural languages [20, 1]. Recently, a systematic study about various pseudorandom encodings including DTE was given in [2].

2 Preliminaries and Honey Encryption Background

Notations. Let \mathcal{S} be a set, a distribution on \mathcal{S} is defined to be a function $p : \mathcal{S} \rightarrow [0, 1]$ such that $\sum_{s \in \mathcal{S}} p(s) = 1$. Denote $U_{\mathcal{S}}$ to be the uniform distribution on \mathcal{S} . For a set $B \subseteq \mathcal{S}$, define $p(B) = \sum_{s \in B} p(s)$. By $s \leftarrow_p \mathcal{S}$ we mean sampling an element s from \mathcal{S} according to distribution p , and by $s \leftarrow_{\$} \mathcal{S}$ we mean s is sampled uniformly from \mathcal{S} . Let \mathcal{A} be a randomized algorithm, then by $y \leftarrow_{\$} \mathcal{A}(X)$ we mean y is the output of algorithm \mathcal{A} running on input X . We use $y \leftarrow \mathcal{A}(X)$ if \mathcal{A} is a deterministic algorithm instead. For a game G , we use $\Pr[G \Rightarrow \text{true}]$ to denote the probability that G outputs true.

Min-entropy. Let $X \leftarrow_p \mathcal{S}$ be a random variable with distribution p . The min-entropy of X is $H_{\infty}(X) = -\log \max_{s \in \mathcal{S}} p(s)$. We also use notations $H_{\infty}(p) = H_{\infty}(X)$ for simplicity.

q -wise independent hash functions. A family of hash functions $\{H_i : \mathcal{M} \rightarrow \mathcal{S}\}_{i \in \mathcal{I}}$ is called universal hash family if for all $m_1, m_2 \in \mathcal{M}, m_1 \neq m_2, \Pr_{i \leftarrow \mathcal{I}}[H_i(m_1) = H_i(m_2)] \leq \frac{1}{|\mathcal{S}|}$. Furthermore, the hash family $\{H_i\}$ is called q -wise independent if for any distinct $m_1, m_2, \dots, m_q \in \mathcal{M}$ and any $t_1, t_2, \dots, t_q \in \mathcal{S}$,

$$\left| \Pr_{i \leftarrow \mathcal{I}}[H_i(m_1) = t_1 \wedge H_i(m_2) = t_2 \wedge \dots \wedge H_i(m_q) = t_q] \right| = \frac{1}{|\mathcal{S}|^q}$$

$\{H_i\}$ is also called pairwise independent when $q = 2$.

Let \mathbb{F} be a field. A (polynomial) q -wise independent hash family $H_{(a_0, \dots, a_{q-1})} : \mathbb{F} \rightarrow \mathbb{F}$, where each $a_i \in \mathbb{F}$, can be constructed [26] as

$$H_{(a_0, \dots, a_{q-1})}(m) = \sum_{i=0}^{q-1} a_i m^i$$

Entropic security. The definition of entropic security was first proposed by Russel and Wang in [21] and later studied by Dodis and Smith[9]. A probabilistic map Y is said to hide all functions of X with leakage ε if for every adversary \mathcal{A} , there exists some adversary \mathcal{A}' such that for all functions f ,

$$|\Pr[\mathcal{A}(Y(X)) = f(X)] - \Pr[\mathcal{A}'() = f(X)]| \leq \varepsilon$$

$\text{TDSS0}_{\text{HE}, p_m, p_k}^{\mathcal{A}_s, f}$	$\text{TDSS1}_{\text{HE}, p_m, p_k}^{\mathcal{A}, f}$
$M \leftarrow_{p_m} \mathcal{M}$	$K \leftarrow_{p_k} \mathcal{K}$
$b \leftarrow_{\$} \mathcal{A}_s$	$M \leftarrow_{p_m} \mathcal{M}$
return $b = f(M)$	$C \leftarrow_{\$} \text{HEnc}(K, M)$
	$b \leftarrow_{\$} \mathcal{A}(C)$
	return $b = f(M)$

■ **Figure 1** TDSS Security Games.

The map $Y()$ is called (μ, ε) -entropically secure if $Y()$ hides all functions of X , whenever the min-entropy of X is at least μ . We say $Y()$ is (μ, ε) -entropically secure for predicates if $Y()$ hides all functions of X that take value in $\{0, 1\}$.

Honey encryption[15, 14]. A honey encryption (HE) scheme $\text{HE} = (\text{HEnc}, \text{HDec})$ is designed for a specific input distribution. We use \mathcal{K} , \mathcal{M} and \mathcal{C} denote the key space, the message space and the ciphertext space, and p_k, p_m denote the key distribution on \mathcal{K} and the message distribution on \mathcal{M} respectively. The encryption algorithm will take p_m as input.

Target distribution semantic security (TDSS) [14]. Since honey encryption is designed only for each specific message distribution, the semantic security type of definition has to be adapted for a targeted distribution. For more detailed discussions, we refer to [14].

Let $f : \mathcal{M} \rightarrow \{0, 1\}$ be a predicate on \mathcal{M} , $p_f(b) = \Pr[f(M) = b \mid M \leftarrow_{p_m} \mathcal{M}]$, and $\omega_f = \max\{p_f(0), p_f(1)\}$. Define security games for HE with respect to distributions p_k, p_m in Figure 1: In game TDSS0 an adversary \mathcal{A}_s called a simulator tries to guess the value of $f(M)$ with no access to ciphertexts, while in game TDSS1 the adversary \mathcal{A} guess $f(M)$ given an encryption of M . The advantage of an adversary \mathcal{A} with respect to HE, distributions p_m, p_k and predicate f is defined as:

$$\text{Adv}_{\text{HE}, p_m, p_k}^{\text{tdss}}(\mathcal{A}, f) = \Pr[\text{TDSS1}_{\text{HE}, p_m, p_k}^{\mathcal{A}, f} \Rightarrow \text{true}] - \Pr[\text{TDSS0}_{\text{HE}, p_m, p_k}^{\mathcal{A}_s, f} \Rightarrow \text{true}]$$

The optimal strategy for \mathcal{A}_s in TDSS0 is to output the most probable value of $f(M)$ given p_m, f , which gives $\Pr[\text{TDSS0}_{\text{HE}, p_m, p_k}^{\mathcal{A}_s, f} \Rightarrow \text{true}] = \omega_f$. Therefore we can rewrite

$$\text{Adv}_{\text{HE}, p_m, p_k}^{\text{tdss}}(\mathcal{A}, f) = \Pr[\text{TDSS1}_{\text{HE}, p_m, p_k}^{\mathcal{A}, f} \Rightarrow \text{true}] - \omega_f.$$

► **Definition 1.** An HE scheme HE with respect to key distribution p_k and message distribution p_m is said to be ε -TDSS secure if

$$\text{Adv}_{\text{HE}, p_m, p_k}^{\text{tdss}} = \max_{\mathcal{A}, f} \text{Adv}_{\text{HE}, p_m, p_k}^{\text{tdss}}(\mathcal{A}, f) \leq \varepsilon$$

where the max is over all (unbounded) adversary \mathcal{A} and arbitrary predicate f .

Distribution-transforming encoder (DTE)[15]. A DTE is a pair of algorithms $\text{DTE} = (\text{encode}, \text{decode})$ defined relative to sets \mathcal{M} and \mathcal{S} , where randomized encoding algorithm encode takes as input a message $M \in \mathcal{M}$ and outputs $S \in \mathcal{S}$, and deterministic decoding algorithm decode takes as input $S \in \mathcal{S}$ and outputs $M \in \mathcal{M}$. DTE should always satisfy correctness: for all $M \in \mathcal{M}$, $\Pr[\text{decode}(\text{encode}(M)) = M] = 1$.

$\text{SAMP0}_{\text{DTE}}^{\mathcal{D}}$	$\text{SAMP1}_{\text{DTE}, p_m}^{\mathcal{D}}$
$S \leftarrow \$ \mathcal{S}$	$M \leftarrow_{p_m} \mathcal{M}$
$b \leftarrow \$ \mathcal{D}(S)$	$S \leftarrow \$ \text{encode}(M)$
return $b = 1$	$b \leftarrow \$ \mathcal{D}(S)$
	return $b = 1$

■ **Figure 2** DTE Security Games.

$\text{HEnc}(K, M)$	$\text{HDec}(K, C)$	$\text{HEnc}(K, M)$	$\text{HDec}(K, C)$
$S \leftarrow \$ \text{encode}(M)$	$S \leftarrow \text{Dec}(K, C)$	$S \leftarrow \$ \text{encode}(M)$	$(R, \tilde{C}) \leftarrow C$
$C \leftarrow \$ \text{Enc}(K, S)$	$M \leftarrow \text{decode}(S)$	$R \leftarrow \$ \{0, 1\}^r$	$S \leftarrow \text{H}_R(K) \oplus \tilde{C}$
return C	return M	$\tilde{C} \leftarrow \text{H}_R(K) \oplus S$	$M \leftarrow \text{decode}(S)$
		$C \leftarrow (R, \tilde{C})$	return M
		return C	

■ **Figure 3** Left: DTE-then-Encrypt; Right: DTE-then-Hash.

The security property for DTE schemes is defined via the security games in Figure 2. The advantage of an adversary \mathcal{D} against DTE and distribution p_m is:

$$\text{Adv}_{\text{DTE}, p_m}^{\text{dte}}(\mathcal{D}) = \Pr[\text{SAMP1}_{\text{DTE}, p_m}^{\mathcal{D}} \Rightarrow \text{true}] - \Pr[\text{SAMP0}_{\text{DTE}}^{\mathcal{D}} \Rightarrow \text{true}]$$

Define DTE advantage as measurement for DTE security as follows:

► **Definition 2.** *The DTE advantage of a scheme DTE with respect to distribution p_m is defined to be $\text{Adv}_{\text{DTE}, p_m}^{\text{dte}} = \max_{\mathcal{D}} \text{Adv}_{\text{DTE}, p_m}^{\text{dte}}(\mathcal{D})$, where the maximization is over all (unbounded) adversary \mathcal{D} .*

Although it is mentioned in [2] that DTE does not exist for *all* distributions, a large number of distributions can be encoded using a DTE. For example, a distribution can be encoded using inverse-sampling DTE in [15] if values of all probability mass functions are explicitly given.

DTE-then-Encrypt[15, 14]. DTE-then-Encrypt serves as a framework to construct HE schemes with respect to a target distribution. A message is first encoded using DTE and then encrypted using a symmetric encryption. The framework is described in Fig 3. In [15, 14], the encryption is instantiated with a hash-based encryption scheme $\text{Enc}(K, M) = (R, \text{H}_R(K) \oplus M)$, where R is randomly drawn. We rename this DTE-then-Hash for clarity. The DTE-then-Hash construction is described in Fig 3.

3 HE Constructions Secure in the Standard Model

Recall that for an n -bit input, one-time pad requires the key to have at least n bits of min-entropy. Entropically secure encryption scheme (ES scheme) relaxes this entropy requirement to $n - t$ by restricting the input with at least t bits of min-entropy. If the ES scheme uses a key with min-entropy $n - t + \delta$, it would achieve a security around $2^{-\delta/2}$ [9]. HE completely removes the key entropy requirement by working on a specific message distribution (called

target distribution). JRT showed that an HE scheme using a key with min-entropy δ would achieve ε -TDSS (targeted distribution semantic security) for ε slightly bigger than $2^{-\delta/2}$ (e.g. ε is around 2^{-13} when $\delta = 30$ [14]), albeit in random oracle model.

Below we present our two attempts at HE constructions in the standard model as well as security analysis: the first using q -wise independent hash family, and the second using an ES scheme. The first construction, which is a natural extension of JR[15] and JRT[14]’s construction in standard model, gives a security bound comparable to the random oracle case, and requires q to be around key length. While this is acceptable for low-entropy key settings, it turns out that our second HE construction using ES scheme can achieve asymptotically optimal security bounds and only requires a pairwise independent hash family.

3.1 HE from q -wise independent hash

Our first attempt to construct HE in standard model follows from the previous works of [15] and [14]. Their construction applies DTE-then-Hash framework where the hash function is modeled as a random oracle. While we cannot use random oracle since we are working in standard model, it is natural to consider the case where the hash function is modeled as a q -wise independent hash family instead. As we will see, the overall structure of the analysis of [15] and [14] still applies with the use of new techniques, yet it results in a relatively weak bound. We give a high level overview in this section and leave the full proof in the appendix.

Let us recall the TDSS security analysis of DTE-then-Hash construction presented in [14, 15], which came in two main steps. In the first step, game transitions are performed on the original TDSS security game to another game where the adversary’s best strategy is to decrypt the ciphertext using all possible keys, compute the predicate value on all decryptions, and output the bit which is supported by larger probability mass of keys. Note that such transition does not rely on random oracle, and incurs an error which is bounded by DTE advantage. In the second step, different balls-into-bins analysis are applied where each decryption attempt is considered as throwing a “ball” into a “bin”. Each decryption result is considered to be independent if random oracle is used; however, we introduce correlations to the decryption results using q -wise independent hash family, and we cannot use majorization lemma to simplify the probability analysis. We overcome these problems by using a more general Chernoff-like bound on q -wise independent random variables[22]. An advantage of this bound is that it does not require the random variables to have the same distribution, so that we can even omit the majorization step in previous proofs.

Game transitions. Similar to previous results [14, 15], the first part of the proof is summarized by the following lemma from [14], which transforms the estimation of the adversary’s advantage $\text{Adv}_{\text{HE}, p_m, p_k}^{\text{tdss}}(\mathcal{A}, f)$ in TDSS games to the expected outcome $\mathbb{E}[L_{p_k}^{\text{H,DTE},f}]$ in an experiment $E_{p_k}^{\text{H,DTE},f}$ defined in 4, via a sequence of games. The bias of the predicate is $\omega_f = \max\{p_f(0), p_f(1)\}$.

Note that experiment $E_{p_k}^{\text{H,DTE},f}$ actually describes a brute-force attack, and the expectation $\mathbb{E}[L_{p_k}^{\text{H,DTE},f}]$ denotes the success probability of this attack. In other words, we are transitioning to an experiment in which a brute-force attack is performed, and we are concerned with the success probability of such attack.

► **Lemma 3** ([14]). *Let HE be defined using DTE-then-Hash construction with respect to distributions p_m, p_k , q -wise independent hash family $\{H_i\}$ and DTE scheme DTE, f be a predicate on \mathcal{M} , \mathcal{A} be any adversary, then*

$$\text{Adv}_{\text{HE}, p_m, p_k}^{\text{tdss}}(\mathcal{A}, f) \leq \text{Adv}_{\text{DTE}, p_m}^{\text{dte}} + \mathbb{E}[L_{p_k}^{\text{H,DTE},f}] - \omega_f$$

where $L_{p_k}^{\text{H,DTE},f}$ is defined via experiment $E_{p_k}^{\text{H,DTE},f}$.

Experiment $E_{p_k}^{\text{H,DTE},f}$
$R \leftarrow_{\$} \{0, 1\}^r, \tilde{C} \leftarrow_{\$} \mathcal{S}$
$B_0 \leftarrow \emptyset, B_1 \leftarrow \emptyset$
for $i = 1, 2, \dots, \mathcal{K} $ do
$S_i \leftarrow \text{H}_R(K_i) \oplus \tilde{C}$
$M_i \leftarrow \text{decode}(S_i)$
$b_i \leftarrow f(M_i)$
$B_{b_i} \leftarrow B_{b_i} \cup \{K_i\}$
endfor
$L_{p_k}^{\text{H,DTE},f} \leftarrow \max_{b \in \{0,1\}} p_k(B_b)$

■ **Figure 4** Experiment used in security analysis of DTE-then-Hash.

Proof for Lemma 3 is given in Appendix A.

Bounding success probability of predicting. In the second part of our proof, we give a bound on $\mathbb{E}[L_{p_k}^{\text{H,DTE},f}]$, which represents the probability of a success brute-force attack. This is the different part of analysis we have to do without the luxury of relying on random oracle to get independence or majorization technique to simplify the balls-into-bins experiment. Our main tool is the following more general Chernoff-like result which is a special case of a theorem from [22]:

► **Lemma 4** ([22]). *Let X_1, \dots, X_n be q -wise independent random variables confined to the interval $[0, 1]$, and $X = \sum_{i=1}^n X_i$ with $\mu = \mathbb{E}[X]$, then*

- 1) *For $\delta \leq 1$ satisfying $q \leq \lfloor \delta^2 \mu e^{-1/3} \rfloor$, $\Pr[|X - \mu| \geq \delta \mu] \leq e^{-\lfloor q/2 \rfloor}$;*
- 2) *For $\delta \geq 1$ satisfying $q \leq \lfloor \delta \mu e^{-1/3} \rfloor$, $\Pr[|X - \mu| \geq \delta \mu] \leq e^{-\lfloor q/2 \rfloor}$.*

Define p_d to be the probability distribution of \mathcal{M} given by sampling a uniformly random seed from \mathcal{S} and then applying `decode`, i.e.

$$p_d(M) = \Pr[M^* = M \mid S \leftarrow_{\$} \mathcal{S}, M^* \leftarrow \text{decode}(S)]$$

The following lemma gives a bound on $\mathbb{E}[L_{p_k}^{\text{H,DTE},f}]$; we defer detailed proof to Appendix B:

► **Lemma 5.** *Let $p_t(b) = \Pr[f(M) = b \mid M \leftarrow_{p_d} \mathcal{M}]$ and $\omega_t = \max\{p_t(0), p_t(1)\}$. Let ω_k denote the maximum key probability. Then for all $q \leq e^{-1/3}/2\omega_k$,*

$$\mathbb{E}[L_{p_k}^{\text{H,DTE},f}] \leq \omega_t + e^{-\lfloor q/2 \rfloor} + (1 - 2e^{-\lfloor q/2 \rfloor})(q\omega_k)^{1/2}e^{1/6}$$

Finalizing the bound. The following theorem sums up the two steps above and gives a TDSS security bound:

► **Theorem 6.** *Let HE be constructed using DTE-then-Hash with respect to distributions p_m, p_k and q -wise independent hash family $\{\text{H}_i\}_{i \in \{0,1\}^r}$, where ω_k denotes the maximum key probability. Then for all $q \leq e^{-1/3}/2\omega_k$,*

$$\text{Adv}_{\text{HE}, p_m, p_k}^{\text{tdss}} \leq 2\text{Adv}_{\text{DTE}, p_m}^{\text{dte}} + e^{-\lfloor q/2 \rfloor} + (1 - 2e^{-\lfloor q/2 \rfloor})(q\omega_k)^{1/2}e^{1/6}$$

In other words, HE is ε -TDSS secure for

$$\varepsilon = 2\text{Adv}_{\text{DTE}, p_m}^{\text{dte}} + e^{-\lfloor q/2 \rfloor} + (1 - 2e^{-\lfloor q/2 \rfloor})(q\omega_k)^{1/2}e^{1/6}$$

The proof for Lemma 5 and Theorem 6 are given in Appendix B.

Note that if we choose $q = \log(1/\omega_k) = H_\infty(p_k)$ in Theorem 6, the security bound is asymptotic to $O((\omega_k \log(1/\omega_k))^{1/2})$, which is close to $O(\omega_k^{1/2})$ in low min-entropy key settings. This implies that we can choose $q \approx \lambda$ if we are using a key of length λ . However, we cannot choose $q = O(1)$ because of the results in Lemma 4. Furthermore, although the $O(\log(1/\omega_k))$ hardly affects the security bound, we can actually push harder towards the $O(\omega_k^{1/2})$ bound, which is considered to be optimal in [14].

3.2 HE from entropically secure encryption

We now give an HE construction satisfying TDSS in the standard model via entropic security. This idea arises when we view both of them as information theoretic encryption candidates. An observation is that entropic security notion and TDSS notion are similar in some way: they both capture the hardness for an unbounded adversary to learn any predicate of the input message given an encryption of this message. The difference is that entropic security expects entropy from the message, while HE scheme further explores the message distribution. Intuitively, we would like to “modify” input message to gain enough entropy.

In order to match the entropy requirement in entropic security, we first encode input messages using DTE (constructed specifically for message distribution, see above definition in Sec.2). It should be pointed out, however, that DTE actually outputs a near-uniform distribution which has almost full entropy. We can think of the DTE output as a uniform distribution in our analysis, which only incurs negligible error. Such ES schemes are easy to find; in fact, any $(n - \alpha, \varepsilon)$ -ES scheme for $\alpha \geq 0$ supports uniform input, since it supports any input with min-entropy at least $n - \alpha$. In this way, the entropy requirements in the ES scheme becomes unimportant since we are using uniform distribution as input; We can even use an ES scheme which only supports uniform input, which leads to better parameters. In fact, according to an observation in JRT[14], our construction achieves asymptotically best TDSS security bound $O(\omega_k^{1/2})$.

More interestingly, our analysis is much simpler than that in JRT[14] and even the message recovery security analysis in JR[15], which is a strictly weaker security notion than TDSS; while at the same time, this simpler analysis is “tighter”: the entropic security path gives an instantiation from *pairwise*-independent hash, but following the more complicated analysis structure in Sec. 3.1, the resulting bound does not allow us to choose q to be as small as 2 in the q -wise independent hash based construction.

► **Theorem 7.** *Let p_m be a distribution on a set \mathcal{M} , p_k be a distribution on a set \mathcal{K} , and n be an integer. Let $\mathbf{e} = (\text{EEnc}, \text{EDec})$ be an $(n - \alpha, \varepsilon)$ -ES scheme for arbitrary $0 \leq \alpha < n$ with key space \mathcal{K} , key distribution p_k and message space $\{0, 1\}^n$, and $\text{DTE} = (\text{encode}, \text{decode})$ a DTE scheme with respect to p_m that outputs an n -bit binary string. Then the DTE-then-Encrypt construction using \mathbf{e} and DTE is an ε' -TDSS secure HE scheme with respect to key distribution p_k and message distribution p_m , where $\varepsilon' = \varepsilon + \text{Adv}_{\text{DTE}, p_m}^{\text{dte}}$.*

Proof. It is easy to check that the HE construction is well defined and satisfies correctness. We show that HE satisfies ε -TDSS security. For every adversary \mathcal{A} and arbitrary predicate f , consider the following sequence of games:

Now game G_0 is exactly the same as game $\text{TDSS1}_{\text{HE}, p_m, p_k}^{\mathcal{A}, f}$. Consider the following adversary $\mathcal{D}(S)$ against DTE security: One can check that \mathcal{D} simulates G_1 when $S \leftarrow_{\$} \mathcal{S}$ and simulates G_0 when S is an encoding of $M \leftarrow_{p_m} \mathcal{M}$. Furthermore, \mathcal{D} returns 1 if and only if \mathcal{A} wins in corresponding games. It follows from DTE advantage definition that

$$\Pr[G_0 \Rightarrow \text{true}] - \Pr[G_1 \Rightarrow \text{true}] \leq \text{Adv}_{\text{DTE}, p_m}^{\text{dte}} \quad (1)$$

23:12 Fooling an Unbounded Adversary with a Short Key, Repeatedly

Game G_0	Game G_1
$K \leftarrow_{p_k} \mathcal{K}$	$K \leftarrow_{p_k} \mathcal{K}$
$M \leftarrow_{p_m} \mathcal{M}$	$S \leftarrow_{\$} \{0, 1\}^n$
$S \leftarrow_{\$} \text{encode}(M)$	$M \leftarrow \text{decode}(S)$
$C \leftarrow_{\$} \text{EEnc}(K, S)$	$C \leftarrow_{\$} \text{EEnc}(K, S)$
$b \leftarrow_{\$} \mathcal{A}(C)$	$b \leftarrow_{\$} \mathcal{A}(C)$
return $b = f(M)$	return $b = f(M)$

■ **Figure 5** Sequence of games used in Theorem 7.

Adversary $\mathcal{D}(S)$
$K \leftarrow_{p_k} \mathcal{K}$
$M \leftarrow \text{decode}(S)$
$C \leftarrow_{\$} \text{EEnc}(K, S)$
$b \leftarrow_{\$} \mathcal{A}(C)$
if $b = f(M)$ return 1
else return 0

■ **Figure 6** Adversary $\mathcal{D}(S)$ against DTE security.

We now work in game G_1 . Note that the random variable S is uniformly sampled from $\{0, 1\}^n$, therefore S has min-entropy n . By the definition of entropic security, there exists some adversary \mathcal{A}' such that for all functions \tilde{f} ,

$$|\Pr[\mathcal{A}(\text{EEnc}(K, S)) = \tilde{f}(S)] - \Pr[\mathcal{A}'() = \tilde{f}(S)]| \leq \varepsilon$$

Setting $\tilde{f}(S) = f(\text{decode}(S))$ we get

$$|\Pr[\mathcal{A}(\text{EEnc}(K, S)) = f(M)] - \Pr[\mathcal{A}'() = f(M)]| \leq \varepsilon$$

Now $\Pr[\mathcal{A}(\text{EEnc}(K, S)) = f(M)]$ is exactly the probability that \mathcal{A} returns true in game G_1 , in other words $\Pr[\mathcal{A}(\text{EEnc}(K, S)) = f(M)] = \Pr[G_1 \Rightarrow \text{true}]$. On the other hand, we have $\Pr[\mathcal{A}'() = f(M)] \leq \Pr[\text{TDSS0}_{p_m}^{\mathcal{A}', f} \Rightarrow \text{true}]$ since the simulator \mathcal{A}_s can simply run \mathcal{A}' and return the same value as \mathcal{A}' does. In other words,

$$|\Pr[G_1 \Rightarrow \text{true}] - \Pr[\text{TDSS0}_{p_m}^{\mathcal{A}', f} \Rightarrow \text{true}]| \leq \varepsilon \quad (2)$$

Combining 1 and 2 we have

$$\begin{aligned} \text{Adv}_{\text{HE}, p_m, p_k}^{\text{tdss}}(\mathcal{A}, f) &= \Pr[\text{TDSS1}_{\text{HE}, p_m, p_k}^{\mathcal{A}, f} \Rightarrow \text{true}] - \Pr[\text{TDSS0}_{p_m}^{\mathcal{A}, f} \Rightarrow \text{true}] \\ &\leq \varepsilon + \text{Adv}_{\text{DTE}, p_m}^{\text{dte}} = \varepsilon' \end{aligned}$$

Since this holds for all \mathcal{A} and f , we have

$$\text{Adv}_{\text{HE}, p_m, p_k}^{\text{tdss}} = \max_{\mathcal{A}, f} \text{Adv}_{\text{HE}, p_m, p_k}^{\text{tdss}}(\mathcal{A}, f) \leq \varepsilon'$$

In other words, HE is ε' -TDSS secure. ◀

As a special case, let \mathbf{e} be the random hashing construction in [9]:

► **Lemma 8** ([9]). *Let $\{H_i\}_{i \in \mathcal{I}}$ be a pairwise independent hash family from $\{0, 1\}^n$ to $\{0, 1\}^n$, and K is sampled according to p_k , then the encryption scheme $\text{Enc}(K, M; i) = (i, H_i(K) \oplus M)$ is (μ, ε) -entropically secure for $\mu = n - H_\infty(p_k) + 2 \log(1/\varepsilon) + 2$. Specifically, $\text{Enc}(K, M; i)$ is $(n - \delta, \varepsilon)$ -entropically secure for $H_\infty(p_k) = \delta + 2 \log(1/\varepsilon) + 2$.*

In our HE construction we only require the entropically secure encryption scheme to support uniform distribution, therefore we can choose $\delta = 0$ for optimal parameters. This leads to the following corollary:

► **Corollary 9.** *Let HE with respect to key distribution p_k and message distribution p_m be constructed using the DTE-then-Hash construction, in which a pairwise independent hash family and an DTE scheme with advantage $\text{Adv}_{\text{DTE}, p_m}^{\text{dte}}$ are applied. Then HE is ε -TDSS secure for $\varepsilon = 2^{(2 - H_\infty(p_k))/2} + \text{Adv}_{\text{DTE}, p_m}^{\text{dte}} = 2\omega_k^{1/2} + \text{Adv}_{\text{DTE}, p_m}^{\text{dte}}$, where ω_k denotes the maximum key probability.*

Corollary 9 shows that we can achieve $O(\omega_k^{1/2})$ TDSS security using a pairwise independent hash family. Comparing this to Remark 5.6 in JRT[14] which states that TDSS security is at least at the order of $\omega_k^{1/2}$, we conclude that our construction achieves asymptotically best security bound while only requiring pairwise independent hash. This is an improvement over JRT[14]'s results, especially since we are working in standard model compared to their random oracle assumption (and also to JR[15] which only considered a weaker message recovery attack with RO).

4 Multi-Message Security

In this section, we are concerned with another drawback of information theoretic encryption besides the key length: the same key must not be used to encrypt multiple messages. Indeed, using one-time pad to encrypt two messages m_1, m_2 with the same key k yields two ciphertexts $m_1 \oplus k, m_2 \oplus k$, from which one can easily recover the value of $m_1 \oplus m_2$. We first analyze the (in)security of key reuse in entropic security, which also has implication that a honey encryption which is not carefully designed for key reuse will also be facing attacks when re-using the same key. Nevertheless, we prove that our HE construction in the standard model using pairwise independent hash further allows one to re-use a short key: this HE construction finally addresses both issues.

4.1 Insecurity of key re-use in entropic security

Entropic security leveraging message entropy helps decrease the key length, however, entropy security does not give a solution to this problem: ES schemes become insecure when a single key is used to encrypt multiple messages, even if these messages are independently sampled. Informally speaking, each encryption requires a slice of fresh randomness from the key, thus the key has to be long enough in order to provide sufficient randomness. We give an analysis on lower bound of the key needed for reuse: generalizing the analysis from single-message settings in [9], first we show that entropic security for multiple messages implies indistinguishability of multiple ciphertexts; then the lower bound can be derived from a Shannon-style bound (when we choose a special representative message distribution). This lower bound implies that a secret key in an ES scheme can only be used to encrypt very limited number of messages.

Key reuse for independent messages in entropy security. We first give a formal definition of entropic security in key reuse scenario, where a single key is used to encrypt multiple independently sampled messages. Note that the security definition becomes stronger if we remove the independence restriction; since we are after a negative result, it suffices to consider this weaker variant.

► **Definition 10.** A probabilistic map $Y()$ is called (t, ε, T) -entropically secure if for all independent random variables X_1, \dots, X_T where each X_i has min-entropy at least t , and for all adversary \mathcal{A} , there exists some adversary \mathcal{A}' such that for all functions f ,

$$|\Pr[\mathcal{A}(Y(X_1), \dots, Y(X_T)) = f(X_1, \dots, X_T)] - \Pr[\mathcal{A}'() = f(X_1, \dots, X_T)]| \leq \varepsilon$$

In the first part of the proof, we will show that for a (t, ε, T) -entropically secure encryption scheme, the joint distribution of T ciphertexts (using the same key) satisfies indistinguishability definition. The latter basically requires that for any two message distributions with the same entropy, the ciphertext distribution would be indistinguishable. An alternative (and equivalent) definition that makes the following easier is that there exists one particular distribution G (that is irrelevant to the system), for all message distributions, the resulting ciphertext is indistinguishable with G . We first generalize those definitions to fit our setting of multiple messages:

► **Definition 11.** A randomized map $Y()$ is (t, ε, T) -indistinguishable, if there is a random variable G , such that for every independent random variables X_1, \dots, X_T over $\{0, 1\}^n$ where each X_i has min-entropy at least t , we have

$$\text{SD}((Y(X_1), \dots, Y(X_T)), G) \leq \varepsilon$$

We prove the following lemma: an entropic secure encryption that can re-use the key for T times implies a form of indistinguishability.

► **Lemma 12.** (t, ε, T) -entropic security for predicates implies $(t - 1, 4T\varepsilon, T)$ -indistinguishability.

Proof. Let (X_1, \dots, X_T) and (X'_1, \dots, X'_T) be two vectors of random variables where each X_i is independent from each X_j , each X'_i is independent from each X'_j , and each X_i, X'_i has min-entropy at least $t - 1$.

First of all, it suffices to prove the indistinguishability of (X_1, \dots, X_T) and (X'_1, \dots, X'_T) when each X_i and each X'_i is a flat distribution on some set of 2^{t-1} points. Otherwise we can rewrite (X_1, \dots, X_T) and (X'_1, \dots, X'_T) as sum of distributions

$$(X_1, \dots, X_T) = \sum_i a_i(X_{i_1}, \dots, X_{i_T}), (X'_1, \dots, X'_T) = \sum_j b_j(X'_{j_1}, \dots, X'_{j_T})$$

where each coordinate X_{i_k}, X'_{j_k} is a flat distribution. $\text{SD}((X_1, \dots, X_T), (X'_1, \dots, X'_T))$ can then be upper bounded by $\sum_{i,j} a_i b_j \text{SD}(X_{i_1}, \dots, X_{i_T}, (X'_{j_1}, \dots, X'_{j_T}))$. Therefore it suffices to show that for every pair of $(X_{i_1}, \dots, X_{i_T}), (X'_{j_1}, \dots, X'_{j_T})$,

$$\text{SD}((X_{i_1}, \dots, X_{i_T}), (X'_{j_1}, \dots, X'_{j_T})) \leq 4T\varepsilon.$$

Now assume (X_1, \dots, X_T) and (X'_1, \dots, X'_T) satisfy that: for each i , X_i and X'_i are two flat distributions over *disjoint* sets of 2^{t-1} points each. Let $\tilde{X} = (\tilde{X}_1, \dots, \tilde{X}_T)$ be sampled as follows: to sample from \tilde{X}_i , first sample a random bit b_i uniformly; if $b_i = 0$, sample \tilde{X}_i according to X_i , and otherwise sample \tilde{X}_i according to X'_i . In this way, each \tilde{X}_i has

min-entropy t , and $\tilde{X}_1, \dots, \tilde{X}_T$ are independent from each other. For every i , let f_i be the predicate which outputs 0 if \tilde{X}_i is sampled according to X_i , and 1 if \tilde{X}_i is sampled according to X'_i , regardless of the choices of other coordinates.

For each i , define an adversary \mathcal{A}_i which, given inputs $y = (Y(\tilde{X}_1), \dots, Y(\tilde{X}_T))$, outputs 0 if $Y(\tilde{X}_i)$ is more likely under the distribution $Y(X_i)$ than $Y(X'_i)$, and 1 otherwise. Note that the output of \mathcal{A}_i is independent of the choices of $\tilde{X}_1, \dots, \tilde{X}_{i-1}, \tilde{X}_{i+1}, \dots, \tilde{X}_T$. Therefore the probability that \mathcal{A}_i successfully predicts f_i is

$$\Pr[\mathcal{A}_i(Y(\tilde{X}_1), \dots, Y(\tilde{X}_T)) = f_i(\tilde{X}_1, \dots, \tilde{X}_T)] = \frac{1}{2} + \frac{1}{2} \text{SD}(Y(X_i), Y(X'_i))$$

On the other hand, for any random variable G over $\{0, 1\}$ independent of \tilde{X}_i , the probability that $G = f_i(\tilde{X}_1, \dots, \tilde{X}_T)$ is exactly $\frac{1}{2}$. By (t, ε, T) -entropic security we get

$$\Pr[\mathcal{A}_i(Y(\tilde{X}_1), \dots, Y(\tilde{X}_T)) = f_i(\tilde{X}_1, \dots, \tilde{X}_T)] \leq \max_G \Pr[G = f_i(\tilde{X}_1, \dots, \tilde{X}_T)] + \varepsilon = \frac{1}{2} + \varepsilon$$

From two inequalities above we get $\text{SD}(Y(X_i), Y(X'_i)) \leq 2\varepsilon$ for every $i \in [1, T]$. Therefore,

$$\text{SD}((Y(X_1), \dots, Y(X_T)), (Y(X'_1), \dots, Y(X'_T))) \leq \sum_{i=1}^T \text{SD}(Y(X_i), Y(X'_i)) \leq 2T\varepsilon$$

For the case where X_i and X'_i are not disjoint, we can find a third vector (X''_1, \dots, X''_T) where each X''_i is a flat distribution on 2^{t-1} points disjoint from both X_i and X'_i . In this way,

$$\text{SD}((Y(X_1), \dots, Y(X_T)), (Y(X''_1), \dots, Y(X''_T))) \leq 2T\varepsilon$$

$$\text{SD}((Y(X'_1), \dots, Y(X'_T)), (Y(X''_1), \dots, Y(X''_T))) \leq 2T\varepsilon$$

We then use the triangle inequality to show that

$$\text{SD}((Y(X_1), \dots, Y(X_T)), (Y(X'_1), \dots, Y(X'_T))) \leq 4T\varepsilon \quad \blacktriangleleft$$

The indistinguishability result can be used to bound the key size: essentially, we will choose a special distribution of vector such that each coordinate has a fixed prefix w_i , while the remaining parts are sampled uniformly. ES ciphertexts can be seen as a statistically secure encryption scheme with all the w_i as input, which gives us the desired bound.

► **Lemma 13.** *Any encryption scheme which is (t, ε, T) -entropically secure for inputs of length n requires a key of length at least $(n - t + 1)T - 1$.*

Proof. For every $w = (w_1, \dots, w_T) \in \{0, 1\}^{(n-t+1)T}$, where each $w_i \in \{0, 1\}^{n-t+1}$, let M_{w_i} be uniformly chosen from $\{w_i\} \times \{0, 1\}^{t-1}$ and $M_w = (M_{w_1}, \dots, M_{w_T})$. Then each M_{w_i} has min-entropy $t - 1$, and any (t, ε, T) -entropically secure encryption scheme Enc produces indistinguishable distributions $(\text{Enc}(M_{w_1}), \dots, \text{Enc}(M_{w_T}))$, and $(\text{Enc}(M_{w'_1}), \dots, \text{Enc}(M_{w'_T}))$ for any pair (w, w') . Therefore $(\text{Enc}(M_{w_1}), \dots, \text{Enc}(M_{w_T}))$ can be seen as an encryption scheme for $(n - t + 1)T$ -bit strings, and thus Enc must have key length $(n - t + 1)T - 1$. ◀

► **Remark 14.** Lemma 13 implies that in an ES scheme, a key of length μ can only be used to encrypt $O(\mu)$ messages even if these messages have entropy $n - O(1)$. Therefore one cannot expect ES to remain secure in multi-message settings, especially with the use of a short key.

5 Conclusions and Future Works

In this paper, we investigate the following problem: is it possible to have an encryption scheme (for a class of messages) that satisfies unbounded semantic type of security, but using only a short key and the key can be re-used. We give an affirmative answer with a construction of honey encryption from pair-wise independent hash that satisfies both.

Approaching the problem via the lens of honey encryption inspires us to explore a nice trade-off between security and generality. We hope our initial positive results can motivate more researches on exploring message distribution for better information theoretic encryption: more general encoding mechanisms, relaxing message independence requirement in key reuse, considering integrity and more.

References

- 1 Esther Omolara Abiodun and Aman Jantan. Modified honey encryption scheme for encoding natural language message. *Int. J. Electr. Comput. Eng.*, 9(3):1871, 2019.
- 2 Thomas Agrikola, Geoffroy Couteau, Yuval Ishai, Stanislaw Jarecki, and Amit Sahai. On pseudorandom encodings. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 639–669. Springer, 2020. doi:10.1007/978-3-030-64381-2_23.
- 3 Yonatan Aumann, Yan Zong Ding, and Michael O. Rabin. Everlasting security in the bounded storage model. *IEEE Trans. Inf. Theory*, 48(6):1668–1680, 2002. doi:10.1109/TIT.2002.1003845.
- 4 Petra Berenbrink, Tom Friedetzky, Zengjian Hu, and Russell A. Martin. On weighted balls-into-bins games. *Theor. Comput. Sci.*, 409(3):511–520, 2008. doi:10.1016/j.tcs.2008.09.023.
- 5 Nikita Borisov, Ian Goldberg, and David A. Wagner. Intercepting mobile communications: the insecurity of 802.11. In Christopher Rose, editor, *MOBICOM 2001, Proceedings of the seventh annual international conference on Mobile computing and networking, Rome, Italy, July 16-21, 2001*, pages 180–189. ACM, 2001. doi:10.1145/381677.381695.
- 6 Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 292–306. Springer, 1997. doi:10.1007/BFb0052243.
- 7 Rahul Chatterjee, Joseph Bonneau, Ari Juels, and Thomas Ristenpart. Cracking-resistant password vaults using natural language encoders. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 481–498. IEEE Computer Society, 2015. doi:10.1109/SP.2015.36.
- 8 Yan Zong Ding and Michael O. Rabin. Hyper-encryption and everlasting security. In Helmut Alt and Afonso Ferreira, editors, *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings*, volume 2285 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2002. doi:10.1007/3-540-45841-7_1.
- 9 Yevgeniy Dodis and Adam D. Smith. Entropic security and the encryption of high entropy messages. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 556–577. Springer, 2005. doi:10.1007/978-3-540-30576-7_30.
- 10 Flávio du Pin Calmon, Muriel Médard, Linda M. Zeger, João Barros, Mark M. Christiansen, and Ken R. Duffy. Lists that are smaller than their parts: A coding approach to tunable secrecy. In *50th Annual Allerton Conference on Communication, Control, and Computing*,

- Allerton 2012, Allerton Park & Retreat Center, Monticello, IL, USA, October 1-5, 2012, pages 1387–1394. IEEE, 2012. doi:10.1109/Allerton.2012.6483380.
- 11 Stefan Dziembowski and Ueli M. Maurer. Optimal randomizer efficiency in the bounded-storage model. *J. Cryptol.*, 17(1):5–26, 2004. doi:10.1007/s00145-003-0309-y.
 - 12 Maximilian Golla, Benedict Beuscher, and Markus Dürmuth. On the security of cracking-resistant password vaults. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1230–1241. ACM, 2016. doi:10.1145/2976749.2978416.
 - 13 Zhicong Huang, Erman Ayday, Jacques Fellay, Jean-Pierre Hubaux, and Ari Juels. Genoguard: Protecting genomic data against brute-force attacks. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 447–462. IEEE Computer Society, 2015. doi:10.1109/SP.2015.34.
 - 14 Joseph Jaeger, Thomas Ristenpart, and Qiang Tang. Honey encryption beyond message recovery security. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 758–788. Springer, 2016. doi:10.1007/978-3-662-49890-3_29.
 - 15 Ari Juels and Thomas Ristenpart. Honey encryption: Security beyond the brute-force bound. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 293–310. Springer, 2014. doi:10.1007/978-3-642-55220-5_17.
 - 16 Burt Kaliski. PKCS #5: Password-based cryptography specification version 2.0. *RFC*, 2898:1–34, 2000. doi:10.17487/RFC2898.
 - 17 Cheuk Ting Li and Abbas El Gamal. Maximal correlation secrecy. *IEEE Trans. Inf. Theory*, 64(5):3916–3926, 2018. doi:10.1109/TIT.2018.2816066.
 - 18 Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *J. Cryptol.*, 5(1):53–66, 1992. doi:10.1007/BF00191321.
 - 19 Ueli M. Maurer and James L. Massey. Local randomness in pseudorandom sequences. *J. Cryptol.*, 4(2):135–149, 1991. doi:10.1007/BF00196773.
 - 20 Abiodun Esther Omolara, Aman Jantan, Oludare Isaac Abiodun, and Howard Eldon Poston. A novel approach for the adaptation of honey encryption to support natural language message. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 2018.
 - 21 Alexander Russell and Hong Wang. How to fool an unbounded adversary with a short key. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 133–148. Springer, 2002. doi:10.1007/3-540-46035-7_9.
 - 22 Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. *SIAM J. Discret. Math.*, 8(2):223–250, 1995. doi:10.1137/S089548019223872X.
 - 23 Claude E. Shannon. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 28(4):656–715, 1949. doi:10.1002/j.1538-7305.1949.tb00928.x.
 - 24 Junji Shikata. Formalization of information-theoretic security for key agreement, revisited. In *Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey, July 7-12, 2013*, pages 2720–2724. IEEE, 2013. doi:10.1109/ISIT.2013.6620721.
 - 25 Mathy Vanhoef and Frank Piessens. Key reinstallation attacks: Forcing nonce reuse in WPA2. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS*

2017, Dallas, TX, USA, October 30 - November 03, 2017, pages 1313–1328. ACM, 2017. doi:10.1145/3133956.3134027.

- 26 Mark N. Wegman and Larry Carter. New classes and applications of hash functions. In *20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29-31 October 1979*, pages 175–182. IEEE Computer Society, 1979. doi:10.1109/SFCS.1979.26.

A Proof for Lemma 3

Proof. We use the sequence of games in Figure 7:

Game G_0	Game G_1	Game G_2
1 : $K \leftarrow_{p_k} \mathcal{K}$	$K \leftarrow_{p_k} \mathcal{K}$	$R \leftarrow_{\$} \{0, 1\}^r$
2 : $M \leftarrow_{p_m} \mathcal{M}$	$S \leftarrow_{\$} \mathcal{S}$	$\tilde{C} \leftarrow_{\$} \mathcal{S}$
3 : $S \leftarrow_{\$} \text{encode}(M)$	$M \leftarrow \text{decode}(S)$	$C \leftarrow (R, \tilde{C})$
4 : $R \leftarrow_{\$} \{0, 1\}^r$	$R \leftarrow_{\$} \{0, 1\}^r$	$b \leftarrow_{\$} \mathcal{A}(C)$
5 : $\tilde{C} \leftarrow \text{H}_R(K) \oplus S$	$\tilde{C} \leftarrow \text{H}_R(K) \oplus S$	$K \leftarrow_{p_k} \mathcal{K}$
6 : $C \leftarrow (R, \tilde{C})$	$C \leftarrow (R, \tilde{C})$	$S \leftarrow \text{H}_R(K) \oplus \tilde{C}$
7 : $b \leftarrow_{\$} \mathcal{A}(C)$	$b \leftarrow_{\$} \mathcal{A}(C)$	$M \leftarrow \text{decode}(S)$
8 : return $b = f(M)$	return $b = f(M)$	return $b = f(M)$

■ **Figure 7** Sequence of games used in Lemma 3.

First of all, game G_0 is exactly game TDSS1 with the HEnc part written in details. Therefore, we have $\Pr[G_0 \Rightarrow \text{true}] = \Pr[\text{TDSS1}_{\text{HE}, p_m, p_k}^{\mathcal{A}, f} \Rightarrow \text{true}]$. By definition we have

$$\text{Adv}_{\text{HE}, p_m, p_k}^{\text{tdss}}(\mathcal{A}, f) = \Pr[G_0 \Rightarrow \text{true}] - \omega_f \quad (3)$$

Next, the gap between G_0 and G_1 can be reduced to DTE security. Note that the only difference between G_0 and G_1 appears in line 2 and line 3. For any adversary \mathcal{A} , consider the following adversary \mathcal{D} (in Fig 8) against DTE security: It is clear that \mathcal{D} outputs 1 if and only if \mathcal{A} outputs the correct bit. Furthermore, \mathcal{D} simulates G_1 when $S \leftarrow_{\$} \mathcal{S}$, and \mathcal{D} simulates G_0 when S is the DTE encoding of message $M \leftarrow_{p_m} \mathcal{M}$. It follows that:

$$\Pr[G_0 \Rightarrow \text{true}] - \Pr[G_1 \Rightarrow \text{true}] \leq \text{Adv}_{\text{DTE}, p_m}^{\text{dte}} \quad (4)$$

Adversary $\mathcal{D}(S)$
$K \leftarrow_{p_k} \mathcal{K}$
$M \leftarrow \text{decode}(S)$
$R \leftarrow_{\$} \{0, 1\}^r$
$\tilde{C} \leftarrow \text{H}_R(K) \oplus S$
$C \leftarrow (R, \tilde{C})$
$b \leftarrow_{\$} \mathcal{A}(C)$
if $b = f(M)$ return 1
else return 0

■ **Figure 8** Adversary $\mathcal{D}(S)$ against DTE security games.

<p style="margin: 0;">Adversary $\mathcal{A}^*(C)$</p> <hr style="border: 0.5px solid black; margin: 2px 0;"/> <p style="margin: 0;">$(R, \tilde{C}) \leftarrow C$</p> <p style="margin: 0;">$L_0 \leftarrow 0, L_1 \leftarrow 0$</p> <p style="margin: 0;">for $K \in \mathcal{K}$ do</p> <p style="margin: 0;">$S \leftarrow \mathbf{H}_R(K) \oplus \tilde{C}$</p> <p style="margin: 0;">$M \leftarrow \text{decode}(S)$</p> <p style="margin: 0;">$L_{f(M)} \leftarrow L_{f(M)} + p_k(K)$</p> <p style="margin: 0;">endfor</p> <p style="margin: 0;">$b^* \leftarrow \text{argmax}_{b \in \{0,1\}} L_b$</p> <p style="margin: 0;">return b^*</p>
--

■ **Figure 9** Adversary $\mathcal{A}^*(C)$ in game G_2 .

The next step is to show that G_2 is equivalent to G_1 . Note that in G_1 we first sample S uniformly from \mathcal{S} and independently from K , which guarantees \tilde{C} also to be a uniform sample from \mathcal{S} independent from K . Therefore, we can first sample \tilde{C} uniformly and choose K after the execution of \mathcal{A} , which is exactly the case in G_2 . Therefore,

$$\Pr[G_1 \Rightarrow \text{true}] = \Pr[G_2 \Rightarrow \text{true}] \quad (5)$$

Now consider the following adversary \mathcal{A}^* in G_2 : Adversary \mathcal{A}^* adds up the probability mass of all the keys resulting in $f(M) = 0$ and $f(M) = 1$ respectively; therefore, we have $L_0 = \Pr[f(M) = 0]$ and $L_1 = \Pr[f(M) = 1]$. This implies that \mathcal{A}^* is the best possible adversary in game G_2 . If we denote $\Pr[G_2^* \Rightarrow \text{true}]$ to be the probability that \mathcal{A}^* succeeds in G_2 , we have

$$\Pr[G_2 \Rightarrow \text{true}] \leq \Pr[G_2^* \Rightarrow \text{true}] \quad (6)$$

Finally, consider Experiment $E_{p_k}^{\text{H,DTE},f}$. For fixed choice of (R, \tilde{C}) , the value $L_{p_k}^{\text{H,DTE},f}$ is exactly L_{b^*} in adversary \mathcal{A}^* , which is the probability that \mathcal{A}^* succeeds conditioned on the choice of (R, \tilde{C}) . Taking expectation over all (R, \tilde{C}) gives

$$\Pr[G_2^* \Rightarrow \text{true}] = \mathbb{E}[L_{p_k}^{\text{H,DTE},f}] \quad (7)$$

Combining 3, 4, 5, 6 and 7 gives the proof for Lemma 3. ◀

B Proof for Lemma 5 and Theorem 6

Proof. In order to bound $\mathbb{E}[L_{p_k}^{\text{H,DTE},f}]$, we would like to give an upper bound of $\Pr[L_{p_k}^{\text{H,DTE},f} \geq \alpha]$ for some $\alpha \in (0, 1)$. Recall that $L_{p_k}^{\text{H,DTE},f} = \max\{p_k(B_0), p_k(B_1)\}$, where $p_k(B_0), p_k(B_1)$ represents the probability that predicate f returns 0 or 1 respectively, under random choices of K . by union bound

$$\begin{aligned} \Pr[L_{p_k}^{\text{H,DTE},f} \geq \alpha] &= \Pr[\max\{p_k(B_0), p_k(B_1)\} \geq \alpha] \\ &\leq \Pr[p_k(B_0) \geq \alpha] + \Pr[p_k(B_1) \geq \alpha] \end{aligned} \quad (8)$$

It turns out that we only need to bound $\Pr[p_k(B_0) \geq \alpha]$ and $\Pr[p_k(B_1) \geq \alpha]$, where $p_k(B_1) = \sum_{f(M_i)=1} p_k(K_i) = \sum_{i=1}^{|\mathcal{K}|} f(M_i)p_k(K_i)$, $p_k(B_0) = 1 - p_k(B_1) = \sum_{i=1}^{|\mathcal{K}|} (1 - f(M_i))p_k(K_i)$. At this point, the value of each $p_k(K_i)$ is fixed given p_k , therefore we are only concerned with the distributions of $f(M_i)$.

23:20 Fooling an Unbounded Adversary with a Short Key, Repeatedly

► **Lemma 15.** *The random variables $M_1, \dots, M_{|\mathcal{K}|}$ are q -wise independent, and each M_i has the same distribution p_d .*

Here the q -wise independence follows from the fact that $H_R(K_1), \dots, H_R(K_{|\mathcal{K}|})$ are q -wise independent (for randomly chosen R), and that each M_i is a function of $H_R(K_i)$. Each M_i has distribution p_d since \tilde{C} is uniformly chosen from \mathcal{S} and independent from R , and therefore each S_i is uniformly chosen from \mathcal{S} .

Now let $p_t(b) = \Pr[f(M) = b \mid M \leftarrow_{p_d} \mathcal{M}]$ and $\omega_t = \max\{p_t(0), p_t(1)\}$. We can assume without loss of generality that $\omega_t = p_t(1) \geq p_t(0)$. It follows that $1/2 \leq \omega_t \leq 1$. In this way, $f(M_1), \dots, f(M_{|\mathcal{K}|})$ are q -wise independent random variables satisfying for every i , $\Pr[f(M_i) = 1] = \omega_t, \Pr[f(M_i) = 0] = 1 - \omega_t$.

We can apply Lemma 4 to prove the following proposition:

► **Proposition 16.** *Let $\omega_k = \max_{1 \leq i \leq |\mathcal{K}|} p_k(K_i)$, and $\alpha = \omega_t + (q\omega_k\omega_t)^{1/2}e^{1/6}$. For $q \leq \omega_t e^{-1/3}/\omega_k$,*

$$\Pr[p_k(B_1) \geq \alpha] \leq e^{-\lfloor q/2 \rfloor}, \Pr[p_k(B_0) \geq \alpha] \leq e^{-\lfloor q/2 \rfloor}$$

Proof. We first prove the proposition for $p_k(B_1)$. Define $X_i = f(M_i)p_k(K_i)/\omega_k$ for $1 \leq i \leq |\mathcal{K}|$, $X = \sum_{i=1}^{|\mathcal{K}|} X_i$ and $\mu = \mathbb{E}[X]$. Since for each i , $p_k(K_i)/\omega_k$ is a constant value independent of M_i , the random variables $X_1, X_2, \dots, X_{|\mathcal{K}|}$ are q -wise independent satisfying $\Pr[X_i = p_k(K_i)/\omega_k] = \omega_t, \Pr[X_i = 0] = 1 - \omega_t$ for all $1 \leq i \leq |\mathcal{K}|$. Therefore

$$\mu = \mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^{|\mathcal{K}|} X_i\right] = \sum_{i=1}^{|\mathcal{K}|} \mathbb{E}[X_i] = \sum_{i=1}^{|\mathcal{K}|} \frac{p_k(K_i)}{\omega_k} \omega_t = \frac{\omega_t}{\omega_k}$$

Furthermore, $p_k(B_1) = \sum_{i=1}^{|\mathcal{K}|} f(M_i)p_k(K_i) = \sum_{i=1}^{|\mathcal{K}|} X_i\omega_k = \omega_k X$.

We apply Lemma 4 on X by choosing $\delta = (q/\mu e^{-1/3})^{1/2} = (q\omega_k/\omega_t)^{1/2}e^{1/6}$ and assuming that $\delta \leq 1$, which is equivalent to $q \leq \omega_t e^{-1/3}/\omega_k$. One can check that $\alpha = (1 + \delta)\omega_t$. Lemma 4 states that $\Pr[X \geq (1 + \delta)\mu] \leq e^{-\lfloor q/2 \rfloor}$. On the other hand,

$$\begin{aligned} \Pr[X \geq (1 + \delta)\mu] &= \Pr\left[X \geq (1 + \delta)\frac{\omega_t}{\omega_k}\right] \\ &= \Pr[\omega_k X \geq (1 + \delta)\omega_t] = \Pr[p_k(B_1) \geq \alpha] \end{aligned}$$

Therefore $\Pr[p_k(B_1) \geq \alpha] \leq e^{-\lfloor q/2 \rfloor}$. This proves the first part of the proposition.

The second part of the proof for $p_k(B_0)$ comes in a similar fashion. This time we redefine $X_i = (1 - f(M_i))p_k(K_i)/\omega_k$, $X = \sum_{i=1}^{|\mathcal{K}|} X_i$ and $\mu = \mathbb{E}[X] = (1 - \omega_t)/\omega_k$. Note that $p_k(B_0) = \omega_k X$. Consider both cases of Lemma 4:

1) If $q \leq \mu e^{-1/3} = (1 - \omega_t)e^{-1/3}/\omega_k$, we choose $\delta = (q/\mu e^{-1/3})^{1/2} \leq 1$, ensuring that $q = \delta^2 \mu e^{-1/3}$. Conditions of the first inequality of Lemma 4 are satisfied since q is always an integer. Therefore,

$$\Pr[p_k(B_0) \geq (1 + \delta)\mu\omega_k] = \Pr[X \geq (1 + \delta)\mu] \leq e^{-\lfloor q/2 \rfloor}$$

One can check that $(1 + \delta)\mu\omega_k = (1 - \omega_t) + (q\omega_k(1 - \omega_t))^{1/2}e^{1/6} \leq \omega_t + (q\omega_k\omega_t)^{1/2}e^{1/6} = \alpha$. (inequality follows from $1/2 \leq \omega_t \leq 1$) Thus

$$\Pr[p_k(B_0) \geq \alpha] \leq \Pr[p_k(B_0) \geq (1 + \delta)\mu\omega_k] \leq e^{-\lfloor q/2 \rfloor}$$

- 2) If $q \geq \mu e^{-1/3} = (1 - \omega_t)e^{-1/3}/\omega_k$, we choose $\delta = q/\mu e^{-1/3} \geq 1$, ensuring that $q = \delta \mu e^{-1/3}$. Again, conditions of the second inequality of Lemma 4 are satisfied since q is an integer. Therefore,

$$\Pr[p_k(B_0) \geq (1 + \delta)\mu\omega_k] = \Pr[X \geq (1 + \delta)\mu] \leq e^{-\lfloor q/2 \rfloor}$$

This time we have $(1 + \delta)\mu\omega_k = (1 - \omega_t) + q\omega_k e^{1/3} \leq \omega_t + (q\omega_k\omega_t)^{1/2}e^{1/6} = \alpha$. (inequality follows from the assumption $q \leq \omega_t e^{-1/3}/\omega_k$ and $1/2 \leq \omega_t \leq 1$) Thus

$$\Pr[p_k(B_0) \geq \alpha] \leq \Pr[p_k(B_0) \geq (1 + \delta)\mu\omega_k] \leq e^{-\lfloor q/2 \rfloor}$$

Combining both cases, we conclude that for $q \leq \omega_t e^{-1/3}/\omega_k$, $\Pr[p_k(B_0) \geq \alpha] \leq e^{-\lfloor q/2 \rfloor}$. This ends the proof for the second part of the proposition. ◀

For the rest of the proof assume $q \leq \omega_t e^{-1/3}/\omega_k$. From Proposition 16 and eq.8, $\Pr[L_{p_k}^{\text{H,DTE},f} \geq \alpha] \leq 2e^{-\lfloor q/2 \rfloor}$. In this way

$$\begin{aligned} \mathbb{E}[L_{p_k}^{\text{H,DTE},f}] &\leq \alpha(1 - \Pr[L_{p_k}^{\text{H,DTE},f} \geq \alpha]) + \Pr[L_{p_k}^{\text{H,DTE},f} \geq \alpha] \\ &\leq \alpha + 2e^{-\lfloor q/2 \rfloor}(1 - \alpha) \\ &= \omega_t + 2e^{-\lfloor q/2 \rfloor}(1 - \omega_t) + (1 - 2e^{-\lfloor q/2 \rfloor})(q\omega_k\omega_t)^{1/2}e^{1/6} \end{aligned}$$

Since $1/2 \leq \omega_t \leq 1$, $\mathbb{E}[L_{p_k}^{\text{H,DTE},f}]$ is further bounded by

$$\mathbb{E}[L_{p_k}^{\text{H,DTE},f}] \leq \omega_t + e^{-\lfloor q/2 \rfloor} + (1 - 2e^{-\lfloor q/2 \rfloor})(q\omega_k)^{1/2}e^{1/6} \quad (9)$$

This finishes the proof for Lemma 5.

From Lemma 3 and 9,

$$\begin{aligned} \text{Adv}_{\text{HE},p_m,p_k}^{\text{tdss}}(\mathcal{A}, f) &\leq \text{Adv}_{\text{DTE},p_m}^{\text{dte}} + \mathbb{E}[L_{p_k}^{\text{H,DTE},f}] - \omega_f \\ &\leq \text{Adv}_{\text{DTE},p_m}^{\text{dte}} + \omega_t + e^{-\lfloor q/2 \rfloor} + (1 - 2e^{-\lfloor q/2 \rfloor})(q\omega_k)^{1/2}e^{1/6} - \omega_f \end{aligned} \quad (10)$$

for all \mathcal{A}, f and $q \leq e^{-1/3}/2\omega_k$.

Now consider the following adversary \mathcal{D}_f against the DTE security game: on input S , \mathcal{D}_f decodes S , applies f to the decoded message and outputs the f -value obtained. One can check that $\Pr[\text{SAMP1}_{\text{DTE},p_m}^{\mathcal{D}_f} \Rightarrow \text{true}] = \omega_f$ and $\Pr[\text{SAMP0}_{\text{DTE}}^{\mathcal{D}_f} \Rightarrow \text{true}] = \omega_t$. By the definition of DTE advantage $|\omega_f - \omega_t| \leq \text{Adv}_{\text{DTE},p_m}^{\text{dte}}$. This combining with inequality (10) gives

$$\text{Adv}_{\text{HE},p_m,p_k}^{\text{tdss}}(\mathcal{A}, f) \leq 2\text{Adv}_{\text{DTE},p_m}^{\text{dte}} + e^{-\lfloor q/2 \rfloor} + (1 - 2e^{-\lfloor q/2 \rfloor})(q\omega_k)^{1/2}e^{1/6} \quad (11)$$

Notice that the right hand side of inequality (11) is independent of the choice of (\mathcal{A}, f) . This finishes the proof of Theorem 6. ◀

T₅: Hashing Five Inputs with Three Compression Calls

Yevgeniy Dodis ✉

New York University, NY, USA

Dmitry Khovratovich ✉

Ethereum Foundation, Luxembourg, Luxembourg

Dusk Network, Luxembourg, Luxembourg

Nicky Mouha ✉

Stratavia, Largo, MD, USA

Mridul Nandi ✉

Indian Statistical Institute, Kolkata, India

Abstract

Given $2n$ -to- n compression functions h_1, h_2, h_3 , we build a new $5n$ -to- n compression function T_5 , using only 3 compression calls:

$$T_5(m_1, m_2, m_3, m_4, m_5) := h_3(h_1(m_1, m_2) \oplus m_5, h_2(m_3, m_4) \oplus m_5) \oplus m_5$$

We prove that this construction matches Stam’s bound, by providing $\tilde{O}(q^2/2^n)$ collision security and $O(q^3/2^{2n} + nq/2^n)$ preimage security (the latter term dominates in the region of interest, when $q < 2^{n/2}$). In particular, it provides birthday security for hashing 5 inputs using three $2n$ -to- n compression calls, instead of only 4 inputs in prior constructions. Thus, we get a sequential variant of the Merkle-Damgård (MD) hashing, where t message blocks are hashed using only $3t/4$ calls to the $2n$ -to- n compression functions; a 25% *saving* over traditional hash function constructions. This time reduces to $t/4$ (resp. $t/2$) sequential calls using 3 (resp. 2) parallel execution units; saving a factor of 4 (resp. 2) over the traditional MD-hashing, where parallelism does not help to process one message. We also get a novel variant of a Merkle tree, where t message blocks can be processed using $0.75(t - 1)$ compression function calls and depth $0.86 \log_2 t$, thereby *saving 25% in the number of calls and 14% in the update time* over Merkle trees. We provide two modes for a local opening of a particular message block: conservative and aggressive. The former retains the birthday security, but provides longer proofs and local verification time than the traditional Merkle tree. For the aggressive variant, we reduce the proof length to a 29% overhead compared to Merkle trees ($1.29 \log_2 t$ vs $\log_2 t$), but the verification time is now 14% *faster* ($0.86 \log_2 t$ vs $\log_2 t$). However, birthday security is only shown under a plausible conjecture related to the 3-XOR problem, and only for the (common, but not universal) setting where the root of the Merkle tree is known to correspond to a valid t -block message.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases hash functions, Merkle trees, Merkle-Damgård, collision resistance

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.24

Related Version *Full Version:* <https://eprint.iacr.org/2021/373.pdf> [11]

Funding *Yevgeniy Dodis:* Partially supported by gifts from VMware Labs, Facebook and Google, and NSF grants 1314568, 1619158, 1815546.

Acknowledgements Thanks to the organizers and participants of Dagstuhl Seminar 18021 “Symmetric Cryptography,” held from January 7-12, 2018 at Schloss Dagstuhl – Leibniz Center for Informatics. The T_5 construction was first presented there by one of the authors of this paper, and the fruitful discussions there provided an initial starting point for this paper.



© Yevgeniy Dodis, Dmitry Khovratovich, Nicky Mouha, and Mridul Nandi; licensed under Creative Commons License CC-BY 4.0

2nd Conference on Information-Theoretic Cryptography (ITC 2021).

Editor: Stefano Tessaro; Article No. 24; pp. 24:1–24:23



Leibniz International Proceedings in Informatics

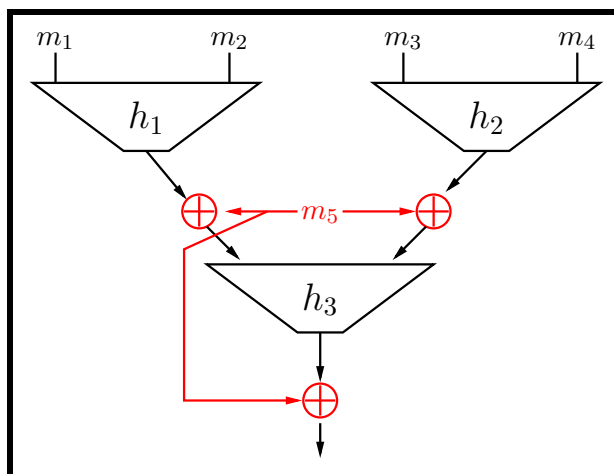
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

A fundamental problem in cryptography is the construction of a hash function using idealized building blocks. A natural way to approach this problem is to use λn -to- n -bit compression functions. Two well-known and widely-deployed constructions follow this approach:

- the *Merkle-Damgård construction* [10, 21], a sequential construction that is used in hash functions such as MD5, SHA-1 and SHA-2, and
- the *Merkle tree* [20], a parallel construction used in hash-based signatures (of interest due to their post-quantum security), version control systems such as git, and cryptocurrencies such as Ethereum.

The collision resistance of the Merkle-Damgård construction and the Merkle tree can be proven, based on the collision-resistance of the compression functions. The number of compression function calls is (essentially) the same for both constructions. For example, setting $\lambda = 2$, which is the focus of this work,¹ they both process t message blocks using t and $(t - 1)$ compression function calls, respectively.



■ **Figure 1** The T_5 construction with five message blocks m_1, m_2, m_3, m_4, m_5 and three compression function calls.

New Compression Function T_5 . In this paper, we introduce the T_5 construction (see Fig. 1) that processes five message blocks using three $2n$ -to- n -bit compression function calls, thereby improving over the state-of-the-art of Merkle-Damgård (with IV counted as message block) and Merkle trees by processing an additional message block with the same number of compression function calls and essentially the same level of collision security.

Although T_5 is of independent theoretical interest to the construction of a compression function, we will also investigate Merkle-Damgård and Merkle trees when instantiated with T_5 .

T_5 with Merkle-Damgård. Our variant of the Merkle-Damgård construction, depicted in Figure 4, processes t message blocks using $3t/4$ calls to the $2n$ -to- n compression functions. If the chaining value is provided as m_5 in T_5 , then h_1 and h_2 can not only be computed in

¹ Without loss of generality, all our results easily generalize to any $\lambda \geq 2$.

parallel, but independently of the chaining value. This allows a fully parallel implementation of h_1 , h_2 , and h_3 (with h_3 “one-round behind”), which is four times faster than MD which requires four sequential compression function calls to process a single input of the same length.

T_5 with Merkle Trees. For our variant of Merkle trees, depicted in Figure 5, we will consider how they are often used in practice: for proof-of-inclusion of data in a larger set.

A *full opening* of a Merkle tree corresponds to the list of all message blocks, which can be used to verify that the message corresponds to a given hash value. An advantage of Merkle trees is it is possible to provide a *local opening*: to verify that one message block belongs to the tree, it suffices to provide a list of compression function outputs that is proportional to the depth of the tree.

These two types of openings give rise to three different notions of collision resistance for trees:

- *full-full* collision resistance, the “traditional” notion that requires finding two distinct messages that result in the same hash value,
- *local-local* collision resistance, where the goal is to find two local openings with the same hash value,
- *full-local* collision resistance, the setting of finding a collision between a full tree and a local opening.

The full-local setting is relevant in the common scenario where a hash value is honestly computed, but the proof is composed by an untrusted party. This happens, for example, when a user sends a message to a cloud server after hashing it, and then later wants to retrieve some message block from the server. Another natural application is the Merkle accumulator, where a protocol accumulates message blocks using a Merkle tree, and later parties provide proofs that a message block is in the tree.

Standard Merkle trees provide the same level of security under all three notions of collision resistance, and the same holds for our Merkle tree variant using T_5 if all four siblings are opened. In this case, our variant of the Merkle tree will process t message blocks using $0.75(t - 1)$ instead of $t - 1$ compression function calls, and depth $0.86 \log_2 t$ instead of $\log_2 t$. Due to the need to open four siblings, the opening proof will increase from $\log_2 t$ to $1.72 \log_2 t$, and the verification time increases as well from $\log_2 t$ to $1.29 \log_2 t$.

However, we also propose an aggressive variant of our construction that only opens three siblings. See Figure 3. When this saving of one element for T_5 is translated to the whole Merkle tree, one gets a smaller local opening proof of $1.29 \log_2 t$, and a shorter verification time of $0.86 \log_2 t$. We prove that this more aggressive variant nevertheless provides the same full-local collision resistance, under a conjecture related to the 3-XOR problem. For local-local collision resistance, we prove security up to $2^{n/3}$ under a conjecture related to the 4-XOR problem. The k -XOR problem is the subject of the well-studied generalized birthday problem by Wagner [33], and used in proof-of-work algorithms such as Equihash. A full comparison of these three constructions will be given in Table 1 of Section 7.2.

Our Results for T_5 . We prove the following security results for T_5 in terms of adversarial advantage after q queries to the inner compression functions:²

² For simplicity of reading, we only list dominant terms, and ignore constant and even small $\text{poly}(n)$ factors; i.e., omit \tilde{O} notation below.

24:4 T₅: Hashing Five Inputs with Three Compression Calls

- $q^2/2^n$ collision resistance, i.e., full-full collision resistance (CR) security (Theorem 2);
- $q^3/2^{2n} + q/2^n$ preimage resistance (Theorem 3).
- $q^2/2^n$ full-local CR security under a conjecture related to the 3-XOR problem (Proposition 7);
- $q^3/2^n$ full-local CR security unconditionally, i.e., 128-bit security for $n = 384$ (Theorem 4);
- $q^3/2^n$ local-local CR security under a conjecture related to the 4-XOR problem (Proposition 10);
- $q^4/2^n$ local-local CR security unconditionally, i.e., 128-bit security for $n = 512$ (Theorem 4).

These results almost immediately imply corresponding security claims for our Merkle-Damgård variant (see Section 6) and our Merkle tree variant (see Section 7 and Table 1). Matching attacks for these security results are provided in the full version of this paper [11].

2 Related Work

The design of a hash function is usually based on one or more *primitives* with fixed-length inputs and outputs. Historically, the most common choice for these primitives were block ciphers. This gives rise to the following question: how can we construct a hash function with the minimum number of block cipher calls?

This question motivated a significant research effort into efficient block-cipher-based hash function constructions. Block ciphers such as Triple-DES and AES have a block size of 64 and 128 bits respectively, which may not provide sufficient collision security when used in the Merkle-Damgård construction. Therefore, one line of work focuses on combining smaller primitives to produce a wider hash function. Results of interest include the Knudsen-Preneel construction based on linear error correcting codes [16], and a double-length construction by Nandi et al. [22] that was generalized by Peyrin et al. [26], and by Seurin and Peyrin [28], which interestingly also links the security to a conjecture related to the 3-sum problem.

A related line of research attempted to improve upon the Merkle-Damgård construction to process additional message blocks. A brief overview of some constructions and an impossibility result was given by Black et al. [6, 7]. More specifically, they considered hash functions that make one block cipher call (under a small set of keys) for each message block to be hashed, and showed that all such constructions are vulnerable to a simple attack.

This work was later generalized by Rogaway and Steinberger [27], and refined in subsequent papers by Stam [29], and by Steinberger et al. [30, 31]. This result, commonly known as “Stam’s bound,” puts a limit on the efficiency (in terms of primitive calls) of any secure hash function construction.

Stam’s bound states that there always exists a collision attack and a preimage attack with at most $2^{n(\lambda-(t-0.5)/r)}$ and $2^{n(\lambda-(t-1)/r)}$ queries respectively on a tn -to- n -bit hash function making r calls to λn -to- n -bit compression functions. We have $t = 5$, $\lambda = 2$, and $r = 3$ in the case of T₅, thereby showing that we cannot hope to do better than $2^{n/2}$ collision and $2^{2n/3}$ preimage security.

As explained by Stam [29], the bound applies to hash functions that satisfy the *uniformity assumption*, and applies to cryptographic permutations ($\lambda = 1$) as well as compressing primitives ($\lambda \geq 1$). However, the main focus of this line of work had been on combining smaller non-compressing primitives (see e.g., Mennink and Preneel [18, 19]).

Recently, McQuoid et al. [17] provided a general framework to prove the collision and second-preimage security of various hash functions. Our T₅ construction is covered by their framework. Unfortunately, their framework does not provide tight collision and second-preimage security bounds for T₅.

Lastly, we recall that a series of papers have investigated optimal trade-offs between time and space for Merkle tree traversal, e.g., Jakobsson et al. [15], Szydło [32], and Berman et al. [4]. Given that we propose T_5 inside a standard Merkle tree, these trade-offs can also be directly applied to the constructions in this paper. We would also like to mention Haitner et al. [13]’s construction which only has depth one, at the cost of making significantly more calls than the standard Merkle tree.

3 Preliminaries

3.1 Notation

If S is a set, $x \stackrel{\$}{\leftarrow} S$ denotes the uniformly random selection of an element from S . We let $y \leftarrow A(x)$ and $y \stackrel{\$}{\leftarrow} A(x)$ be the assignment to y of the output of a deterministic and randomized algorithm $A(x)$, respectively.

For positive integers m, n , we let $\text{Func}(m, n)$ denote the set of all functions mapping $\{0, 1\}^m$ into $\{0, 1\}^n$. We write $h \stackrel{\$}{\leftarrow} \text{Func}(m, n)$ to denote random sampling from the set $\text{Func}(m, n)$ and assignment to h , and say that h is modeled as an ideal hash function. For fixed m and n , such modeling is attempting to approximate the security of real-world, keyless, fixed-input-size compression functions, such as the compression function of SHA-2.

3.2 Security Definitions of Hash Functions

An *adversary* A is a probabilistic algorithm, possibly with access to oracles $\mathcal{O}_1, \dots, \mathcal{O}_\ell$ denoted by $A^{\mathcal{O}_1, \dots, \mathcal{O}_\ell}$. Our definitions of collision (Coll), and preimage (Pre) security are given for any general fixed-input length hash function H built upon the compression functions h_i for $i = 1, \dots, \ell$ where h_i are modeled as ideal functions. Namely, for a fixed adversary A and for all $i = 1$ to ℓ with $h_i \stackrel{\$}{\leftarrow} \text{Func}(2n, n)$, we define the following advantage functions:

$$\text{Adv}_H^{\text{Coll}}(A) = \Pr \left[H^{h_1, \dots, h_\ell}(M) = H^{h_1, \dots, h_\ell}(M') \text{ and } M \neq M' \mid (M, M') \stackrel{\$}{\leftarrow} A^{h_1, \dots, h_\ell}(\cdot) \right]$$

and

$$\text{Adv}_H^{\text{Pre}}(A) = \Pr \left[H^{h_1, \dots, h_\ell}(M) = H^{h_1, \dots, h_\ell}(M') \mid M \stackrel{\$}{\leftarrow} \mathcal{M}_H, M' \stackrel{\$}{\leftarrow} A^{h_1, \dots, h_\ell}(H^{h_1, \dots, h_\ell}(M)) \right]$$

We define the $\text{Adv}_H^{\text{atk}}(q)$ against the $\text{atk} = \{\text{Coll}, \text{Pre}\}$ -security of H as the maximum advantage over all adversaries making at most q total queries to its oracles.

3.3 Local Opening Security

We define local opening security of a hash function output (viewed as a commitment of a message). Given a function H built upon compression functions h_1, h_2, \dots where h_i are all modeled as ideal functions, a local opening $\text{Open}^{h_1, \dots}(\cdot, \cdot)$ for $H^{h_1, \dots}$ maps a pair (M, i) to π (called proof) where $M = (m_1, m_2, \dots, m_c)$ is a message (a tuple of blocks) and $1 \leq i \leq c$ is an index.

24:6 T₅: Hashing Five Inputs with Three Compression Calls

Correctness of Local Opening. There is an efficient function $\text{Ver}^{h_1, \dots}$ such that for all message M , all index i ,

$$\text{Ver}^{h_1, \dots}(i, m_i, \text{Open}^{h_1, \dots}(M, i), H(M)) = 1.$$

Security of Local Opening. We provide two notions of local opening security. For the stronger variant, which we call “local-local,” the adversary wins if it produces an output f corresponding to two contradicting local openings for some position i . For the weaker variant, which we call “full-local,” the adversary wins if it produces an output f corresponding to a local opening contradicting a full opening.

► **Definition 1** (local-local and full-local opening advantage). *Let H be a hash function and Open is a correct local opening for H with Ver is the verification function. For any adversary A , we define the local-local opening advantage as*

$$\text{Adv}_H^{\text{local-local}}(A) = \Pr \left[\text{Ver}(i, m, \pi, f) = \text{Ver}(i, m', \pi', f) = 1, m \neq m' \mid (i, m, m', \pi, \pi', f) \stackrel{\$}{\leftarrow} A^{h_1, \dots} \right]$$

We define full-local opening advantage (a weaker variant of the above) of A as

$$\text{Adv}_H^{\text{full-local}}(A) = \Pr \left[\text{Ver}(i, m', \pi', H(M)) = 1, m' \neq m_i \mid (i, M, m', \pi') \stackrel{\$}{\leftarrow} A^{h_1, \dots} \right]$$

And finally, for a function H with local opening algorithms $(\text{Open}, \text{Ver})$ and attack $z \in \{\text{local-local}, \text{full-local}\}$, we define $\text{Adv}_H^z(q) = \max_A \text{Adv}_H^z(A)$ as the maximum advantage over all adversaries making at most q total queries to its oracles.

Intuitively, the weaker definition protects against situations where the initial commitment f was honestly produced, using some long message M . Thus, a contradictory local opening will result in finding a collision between a local opening and a full opening. In contrast, the (traditional) stronger definition is directly concerned with somebody producing two contradictory local openings.

By-Pass Hash Computation. We say that H has a *by-pass computation* H_i corresponding to a local opening Open for a fixed index $1 \leq i \leq c$, if for all M ,

$$H_i^{h_1, \dots}(m_i, \text{Open}^{h_1, \dots}(M, i)) = H^{h_1, \dots}(M).$$

In other words, given a proof (output of the Open) and the message block for the index (for which the proof is produced), we can compute the hash output of the message (without knowing the other blocks of the message).

The presence of by-pass computations $\mathcal{H} = \{H_i\}$ for all indices lead to a natural verification algorithm as follows: $\text{Ver}(i, m, \pi, f) = 1$ whenever $H_i^{h_1, \dots}(m, \pi) = f$. For a fixed index i , we define the *cross-collision* advantage between the hash function H and a by-pass computation H_i as

$$\text{Adv}_{H, H_i}^{\text{Coll}}(A) = \Pr \left[H(M) = H_i(m', \pi) \text{ and } M_i \neq m' \mid (M, m', \pi) \stackrel{\$}{\leftarrow} A^{h_1, \dots} \right]$$

Similarly, we define the *inter-collision* advantage for by-pass computation H_i as

$$\text{Adv}_{H_i}^{\text{Coll}}(A) = \Pr \left[H_i(m, \pi) = H_i(m', \pi') \text{ and } m \neq m' \mid (m, \pi, m', \pi) \stackrel{\$}{\leftarrow} A^{h_1, \dots} \right]$$

Let $\mathcal{H} = \{H_i\}$ be the family of by-pass computations for all indices i . We define

$$\mathbf{Adv}_{H, \mathcal{H}}^{\text{Coll}}(q) = \max_A \max_i \mathbf{Adv}_{H, H_i}^{\text{Coll}}(A) \quad \text{and} \quad \mathbf{Adv}_{\mathcal{H}}^{\text{Coll}}(q) = \max_A \max_i \mathbf{Adv}_{H_i}^{\text{Coll}}(A)$$

as the maximum advantage over all by-pass computations for all adversaries making at most q total queries to its oracles.

Now we make a simple observation when by-pass computations $\mathcal{H} = \{H_i\}$ for all indices exist for a hash function H , the induced verification procedure Ver satisfies

$$\mathbf{Adv}_H^{\text{full-local}}(q) \leq \mathbf{Adv}_{H, \mathcal{H}}^{\text{Coll}}(q) \quad \text{and} \quad \mathbf{Adv}_H^{\text{local-local}}(q) \leq \mathbf{Adv}_{\mathcal{H}}^{\text{Coll}}(q) \tag{1}$$

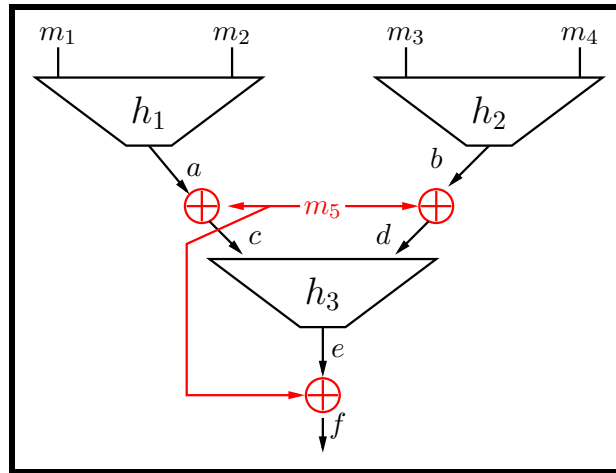
The above observation helps us to reduce the local opening security to cross-collision or inter-collision security problem for the family \mathcal{H} . As all the function H_i in this family are often symmetric, a proof for a fixed function H_i implies the one for the entire family \mathcal{H} .

4 Construction T_5

We define $T_5 : \{0, 1\}^{5n} \rightarrow \{0, 1\}^n$ based on the $2n$ -to- n -bit compression functions h_1, h_2, h_3 as follows:

$$T_5(m_1, m_2, m_3, m_4, m_5) := h_3(h_1(m_1, m_2) \oplus m_5, h_2(m_3, m_4) \oplus m_5) \oplus m_5$$

For all our proofs we will assume that the compression functions h_i for $i = 1$ to 3 are ideal functions.



■ **Figure 2** Modified 2-level Merkle tree $T_5(m_1, m_2, m_3, m_4, m_5)$ with an extra input m_5 for the same 3 hash calls.

Notation. As shown in Figure 2, we use the variables

- m_1 and m_2 (resp. m_3 and m_4) to denote the left and right halves of various inputs to h_1 (resp. h_2);
- a (resp. b) to denote various outputs of h_1 (resp. h_2);
- c and d to denote the left and right halves of various inputs to h_3 ;
- e to denote various outputs of h_3 ;
- $M = (m_1, m_2, m_3, m_4, m_5)$ to denote various inputs to T_5 ;

■ f to denote various outputs of T_5 .

Hence, a valid computation of $T_5(M) = T_5(m_1, \dots, m_5)$ proceeds as follows:

1. Set $a = h_1(m_1, m_2)$, $b = h_2(m_3, m_4)$.
2. Set $c = a \oplus m_5$, $d = b \oplus m_5$.
3. Set $e = h_3(c, d)$, and output $f = e \oplus m_5$.

We say that a triple of queries $((m_1, m_2), a)$ to h_1 , $((m_3, m_4), b)$ to h_2 , and $((c, d), e)$ to h_3 is *consistent* if

$$a \oplus b = c \oplus d, \tag{2}$$

in which case we define $m_5 = a \oplus c = b \oplus d$, and say that this consistent triple of queries (uniquely) defines a valid T_5 evaluation (M, f) , where $M = (m_1, m_2, m_3, m_4, m_5)$ and $f = e \oplus a \oplus c = e \oplus b \oplus d$.

Main Results of the Section. The main result of this paper is to provide the collision and preimage security of the T_5 hash function. The following theorem shows that T_5 achieves nearly birthday collision security, despite hashing one more input than the traditional Merkle-Damgård function of depth 2.

► **Theorem 2.** *The T_5 construction achieves nearly birthday-bound collision security:*

$$\text{Adv}_{T_5}^{\text{Coll}}(q) \leq \frac{(n^2 + 10)q^2}{2^n} \tag{3}$$

The full formal proof of this result is somewhat subtle, and will be given in the full version of this paper [11]. But an informal proof intuition for a representative special case will be given in Section 4.1.

As our second main result, we also show that T_5 maintains nearly optimal preimage security $\tilde{O}(q/2^n)$ for $q < 2^{n/2}$, which means it offers optimal preimage *and* collision-resistance security in the common range of $q < 2^{n/2}$.

However, even when q grows above $2^{n/2}$, T_5 still offers non-trivial security $O(q^3/2^{2n})$ for values of $q < 2^{2n/3-1}$, which is likely sufficient for most applications. As we show in the full version of this paper [11], T_5 is indeed not preimage resistant when $q > 2^{2n/3}$, so our result is tight.

► **Theorem 3.** *Assuming $q \leq 2^{2n/3-1}$, the T_5 construction achieves the following preimage security:*

$$\text{Adv}_{T_5}^{\text{Pre}}(q) \leq \frac{2q^3}{2^{2n}} + O\left(\frac{qn}{2^n}\right) \tag{4}$$

We give a formal proof in the full version of this paper [11], but present some proof intuition (for an important special case) in Section 4.2, similar to what was done in Section 4.1.

4.1 Proof Intuition for Collision Resistance of T_5

Below we give the proof intuition for the simple, but natural special case where the adversary A makes all of its queries to h_1 and h_2 before any query to h_3 is made. As we explain in the formal proof in the full version of this paper [11], this assumption is with a *significant* loss of generality, and several special arguments are needed to cover the fully general case. However, this simplified case will already demonstrate some of the main arguments of our analysis.

The proof will roughly consist in arguing that A , making q total hash queries, is unlikely – up to birthday advantage – to succeed in the following four tasks. (These tasks are formalized in the full version of this paper [11].)

1. **Task 1:** finding any simple collision in h_1 or h_2 .³
2. **Task 2:** finding some value z for which there exist more than n distinct pairs of queries $((m_1, m_2), (m_3, m_4))$ to h_1 and h_2 result in⁴

$$h_1(m_1, m_2) \oplus h_2(m_3, m_4) = z$$

3. **Task 3:** generate more than nq valid evaluations of T_5 .
4. **Task 4:** generate a non-trivial collision of T_5 .

The argument of each subsequent task will inductively assume that the adversary indeed failed in the previous task.

For Task 1, this is the trivial birthday bound on h_1 or h_2 .

For Task 2, we will formally define the set $C_{12}(z)$ to consist of pairs of queries $((m_1, m_2), a)$ to h_1 and $((m_3, m_4), b)$ satisfying $a \oplus b = z$. Using the fact that no simple collisions in h_1 and h_2 are found, it is easy to see that each of the n queries to h_1 and h_2 inside $C_{12}(z)$ must be distinct. Moreover, the latter of the two queries $((*, a), (*, b)) \in C_{12}(z)$ must collide with a fixed value z plus the former of the two queries. E.g., if the query $(*, a)$ to h_1 was made before $(*, b)$ to h_2 , this tuple will fall inside $C_{12}(z)$ only if $b = z \oplus a$, which happens with probability 2^{-n} . Taking the union bound over all values z and all possible choices of $2n$ out of q queries to be included inside $C_{12}(z)$, we see that

$$\Pr[\exists z \text{ s.t. } |C_{12}(z)| \geq n] \leq 2^n \cdot q^{2n} \cdot \left(\frac{1}{2^n}\right)^n \leq \left(\frac{2q^2}{2^n}\right)^n \ll O\left(\frac{q^2}{2^n}\right)$$

For Task 3, we will use our simplifying assumption that A makes all of its queries to h_1 and h_2 before any query to h_3 is made. In this case, all consistent triples of queries to h_1 , h_2 and h_3 defining a valid input-output (M, f) to T_5 get created by making a call to h_3 . For each of at most q such queries to h_3 on some input (c, d) , we claim that this query will “match up” with a pair of earlier queries $(*, a)$ to h_1 and $(*, b)$ to h_2 only if $m_5 = a \oplus c = b \oplus d$, which is equivalent to $a \oplus b = c \oplus d$, which means that

$$((*, a), (*, b)) \in C_{12}(c \oplus d)$$

But we already assumed that $|C_{12}(z)| \leq n$ for all z , meaning that each query (c, d) can form a consistent tuple with at most n pairs of queries to h_1 and h_2 . Summing over all (up to) q queries to h_3 , the total number of evaluations will be at most nq .⁵

Finally, for Task 4 that we care about, we will once again use our simplifying assumption. In particular, under our assumption, such a collision can only be caused by a call to h_3 on some input (c, d) . From the previous argument, we already know that this query will “match up” with a pair of earlier queries $(*, a)$ to h_1 and $(*, b)$ to h_2 only if $((*, a), (*, b)) \in C_{12}(c \oplus d)$, meaning there are at most n new evaluations of T_5 caused by this query. Also, any two of these n new evaluations cannot collide among themselves, as they have two different values $a \neq a'$ (remember, no collisions in h_1), so the final outputs $f = h_3(c, d) \oplus c \oplus a \neq h_3(c, d) \oplus c \oplus a' = f'$. Thus, the only chance the adversary has is if one of these n new evaluations of T_5 (call the output f) collides with one of at most nq already defined previous evaluations f' of T_5 .

But each of the n new output values f will be *individually random*, as it is equal to random $e = h_3(c, d)$ plus $m_5 = a \oplus c$. Hence, this individually random f can collide with

³ For the general case, we will also need no collisions in a slightly modified variant of h_3 .

⁴ For the general case, we will also need similar guarantees for the combinations of $h_1 + h_3$ and $h_2 + h_3$.

⁵ As we will see, in the general case a very different proof strategy will be needed to extend this argument to valid evaluations of T_5 completed by calls to h_1 and h_2 .

24:10 T₅: Hashing Five Inputs with Three Compression Calls

the previously defined output f' of T₅ with probability at most $nq/2^n$, because from failing Task 3 we know there are at most nq previous evaluations of T₅ completed so far. Taking the union bound over n values of f , and q queries to h_3 , the final bound $O(n^2q^2/2^n)$ follows.

General Case. We will not fully detail the general case (see full argument in the full version of this paper [11]), but briefly demonstrate that making queries to h_1 or h_2 after some queries to h_3 could be potentially helpful to the adversary, and will require adjustments to our proof strategy above.

For example, our proof sketch above showed that, modulo very rare events, the number of valid evaluations of T₅ can increase by at most n for each new query to h_3 . However, imagine that A first makes a query to h_2 with output b . Then A can make $\Omega(q)$ queries (c_i, d_i) all satisfying $c_i \oplus d_i = z$ (for some z). Now, a query to h_1 (made after these $\Omega(q)$ queries to h_3) has a chance to simultaneously match with $\Omega(q) \gg n$ tuples $((*, b), ((c_i, d_i), *))$. Of course, in order for this to happen, the random answer a must match $b \oplus z$, which happens with tiny probability. In fact, we could apply Markov's equality to argue that the probability a query to h_1 will produce more than n new evaluation points is at most (what turns out to be by an easy calculation) $O(q^2/2^n)$. By itself, this is good enough, but it will not “survive” a union bound over up to q potential queries to h_1 . Instead, we will use linearity of expectation to make a global, “stochastic” argument that *all* such (up to) q queries to h_1 and h_2 will define more than nq new evaluations with at most “birthday” probability. See the full version of this paper [11] for the details.

Overall, the full proof in the general case will be noticeably more subtle than the proof intuition given above, but will still follow the same high-level structure.

4.2 Proof Intuition for Preimage Resistance of T₅

As in Section 4.1, we will only consider the special case when the adversary A makes all of its queries to h_1 and h_2 before any query to h_3 is made, as it will contain most of the ideas needed in the general proof. Also, we will assume that $q = \Omega(\sqrt{n} \cdot 2^{n/2})$, as this is the case where the “unexpected” term $q^3/2^{2n}$ appears.⁶

In this setting, there are several differences from the case of collision-resistance we considered so far. First, A is given a specific target f to invert. In particular, we will not care about local collisions in functions h_1 , h_2 , and the c - or d -“shifted” versions of h_3 , as such collisions will happen, but will not help the adversary invert f . Instead, we will care that in each new call to $h_3(c, d)$, the number of valid new evaluations of T₅ will be not much higher than what we expect. Recall, in our special case of only h_3 queries causing all new evaluations of T₅, this number of new evaluations is bounded by $|C_{12}(c \oplus d)|$, where $C_{12}(z)$ is the set of pairs of queries $((m_1, m_2), a)$ to h_1 and $((m_3, m_4), b)$ satisfying $a \oplus b = z$. And since each such evaluation defines an individually random output value $f' = h_3(c, d) \oplus c \oplus a$, the probability this f' matches f is 2^{-n} , meaning A's overall chance to invert f in this query is $|C_{12}(c \oplus d)|/2^n$.

Of course, the adversary can select *any* value of $z = c \oplus d$ to make sure it chooses the largest set $C_{12}(z)$. Thus, if we want to upper bound A's probability of success, we must argue that $K = \max_z |C_{12}(z)|$ is not much higher than its expectation with high probability. In the collision resistance proof, we manage to upper bound $K \leq n$ with “good enough”

⁶ Our general proof will not treat this case separately, but the calculations are slightly easier to write for the “beyond-birthday” case.

probability $(q^2/2^n)^n$. Indeed, this was enough to withstand the union bound over z to give the final probability $(2q^2/2^n)^n \leq 2q^2/2^n$ that $K \geq n$ in that setting.

We will do a similar union bound in our case as well, except that we have $q \gg 2^{n/2}$, so even in the best case scenario we expect $|C_{12}(z)| \geq q^2/2^n \gg n$ for any given z , let alone the z chosen by the adversary. Moreover, the final birthday bound will not be good enough for us in this setting as well. But first let us optimistically assume that for any fixed z , we managed to get the following very strong concentration bound:⁷

$$\Pr \left[|C_{12}(z)| \geq \frac{2q^2}{2^n} \right] \leq \frac{1}{2^{2n}} \quad (5)$$

Then we will be done, because we can take the union bound over z to conclude that $\Pr [K \geq 2q^2/2^n] \leq 2^{-n}$. And, finally, there will be at most $2q^2/2^n$ new evaluations f' per each query to h_3 . Thus, taking the union bound over at most q such queries, A's overall inversion probability (ignoring 2^{-n} failure event above) is upper bounded by:

$$q \cdot \frac{2q^2}{2^n} \cdot \frac{1}{2^n} = \frac{2q^3}{2^{2n}}$$

High Concentration Bound via Tabulation Hashing. So it remains to argue the high concentration bound in (5). This turns out to be much harder than in the collision-resistance case, where the bound we needed was the much weaker $O((q^2/2^n)^n)$, which is meaningless when $q > 2^{n/2}$.

The next naive attempt is to write $|C_{12}(z)|$ as a sum of q^2 indicator variables X_{ij} , equal to 1 is the i -th output a_i of h_1 and the j -th output b_j of h_2 satisfy $a_i \oplus b_j = z$. And then try to use a Chernoff bound to argue that the probability that the sum of these indicator variables is twice as large as its expectation is exponentially low. Unfortunately, the random variables X_{ij} are not even 4-wise independent; e.g., $(a_1 \oplus b_1) \oplus (a_1 \oplus b_2) \oplus (a_2 \oplus b_1) \oplus (a_2 \oplus b_2) = 0$. So we cannot apply the Chernoff bound, and the Chebyshev inequality for pairwise independent random variables is not strong enough. Indeed, the question of getting our concentration bound turned out to be quite deep.

Fortunately, the setting we need turns out to be equivalent to the classical hashing problem, called *simple tabulation hashing*, introduced in the seminal paper of Carter and Wegman [9]. Applied to our setting, given two random “hash tables” T_1 and T_2 with range of size $N = 2^n$, tabulation hashing would map a “ball” $y = (u, v)$ into a “bin” $z = T_1[u] \oplus T_2[v]$. The classical question studied by tabulation hashing is to upper bound occupancy of any such bin z after some Q balls are thrown using tabulation hashing. We defer the details to the formal proof in Section 4.2, but point out that in our setting the tables are implemented using hash functions h_1 and h_2 , and the number of balls $Q \leq q^2$ corresponds to all pairs of queries to h_1 and h_2 .

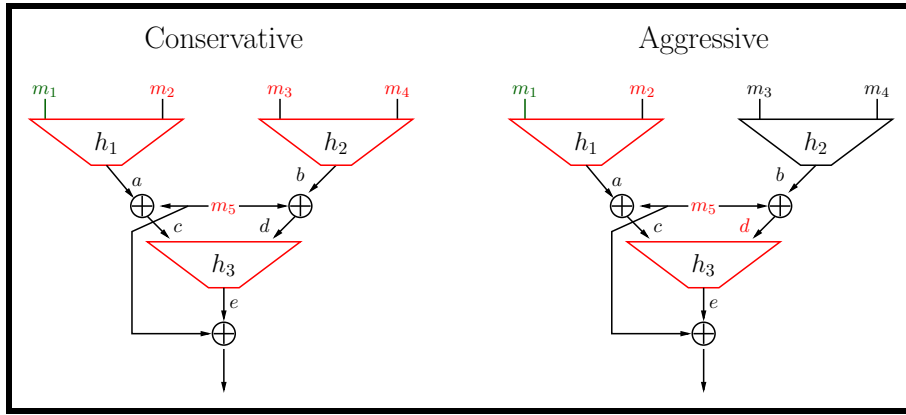
Designing strong enough concentration bound for tabulation hashing (which is exactly what we need!) was an open problem for many years, until the breakthrough result of Pătraşcu and Thorup [24] showed a Chernoff-type concentration bound which enables us to show (5), and thus complete the proof.

⁷ This bound, as stated, is only true if $q = \Omega(\sqrt{n} \cdot 2^{n/2})$. In the general case the term becomes $|C_{12}(z)| \geq \Omega(n) + 2q^2/2^n$, as we expect $\Omega(n)$ multi-collisions already when $q \approx 2^{n/2}$.

5 Aggressive Opening for T_5

In this section we describe a non-trivial opening for T_5 . We first note that a straightforward way to open a block m_i in T_5 is to provide all four siblings $m_{j \neq i}$. For a single T_5 the full-local and the local-local security (Section 3.3) definitions are the same and correspond to the collision security of T_5 which we have already studied. The performance of this method in a full tree is somewhat less attractive and is given in detail in Section 7.2. Thus we call it *conservative*.

Now we provide another point on the security-performance tradeoff for T_5 , which we call *aggressive*. We see that even though the provable security bounds decrease, the heuristic security (as the complexity of best attacks) remains the same under plausible conjectures. Both openings are depicted in Figure 3.



■ **Figure 3** Conservative and aggressive openings for T_5 . Green m_1 is opened. Red are opening elements (4 vs 3) and recomputed compression functions (3 vs 2).

Our (aggressive) local opening Open for T_5 is defined as follows, where we have $m = (m_1, m_2, m_3, m_4, m_5)$.

1. $\text{Open}^{h_1, h_2, h_3}(1, m) = (m_2, m_5, h_2(m_3, m_4) \oplus m_5)$.
2. $\text{Open}^{h_1, h_2, h_3}(2, m) = (m_1, m_5, h_2(m_3, m_4) \oplus m_5)$.
3. $\text{Open}^{h_1, h_2, h_3}(3, m) = (m_4, m_5, h_1(m_1, m_2) \oplus m_5)$.
4. $\text{Open}^{h_1, h_2, h_3}(4, m) = (m_3, m_5, h_1(m_1, m_2) \oplus m_5)$.
5. $\text{Open}^{h_1, h_2, h_3}(5, m) = (m_1, m_2, h_2(m_3, m_4) \oplus m_5)$.

We first show that the above defined opening has a by-pass computation T'_5 for index 1 (one can similarly show for the other indices) and hence it satisfies the correctness condition of an opening. For the sake of simplicity, we skip the hash oracles notation h_1, h_2, h_3 . We define $T'_5 : \{0, 1\}^{4n} \rightarrow \{0, 1\}^n$ based on the $2n$ -to- n -bit compression functions h_1, h_3 as follows:

$$T'_5(m'_1, m'_2, m'_3, m'_4) := h_3(h_1(m'_1, m'_2) \oplus m'_3, m'_4) \oplus m'_3$$

A straightforward calculation shows that $T'_5(m_1, \text{Open}(1, m)) = T_5(m)$. Hence T'_5 is a by-pass computation of T_5 for the opening function defined above.

► **Theorem 4.**

$$\text{Adv}_{T_5}^{\text{full-local}}(q) \leq \text{Adv}_{T_5, T'_5}^{\text{Coll}}(q) \leq \frac{nq^3 + 9q^2}{2^n} \quad (6)$$

and

$$\mathbf{Adv}_{T_5}^{\text{local-local}}(q) \leq \mathbf{Adv}_{T_5}^{\text{Coll}}(q) \leq \frac{q^4}{2^n} \quad (7)$$

We note that the relation between the collision advantage and the opening advantage is already described in (1). So it only remains to bound the cross-collision probability and the collision probability for a family of by-pass hash computations. The full formal proof of the cross-collision bound is somewhat similar to the collision security analysis of T_5 , and is provided in the full version of this paper [11]. But an informal proof intuition for a representative special case will be given in Section 5.1 below. However, the bound for collision advantage of the by-pass family is more or less straightforward and described afterwards.

5.1 Proof Intuition for Theorem 4 (Cross-Collision Bound)

Here, we give a proof sketch of the cross-collision advantage between T_5 and T'_5 . We first note that in the case of T'_5 , all queries are consistent. Hence q queries each to h_1 and h_3 can generate a maximum of q^2 T'_5 evaluations. And, as we have already seen before, q oracle queries can generate at most nq evaluations of T_5 if bad events B_1 and B_2 don't happen. Now, we approach as in the earlier theorem. We break the event that a collision between T_5 and T'_5 has occurred into three parts, depending upon which oracle was queried by the i -th query and give an upper bound to the collision probability. Then, we apply a union bound over the q queries to give the intended result. Let X_i be the event that none of bad events B_1 , B_2 or B_3 happened. The three cases are as follows:

- The i -th query is $((m_1, m_2), a)$ to h_1 . As in the earlier proof, a collision can happen in two ways, either this output a induces a T_5 tuple and a T'_5 tuple which collide, or there is some *previous* value $(*, f) \in \mathbf{Eval}^{i-1}$ such that the random answer $h_1(m_1, m_2) = a$ caused the collision with this f . The former case implies only the trivial collision, which we have ruled out. In the latter case, for a collision to happen, there are two subcases depending on whether $(*, f)$ comes from a previous evaluation of T_5 or T'_5 evaluation. If $(*, f)$ comes from a T_5 evaluation, the random answer a can combine with any of the queries $((c, d), e)$ of h_3 such that $f = a \oplus c \oplus e$. We note that there are a maximum of nq prior T_5 evaluations, and q h_3 queries. The probability that $f = a \oplus c \oplus e$ is $1/2^n$ for each such query. Therefore,

$$\Pr[X_i] \leq nq \cdot q \cdot \frac{1}{2^n} = \frac{nq^2}{2^n}$$

If $(*, f)$ comes from a T'_5 evaluation, we note that there are at most q^2 such evaluations. There exist tuples $((*, b), ((c, d), e)) \in C_{23}(f)$, which can be at most n in number, and the random answer $a = h_1(m_1, m_2)$ should be equal to $b \oplus c \oplus d$. This again gives

$$\Pr[X_i] \leq q^2 \cdot n \cdot \frac{1}{2^n} = \frac{nq^2}{2^n}$$

- The i -th query is $((m_3, m_4), b)$ to h_2 . The only way this query can cause a collision is that there exists some previous value $(*, f) \in \mathbf{Eval}_{T'_5}^{i-1}$ such that the random answer creates a T_5 output equal to f . This case is exactly the same as the first subcase of Case 1.
- The i -th query is $((c, d), e)$ to h_3 . The case generated by this query is the same as that in the first case. If this query generates a T_5 tuple and a T'_5 tuple that collide, we again get only the trivial collision. If there is some previous T_5 evaluation with which this

24:14 T₅: Hashing Five Inputs with Three Compression Calls

query collides, then again, q such queries can combine with n evaluations of T_5 , and the collision probability for each combination is $1/2^n$, resulting in the same probability as in the first case. If this query collides with some previous T_5' evaluation, which are nq in number, we note that there exist tuples $((*, a), (*, b)) \in C_{12}(c \oplus d)$ which can be at most n in number. Again, we get the same probability.

Taking a union bound over the q queries, we find that

$$\Pr [B_4 \cap \overline{B_1 \cap B_2 \cap B_3}] \leq q \cdot \max_i \Pr[X_i] \leq \frac{nq^3}{2^n}.$$

5.2 Proof of (7) (Collision Bound of Family of By-Pass Hash)

Now we prove the second part of the result (collision of by-pass hash family). Here we show the collision probability for T_5' (i.e., for the index 1). The proof for the other indices will be very similar and will have the same bound. As we take the maximum collision probability for all indices, the result will follow. Following a similar notation, let $h_1(m_1, m_2) = b$, $h_1(m'_1, m'_2) = b'$, $c = b \oplus m_5$ and $c' = b' \oplus m'_5$. So, the hash outputs are $T_5'(m_1, m_2, m_5, d) = f = h_3(c, d) \oplus m_5$ and $T_5'(m'_1, m'_2, m'_5, d') = f' = h_3(c', d) \oplus m'_5$. If $f = f'$ with $(m'_1, m'_2, m'_5, d') \neq (m_1, m_2, m_5, d)$ (i.e., collision happens) then we have

$$h_1(m'_1, m'_2) \oplus h_1(m_1, m_2) \oplus (c \oplus h_3(c, d)) \oplus (c' \oplus h_3(c', d')) = 0 \quad (8)$$

Let us write $h_3(x, y) \oplus x$ as $h'_3(x, y)$. It is obvious that h'_3 behaves exactly like a random function (independent with h_1, h_2). Thus, we have shown that collision problem of T_5' is reduced to finding $((m_1, m_2), (c, d)) \neq ((m'_1, m'_2), (c', d'))$ such that

$$h_1(m_1, m_2) \oplus h_1(m'_1, m'_2) \oplus h'_3(c, d) \oplus h'_3(c', d') = 0 \quad (9)$$

We call this problem 4-XOR' (a variant of 4-XOR problem described in the following section). It is also not difficult to construct a collision pair from four pairs satisfying a 4-XOR' relation. In other words, 4-XOR' is equivalent to finding a collision of T_5' . Now, by applying a union bound, the collision probability can be simply bounded above by $q^4/2^n$.

5.3 Reduction of Local Opening Security to 3-XOR/4-XOR Problem

k-XOR Problem. Let F_1, F_2, \dots, F_k be k oracles that output strings of n bits. Find x_1, x_2, \dots, x_k such that

$$F_1(x_1) \oplus F_2(x_2) \oplus \dots \oplus F_k(x_k) = 0.$$

Reduction for full-local. When $k = 3$, the k -XOR problem has an information-theoretical security of $n/3$ -bits, however the best algorithm requires more than $2^{n/2}/\sqrt{n}$ time (ignoring a $\log n$ factor) and queries [8, 23]. Bridging this gap would imply a solution to the long-standing 3-XOR problem, which would be a substantial breakthrough.

► **Conjecture 5.** (*3-XOR Hardness.*) For any algorithm \mathcal{A} that makes less than $q < 2^{n/2}$ queries to F_1, F_2, F_3 and runs for time q , the probability that for randomly chosen F_1, F_2, F_3 it solves the 3-XOR problem is at most $n^2 q^2 / 2^n$.

Note that we have kept some margin on the power of n in our conjecture. Until now we have attack with advantage about $nq^2/2^n$. Now we provide a heuristic reduction of full-local

security to the 3-XOR problem. Let us look at the full-local security definition. To break it, we are required to find $(m_1, m_2, m_3, m_4, m_5, m'_1, m'_2, m'_5, d')$ such that

$$T_5(m_1, m_2, m_3, m_4, m_5) = h_3(h_1(m'_1, m'_2) \oplus m'_5, d') \oplus m'_5$$

or equivalently

$$T_5(m_1, m_2, m_3, m_4, m_5) = h_3(c', d') \oplus h_1(m'_1, m'_2) \oplus c'. \quad (10)$$

In our reduction we consider only algorithms which we call *valid-aware*. Those (1) make all queries to h_1, h_2 before h_3 and (2) for any query (c, d) they make to h_3 are aware of all valid T_5 executions that are created this way. Concretely, with any query (c, d) they attach (a potentially empty) list of all pairs h_1, h_2 that are valid with c, d . This list cannot be long as we had argued for Theorem 2 where there cannot be more than n of them. We stress that this is a natural assumption as every valid execution ever considered by an algorithm must be discovered in some way, and if all h_3 queries are made last, it only makes sense to make queries that yield valid executions. The speculative nature of this argument makes us claim that the reduction is only heuristic.

► **Lemma 6.** *Given any valid-aware algorithm A, which makes at most q queries to oracles h_1, h_2, h_3 , which solves (10) with probability ϵ , there is an algorithm A' making at most $q < 2^{n/2}$ queries to oracles F_1, F_2, F_3 (with runtime almost the same as A) that solves 3-XOR problem with probability at least $\epsilon/(4n)$.*

Proof. First we note that a solution to (10) must have $(c, d) \neq (c', d')$, i.e., h_3 gets a non-zero difference. Indeed, otherwise the output e of h_3 and thus m_5 must both have a zero difference, which in turn implies that a and b have a zero difference, i.e., we have found a collision in h_1 or h_2 which we are ruling out (or we can extend our reduction with this outcome).

Now, A works as follows:

- It calls A to find a collision between T_5 and T'_5 .
- When A queries $h_1(m_1, m_2)$ it is given $F_1(0||m_1||m_2)$.
- When A queries $h_2(m_3, m_4)$ it is given $F_1(1||m_3||m_4)$.
- When A queries $h_3(c, d)$ with list L of valid tuples (m_1, m_2, m_3, m_4) such that $h_1 \oplus h_2 = c \oplus d$. If the list is empty, A' returns $F_2(c||d) \oplus c$ to A. Otherwise A' flips a coin:
 - For tails A' selects a random entry of L and returns $F_3(c||d) \oplus c \oplus F_1(0||m_1||m_2)$ to A.
 - For heads A' returns $F_2(c||d) \oplus c$ to A.

Now suppose A finds a solution to (10). This implies that

$$h_1(m_1, m_2) \oplus h_3(c, d) \oplus c = h_1(m'_1, m'_2) \oplus h_3(c', d') \oplus c'.$$

Recall that $(c, d) \neq (c', d')$. Now note that:

- (c, d) yields a valid execution of T_5 . Note that the validity was known at the time of query.
- With probability at least $1/(2n)$ the value for $h_3(c, d)$ is selected as $F_3(c||d) \oplus c \oplus F_1(0||m_1||m_2)$ (i.e., we had selected the same m_1, m_2 as in h_1 of the solution) since we have at most n pairs (h_1, h_2) with given difference $c \oplus d$.
- With probability $1/2$ the value for $h_3(c', d')$ is selected as $F_3(c'||d') \oplus c'$.
- The value for $h_1(m_1, m_2)$ is $F_1(0||m_1||m_2)$ and should cancel out due to our outcome for $h_3(c, d)$.

24:16 T₅: Hashing Five Inputs with Three Compression Calls

Therefore with total probability at least $1/(4n)$ the solution found by A is translated into

$$F_3(c||d) = F_1(0||m'_1||m'_2) \oplus F_2(c'||d')$$

which yields a solution for the 3-XOR problem. ◀

Together with the hardness conjecture, we obtain the following proposition.

► **Proposition 7.** *Assuming the 3-XOR hardness and that the best algorithm that breaks full-local aggressive collision security is valid-aware, the adversary that runs in time q finds a full-local collision with probability at most $4n^3q^2/2^n$.*

Reduction for Local-Local. The best known algorithm for solving 4-XOR problem runs in $O(n2^{n/3})$ time and $O(2^{n/3})$ queries [33]. So, we pose a similar conjecture for the 4-XOR problem.

► **Conjecture 8.** (*4-XOR Hardness.*) *For any algorithm \mathcal{A} that makes $q < 2^{n/3}$ queries to random oracles F_1, F_2, F_3, F_4 , runs for time q , the probability that it solves the 4-XOR problem is at most $q^3/2^n$.*

Now we consider a simple variant of 4-XOR problem, called 4-XOR', which uses two lists. Let F, F' be oracles that output random n -bit strings. Find x, y, z, w with $(x, y) \neq (z, w)$ such that

$$F(x) \oplus F(y) \oplus F'(z) \oplus F'(w) = 0.$$

► **Lemma 9.** *Given any algorithm A' making at most q queries to all its oracles which solves the 4-XOR' problem with probability ϵ there is an algorithm A making at most q queries to all its oracles (with run time almost same as A') which solves 4-XOR problem with probability at least $\epsilon/4$.*

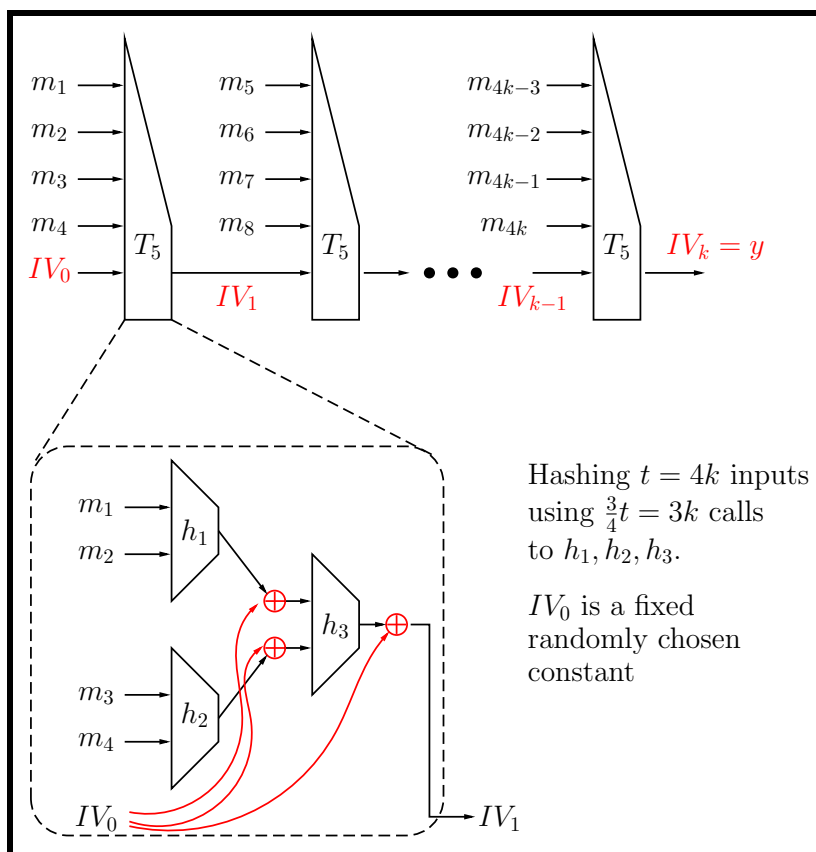
Proof. The reduction from A' to A works as follows. We run A' and it makes two types of queries, namely to F and to F' . For each query we choose b randomly from $\{1, 2\}$. If it is an $F(x)$ query, then A returns $F_b(x)$. Similarly, if it is an $F'(z)$ query, A returns $F_{2+b}(z)$ to A' . Finally, A' returns (x, y, z, w) such that $F(x) \oplus F(y) \oplus F'(z) \oplus F'(w) = 0$. Now, A succeeds if $F(x) = F_b(x)$, $F(y) = F_{3-b}(y)$ and $F'(z) = F_{2+b'}(z)$, $F'(w) = F_{5-b'}(w)$. We note that the b values are chosen randomly. As F and F' are independent random functions, the output to A' is independent of b . In other words, the view of A' remains independent with the b values chosen by A. Thus, A succeeds with probability $1/4$ given that A' succeeds. ◀

We have already seen that given a collision adversary B of T₅' we can construct an algorithm A' for solving the 4-XOR' problem and hence we can construct an algorithm A solving the 4-XOR problem. Moreover the success probability of solving the 4-XOR is at least $\frac{1}{4} \cdot \text{Adv}_{T_5}^{\text{Coll}}(\text{B})$. This leads us to conclude with the following claim.

► **Proposition 10.** *Assuming the 4-XOR hardness and that the best algorithm that breaks local-local aggressive collision security is valid-aware, the adversary that runs in time q finds a local-local collision with probability at most $4q^3/2^n$.*

6 Merkle-Damgård Variant

We can plug in our 5-to-1 compression function to the standard Merkle-Damgård (MD) mode to get a sequential hash t -to-1 function making only $3t/4$ calls to the underlying $2n$ -to- n compression functions h_1, h_2 and h_3 , and inheriting the birthday security of T₅. This function is depicted in Figure 4.



■ **Figure 4** Merkle-Damgård Hash based on T_5 .

Parallel Implementation. In addition to providing a 25% speedup when compared to the traditional MD mode applied to a $2n$ -to- n -compression function h , another advantage of our new variant is that it is easily parallelizable for architectures that support parallel execution.

For example, if we have three execution units P_1, P_2, P_3 , the unit P_i can be responsible for all h_i computations. This allows to compute a hash of $t = 4k$ message blocks using only $(k + 2) = (t/4 + 2)$ parallel rounds of hashing, saving a factor (almost) 4 over the traditional MD mode. The execution unit P_3 will be “one-round behind” units P_1 and P_2 (so that in the first round only P_1 and P_2 hash (m_1, m_2) and (m_3, m_4) , and in the last round only P_3 produces the final hash). Namely, when P_3 just completed the computation of the previous initial value IV_j , P_1 completed $a_j = h_1(m_{4j+1}, m_{4j+2})$, and P_2 completed $b_j = h_2(m_{4j+3}, m_{4j+4})$, in the next round P_3 will set the next initial value

$$IV_{j+1} = h_3(IV_j \oplus a, IV_j \oplus b) \oplus IV_j,$$

while P_1 and P_2 respectively compute $a_{j+1} = h_1(m_{4j+5}, m_{4j+6})$ and $b_{j+1} = h_2(m_{4j+7}, m_{4j+8})$.

Similarly, two execution units P'_1, P'_2 can perform the same task in only $2k = t/2$ parallel rounds, saving a factor 2 over the traditional MD mode.

Larger Compression. Although our results are stated for $2n$ -to- n compression functions, we can also easily extend them to the λn -to- n case, by simply adding $(\lambda - 2)$ “dummy” inputs to each of h_1, h_2, h_3 . For example, for $\lambda = 3$, this gives us an $8n$ -to- n analog of T_5 , using

three calls to some $3n$ -to- n hash function. Which gives a new MD mode for $t = 7k$ block messages, using only $3k = 3t/7$ compression calls. In contrast, a naive MD mode will use $t/2$ calls, saving a factor $(1 - (3t/7)/(t/2)) = 1/7 \approx 14.2\%$. Similar calculations can be done for larger λ .

7 Merkle Tree Variant

Our 2-level construction extends straightforwardly to a full-blown t -to-1 tree H . In Section 7.1 we briefly recall how to build Merkle Trees (MT) from smaller compression functions (such as T₅). We then present our faster and shallower MT variant and its properties in Section 7.2.

7.1 General Merkle Trees and Their Security

The Merkle tree is a data structure to store long lists of t n -bit elements, so that insertion, deletion, update, or proof of membership for an element need only $O(\log t)$ time and space. It is built on a λn -to- n compression function H_λ (typically $\lambda = 2$ but other values are used too) and retains all its security properties regarding collision and preimage resistance from the compression function. The tree is defined recursively:

$$\begin{aligned} \text{MT}_\lambda(\underbrace{m_1, \dots, m_\lambda}_{m_{[1:\lambda]}}) &= H_\lambda(m_1, \dots, m_\lambda); \\ \text{MT}_{\lambda t}(m_{[1:\lambda t]}) &= H_\lambda(\text{MT}_\lambda(m_{[1:\lambda]}), \dots, \text{MT}_\lambda(m_{[\lambda t - t + 1:\lambda t]})) \end{aligned}$$

where $t = \lambda^k \geq 2$, with the last hash called a root and m_i called leafs.

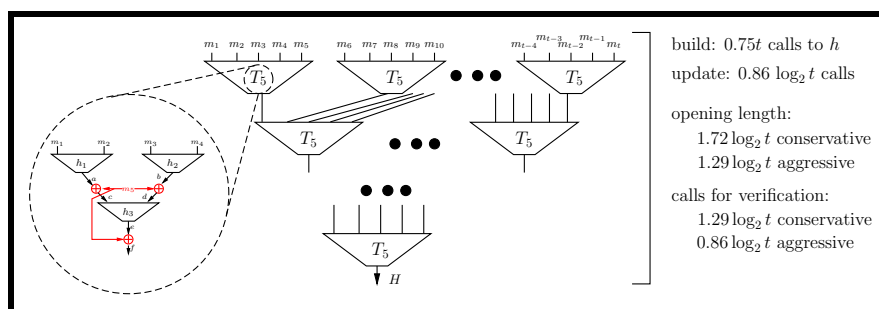
As for the compression function, we define the terms full opening and local opening for the entire MT, with the former being all t elements and the latter for a leaf in a tree is a sequence of $\log_\lambda t$ local openings of an element in all compression functions on the path from the leaf to the root. In the simplest case $\lambda = 2$ an opening is one element per tree layer, whereas for wider compression functions it is $\lambda - 1$ or fewer (in case H_λ has “aggressive” local opening). The full-local security for the tree is defined analogously to the compression function and corresponds to the case of a public tree to which an adversary makes a forged membership proof. The local-local security matches the case when the adversary provides two valid openings for an alleged tree root but the full tree is unknown. Both full-local and local-local security for the tree follows from their compression function counterparts. So overall we have the following parameters:

- Efficiency $E(t)$ as the number of compression function calls is $(t - 1)/(\lambda - 1)$.
- Depth $D(t)$ of the tree is $\log_\lambda t$.
- Update/insert/delete complexity as the number of compression function recomputations equals $D(t)$.
- The total length of *conservative* opening $L(t)$ is $(\lambda - 1) \log_\lambda t$.
- In case H_λ might have a more compact (i.e., “aggressive”) local opening of length $\ell \leq \lambda - 1$, then the total length of the resulting “aggressive” local opening for MT_λ becomes $L(t) = \ell \log_\lambda t$.
- Number of calls to F needed to verify the opening: $V(t) = \log_\lambda t$.
- Collision, full-local, and local-local opening security the same as that of H_λ (ideally $2^{n/2}$, if one uses a $2n$ -to- n hash function as last building block).
- Preimage security the same as that of H_λ (ideally 2^n , if one uses $2n$ -to- n hash function as last building block).

■ **Table 1** We compare standard Merkle trees to two variants of Merkle trees with T_5 . The conservative variant requires opening four siblings in a local opening proof, compared to three siblings in the aggressive variant. The boxed formulas are conjectures based on the 3-XOR and 4-XOR problems. We have $2/\log_2 5 \approx 0.86$, $3/\log_2 5 \approx 1.29$, $4/\log_2 5 \approx 1.72$. Note that full-full collision resistance (CR) security is listed for completeness, this is the “traditional” collision resistance involving two distinct messages (and the entire corresponding Merkle trees).

	Standard Merkle	Merkle with T_5 (conservative)	Merkle with T_5 (aggressive)
build calls/ t	1	0.75	0.75
update/ $\log_2 t$	1	0.86	0.86
verify/ $\log_2 t$	1	1.29	0.86
opening/ $\log_2 t$	1	1.72	1.29
full-full CR security	$n/2$	$n/2$	$n/2$
full-local CR security	$n/2$	$n/2$	$n/3 \rightarrow \boxed{n/2}$
local-local CR security	$n/2$	$n/2$	$n/4 \rightarrow \boxed{n/3}$

7.2 Faster and Shallower Merkle Tree based on T_5



■ **Figure 5** Full-blown tree based on T_5 . Security is $2^{n/2}$ (tight) for conservative openings, $2^{n/3}$ provable and $2^{n/2}$ heuristic for full-local aggressive opening, $2^{n/4}$ provable and $2^{n/3}$ heuristic for local-local aggressive opening.

Our construction extends straightforwardly to a full-blown t -to-1 tree H of any depth k , as depicted in Figure 5, with optimal $t = 5^k$. Notice, unlike our compression function T_5 for H , which needed domain separation for functions h_1, h_2, h_3 (see the full version of this paper [11]), the final tree H can reuse the same h_1, h_2, h_3 across all invocations, which follows from general security properties of standard Merkle trees.

The overall trade-offs of our construction are summarized in Table 1, but we expand on it below.

Efficiency. Let us summarize the performance of our tree construction:

- **Build:** In H we compress every 5 inputs using our $5n$ -to- n compression function T_5 , therefore using $(t - 1)/4$ calls to T_5 , which is equivalent to

$$E(t) = 0.75(t - 1)$$

24:20 T₅: Hashing Five Inputs with Three Compression Calls

calls to the h_j 's, thus giving us 25% improvement over the regular Merkle tree.

- **Depth/update:** The depth $D(t)$ of the tree, measured in the calls to the h_j 's, reduces from $\log_2 t$ to

$$D(t) = 2 \log_5 t \approx 0.86 \log_2 t$$

which is a 14% saving compared to standard Merkle trees.

- **Opening length (conservative):** In the conservative opening we open all 4 siblings, thus $L(t)$ increases from $\log_2 t$ of standard Merkle trees to $4 \log_5 t \approx 1.72 \log_2 t$ (loss of 72%). It is still useful though when bandwidth is not critical which is sometimes the case.
- **Opening length (aggressive):** In the aggressive opening we open 3 elements, thus $L(t)$ increases to only $3 \log_5 t \approx 1.29 \log_2 t$ (loss of 29%, which may be tolerated for some applications).
- **Verification time (conservative):** the number of h_i calls needed to verify the proof increases to $V(t) = 3 \log_5 t \approx 1.29 \log_2 t$.
- **Verification time (aggressive):** the number of h_i calls needed to verify the proof decreases to $V(t) = 2 \log_5 t \approx 0.86 \log_2 t$. This is very handy in applications when opening verification time is crucial (such as zero-knowledge membership proofs where proving time linearly depend on the circuit size needed for the opening verification). However we pay for this with security (see below).

Security. Since our construction applies the standard Merkle paradigm to the 5-to-1 compression function T₅, the resulting hash function H inherits the same global (or local opening) collision and preimage security as the compression function T₅:

- **Full-local (aggressive) security:** provable security up to (taking $poly(n)$ factors aside) $2^{n/3}$ queries (Theorem 4), heuristic security up to $2^{n/2}$ running time (Proposition 7).
- **Local-local (aggressive) security:** provable security up to $2^{n/4}$ queries (Theorem 4), heuristic security up to $2^{n/3}$ running time (Proposition 10).
- **Both full-local and local-local (conservative) security:** provable security up to $2^{n/2}$ queries (Theorem 2).

Additionally, non-trivial preimage security holds up to $2^{2n/3}$ queries (and at optimal level $O(q/2^n)$, in the region of interest where $q \leq 2^{n/2}$).

Applications. Merkle trees are used in a number of protocols, but their exhaustive list is beyond the scope of this paper. To name just a few: anonymous cryptocurrencies and mixers [14, 25], interactive oracle proof (IOP) compilers [2, 3], and post-quantum hash-based signatures [5]. Among them, cryptocurrencies and mixers are the examples of publicly controlled Merkle trees, i.e., where the full-local opening makes sense and where the aggressive opening with its improved verification time is appealing.

From all this diverse range of applications, the ones who benefit also from the conservative opening strategy are those for which the performance and depth are more important than the local opening size. Of course, basic hashing by itself is a very important example of such an application. The next examples are zero-knowledge proof systems where circuit depth is a major performance factor [34, 35]. Finally, a more speculative area where our construction could help is multiparty computation protocols applied to functionalities involving hashing, whose complexity depends on the circuit depth (e.g., variants of the original BGW [1] and GMW [12] protocols).

Summary. Our construction has clear advantage over the regular Merkle tree in build and update efficiency, but loses in the opening length. For the verification time and security we have a tradeoff: an aggressive opening needs fewer h_i calls but only heuristic security argument of $2^{n/2}$ with provable security reaching $2^{n/3}$ only, whereas the conservative opening has the same security properties as in the regular tree but requires 30% more verification calls. Thus whether or not our construction outperforms the regular Merkle tree depends on the setting.

8 Generalizations and Future Work

Our construction is likely to be extended to wider compression functions. For example, a natural generalization of our construction can hash $T = 3 \cdot 2^k - 1$ inputs using only $E = 2 \cdot 2^k - 1 = (2T - 1)/3$ evaluations, where standard $2n$ -to- n hash h corresponds to $k = 0$, and our T_5 corresponds to $k = 1$. As k increases, $E(k)/T(k) \rightarrow 2/3$, which matches Stam's bound for building Tn -to- n hash functions from $2n$ -to- n compression functions. Unfortunately, as k grows, the local opening size also grows, so it is unclear if this overall hash saving is worthwhile for applications. We leave the security analysis of this, and other optimized hashing constructions to future work.

Finally, it remains to prove the reduction of the full-local aggressive security to the 3-XOR problem unconditionally, i.e., without restrictions on the adversary.

References

- 1 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10. ACM, 1988.
- 2 Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon interactive oracle proofs of proximity. In *ICALP*, volume 107 of *LIPICs*, pages 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.
- 3 Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *TCC (B2)*, volume 9986 of *Lecture Notes in Computer Science*, pages 31–60, 2016.
- 4 Piotr Berman, Marek Karpinski, and Yakov Nekrich. Optimal trade-off for merkle tree traversal. *Theor. Comput. Sci.*, 372(1):26–36, 2007.
- 5 Daniel J Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O’Hearn. SPHINCS: practical stateless hash-based signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 368–397. Springer, 2015.
- 6 John Black, Martin Cochran, and Thomas Shrimpton. On the impossibility of highly-efficient blockcipher-based hash functions. In *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 526–541. Springer, 2005.
- 7 John Black, Martin Cochran, and Thomas Shrimpton. On the impossibility of highly-efficient blockcipher-based hash functions. *J. Cryptol.*, 22(3):311–329, 2009.
- 8 Charles Bouillaguet, Claire Delaplace, and Pierre-Alain Fouque. Revisiting and improving algorithms for the 3xor problem. *IACR Trans. Symmetric Cryptol.*, 2018(1):254–276, 2018.
- 9 Larry Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.
- 10 Ivan Damgård. A design principle for hash functions. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.
- 11 Yevgeniy Dodis, Dmitry Khovratovich, Nicky Mouha, and Mridul Nandi. T5: hashing five inputs with three compression calls. *IACR Cryptology ePrint Arch.*, 2021:373, 2021. Full version of this paper.

24:22 T₅: Hashing Five Inputs with Three Compression Calls

- 12 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.
- 13 Iftach Haitner, Yuval Ishai, Eran Omri, and Ronen Shaltiel. Parallel hashing via list recoverability. In *CRYPTO*, volume 9216 of *Lecture Notes in Computer Science*, pages 173–190. Springer, 2015.
- 14 Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification: Version 2019.0-beta-37 [overwinter+sapling]. Technical report, Zerocoin Electric Coin Company, 2019. available at <https://github.com/zcash/zips/blob/master/protocol/protocol.pdf>.
- 15 Markus Jakobsson, Frank Thomson Leighton, Silvio Micali, and Michael Szydlo. Fractal merkle tree representation and traversal. In *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 314–326. Springer, 2003.
- 16 Lars R. Knudsen and Bart Preneel. Construction of secure and fast hash functions using nonbinary error-correcting codes. *IEEE Trans. Inf. Theory*, 48(9):2524–2539, 2002.
- 17 Ian McQuoid, Trevor Swope, and Mike Rosulek. Characterizing collision and second-preimage resistance in linicrypt. In Dennis Hofheinz and Alon Rosen, editors, *TCC*, volume 11891 of *Lecture Notes in Computer Science*, pages 451–470. Springer, 2019.
- 18 Bart Mennink and Bart Preneel. Hash functions based on three permutations: A generic security analysis. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 330–347. Springer, 2012.
- 19 Bart Mennink and Bart Preneel. Efficient parallelizable hashing using small non-compressing primitives. *Int. J. Inf. Sec.*, 15(3):285–300, 2016.
- 20 Ralph C. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and Privacy*, pages 122–134. IEEE Computer Society, 1980.
- 21 Ralph C. Merkle. One way hash functions and DES. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.
- 22 Mridul Nandi, Wonil Lee, Kouichi Sakurai, and Sangjin Lee. Security analysis of a 2/3-rate double length compression function in the black-box model. In *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 243–254. Springer, 2005.
- 23 Ivica Nikolic and Yu Sasaki. Refinements of the k-tree algorithm for the generalized birthday problem. In *ASIACRYPT*, volume 9453 of *Lecture Notes in Computer Science*, pages 683–703. Springer, 2015.
- 24 Mihai Patrascu and Mikkel Thorup. The power of simple tabulation hashing. *J. ACM*, 59(3):14:1–14:50, 2012.
- 25 Alexey Pertsev, Roman Semenov, and Roman Storm. Tornado cash privacy solution version 1.4, 2019. URL: https://tornado.cash/Tornado.cash_whitepaper_v1.4.pdf.
- 26 Thomas Peyrin, Henri Gilbert, Frédéric Muller, and Matthew J. B. Robshaw. Combining compression functions and block cipher-based hash functions. In *ASIACRYPT*, volume 4284, pages 315–331. Springer, 2006.
- 27 Phillip Rogaway and John P. Steinberger. Security/efficiency tradeoffs for permutation-based hashing. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 220–236. Springer, 2008.
- 28 Yannick Seurin and Thomas Peyrin. Security analysis of constructions combining FIL random oracles. In *FSE*, volume 4593, pages 119–136. Springer, 2007.
- 29 Martijn Stam. Beyond uniformity: Better security/efficiency tradeoffs for compression functions. In *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 397–412. Springer, 2008.
- 30 John P. Steinberger. Stam’s collision resistance conjecture. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 597–615. Springer, 2010.
- 31 John P. Steinberger, Xiaoming Sun, and Zhe Yang. Stam’s conjecture and threshold phenomena in collision resistance. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 384–405. Springer, 2012.

- 32 Michael Szydlo. Merkle tree traversal in log space and time. In *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 541–554. Springer, 2004.
- 33 David A. Wagner. A generalized birthday problem. In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.
- 34 Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society, 2018.
- 35 Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In *CRYPTO*, volume 11694 of *Lecture Notes in Computer Science*, pages 733–764. Springer, 2019.

Post-Compromise Security in Self-Encryption

Gwangbae Choi

Fasoo, Seoul, Korea

F. Betül Durak

Robert Bosch LLC – Research and Technology Center, Pittsburgh PA, USA

Serge Vaudenay

Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

Abstract

In self-encryption, a device encrypts some piece of information for itself to decrypt in the future. We are interested in security of self-encryption when the state occasionally leaks. Applications that use self-encryption include cloud storage, when a client encrypts files to be stored, and in 0-RTT session resumptions, when a server encrypts a resumption key to be kept by the client. Previous works focused on forward security and resistance to replay attacks. In our work, we study post-compromise security (PCS). PCS was achieved in ratcheted instant messaging schemes, at the price of having an inflating state size. An open question was whether state inflation was necessary. In our results, we prove that post-compromise security implies a super-linear state size in terms of the number of active ciphertexts which can still be decrypted. We apply our result to self-encryption for cloud storage, 0-RTT session resumption, and secure messaging. We further show how to construct a secure scheme matching our bound on the state size up to a constant factor.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases Encryption, Ratchet, Post-Compromise Security, Instant Messaging, Session Resumption, Cloud Storage

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.25

Acknowledgements We deeply thank the anonymous reviewers for their valuable comments.

1 Introduction

In many deployed applications, the design of the application involves various devices communicating with each other securely. They sometimes require one of the devices to encrypt some piece of information that will be used in the future by itself. We call this *self-encryption*. One application is massive client-server connections where millions of clients connect to a server, causing the server being unable to afford to store any client-specific information. On the other hand, recent protocols such as TLS 1.3 offers an alternate way to make the server to resume past sessions without going through a new round-trip handshake when a client reconnects to the server.¹ While clients would surely benefit from a smooth connection experience, the server has to “remember” each session in a secure manner, possibly by keeping a (small or big) size of state. More precisely, when a client connects to a website for the first time, the web server generates a ticket for the client. This ticket is a piece of information that helps the server to remember the session. Somehow, this is a helper that the server encrypts for itself which is to be kept by the client like cookies. When the client reconnects to the same website with her ticket, the server may use the information contained in the ticket to resume their session. As desired, it gives the freedom not to store any client-specific information on the server-side. However, the server needs a secret state for

¹ As of November 2019, 34% of TLS connections use session resumption [11].



25:2 Post-Compromise Security in Self-Encryption

the cryptographic operations which are used in generating and decrypting tickets. From the security point of view, then, the concern becomes to provide security against replay attacks or occasional exposures of the internal state of the server.

In general, the internal state is any type of information that would let a device decrypt (some part of) the communication. In this work, we investigate the security of self-encryption which comes in two forms: forward security (FS) and post-compromise security (PCS). Intuitively, forward security provides security for the *past* communication when exposure happens, whereas post-compromise security aims to heal the *future* communication when exposure occurs [6]. Before going forward with security, we list three applications.

0-RTT in TLS 1.3

In the TLS 1.3 protocol, a client connects to a server and establishes a common secret key through a handshake key agreement protocol. This is succeeded with a full round trip time (1-RTT) communication. Ideally, when the client reconnects to the same server after a while, the connection should be resumed with no round trip time (0-RTT). 0-RTT has been an active research domain in the last few years [2, 7, 10]. It is achieved in practice through two elementary approaches called *session caches* and *session tickets* as described by Aviram, Gellert, and Jager (AGJ) [2]. In the former technique, the server resumes the session by assigning a different resumption key for each connection and sending the client a look-up index that links to the resumption key. The ticket is that index. When the client comes back, it includes the ticket and the payload data. This provides forward security. Nevertheless, the solution depends on maintaining a big database on the server, which is not alluring.

The other approach for 0-RTT in TLS 1.3 configurations is to create session tickets for each client by using a long-term secret key K (the ticket encryption key). Therefore, instead of storing a unique key for each session, the server generates a secret material for each client and encrypts it under K . The secret material is called resumption key whereas the encrypted resumption key is the ticket. The client stores both the resumption key and the ticket. Later on, the client encrypts the payload with the resumption key and includes her ticket in 0-RTT message to remind herself. The server can decrypt the ticket with K and retrieve the resumption secret to decrypt the payload. This approach avoids storing a big database; it is easy to implement and to integrate in existing systems, yet, it does not provide any kind of security in the case of a key exposure.²

In their recent work, Aviram, Gellert, and Jager (AGJ) [2] studied the forward security and the resistance to replay attacks of session resumption, specifically focusing on session tickets. However, they did not consider PCS in their security model.

Cloud Storage

In a single client-server cloud storage, the client wants to outsource her files in a remote storage (cloud) in an encrypted form. The encryption of the files occurs locally on the client who keeps the secret decryption material. The adversary has full access to the cloud and can also keep archives of removed storage. If the client encrypts all files with the same key, the leakage of the key becomes catastrophic as all files (even the removed ones) become compromised. Besides, the client aims to minimize the storage on her local while maintaining

² In TLS 1.3, it is considered good practice to rotate the long-term key K every few hours by assuming that all the clients will resume their sessions in the “life-time” of K . Nevertheless, as soon as the key K is compromised, there is neither FS nor PCS during the active period of K .

strong security in case of a compromise of her internal state. This cloud storage problem shares similarities with the 0-RTT problem: the cloud client and the 0-RTT server want to minimize their storage while conserving security.

On the other hand, keys should not be used more than what the encryption method can guarantee to be secure or age too long. This is part of a common good practice in key management. Regulations actually mandate the encrypted files to be updated from an old key to a new key often enough. This is called *key rotation*. The fundamental motivation, however, comes with the desire to achieve resilience to key exposure. Key rotation was formally studied by Boneh et al. [3]. More recently, Everspaugh et al. [9] considered the integrity problem with key rotation.

The naive way to achieve key rotation is to make the client download the encrypted files on the local, decrypt them with the existing key, generate a new fresh key, re-encrypt, and finally outsource back. However, it is a very cumbersome solution for the client. The main task of key rotation is to avoid the complexity of communication and the complexity of treatment on the client side. In practice, AWS and Google deploy a more practical methods based on hybrid encryption: a header $ct_1 = \text{Enc}_K(\text{eph})$ is formed by encrypting an ephemeral key eph and the rest of the ciphertext $ct_2 = \text{Enc}_{\text{eph}}(\text{pt})$ is formed by encrypting the plaintext pt using eph . Key rotation is done by updating the header as $ct'_1 = \text{Enc}_{K'}(\text{eph})$ but keeping the same ephemeral key so $ct'_2 = ct_2$. This was argued to be a bit cheating with the concept of key rotation as the encryption of data under the same key was remaining in ct_2 .

We tackle the privacy problem differently. Instead of updating a ciphertext to be decryptable with a chosen key, we let ciphertexts unchanged but update the state which is stored by the client³. Naturally, our concern becomes more focused on the storage space on the client side. In our setting, the client stores one state and needs no operation on ciphertexts.

Instant Messaging

Post-compromise security in instant messaging was formally studied during the last few years [14, 12, 13, 1, 8]. It is addressed by the notion of *ratchet*. A ratchet consists of updating a key in a one-way manner (for FS) by using some unpredictable randomness (for PCS). Bidirectional secure communication applications can be transformed into self-encryption. In fact, roughly speaking, we can merge both participants into one single device which would encrypt for itself. A *ratcheted* scheme is normally FS and PCS secure, hence defines an FS and PCS secure self encryption which we call a *self-ratchet*.

Our Perspective

In order to study the security of self-encryption, we consider a scheme which generates ciphertexts with the ability to decrypt later, even when the state to decrypt evolves. We define it in a way that it covers the three (and potentially more) applications we described earlier. Furthermore, we are interested in forward security and post-compromise security of these systems. The former captures that the system generates ciphertexts that should remain decryptable for a limited time and that are not going to be decryptable anymore after they “expire” (it could happen either because the settings allow the ciphertexts to stay alive for a limited time or because there is an inherent latency to rotate keys). The ciphertexts that are still decryptable are called *active ciphertexts*. Making a ciphertext become inactive

³ We do not mean to pick a fresh key to “rotate” the key and update the header as practiced by AWS.

is a way to have forward security: if the state of the scheme is exposed after a ciphertext becomes inactive, this ciphertext is still safe. The PCS defines what happens to the security *after* an exposure of a state. When an exposure takes place, the post-compromise secure system should be able to heal the state such that the ciphertexts which are generated after the healing are secure. In many studies, PCS is interchangeably used with *healing*.

While studying self-ratcheted schemes with PCS guarantees (as well as FS), it was intuitive to expect that the state size of any post-compromise secure self-ratcheted scheme will grow because decryption keys would need to be independent. However, it was not clear why and with what bounds we could achieve it. The first contribution of our work is to show that we cannot achieve post-compromise security better than adding a trivial solution to already existing efficient FS schemes.

As for forward security, AGJ [2] specifically consider the session resumption in TLS 1.3 and they designed solutions for FS and replay attacks without PCS. Their construction is practical. In another study by Günther et al. [10] and Derler et al. [7], the authors consider a solution without any shared secret. In these works, the clients resume connections without having to store any session-specific information on her local. The client keeps only the long-term public key pk of the server. Therefore, they look for forward-secure solutions when the long-term secret key sk evolves throughout time although the associated public key never changes, hence the clients never updates its state. Although it is remarkable that such schemes with forward security exist, both constructions are less practical due to the heavy cryptographic tools they use. Therefore, we rather focus on the FS scheme AGJ to add PCS.

In their seminal paper on PCS, Cohn-Gordon, Cremers, and Garratt [6] focus on Authenticated Key Exchange (AKE). In AKE, the protocol starts with a state and ends when both participants have obtained the exchanged key. The typical exposure threats happen before or after the protocol but not during it because the protocol is rather short. The AKE protocol proposed by Cohn-Gordon et al. [6] requires to store nonces and ephemeral secrets during the execution, which inflate the state. Deflation happens when the protocol is fully complete. In our perspective (and specially about instance messaging), communication is asynchronous and it can take some time before a protocol fully terminates. Hence, there is the case when several protocols run concurrently. This is the case where the state would grow with the number of incomplete sessions, just like in the instance messaging case (which we illustrate on Fig. 15).

Our Contribution

In the present work, we start with the definition of a minimal primitive called Self-Encrypted Queue (SEQ) with correctness and one-way (OW) security. It gives the minimal functionality for any PCS construction, more particularly self-encryption schemes. Then, we prove that for every SEQ primitive with states of bounded length, there is an adversary with small complexity and high probability of success to break OW security. More precisely, the probability of success is at least $\frac{1}{4n} 2^{-2 \lceil \frac{\ell+1}{n/\Delta} \rceil}$ when the state size is bounded by ℓ , n is the number of active ciphertexts, and $\Delta = 1$ (or defined below). This result led us to conclude that when self-encryption is post-compromise secure, it must have a state which grows more than linearly in n .⁴ This does not provide the practicality we were hoping for. Therefore, we define a refinement which is a relaxed version of post-compromise security. In layman

⁴ It grows linearly if we take the key size as a memory unit. (The key size cannot have a constant bit length. Otherwise, exhaustive search breaks it with constant complexity.)

terms, we look into the following case: Maybe the first ciphertext that will be generated after an exposure is not secure, but the system could be designed to heal the security after the generation of Δ ciphertexts, where Δ is a constant parameter of our scheme. We call it Δ -PCS. We show that in refined definitions, the state size is super-linear in $\frac{n}{\Delta}$ as opposed to growing super-linear in n .

We prove that this impossibility result applies both in self-encryption and in secure messaging. In addition to this, we prove that this result is tight by constructing a simple self-encryption scheme achieving Δ -PCS with a state size matching our bounds.

After our impossibility results, we focus on few applications by borrowing already existing formal interfaces from AGJ [2] in order to add PCS security in the discussed settings. We modify the interface in a way that decryption and puncturing happens with separate function calls in case the puncturing is not always necessary. Later on, we look at secure ratcheted protocols which provides PCS security from the literature. We show that the state of these protocols grows linearly (in terms of number of keys) as they “ratchet” every time a new message is generated, hence falling into the case where $\Delta = 1$. On the other hand, we have two secure communication protocols given by Alwen, Coretti, and Dodis (called ACD and ACD-PK) [1] which model well what Signal is deploying. We observe that the state in both schemes does not grow linearly like other PCS schemes. This is due to the fact that these two protocols do not guarantee Δ -PCS for any constant Δ . In fact, healing happens only when the direction of communication changes.

We conclude that adding PCS to FS-secure systems can be succeeded at the price of a minimal state growth with proven bounds and we cannot hope for better.

Structure of the Paper

In Section 2, we define a basic PCS-secure primitive called SEQ and we prove that its state size must grow super-linearly. In Section 3, we apply this result to self-encryption. We construct a scheme based on AGJ with super-linear growth and PCS security. Finally, in Section 4, we show how to apply our result to instant secure messaging.

2 Impossibility Result

In this section, we first define a minimal primitive called Self Encrypted Queue (SEQ) achieving post-compromise security. This primitive is not meant to have any concrete application. However, we will prove that (examples of) useful primitives imply SEQ, and that SEQ must have a linearly growing state.

2.1 Definition of a Minimal Primitive

We define below a *minimal* primitive which works in two phases: It iteratively generates a sequence of plaintext/ciphertext pairs (pt, ct) by updating its state. Then, it takes the sequence of ct in the same order as generated and recovers the exact sequence of pt . The primitive is minimal in the sense that all considered applications which claim PCS must achieve this functionality and even more (such as being able to receive the list of ct in different order, or to have encryption and decryption steps mixed up). We build a limited self-encryption (actually, we build a KEM) which we call a SEQ.

► **Definition 1 (SEQ).** A *Self Encrypted Queue (SEQ)* is a primitive defined by $\text{Gen}(1^\lambda) \rightarrow \text{st}$ which generates an initial state;

25:6 Post-Compromise Security in Self-Encryption

Correctness at level- n :	Game $\text{OW}_{m,\Delta,\lambda}(\mathcal{A})$:
1: $\text{Gen}(1^\lambda) \rightarrow \text{st}_0$	1: $\text{Gen}(1^\lambda) \rightarrow \text{st}_0$
2: for $i = 1$ to n do \triangleright fill up the queue	2: for $i = 1$ to m do
3: $\text{Enc}(\text{st}_{i-1}) \rightarrow (\text{st}_i, \text{pt}_i, \text{ct}_i)$	3: $\text{Enc}(\text{st}_{i-1}) \rightarrow (\text{st}_i, \text{pt}_i, \text{ct}_i)$
4: end for	4: end for
5: for $i = 1$ to n do \triangleright empty the queue	5: $\mathcal{A}(1^\lambda, \text{st}_{m-\Delta}, \text{ct}_1, \dots, \text{ct}_m) \rightarrow z$
6: $\text{Dec}(\text{st}_{n+i-1}, \text{ct}_i) \rightarrow (\text{st}_{n+i}, \text{pt}'_i)$	6: return $1_{z=\text{pt}_m}$
7: if $\text{pt}_i \neq \text{pt}'_i$ then return 0	
8: end for	
9: return 1	

■ **Figure 1** Correctness and OW games for SEQ.

$\text{Enc}(\text{st}) \rightarrow (\text{st}', \text{pt}, \text{ct})$ which updates the state and adds to the queue a new message which is pt in clear and ct in encrypted form;

$\text{Dec}(\text{st}, \text{ct}) \rightarrow (\text{st}', \text{pt}/\perp)$ which updates the state and decrypts ct which leads the queue. This is deterministic.

We say that SEQ is **correct to level- n** if the correctness game in Fig. 1 always return 1.⁵

The principle of this primitive is that a state is updated at every encryption/decryption so that the new state can decrypt the released ciphertext in the order they have been released. In the correctness game, the queue is filled up with $(\text{ct}_1, \dots, \text{ct}_n)$, then emptied.

► **Definition 2.** Let $n(\lambda)$ and $\Delta(\lambda)$ be polynomially bounded positive integer functions of a security parameter λ . We consider the $\text{OW}_{m,\Delta,\lambda}$ game in Fig. 1. We say that **SEQ with level n is Δ -secure** if for any PPT adversary \mathcal{A} , $\lambda \mapsto \max_{1 \leq m \leq n} \Pr[\text{OW}_{m,\Delta,\lambda}(\mathcal{A}) \rightarrow 1]$ is a negligible function.

The value of Δ represents the time the scheme needs to heal security after an exposure. This means that Δ steps after exposing the state, the new state has become safe again and the encryptions to follow will protect confidentiality. In the game, $\text{st}_{m-\Delta}$ is exposed and the goal of the adversary is to decrypt ct_m . Most secure schemes are 1-secure, because security heals after $\Delta = 1$ encryption.

It is easy to design a secure SEQ of level n with a state with $\mathcal{O}(n)$ keys inside. For instance, for any n , the scheme in Fig. 2 is a 1-secure SEQ to level n with state of size $n\lambda$, where λ is the security parameter. This SEQ is trivially correct: st accumulates all pt in a queue during encryption and releases them during decryption. It is also perfectly secure: pt is independent from the corresponding ct and from the previous states. Hence, any $\text{OW}_{m,\Delta,\lambda}$ adversary has an advantage of $2^{-\lambda}$.

Ideally, states should not inflate. For that, one can count on ct to transport a helper to recover pt without having to store it in st . However, we prove next that a correct and OW-secure SEQ primitive with st in a space \mathcal{ST} of size $2^{o(n \log n)}$ does not exist.

2.2 Impossibility Result

► **Theorem 3.** There exists a (small) constant c such that for every probability $\alpha \in]0, 1]$ and integers $\lambda, n, \ell, \Delta, k$, for every correct SEQ primitive of level n as in Def. 1 with st in a

⁵ Throughout this paper, 1_P denotes a function returning 1 if the predicate P is true, and 0 otherwise.

<p>Gen(1^λ):</p> <p>1: $\text{st} \leftarrow (\lambda, \emptyset)$</p> <p>2: return st</p> <p>Enc(st):</p> <p>3: parse $\text{st} = (\lambda, L)$</p> <p>4: pick pt of length λ at random</p> <p>5: $L \leftarrow (L, \text{pt})$</p> <p>6: $\text{st} \leftarrow (\lambda, L)$</p> <p>7: $\text{ct} \leftarrow \perp$</p> <p>8: return $(\text{st}, \text{pt}, \text{ct})$</p>	<p>\triangleright a list of length 0</p> <p>\triangleright append pt in L</p>	<p>Dec(st, ct):</p> <p>9: parse $\text{st} = (\lambda, L)$</p> <p>10: parse $L = (\text{pt}, L')$</p> <p>\triangleright pt is the first length-λ element of st</p> <p>11: $\text{st} \leftarrow (\lambda, L')$</p> <p>12: return (st, pt)</p>
---	--	--

■ **Figure 2** A trivial SEQ.

space \mathcal{ST} of size $|\mathcal{ST}| \leq 2^\ell$, there exist $m \leq n$ and an $\text{OW}_{m,\Delta,\lambda}$ adversary \mathcal{A} of complexity $(n - m + \Delta)T_{\text{Enc}} + mT_{\text{Dec}} + c$, and advantage at least

$$\Pr[\text{OW}_{m,\Delta,\lambda}(\mathcal{A}) \rightarrow 1] > \frac{\alpha}{n} \left(1 - \left(\frac{1}{k} + \frac{k-1}{2} \alpha \right)^{\lfloor \frac{n}{\Delta} \rfloor} 2^\ell \right)$$

where T_{Enc} and T_{Dec} are the complexities of **Enc** and **Dec**.

Interestingly, for $k = 2$ and $\alpha = \frac{1}{\lfloor n/\Delta \rfloor}$, this theorem gives $\Pr[\text{OW}_{m,\Delta,\lambda}(\mathcal{A}) \rightarrow 1] > \frac{\Delta}{n^2} (1 - e^{2^\ell - \lfloor \frac{n}{\Delta} \rfloor})$. Thus, it is clear that $\ell \leq \lfloor \frac{n}{\Delta} \rfloor - 2$ is insecure.

We can be more precise and obtain insecurity when $\frac{\ell\Delta}{n}$ is bounded by a logarithmic term (of the security parameter). Let $\varepsilon = 2^{-\frac{\ell+1}{\lfloor n/\Delta \rfloor}}$. Th. 3 with $\alpha = \frac{\varepsilon^2}{2}$ and $k = \lceil \frac{2}{\varepsilon} \rceil$ gives the following result:

► **Corollary 4.** *There exists a (small) constant c such that for every integers λ , n , ℓ , and Δ , for every correct SEQ primitive of level n as in Def. 1 with st in a space \mathcal{ST} of size $|\mathcal{ST}| \leq 2^\ell$, there exist $m \leq n$ and an $\text{OW}_{m,\Delta,\lambda}$ adversary \mathcal{A} of complexity $(n - m + \Delta)T_{\text{Enc}} + mT_{\text{Dec}} + c$, and advantage at least*

$$\Pr[\text{OW}_{m,\Delta,\lambda}(\mathcal{A}) \rightarrow 1] > \frac{1}{4n} 2^{-2 \frac{\ell+1}{\lfloor n/\Delta \rfloor}}$$

where T_{Enc} and T_{Dec} are the complexities of **Enc** and **Dec**.

This means that the state needs a size ℓ such that

$$\ell > \frac{1}{2} \left\lfloor \frac{n}{\Delta} \right\rfloor \log_2 \frac{1}{4n\varepsilon} - 1 \quad (1)$$

to achieve Δ -security up to n encryptions with advantage bounded by ε . For $\varepsilon = 2^{-\lambda}$ and $n = \text{Poly}(\lambda)$, the dominant term is $\frac{\lambda n}{2\Delta}$.

We can now prove Th. 3:

Proof. Let us consider a correct primitive of level n with st in a space \mathcal{ST} such that $|\mathcal{ST}| \leq 2^\ell$. We will show that it is insecure. To do so, we will first express that the state st after n encryptions are *constrained*. Namely, constraints are that st must decrypt the generated sequence of ct correctly. The constraints increase with n , and the set of possible st values which make decryption correct decreases. The set of constrained states does *not* decrease exponentially because of the surprising existence of “super states” which are able to

decrypt more than their constraints. Namely, super states can decrypt universally, including encryptions from the “future” which have not been generated yet. This is counter-intuitive. This set of super states is a hard core in the set of constrained states. We show that the set of constrained which are non-super states *does* decrease exponentially. Hence, by taking n large enough, constrained states become all super states: the state after n encryptions must be a super state. We use the property of the super state to mount an attack.

We first define notations. We extend the Enc and Dec functions. First of all, with random coins ρ , we write $\text{Enc}(\text{st}; \rho) = (\text{st}', \text{pt}, \text{ct})$ and consider Enc as deterministic with explicit coins. For $X \in \{\text{Enc}, \text{Dec}\}$ and $y \in \{\text{st}, \text{pt}, \text{ct}\}$, we denote by X_{o_y} the generated output of type y by the X operation: for both Enc and Dec , the output components define subfunctions $\text{Enc}_{\text{o}_\text{st}}, \text{Enc}_{\text{o}_\text{pt}}, \text{Enc}_{\text{o}_\text{ct}}, \text{Dec}_{\text{o}_\text{st}}, \text{Dec}_{\text{o}_\text{pt}}$ by

$$\begin{aligned}\text{Enc}(\text{st}; \rho) &= (\text{Enc}_{\text{o}_\text{st}}(\text{st}; \rho), \text{Enc}_{\text{o}_\text{pt}}(\text{st}; \rho), \text{Enc}_{\text{o}_\text{ct}}(\text{st}; \rho)) \\ \text{Dec}(\text{st}, \text{ct}) &= (\text{Dec}_{\text{o}_\text{st}}(\text{st}, \text{ct}), \text{Dec}_{\text{o}_\text{pt}}(\text{st}, \text{ct}))\end{aligned}$$

We further extend those functions with a variable number of inputs ρ or ct . We define

$$\begin{aligned}\text{Enc}_{\text{o}_\text{st}}(\text{st}, \rho_1, \dots, \rho_i) &= \text{Enc}_{\text{o}_\text{st}}(\text{Enc}_{\text{o}_\text{st}}(\text{st}, \rho_1, \dots, \rho_{i-1}); \rho_i) \\ \text{Dec}_{\text{o}_\text{st}}(\text{st}, \text{ct}_1, \dots, \text{ct}_i) &= \text{Dec}_{\text{o}_\text{st}}(\text{Dec}_{\text{o}_\text{st}}(\text{st}, \text{ct}_1, \dots, \text{ct}_{i-1}), \text{ct}_i)\end{aligned}$$

with the convention that $\text{Enc}_{\text{o}_\text{st}}(\text{st}) = \text{st}$ and $\text{Dec}_{\text{o}_\text{st}}(\text{st}) = \text{st}$, i.e., the functions with zero coins do nothing but returning st unchanged. Next, $\text{Enc}_{\text{o}_\text{pt}}(\text{st}, \rho_1, \dots, \rho_i)$ is the list of generated pt , $\text{Enc}_{\text{o}_\text{ct}}(\text{st}, \rho_1, \dots, \rho_i)$ is the list of generated ct , and $\text{Dec}_{\text{o}_\text{pt}}(\text{st}, \text{ct}_1, \dots, \text{ct}_i)$ is the list of decrypted pt :

$$\begin{aligned}\text{Enc}_{\text{o}_\text{pt}}(\text{st}, \rho_1, \dots, \rho_i) &= (\text{Enc}_{\text{o}_\text{pt}}(\text{Enc}_{\text{o}_\text{st}}(\text{st}, \rho_1, \dots, \rho_{j-1}); \rho_j))_{j=1, \dots, i} \\ \text{Enc}_{\text{o}_\text{ct}}(\text{st}, \rho_1, \dots, \rho_i) &= (\text{Enc}_{\text{o}_\text{ct}}(\text{Enc}_{\text{o}_\text{st}}(\text{st}, \rho_1, \dots, \rho_{j-1}); \rho_j))_{j=1, \dots, i} \\ \text{Dec}_{\text{o}_\text{pt}}(\text{st}, \text{ct}_1, \dots, \text{ct}_i) &= (\text{Dec}_{\text{o}_\text{pt}}(\text{Dec}_{\text{o}_\text{st}}(\text{st}, \text{ct}_1, \dots, \text{ct}_{j-1}), \text{ct}_j))_{j=1, \dots, i}\end{aligned}$$

Let st_n be the state which is obtained after n encryptions, before starting the decryption phase. In order to characterize the constraints on st_n coming from the first i encryptions, we introduce a set $C[r_i]$ corresponding to (and indexed with) each update operation $r_i = (\text{st}_0, \rho_1, \dots, \rho_i)$. Due to correctness, st_n must decrypt $\text{Enc}_{\text{o}_\text{ct}}(r_i)$ to $\text{Enc}_{\text{o}_\text{pt}}(r_i)$. Hence, we define the set of states which are constrained to r_i by

$$C[r_i] = \{\text{st} \in \mathcal{ST}; \text{Dec}_{\text{o}_\text{pt}}(\text{st}, \text{Enc}_{\text{o}_\text{ct}}(r_i)) = \text{Enc}_{\text{o}_\text{pt}}(r_i)\}$$

Clearly, for any i and any $\text{st}_0, \rho_1, \dots, \rho_n$, we have

$$\text{Enc}_{\text{o}_\text{st}}(\text{st}_0, \rho_1, \dots, \rho_n) \in C[\text{st}_0, \rho_1, \dots, \rho_n]$$

We note that $C[r_0]$, where $r_0 = \text{st}_0$ is the set of states subject to no restriction, hence $C[\text{st}_0] = \mathcal{ST}$. Furthermore, we note that

$$C[r_n] \subseteq \dots \subseteq C[r_2] \subseteq C[r_1] \subseteq C[r_0] = \mathcal{ST}$$

A state in $C[r_{i-\Delta}]$ decrypts well the first $i - \Delta$ ciphertexts. It may also be element of $C[r_{i-\Delta}, \rho_{i-\Delta+1}, \dots, \rho_i]$ if it decrypts the next Δ ciphertexts which are produced with coins $\rho_{i-\Delta+1}, \dots, \rho_i$. It may also be in $C[r_{i-\Delta}, \rho'_{i-\Delta+1}, \dots, \rho'_i]$ and decrypt Δ ciphertexts produced with other coins. With good probability, some state may actually have the “super-power” to decrypt ciphertexts produced with Δ more random coins. We call those states the *super states*. Intuitively, this is unexpected to happen but we show below that super-states exist and an adversary can build some easily.

More concretely, let $\alpha > 0$ be the probability from the statement of the theorem. We define a set of super states for $r_{j-\Delta} = (\mathbf{st}_0, \rho_1, \dots, \rho_{j-\Delta})$:

$$S[r_{j-\Delta}] = \left\{ \mathbf{st} \in \mathcal{ST}; \Pr_{\rho'_{j-\Delta+1}, \dots, \rho'_j} [\mathbf{st} \in C[r_{j-\Delta}, \rho'_{j-\Delta+1}, \dots, \rho'_j]] > \alpha \right\}$$

This set $S[r_{j-\Delta}]$ defines a set of states which are α -likely to decrypt a “fork” in the sequence of random coins. (See Fig. 3.)

We note that $S[r_{j-\Delta}] \subseteq C[r_{j-\Delta}]$ since for $\mathbf{st} \in S[r_{j-\Delta}]$, there must exist (due to a non-zero probability) $\rho'_{j-\Delta+1}, \dots, \rho'_j$ such that

$$\mathbf{st} \in C[r_{j-\Delta}, \rho'_{j-\Delta+1}, \dots, \rho'_j] \subseteq C[r_{j-\Delta}]$$

We define a union of super states as follows:

$$S^\cup[\mathbf{st}_0, \rho_1, \dots, \rho_{n-\Delta}] = S[\mathbf{st}_0] \cup S[\mathbf{st}_0; \rho_1] \cup \dots \cup S[\mathbf{st}_0; \rho_1, \dots, \rho_{n-\Delta}]$$

Clearly

$$S^\cup[r_{n-\Delta}] \supseteq \dots \supseteq S^\cup[r_1] \supseteq S^\cup[r_0]$$

The idea of the proof is to show that states with too many constraints tend to become super-states. Namely, we first prove that for n large enough, $C[r_n]$ is included in $S^\cup[r_{n-\Delta}]$ with large probability p . This means that after n encryptions, a state becomes a super-state. Hence, this state belongs to some $S[r_{m-\Delta}]$, with a random $m \leq n$. We now take a fixed value of m which is taken with probability at least $\frac{1}{n}$. (It exists, due to the pigeon-hole principle.) We take n encryptions from random coins $\mathbf{st}_0, \rho_1, \dots, \rho_{m-\Delta}, \rho'_{m-\Delta+1}, \dots, \rho'_n$. We deduce that there is a probability at least $\frac{p}{n}$ to get a state \mathbf{st}'_n in $S[r_{m-\Delta}]$. If it happens, \mathbf{st}'_n decrypts what is generated by the fork $\mathbf{st}_0, \rho_1, \dots, \rho_m$ with probability at least α (by definition of the super states). We define an adversary that exploits this fact in Fig. 3. The m encryptions with $\mathbf{st}_0, \rho_1, \dots, \rho_m$ are generated by the game, the state $\mathbf{st}_{m-\Delta}$ leaks, and the adversary can fork to construct \mathbf{st}'_n from it. We obtain the success probability of the adversary in the $\text{OW}_{m,\Delta,\lambda}$ game:

$$\Pr[\text{OW}_{m,\Delta,\lambda}(\mathcal{A}) \rightarrow 1] > \frac{\alpha p}{n} \quad (2)$$

In what follows, we show that $p \geq 1 - \left(\frac{1}{k} + \frac{k-1}{2}\alpha\right)^{\lfloor \frac{n}{\Delta} \rfloor} 2^\ell$.

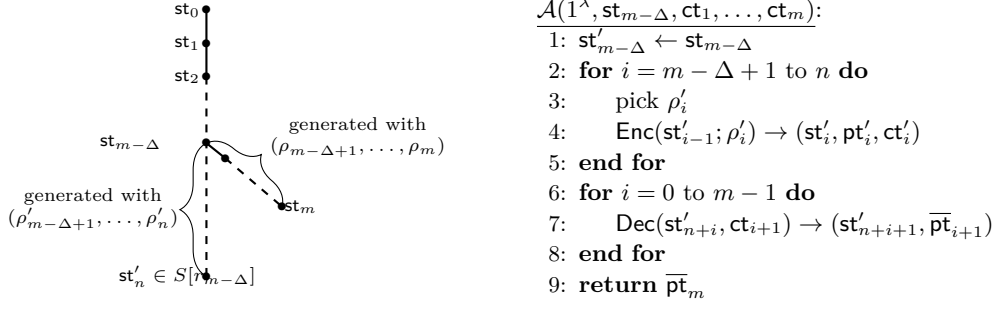
Let i be an integer. We consider for the moment that $\mathbf{st}_0, \rho_1, \dots, \rho_{i-\Delta}$ are fixed. For simplicity, we denote

$$\begin{aligned} C_{i-\Delta} &= C[\mathbf{st}_0, \rho_1, \dots, \rho_{i-\Delta}] & S_{i-\Delta}^\cup &= S^\cup[\mathbf{st}_0, \rho_1, \dots, \rho_{i-2\Delta}] \\ C_i(\vec{\rho}) &= C[\mathbf{st}_0, \rho_1, \dots, \rho_{i-\Delta}, \vec{\rho}] & S_i^\cup &= S^\cup[\mathbf{st}_0, \rho_1, \dots, \rho_{i-\Delta}] \end{aligned}$$

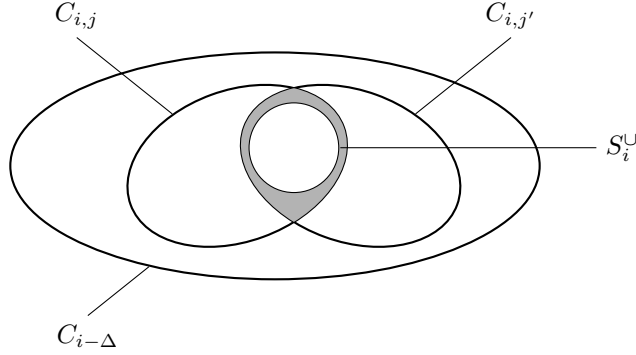
for a vector $\vec{\rho}$ of dimension Δ . We take k independent random Δ -dimensional vectors $\vec{\rho}_j$, for integers $j = 1, \dots, k$ and we define $C_{i,j} = C_i(\vec{\rho}_j)$. (k is defined in the statement of the Lemma.) Given $\vec{\rho}_j$ fixed and some $\mathbf{st} \in C_{i,j} = S_i^\cup$ fixed, we have $\mathbf{st} \notin S_i^\cup$ meaning that $\mathbf{st} \notin S[\mathbf{st}_0, \rho_1, \dots, \rho_{i-\Delta}]$, thus

$$\Pr_{\vec{\rho}_{j'}}[\mathbf{st} \in C_{i,j'}] \leq \alpha$$

25:10 Post-Compromise Security in Self-Encryption



■ **Figure 3** Starting from state st_0 and applying m encryption that generates ct_1, \dots, ct_m , we hope that leaking st_m and forking to n encryptions in total will end up in $st'_n \in S[r_{m-\Delta}]$. Therefore, st'_n decrypts all the ciphertext with probability at least α .



■ **Figure 4** Illustration of the intersection $(C_{i,j'} - S_i^U) \cap (C_{i,j} - S_i^U)$.

for any $\vec{\rho}_{j'}$ independent vector indexed with $j' \neq j$, by definition of S_i and $C_{i,j'}$. We count

$$|(C_{i,j'} - S_i^U) \cap (C_{i,j} - S_i^U)| = \sum_{st \in C_{i,j} - S_i^U} \mathbb{1}_{st \in C_{i,j'}}$$

We obtain

$$\mathbb{E}_{\vec{\rho}_{j'}} [|(C_{i,j'} - S_i^U) \cap (C_{i,j} - S_i^U)|] \leq \alpha |C_{i,j} - S_i^U| \leq \alpha |C_{i-\Delta} - S_{i-\Delta}^U|$$

for any j, j' , and $\vec{\rho}_j$ with $j \neq j'$. This is illustrated in Fig. 4. Clearly, we can then randomize $\vec{\rho}_j$ and obtain

$$\mathbb{E} [|(C_{i,j'} - S_i^U) \cap (C_{i,j} - S_i^U)|] \leq \alpha |C_{i-\Delta} - S_{i-\Delta}^U|$$

for any j and j' with $j \neq j'$.

Let $A_j = C_{i,j} - S_i^U$. This denotes one of the k subsets of $A = C_{i-\Delta} - S_{i-\Delta}^U$. We have

$$\sum_{j=1}^k |A_j| \leq |A| + \sum_{1 \leq j < j' \leq k} |A_j \cap A_{j'}|$$

Indeed, any element x of A occurring in exactly m subsets A_j is counted m times on the left-hand side and $1 + \frac{m(m-1)}{2}$ times on the right-hand side. However, $m \leq 1 + \frac{m(m-1)}{2}$ for

every integer m . We deduce

$$\mathbb{E} \left[\sum_{j=1}^k |C_{i,j} - S_i^{\cup}| \right] \leq \left(1 + \frac{k(k-1)}{2} \alpha \right) |C_{i-\Delta} - S_{i-\Delta}^{\cup}|$$

Given that all $\mathbb{E}[|C_{i,j} - S_i^{\cup}|]$ are equal, we have proven that

$$\mathbb{E}_{\vec{\rho}} [|C_i(\vec{\rho}) - S_i^{\cup}|] \leq \left(\frac{1}{k} + \frac{k-1}{2} \alpha \right) |C_{i-\Delta} - S_{i-\Delta}^{\cup}|$$

We can now randomize $\rho_1, \dots, \rho_{n-\Delta}$ as well and obtain

$$\mathbb{E} [|C[\text{st}_0, \rho_1, \dots, \rho_n] - S^{\cup}[\text{st}_0, \rho_1, \dots, \rho_{n-\Delta}]|] \leq \left(\frac{1}{k} + \frac{k-1}{2} \alpha \right)^{\lfloor \frac{n}{\Delta} \rfloor} |\mathcal{ST}|$$

We bound $|\mathcal{ST}| \leq 2^\ell$ and $\Pr[E \neq \emptyset] \leq \mathbb{E}[|E|]$ (due to the Markov inequality) for a random set E and obtain

$$\Pr[C[\text{st}_0, \rho_1, \dots, \rho_n] - S^{\cup}[\text{st}_0, \rho_1, \dots, \rho_{n-\Delta}] \neq \emptyset] \leq \left(\frac{1}{k} + \frac{k-1}{2} \alpha \right)^{\lfloor \frac{n}{\Delta} \rfloor} 2^\ell$$

By assumption on the size of \mathcal{ST} , for n large enough, we obtain that the set difference $C[\text{st}_0, \rho_1, \dots, \rho_n] - S^{\cup}[\text{st}_0, \rho_1, \dots, \rho_{n-\Delta}]$ is likely to be empty which means that the states in $C[\text{st}_0, \rho_1, \dots, \rho_n]$ are super states. By the definition of $C[r_n]$, $\text{Enc}_{\text{o_st}}(\text{st}_0; \rho_1, \dots, \rho_n) \in C[\text{st}_0; \rho_1, \dots, \rho_n]$. Hence, $\text{Enc}_{\text{o_st}}(\text{st}_0; \rho_1, \dots, \rho_n)$ is likely to be in $S^{\cup}[\text{st}_0, \rho_1, \dots, \rho_{n-\Delta}]$. More precisely,

$$\Pr[\text{Enc}_{\text{o_st}}(\text{st}_0, \rho_1, \dots, \rho_n) \notin S^{\cup}[\text{st}_0, \rho_1, \dots, \rho_{n-\Delta}]] \leq \left(\frac{1}{k} + \frac{k-1}{2} \alpha \right)^{\lfloor \frac{n}{\Delta} \rfloor} 2^\ell$$

If $\text{Enc}_{\text{o_st}}(\text{st}_0, \rho_1, \dots, \rho_n) \in S^{\cup}[\text{st}_0, \rho_1, \dots, \rho_{n-\Delta}]$, it means there exists (at least) one $m \leq n$ such that

$$\Pr_{\vec{\rho}'} [\text{Enc}_{\text{o_st}}(\text{st}_0, \rho_1, \dots, \rho_n) \in C(\text{st}_0, \rho_1, \dots, \rho_{m-\Delta}, \vec{\rho}')] > \alpha$$

Therefore, we obtain the success probability in the $\text{OW}_{m,\Delta,\lambda}$ game (from Eq. (2)):

$$\Pr[\text{OW}_{m,\Delta,\lambda}(\mathcal{A}) \rightarrow 1] > \frac{\alpha}{n} \left(1 - \left(\frac{1}{k} + \frac{k-1}{2} \alpha \right)^{\lfloor \frac{n}{\Delta} \rfloor} 2^\ell \right)$$

The complexity of \mathcal{A} is $n - m + \Delta$ encryptions and m decryptions. ◀

Uniform Impossibility Result

Our Th. 3 and Cor. 4 are *non-uniform* in the sense that the parameter m depends on λ in an unknown manner. However, \mathcal{A} is constructed in a polynomially bounded manner based on m . Thus, by guessing m , we obtain a uniform result with advantage divided by n .

3 Self-Ratchet

3.1 Definitions

Consider a self-ratcheted scheme $\text{SR} = (\text{lg}, \text{Init}, \text{Enc}, \text{Dec}, \text{Punc})$ with the following syntax:

25:12 Post-Compromise Security in Self-Encryption

- $\lg(\lambda)$ (length of the plaintext)
- $\text{SR.Init}(1^\lambda) \xrightarrow{\$} \text{st}$ (output an initial state for the device)
- $\text{SR.Enc}(\text{st}, \text{pt}) \xrightarrow{\$} (\text{st}', \text{ct})$ (update the state while encrypting $\text{pt} \in \{0, 1\}^{\lg(\lambda)}$)
- $\text{SR.Dec}(\text{st}, \text{ct}) \rightarrow \text{pt}$ or \perp (decrypt ct into pt)
- $\text{SR.Punc}(\text{st}, \text{ct}) \rightarrow \text{st}'$ (update the state by puncturing ct in st)

In our settings, there exists a device following a protocol which produces some pt/ct for itself so that it can eventually decrypt ct to recover pt in the future. Encryption is stateful. The protocol makes sure that when the device should no longer be able to decrypt ct and should be secure against any future state exposure, it can “puncture” the state. This means that the state st which can decrypt ct is replaced by a new (punctured) state st' so that ct is not decryptable by st' . With this notion, we aim at forward security and PCS.

► **Definition 5 (SR).** A *self-ratcheted scheme (SR) of level n* is a primitive $\text{SR} = (\text{Init}, \text{Enc}, \text{Dec}, \text{Punc})$ which is *n -correct* in the sense that for any sequence sched , the game in Fig. 5 never returns 1. Here, sched is a sequence of scheduled instructions which can be of three different types: (“Enc”, pt) (encrypt plaintext pt), (“Dec”, j) (decrypt the j -th produced ciphertext), and (“Punc”, j) (puncture the j -th produced ciphertext).

The correctness notion must consider any order of Enc/Dec/Punc instructions. This is what sched is modeling. We describe what should happen when this sequence of instructions is sched . Actually, we declare in L_{ct} the ciphertexts which are “active” and we put in L_{pt} how they are expected to decrypt.

This definition assumes that the number of “active” ciphertexts remains bounded by a parameter n (line number 5).

Compared to SEQ, an SR does not update the state during decryption (this is rather done by a separate function) and decryption can be done in any order of the ciphertexts (i.e., not only in the order they have been created). As applications will show, SR appears to be a most wanted primitive.

Application to Cloud Storage

SR schemes can be used for cloud storage where a client wants to store her files on the cloud in an encrypted form. Ideally, a single file is encrypted with SR.Enc to obtain a ct . For retrieval, the SR.Dec is run to decrypt the file. Eventually, when the client wants to remove the file from the cloud, the protocol will puncture her state for ct . The first desired security is that after a client erases an encrypted file, even though a copy was illegally kept and the state of the client later leaks, the file is unrecoverable. This is forward security. With SR, it is achieved by puncturing. The second desired security is that after the state of a client has leaked, if the client wants to store a new file in the cloud, this file should be safe, as long as no exposure occurs during the activity time of this file. This is post-compromise security. It is achieved by what we call self-ratchet.

One problem specific to cloud storage is that files are typically big and SR should handle them in encryption, decryption, and puncturing. One common approach is to use a domain expander based on a hybrid construction. Like the KEM/DEM hybrid cryptosystems, we can use SR to encrypt an ephemeral key K and symmetrically encrypt the plaintext with K .

We could also add key rotation, if required, by using SR to encrypt the encryption key: to encrypt a file pt , we pick a random key k (in the key domain of the key rotation scheme) and we run $\text{ct}_1 \leftarrow \text{SR.Enc}(\text{st}, k)$. Then, we encrypt pt with k following the key rotation scheme and obtain a header ct_2 and a ciphertext ct_3 . The final ciphertext is $\text{ct} = (\text{ct}_1, \text{ct}_2, \text{ct}_3)$. To

```

1: SR.Init( $1^\lambda$ )  $\xrightarrow{\$}$  st
2: set lists  $L_{pt}$  and  $L_{ct}$  to empty
3: for  $i = 1$  to  $|\text{sched}|$  do
4:   if  $\text{sched}_i$  parses as (“Enc”, pt) for some pt then
5:     if the number of  $L_{ct}$  entries which are different from  $\perp$  is at most  $n - 1$  then
6:       SR.Enc(st, pt)  $\rightarrow$  (st, ct)
7:        $L_{pt} \leftarrow (L_{pt}, \text{pt})$ 
8:        $L_{ct} \leftarrow (L_{ct}, \text{ct})$ 
9:     end if
10:  else if  $\text{sched}_i$  parses as (“Dec”,  $j$ ) for some  $j$  then
11:    if  $L_{ct}[j]$  exists and  $L_{ct}[j] \neq \perp$  then
12:      SR.Dec(st,  $L_{ct}[j]$ )  $\rightarrow$  pt
13:      if  $\text{pt} \neq L_{pt}[j]$  then return 1
14:    end if
15:  else if  $\text{sched}_i$  parses as (“Punc”,  $j$ ) for some  $j$  then
16:    if  $L_{ct}[j]$  exists and  $L_{ct}[j] \neq \perp$  then
17:      SR.Punc(st,  $L_{ct}[j]$ )  $\rightarrow$  st
18:       $L_{ct}[j] \leftarrow \perp$ 
19:    end if
20:  end if
21: end for
22: return 0

```

■ **Figure 5** Correctness game for SR of level n .

rotate the key k , we puncture st with ct_1 , run the key rotation scheme on $(\text{ct}_2, \text{ct}_3)$ to get a new key k' and new $(\text{ct}'_2, \text{ct}'_3)$, and run $\text{ct}'_1 \leftarrow \text{SR.Enc}(\text{st}, k')$ to form $\text{ct}' = (\text{ct}'_1, \text{ct}'_2, \text{ct}'_3)$.

Application to 0-RTT Session Resumption

SR schemes can be used for 0-RTT session resumption. Essentially, a server having a secure connection with a client using a key K would use $\text{SR.Enc}(\text{st}, K)$ to issue a ticket ct and send ct to the client. To resume a session, the client, who kept K and ct , would resend ct to the server who would use SR.Dec to recover K . The server might also immediately puncture it to avoid any replay of the ticket ct and for forward security.

Previous Work on 0-RTT Session Resumption

Def. 5 is more general than the definition of 0-RTT session resumption [2]. The differences are as follows:

- the notations for 0-RTT session resumption are Setup, TicketGen, and ServerRes instead of Init, Enc, Dec;
- SR separates SR.Dec and SR.Punc instead of having both functionalities in ServerRes.

There is no formal definition of correctness for 0-RTT session resumption in Aviram et al. [2]. However, we can fairly assume it is the same as our notion of correctness in Def. 5, but when sequences sched are limited such that every decryption is followed by puncturing: for all i and j , if $\text{sched}_i = (\text{“Dec”}, j)$ then $\text{sched}_{i+1} = (\text{“Punc”}, j)$. In 0-RTT session resumption, it makes sense to merge SR.Dec with SR.Punc as one of the security goal is precisely to prevent a ct to be replayed. For cloud storage, the client may need to decrypt the same ct several times before she removes the file from the cloud. Hence, we keep SR.Dec and SR.Punc separate.

We adapt the security definition of 0-RTT session resumption with our notations to which we add specific instructions for post-compromise security. We define the $\text{IND}_{b,n,\Delta,\lambda}^{\text{SR,opt}}(\mathcal{A})$ game in Fig. 6. We also generalize it to adaptive security. In the AGJ security model, the game

25:14 Post-Compromise Security in Self-Encryption

starts with many `OEnc` and only after that, the adversary can play with oracles except `OEnc` (it is somehow non-adaptive). The AGJ model uses $\text{opt} = \{\text{noPCS}, \text{replay}\}$ and it is formalized for key establishment rather than encryption. (This means that there is a `Test` oracle to test a decryption instead of a `Challenge` oracle to get an encryption challenge.)

► **Definition 6** (SR security). *Let $n(\lambda)$ and $\Delta(\lambda)$ be polynomially bounded positive integer functions of a security parameter λ . The option set opt specifies some variants in the game in Fig. 6. The advantage is*

$$\text{Adv}_{n,\Delta,\lambda}^{\text{IND}^{\text{SR,opt}}}(\mathcal{A}) = \left| \Pr \left[\text{IND}_{1,n,\Delta,\lambda}^{\text{SR,opt}}(\mathcal{A}) \rightarrow 1 \right] - \Pr \left[\text{IND}_{0,n,\Delta,\lambda}^{\text{SR,opt}}(\mathcal{A}) \rightarrow 1 \right] \right|$$

We say that **SR is IND-opt secure at level n with delay Δ** if for any PPT adversary \mathcal{A} , $\lambda \mapsto \text{Adv}_{n,\Delta,\lambda}^{\text{IND}^{\text{SR,opt}}}(\mathcal{A})$ is a negligible function.

When “replay” \in opt , the security notion aims to address replay attacks. It enforces puncturing after decryption. Hence, decryption must puncture, as well. When “noPCS” \in opt , the security notion aims to capture forward security *without* post-compromise security. Absence of noPCS in opt is a stronger security notion as it captures FS and PCS together.

Game $\text{IND}_{b,n,\Delta,\lambda}^{\text{SR,opt}}(\mathcal{A})$:

- 1: $\text{Init}(1^\lambda) \rightarrow \text{st}$
- 2: $\text{Active}, \text{Revealed} \leftarrow \emptyset$
- 3: $\text{challenged} \leftarrow \text{false}$
- 4: $\text{AfterExp} \leftarrow \Delta$
- 5: $\mathcal{A}^{\text{OEnc}, \text{ODec}, \text{Challenge}, \text{OPunc}, \text{OExp}}(1^\lambda) \rightarrow b^*$
- 6: **return** b^*

Oracle $\text{OEnc}(\text{pt})$:

- 7: **if** $|\text{Active}| \geq n$ **then return** \perp
- 8: $\text{SR.Enc}(\text{st}, \text{pt}) \rightarrow (\text{st}, \text{ct})$
- 9: $\text{Active} \leftarrow \text{Active} \cup \{\text{ct}\}$
- 10: $\text{Revealed} \leftarrow \text{Revealed} \cup \{\text{ct}\}$
- 11: $\text{AfterExp} \leftarrow \text{AfterExp} + 1$
- 12: **return** ct

Oracle $\text{ODec}(\text{ct})$:

- 13: **if** $\text{ct} \in \text{Active} - \text{Revealed}$ **then**
- 14: **return** \perp
- 15: **end if**
- 16: $\text{SR.Dec}(\text{st}, \text{ct}) \rightarrow r$
- 17: **if** “replay” $\in \text{opt}$ **then** $\text{OPunc}(\text{ct})$
- 18: **return** r

Oracle $\text{OPunc}(\text{ct})$:

- 19: $\text{SR.Punc}(\text{st}, \text{ct}) \rightarrow \text{st}$
- 20: $\text{Active} \leftarrow \text{Active} - \{\text{ct}\}$
- 21: $\text{Revealed} \leftarrow \text{Revealed} - \{\text{ct}\}$
- 22: **return**

Oracle $\text{Challenge}(\text{pt}_1)$:

- 23: **if** challenged **then return** \perp
- 24: **if** $|\text{Active}| \geq n$ **then return** \perp
- 25: **if** $\text{AfterExp} < \Delta$ **then return** \perp
- 26: pick pt_0 of same length as pt_1 at random
- 27: $\text{SR.Enc}(\text{st}, \text{pt}_b) \rightarrow (\text{st}, \text{ct})$
- 28: $\text{Active} \leftarrow \text{Active} \cup \{\text{ct}\}$
- 29: $\text{AfterExp} \leftarrow \text{AfterExp} + 1$
- 30: $\text{challenged} \leftarrow \text{true}$
- 31: **return** ct

Oracle $\text{OExp}()$:

- 32: **if** ($\neg \text{challenged}$ and “noPCS” $\in \text{opt}$) or $(\text{Active} - \text{Revealed} \neq \emptyset)$ **then**
- 33: **return** \perp
- 34: **end if**
- 35: $\text{AfterExp} \leftarrow 0$
- 36: **return** st

■ **Figure 6** Indistinguishability game for self-ratchet.

$S.Gen = SR.Init$

$S.Enc(st)$:

- 1: pick $K \in \{0, 1\}^{\lg(\lambda)}$ at random
- 2: $SR.Enc(st, K) \rightarrow (st', ct)$
- 3: **return** (st', K, ct)

$S.Dec(st, ct)$:

- 4: $SR.Dec(st, ct) \rightarrow K$
- 5: **if** $K \neq \perp$ **then** $SR.Punc(st, ct) \rightarrow st$
- 6: **return** (st, K)

■ **Figure 7** SEQ from SR.

$\mathcal{A}^{OEnc, ODec, Challenge, OPunc, OExp}(1^\lambda)$:

- 1: pick $m \in \{\Delta, \dots, n\}$
- 2: **for** $i = 1$ to $m - \Delta$ **do**
- 3: pick pt_i at random
- 4: $OEnc(pt_i) \rightarrow ct_i$
- 5: **end for**
- 6: $OExp() \rightarrow st_{m-\Delta}$

7: **for** $i = m - \Delta + 1$ to $m - 1$ **do**

8: pick pt_i at random

9: $OEnc(pt_i) \rightarrow ct_i$

10: **end for**

11: pick pt_m at random

12: $Challenge(pt_m) \rightarrow ct_m$

13: $\mathcal{B}(1^\lambda, st_{m-\Delta}, ct_1, \dots, ct_m) \rightarrow z$

14: **return** $1_{z=pt_m}$

■ **Figure 8** Adversary against SR based on an adversary for SEQ.

3.2 Impossibility Result

► **Theorem 7.** *For every integer $n, \ell, \Delta > 0$ and any n -correct self-ratcheted scheme SR following Def. 5, and such that st belongs to a space of size bounded by 2^ℓ , there exist a (small) constant c and an adversary of complexity bounded by $(n + \Delta)(T_{Enc} + T_{Dec} + T_{Punc} + 1) + c$ having advantage*

$$\text{Adv}_{n, \Delta, \lambda}^{\text{IND}^{\text{SR}, \text{opt}}}(\mathcal{A}) > \frac{1}{4n^2} 2^{-2 \frac{\ell+1}{\lceil n/\Delta \rceil}} - 2^{-\lg(\lambda)}$$

for $\text{opt} = \perp$ and $\text{opt} = \text{replay}$, and where T_{\S} is the complexity to pick an element of $\{0, 1\}^{\lg(\lambda)}$ at random and T_{Enc}, T_{Dec} and T_{Punc} are the complexities of Enc, Dec and Punc.

Proof. We construct a SEQ from a self-ratcheted protocol SR in Fig. 7. Clearly, the n -correctness of SR implies the n -correctness of S for any n . The SEQ scheme only imposes ciphertexts to be received in the same order as they have been produced.

Due to Cor. 4, there exists m and an $OW_{m, \Delta, \lambda}$ adversary \mathcal{B} such that $\Pr[OW_{m, \Delta, \lambda} \rightarrow 1] = p$ with $p > \frac{1}{4n} 2^{-2 \frac{\ell+1}{\lceil n/\Delta \rceil}}$. \mathcal{B} is constructed uniformly from m . Then, we can construct an $\text{IND}_{b, n, \Delta, \lambda}^{\text{SR}, \text{opt}}$ adversary \mathcal{A} who guesses m as in Fig. 8.

The Challenge oracle encrypts pt_m which is either pt_m or random. Since \mathcal{A} simulates well the $OW_{m, \Delta, \lambda}$ game, we have $\Pr[z = pt_m] \geq \frac{p}{n}$. Hence, $\Pr[\text{IND}_{1, n, \Delta}^{\text{SR}, \text{opt}} \rightarrow 1] = \frac{p}{n}$ and $\Pr[\text{IND}_{0, n, \Delta}^{\text{SR}, \text{opt}} \rightarrow 1] = 2^{-\lg(\lambda)}$. Hence, the advantage is $\frac{p}{n} - 2^{-\lg(\lambda)}$.

The adversary \mathcal{A} picks m plaintexts and issues $m - 1$ $OEnc$ queries, one $OExp$ query and one Challenge query, and then simulates an $OW_{m, \Delta, \lambda}$ adversary \mathcal{B} . The complexity of \mathcal{B} is the complexity of $n - m + \Delta$ encryptions and m decryptions, and the complexities of $S.Enc$ and $S.Dec$ are respectively $T_{Enc} + T_{\S}$ and $T_{Dec} + T_{Punc}$. The complexity of \mathcal{A} therefore is $n - m + \Delta$ encryptions, m decryptions, m punctuations, $m + 1$ oracle calls and $(n + \Delta)$ random selections. ◀

25:16 Post-Compromise Security in Self-Encryption

<p>SR.Init(1^λ):</p> <p>1: $\text{st} \leftarrow (0, [])$</p> <p style="padding-left: 100px;">\triangleright a counter set to 0 and an empty list</p> <p>2: return st</p> <p>SR.Dec(st, ct):</p> <p>3: parse $\text{st} = (c, L)$ and $\text{ct} = (i, \text{ct}_0)$</p> <p>4: $\text{FSSR.Dec}(L[i], \text{ct}_0) \rightarrow \text{pt}$</p> <p>5: return pt</p> <p>SR.Punc(st, ct):</p> <p>6: parse $\text{st} = (c, L)$ and $\text{ct} = (i, \text{ct}_0)$</p> <p>7: $\text{FSSR.Punc}(L[i], \text{ct}_0) \rightarrow L[i]$</p> <p style="padding-left: 100px;">$\triangleright L[i]$ is updated</p> <p>8: $\text{st} \leftarrow (c, L)$</p> <p>9: return st</p>	<p>SR.Enc(st, pt):</p> <p>10: parse $\text{st} = (c, L)$</p> <p>11: if $c = 0$ then</p> <p>12: $c \leftarrow \Delta$</p> <p>13: $\text{FSSR.Init}(1^\lambda) \rightarrow s$</p> <p>14: $L \leftarrow (L, s)$</p> <p style="padding-left: 100px;">\triangleright add a new FSSR state in L</p> <p>15: end if</p> <p>16: $c \leftarrow c - 1$</p> <p>17: set ℓ to the length of L</p> <p>18: $\text{FSSR.Enc}(L[\ell], \text{pt}) \rightarrow (L[\ell], \text{ct}_0)$</p> <p style="padding-left: 100px;">$\triangleright L[\ell]$ is updated</p> <p>19: $\text{st} \leftarrow (c, L)$</p> <p>20: $\text{ct} \leftarrow (\ell, \text{ct}_0)$</p> <p>21: return (st, ct)</p>
--	---

■ **Figure 9** Post-compromise secure self-ratchet from forward secure self-ratchet.

3.3 Constructions

We provide a generic construction SR from an FSSR scheme⁶ providing forward security. For every Δ , we create a new structure with forward security and store it. Given a scheme FSSR offering only forward security, we construct SR as in Fig. 9.

► **Theorem 8.** *Let $n(\lambda)$ and $\Delta(\lambda)$ be polynomially bounded positive integer functions of a security parameter λ . Let opt be either \perp or $\{\text{replay}\}$. Let FSSR be a self-ratcheted scheme which is $\text{IND}-(\text{opt} \cup \{\text{noPCS}\})$ secure at level Δ . Then, SR (in Fig. 9, with parameter Δ) is a self-ratcheted scheme which is IND-opt secure at level n with delay Δ .*

Proof. Let opt be either \perp or $\{\text{replay}\}$ and \mathcal{B} be an IND-opt adversary against SR with delay Δ . Assume that \mathcal{B} queries at most q encryption and challenge queries. Then, we can construct an $\text{IND}-(\text{opt} \cup \{\text{noPCS}\})$ adversary \mathcal{A} against FSSR at level Δ as shown on Fig. 10.

By the construction, SR generates a new state of FSSR for each Δ encryptions. The adversary \mathcal{A} therefore simulates the IND-opt security game with delay Δ while trying to replace Δ ciphertexts by the ciphertexts that the adversary is challenging. If the oracle $\text{Challenge}'$ does not abort the game, the adversary \mathcal{A} can correctly guess b if \mathcal{B} can correctly guess it. The probability that the game is not aborted by $\text{Challenge}'$ is about Δ/q . Then, the advantage of \mathcal{A} is

$$\text{Adv}_{\Delta, \cdot, \lambda}^{\text{IND}^{\text{FSSR}, (\text{opt} \cup \{\text{noPCS}\})}}(\mathcal{A}) = \frac{1}{\lceil q/\Delta \rceil} \text{Adv}_{n, \Delta, \lambda}^{\text{IND}^{\text{SR}, \text{opt}}}(\mathcal{B})$$

Since q is polynomially bounded and $\Delta \geq 1$, if $\text{Adv}_{\Delta, \cdot, \lambda}^{\text{IND}^{\text{FSSR}, (\text{opt} \cup \{\text{noPCS}\})}}(\mathcal{A})$ is negligible, then $\text{Adv}_{n, \Delta, \lambda}^{\text{IND}^{\text{SR}, \text{opt}}}(\mathcal{B})$ is negligible too. Hence, SR is IND-opt secure at level n with delay Δ if FSSR is $\text{IND}-(\text{opt} \cup \{\text{noPCS}\})$ secure at level Δ . ◀

⁶ FSSR means FS-secure self-ratcheted scheme.

$\mathcal{A}^{\text{OEnc, ODec, Challenge, OPunc, OExp}}(1^\lambda)$:

```

1:  $\text{idx} \xleftarrow{\$} \{1, \dots, \lceil q/\Delta \rceil\}$ 
2:  $\text{SR.Init}(1^\lambda) \rightarrow \text{st}$ 
3:  $\text{Active, Revealed} \leftarrow \emptyset$ 
4:  $\text{challenged} \leftarrow \text{false}$ 
5:  $\text{AfterExp} \leftarrow \Delta$ 
6:  $\mathcal{B}^{\text{OEnc}', \text{ODec}', \text{Challenge}', \text{OPunc}', \text{OExp}'}(1^\lambda) \rightarrow b'$ 
7: return  $b'$ 

```

Subroutine $\text{OEnc}'(\text{pt})$:

```

8: if  $|\text{Active}| \geq n$  then return  $\perp$ 
9:  $\text{SR.Enc}(\text{st}, \text{pt}) \rightarrow (\text{st}, \text{ct})$ 
10:  $\text{parse ct} = (\ell, \text{ct}_0)$ 
11: if  $\ell = \text{idx}$  then
12:    $\text{OEnc}(\text{pt}) \rightarrow \text{ct}_0$ 
13: end if
14:  $\text{ct} \leftarrow (\ell, \text{ct}_0)$ 
15:  $\text{Active} \leftarrow \text{Active} \cup \{\text{ct}\}$ 
16:  $\text{Revealed} \leftarrow \text{Revealed} \cup \{\text{ct}\}$ 
17:  $\text{AfterExp} \leftarrow \text{AfterExp} + 1$ 
18: return  $\text{ct}$ 

```

Subroutine $\text{ODec}'(\text{ct})$:

```

19: if  $\text{ct} \in \text{Active} - \text{Revealed}$  then
20:   return  $\perp$ 
21: end if
22:  $\text{parse ct} = (\ell, \text{ct}_0)$ 
23: if  $\ell = \text{idx}$  then
24:    $\text{ODec}(\text{ct}_0) \rightarrow \text{pt}$ 
25: else
26:    $\text{SR.Dec}(\text{st}, \text{ct}) \rightarrow (\text{st}, \text{pt})$ 
27: end if
28: if “replay”  $\in \text{opt}$  then  $\text{OPunc}'(\text{ct})$ 
29: return  $\text{pt}$ 

```

Subroutine $\text{OPunc}'(\text{ct})$:

```

30:  $\text{parse ct} = (\ell, \text{ct}_0)$ 
31: if  $\ell = \text{idx}$  then
32:    $\text{OPunc}(\text{ct}_0)$ 
33: else
34:    $\text{SR.Punc}(\text{st}, \text{ct}) \rightarrow \text{st}$ 
35: end if
36:  $\text{Active} \leftarrow \text{Active} - \{\text{ct}\}$ 
37:  $\text{Revealed} \leftarrow \text{Revealed} - \{\text{ct}\}$ 
38: return

```

Subroutine $\text{Challenge}'(\text{pt})$:

```

39: if  $|\text{Active}| \geq n$  or  $\text{AfterExp} < \Delta$  then
40:   return  $\perp$ 
41: end if
42:  $\text{parse st} = (c, L)$ 
43: if  $(c \neq 0$  or  $|L| \neq \text{idx} - 1)$  and  $(c = 0$  or  $|L| \neq \text{idx})$  then
44:   abort the game
45: end if
46:  $\text{SR.Enc}(\text{st}, \text{pt}) \rightarrow (\text{st}, \text{ct})$ 
47:  $\text{Challenge}(\text{pt}) \rightarrow \text{ct}$ 
48:  $\text{Active} \leftarrow \text{Active} \cup \{\text{ct}\}$ 
49:  $\text{AfterExp} \leftarrow \text{AfterExp} + 1$ 
50:  $\text{challenged} \leftarrow \text{true}$ 
51: return  $\text{ct}$ 

```

Subroutine $\text{OExp}'()$:

```

52:  $\text{parse st} = (c, L)$ 
53: if  $|L| \geq \text{idx}$  then
54:    $\text{OExp}() \rightarrow \text{st}'$ 
55:   if  $\text{st}' = \perp$  then return  $\perp$ 
56:    $L[\text{idx}] \leftarrow \text{st}'$ 
57: end if
58:  $\text{AfterExp} \leftarrow 0$ 
59: return  $(c, L)$ 

```

■ **Figure 10** FS adversary for FSSR based on an adversary for SR.

Optimization

Our SR scheme can obviously be optimized for storage. For each state $L[i]$, we can add a counter of active ciphertexts with $L[i]$ which is incremented by Enc and decremented by Punc (after checking that decryption works). Then, when the counter becomes 0, $L[i]$ can be erased.

Another convenient optimization holds when the application wants to operate bulk puncturing of too old ciphertexts. This implies to erase all first $L[i]$. It is quite compatible with recent policies of session resumption: a session which is too old cannot be resumed.

<pre> FSSR.Init(1^λ): 1: PPRF.Setup(1^λ) $\rightarrow k_{\text{PPRF}}$ 2: $\text{st} \leftarrow (k_{\text{PPRF}}, 0)$ 3: return st FSSR.Punc(st, ct): 4: parse $\text{st} = (k_{\text{PPRF}}, \text{cnt})$ 5: parse $\text{ct} = (\text{cnt}', \text{ct}_0)$ 6: $\text{pt} \leftarrow \text{FSSR.Dec}(\text{st}, \text{ct})$ 7: if $\text{pt} = \perp$ then return \perp 8: PPRF.Punc($k_{\text{PPRF}}, \text{cnt}'$) $\rightarrow k_{\text{PPRF}}$ 9: if $k_{\text{PPRF}} = \perp$ then return \perp 10: $\text{st} \leftarrow (k_{\text{PPRF}}, \text{cnt})$ 11: return st </pre>	<pre> FSSR.Enc(st, pt): 12: parse $\text{st} = (k_{\text{PPRF}}, \text{cnt})$ 13: $\kappa \leftarrow \text{PPRF.Eval}(k_{\text{PPRF}}, \text{cnt})$ 14: if $\kappa = \perp$ then return \perp 15: $\text{ct}_0 \leftarrow \text{AEAD.Enc}(\kappa, \text{cnt}, \text{pt})$ 16: $\text{ct} \leftarrow (\text{cnt}, \text{ct}_0)$ 17: $\text{st} \leftarrow (k_{\text{PPRF}}, \text{cnt} + 1)$ 18: return st, ct FSSR.Dec(st, ct): 19: parse $\text{st} = (k_{\text{PPRF}}, \text{cnt})$ 20: parse $\text{ct} = (\text{cnt}', \text{ct}_0)$ 21: $\kappa \leftarrow \text{PPRF.Eval}(k_{\text{PPRF}}, \text{cnt}'$) 22: if $\kappa = \perp$ then return \perp 23: $\text{pt} \leftarrow \text{AEAD.Dec}(\kappa, \text{cnt}', \text{ct}_0)$ 24: return pt </pre>
---	--

■ **Figure 11** FS-secure SR.

3.4 FS-Secure Self-Ratcheted Scheme (from AGJ)

We adapt⁷ the generic construction from Aviram et al. [2] based on a puncturable PRF denoted as PPRF. We use authenticated encryption with associated data $\text{AEAD} = (\text{Gen}, \text{Enc}, \text{Dec})$. (In our notation, the second input to Enc and Dec is the associated data i.e. the header to be authenticated.) The construction is in Fig. 11.

AGJ presented two possible PPRF constructions. One is based on the Camenisch-Lysyanskaya RSA accumulator [5]. The other is based on a tree structure.

RSA-based PPRF

The RSA-based construction uses a PPRF key of linear size in terms of the number of encryptions and can only handle a polynomial number of encryptions. This is the total number of encryptions, i.e. not only the ones remaining active. We give the construction in Fig. 12, using a random oracle H and the list of first odd primes (p_1, \dots, p_m) . In the original paper [2], the authors have shown that the above construction is a secure PPRF in the random oracle model, under the strong RSA assumption. The PPRF key is of size $2\lambda + m$. However, the N part of the key can be set as a domain parameter which is common to many keys.

In our construction, the device only needs to encrypt Δ messages per PPRF key. Hence, we can set $m = \Delta$ in the above PPRF, meaning that the FSSR has states of size $\lambda + \Delta + \log_2 \Delta$ plus λ bits of common parameter N . Finally, our SR has states of size

$$\ell = \frac{n}{\Delta} (\lambda + \Delta + \log_2 \Delta) + \log_2 \Delta + \lambda \quad (3)$$

We can see that $\frac{\ell \Delta}{n}$ is at least linear in λ , hence super-logarithmic. Compared to (1), we are within a factor close to 2 to the lower bound.

⁷ The only change is the separation between Dec and Punc.

PPRF.Setup (1^λ): 1: generate an RSA modulus $N = pq$ of length λ using safe primes 2: erase p and q 3: pick $g \in \mathbf{Z}_N$ at random 4: $r \leftarrow (0, 0, \dots, 0) \in \{0, 1\}^m$ 5: $k_{\text{PPRF}} \leftarrow (N, g, r)$ 6: return k_{PPRF}	PPRF.Eval (k_{PPRF}, x): 7: parse $k_{\text{PPRF}} = (N, g, r)$ 8: if $r_x = 1$ then return \perp 9: $P_x \leftarrow \prod_{i=1}^m p_i^{r_i \cdot 1_{i \neq x}}$ 10: $y \leftarrow H(g^{P_x} \bmod N)$ 11: return y	PPRF.Punc (k_{PPRF}, x): 12: parse $k_{\text{PPRF}} = (N, g, r)$ 13: if $r_x = 1$ then return \perp 14: $g \leftarrow g^{p_x} \bmod N$ 15: $r_x \leftarrow 1$ 16: $k_{\text{PPRF}} \leftarrow (N, g, r)$ 17: return k_{PPRF}
---	---	--

■ **Figure 12** RSA-based PPRF.

Tree-based PPRF

The tree-based construction is formed with two functions G_0 and G_1 from $\{0, 1\}^\lambda$ to itself, which we extend to functions G_z for every binary word z by $G_{xy}(L) = G_y(G_x(L))$. Then, the PPRF defines a binary tree of depth d which is partially labeled. The PPRF key is a set of (x, L) pairs where x is a binary word (hence a node in the binary tree) and L is its label in $\{0, 1\}^\lambda$. Initially, the key consists of the label of the root ε . To evaluate on x , one should find a labeled node (y, L) such that y is a prefix of x , write $x = yz$, and return $G_z(L)$. The interface of the PPRF only takes d -bit input x (i.e. leaves), but our evaluation is defined for every node. To puncture a leaf x , one should find this y again and replace (y, L) from the key by the list of (x', L') with $x' = yz_1 \cdots z_{i-1} \bar{z}_i$ and L' being the evaluation on x' , where $z_1 \cdots z_{|z|}$ is the binary expansion of z and \bar{z}_i is the bit complement of z_i . Hence, a PPRF key is an anti-chain with no siblings. In the worst case, it could inflate by d pairs at every puncture, but the maximum length is of 2^{d-1} pairs.

Same as the RSA construction, one only needs to evaluate $2^d = \Delta$ leaves. In the worst case, a PPRF key has length $2^{d-1} \times d\lambda$ which is $\frac{1}{2}\lambda\Delta \log_2 \Delta$. Hence, the FS-secure self-ratcheted scheme has states of size bounded by $\frac{1}{2}\lambda\Delta \log_2 \Delta + \log_2 \Delta$. Finally, our secure self-ratcheted scheme has states of size

$$\ell = \frac{n}{\Delta} \left(\frac{1}{2}\lambda\Delta \log_2 \Delta + \log_2 \Delta \right) + \log_2 \Delta$$

which is larger than with the RSA-based method.

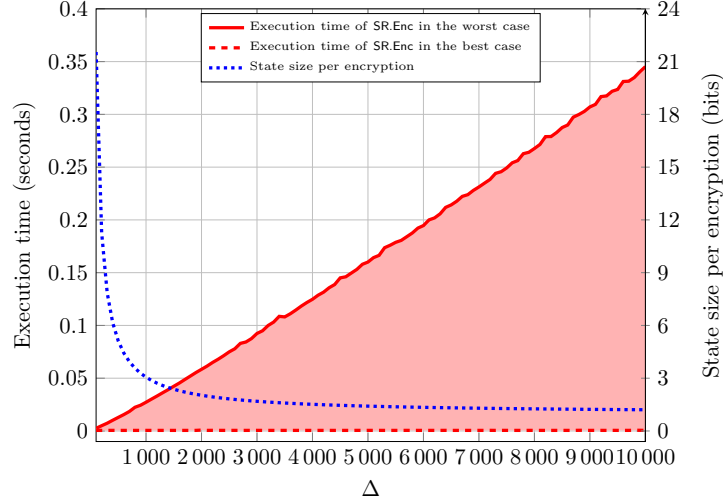
3.5 Experimental Results

We instantiate an SR based on FSSR with the RSA-based PPRF. We assumed that the same RSA modulus is used for all PPRF keys, the RSA modulus so is precomputed and given as a parameter to SR. Hence, the cost of setting up the RSA modulus is not covered in our analysis. For H and AEAD, we used SHA-256 and AES-GCM.

Our experiment was done on a machine with the AMD Opteron 8354 processor and 128 GB of RAM by using the SageMath version 8.7. We picked a common 2048-bit RSA modulus.

We tried many values for Δ from $\Delta = 100$ to $\Delta = 10\,000$ by steps of 100. We measured the worst case complexity of an SR.Enc encryption, which is actually the very first one when nothing is punctured and which includes FSSR.Init, as well as the best case complexity of SR.Enc, which is the very last one after all other values have been punctured. For accuracy, we did it 1000 times for each Δ and took the average. The results are plotted in Fig. 13.

On the plot, we added the total state size divided by the total number n of encryptions as it goes to infinity. This is essentially $\frac{\ell}{n}$ with ℓ given by Eq.(3). As we can see, the execution time grows linearly with Δ while $\frac{\ell}{n} - 1$ is inverse proportional to Δ .



■ **Figure 13** The execution time of SR.Enc in the worst/best case and the state size divided by the number of encryptions with 2048-bit RSA modulus.

4 Bipartite Ratcheted Communication

4.1 Definitions

We consider a ratcheted scheme $S = (\text{Gen}, \text{Enc}, \text{Dec})$ following the syntax

- $S.\text{Gen}(1^\lambda) \rightarrow (\text{st}_A, \text{st}_B)$ (generate a pair of states)
- $S.\text{Enc}(\text{st}) \rightarrow (\text{st}', \text{pt}, \text{ct})$ (update the state while producing a pt/ct pair)
- $S.\text{Dec}(\text{st}, \text{ct}) \rightarrow (\text{st}', \text{pt})$ (update the state while decrypting ct)

To avoid defining a general correctness and security for ratcheted schemes, which is quite lengthy and complicated, we only adopt a definition matching a particular case of our interest. This is the case when one participant Alice desperately tries to reach her counterpart Bob by consistently sending messages without receiving any response, while Bob actually acknowledges for the receipt of every message from Alice but his acknowledgments somehow never make it through. (See Fig. 15.)

► **Definition 9.** A *simple ratcheted scheme* is a primitive S defined by $S = (\text{Gen}, \text{Enc}, \text{Dec})$ which is **n -correct** in the sense that the game in Fig. 14 never returns 1.

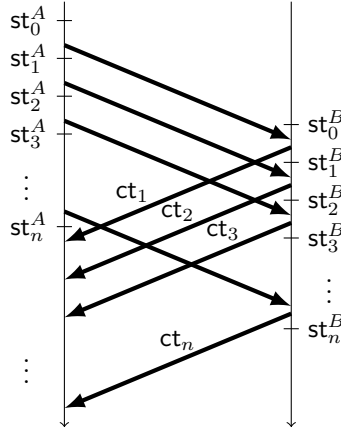
In this communication pattern, protocols such as PR [14], JS [12], JMM [13], and DV [8] have growing states. We can clearly see it on the implementation results by Caforio et al. [4]. Protocols such as Signal [15] or ACD [1] keep constant-size states but offer no post-compromise security in our communication pattern. In fact, in ACD, Alice keeps sending messages in the same “epoch” (following the terminology of ACD [1]) using the forward secure scheme called FS in ACD, while Bob receives those messages from an old epoch (for him) and keeps sending messages in his own epoch, using FS as well. Since the FS scheme is deterministic, it

```

1:  $S.Gen(1^\lambda) \rightarrow (st_0^A, st_0^B)$ 
2: for  $i = 1$  to  $n$  do
3:    $S.Enc(st_{i-1}^A) \rightarrow (st_i^A, pt'_i, ct'_i)$ 
4:    $S.Dec(st_{i-1}^B, ct'_i) \rightarrow (x, pt_i)$ 
5:    $S.Enc(x) \rightarrow (st_i^B, pt_i, ct_i)$ 
6: end for
7:  $x \leftarrow st_n^A$ 
8: for  $i = 1$  to  $n$  do
9:    $S.Dec(x, ct_i) \rightarrow (x, pt)$ 
10:  if  $pt \neq pt_i$  then return 1
11: end for
12: return 0

```

■ **Figure 14** Correctness game for a simple ratcheted scheme of level- n .



■ **Figure 15** Simulation of the level- n correctness game.

offers no post-compromise security. In ACD-PK, there is an extra public-key encryption but the decryption key remains constant within the same epoch. Hence, exposing st_1^A is enough to decrypt all ciphertexts in both ACD and ACD-PK.

Post-compromise security should make impossible to decrypt ct_m which was released after having ratcheted Δ times both participants after the last state exposure which revealed $st_{m-\Delta}^A$ and $st_{m-\Delta}^B$. For instance, with $\Delta = 1$ and $m = 2$, it should be impossible on Fig. 15 to compute pt_2 from $(st_1^A, st_1^B, ct_1, ct_2)$. This is formalized by the following definition.

► **Definition 10.** Let $n(\lambda)$ and $\Delta(\lambda)$ be polynomially bounded positive integer functions of a security parameter λ . For a simple ratcheted scheme S which is n -correct, we define the game in Fig. 16 with parameters $m \leq n$ and $\Delta > 0$: We say that S with level n is Δ -secure if for any PPT adversary \mathcal{A} , $\lambda \mapsto \max_{1 \leq m \leq n} \Pr[\text{OW}_{m,\Delta,\lambda}(\mathcal{A}) \rightarrow 1]$ is a negligible function.

4.2 Impossibility Result

► **Theorem 11.** For every integer $n, \ell, \Delta > 0$ and any n -correct simple ratcheted scheme S following Def. 9, and such that (st_A, st_B) belongs to a space of size bounded by 2^ℓ , there exists an adversary of low complexity having advantage

$$\Pr[\text{OW}_{m,\Delta,\lambda}(\mathcal{A}) \rightarrow 1] > \frac{1}{4n^2} 2^{-2 \frac{\ell+1}{\lfloor n/\Delta \rfloor}}$$

in the security game of Def. 10.

Proof. We construct a SEQ protocol P as shown in Fig. 17. If S is n -correct (in the sense of Def. 9), then this new scheme P is correct to level n (in the sense of Def. 1). This comes

25:22 Post-Compromise Security in Self-Encryption

$\text{OW}_{m,\Delta,\lambda}$:

- 1: $S.\text{Gen}(1^\lambda) \rightarrow (\text{st}_0^A, \text{st}_0^B)$
- 2: **for** $i = 1$ to m **do**
- 3: $S.\text{Enc}(\text{st}_{i-1}^A) \rightarrow (\text{st}_i^A, \text{pt}'_i, \text{ct}'_i)$
- 4: $S.\text{Dec}(\text{st}_{i-1}^B, \text{ct}'_i) \rightarrow (x, \text{pt}'_i)$
- 5: $S.\text{Enc}(x) \rightarrow (\text{st}_i^B, \text{pt}_i, \text{ct}_i)$
- 6: **end for**
- 7: $\mathcal{A}(1^\lambda, \text{st}_{m-\Delta}^A, \text{st}_{m-\Delta}^B, \text{ct}_1, \dots, \text{ct}_m) \rightarrow x$
- 8: **return** $1_{x=\text{pt}_m}$

■ **Figure 16** OW game for a simple ratcheted scheme.

<p>$P.\text{Gen}(1^\lambda) \rightarrow \text{st}$:</p> <ol style="list-style-type: none"> 1: $S.\text{Gen}(1^\lambda) \rightarrow (\text{st}_A, \text{st}_B)$ 2: $\text{st} \leftarrow (\text{st}_A, \text{st}_B)$ 3: return st <p>$P.\text{Dec}(\text{st}, \text{ct}) \rightarrow (\text{st}', \text{pt})$:</p> <ol style="list-style-type: none"> 4: parse $\text{st} = (\text{st}_A, \text{st}_B)$ 5: $S.\text{Dec}(\text{st}_A, \text{ct}) \rightarrow (\text{st}'_A, \text{pt})$ 6: $\text{st}' \leftarrow (\text{st}'_A, \text{st}_B)$ 7: return (st', pt) 	<p>$P.\text{Enc}(\text{st}) \rightarrow (\text{st}', \text{pt}, \text{ct})$:</p> <ol style="list-style-type: none"> 8: parse $\text{st} = (\text{st}_A, \text{st}_B)$ 9: $S.\text{Enc}(\text{st}_A) \rightarrow (\text{st}'_A, \text{pt}', \text{ct}')$ 10: $S.\text{Dec}(\text{st}_B, \text{ct}') \rightarrow (\text{st}''_B, \text{pt}'')$ 11: $S.\text{Enc}(\text{st}''_B) \rightarrow (\text{st}''_B, \text{pt}, \text{ct})$ 12: $\text{st}' \leftarrow (\text{st}'_A, \text{st}''_B)$ 13: return $(\text{st}', \text{pt}, \text{ct})$
--	--

■ **Figure 17** Simple ratchet S to SEQ.

from a direct translation of definitions. Furthermore, any uniform adversary against P (in the sense of Def. 2) translates into an adversary against S in the sense of Def. 10: guess m then given $(\text{st}_{m-\Delta}^A, \text{st}_{m-\Delta}^B)$ the adversary decrypts ct_m . We conclude by applying Cor. 4. ◀

5 Conclusion

We defined a self-encryption mechanism involving a device which encrypts a secret message for herself to use in the future. We are interested in security when the state of a device in such settings leaks causing the leakage of the secret message. We started giving some instances where self-ratcheting finds applications in cloud storage, when a client encrypts files to be stored, and in 0-RTT session resumption, when a server encrypts a resumption key to be kept by the client. Unlike previous works which focused on forward security and resistance to replay attacks, we studied how to add post-compromise security, as well.

We first proved that post-compromise security implies a super-linear state size in terms of the number of ciphertexts which can still be decrypted by the state. We then give formal definitions of self-ratchet. We finally showed how to design a secure scheme satisfying our bound on the state size.

Furthermore, we showed that our results on the growth of state size matches with existing secure bidirectional secure messaging applications. Given the fact that the messaging applications provide different level of PCS, we observed that there exist some protocols such as ACD without growing state size. It is due to the fact that the protocol is secure with a weaker notion of PCS which could allow constant-size states. It would be interesting to investigate weaker PCS notions in self-encryption applications such as cloud storage or 0-RTT.

References

- 1 Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol. In *Advances in Cryptology – EUROCRYPT 2019*, LNCS. Springer, 2019. doi:10.1007/978-3-030-17653-2_5.
- 2 Nimrod Aviram, Kai Gellert, and Tibor Jager. Session Resumption Protocols and Efficient Forward Security for TLS 1.3 0-RTT. In *Advances in Cryptology – EUROCRYPT 2019*, LNCS. Springer, 2019. doi:10.1007/978-3-030-17656-3_5.
- 3 Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key Homomorphic PRFs and Their Applications . In *Advances in Cryptology – CRYPTO 2013*, LNCS. Springer, 2013. doi:10.1007/978-3-642-40041-4_23.
- 4 Andrea Caforio, F. Betül Durak, and Serge Vaudenay. Beyond security and efficiency: On-demand ratcheting with security awareness. In *Public Key Cryptography – PKC 2021*, LNCS. Springer, 2021. Full version: Cryptology ePrint Archive, Report 2019/965 <https://eprint.iacr.org/2019/965>. doi:10.1007/978-3-030-75248-4_23.
- 5 Jan Camenisch and Anna Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *Advances in Cryptology - CRYPTO 2002*, LNCS. Springer, 2002. doi:10.1007/3-540-45708-9_5.
- 6 Katriel Cohn-Gordon, Cas Cremers, and Luke Garratt. On post-compromise security. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 164–178, June 2016. Full version: Cryptology ePrint Archive, Report 2016/221 <https://eprint.iacr.org/2016/221>. doi:10.1109/CSF.2016.19.
- 7 David Derler, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Bloom Filter Encryption and Applications to Efficient Forward-Secret 0-RTT Key Exchange. In *Advances in Cryptology – EUROCRYPT 2018*, LNCS. Springer, 2018. doi:10.1007/s00145-021-09374-3.
- 8 F. Betül Durak and Serge Vaudenay. Bidirectional Asynchronous Ratcheted Key Agreement with Linear Complexity. In *Advances in Information and Computer Security – IWSEC 2019*, LNCS. Springer, 2019. Full version: Cryptology ePrint Archive, Report 2018/889 <https://eprint.iacr.org/2018/889>. doi:10.1007/978-3-030-26834-3_20.
- 9 Adam Everspaugh, Kenneth Paterson, Thomas Ristenpart, and Sam Scott. Key Rotation for Authenticated Encryption. In *Advances in Cryptology – CRYPTO 2017*, LNCS. Springer, 2017. doi:10.1007/978-3-319-63697-9_4.
- 10 Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-RTT Key Exchange with Full Forward Secrecy. In *Advances in Cryptology – EUROCRYPT 2017*, LNCS. Springer, 2017. doi:10.1007/978-3-319-56617-7_18.
- 11 Ralph Holz, Johanna Amann, Abbas Razaghpanah, and Narseo Vallina-Rodriguez. The Era of TLS 1.3: Measuring Deployment and Use with Active and Passive Methods. *CoRR*, 2019. URL: <http://arxiv.org/abs/1907.12762>.
- 12 Joseph Jaeger and Igors Stepanovs. Optimal Channel Security Against Fine-Grained State Compromise: The Safety of Messaging . In *Advances in Cryptology – CRYPTO 2018*, LNCS. Springer, 2018. doi:10.1007/978-3-319-96884-1_2.
- 13 Daniel Jost, Ueli Maurer, and Marta Mularczyk. Efficient Ratcheting: Almost-Optimal Guarantees for Secure Messaging. In *Advances in Cryptology – EUROCRYPT 2019*, LNCS. Springer, 2019. doi:10.1007/978-3-030-17653-2_6.
- 14 Bertram Poettering and Paul Rösler. Towards bidirectional ratcheted key exchange. In *Advances in Cryptology – CRYPTO 2018*, LNCS. Springer, 2018. doi:10.1007/978-3-319-96884-1_1.
- 15 Open Whisper Systems. Signal protocol library for Java/Android. GitHub repository <https://github.com/WhisperSystems/libsignal-protocol-java>, 2017.

Generic-Group Identity-Based Encryption: A Tight Impossibility Result

Gili Schul-Ganz ✉

School of Computer Science and Engineering, Hebrew University of Jerusalem, Israel

Gil Segev ✉

School of Computer Science and Engineering, Hebrew University of Jerusalem, Israel

Abstract

Following the pioneering work of Boneh and Franklin (CRYPTO '01), the challenge of constructing an identity-based encryption scheme based on the Diffie-Hellman assumption remained unresolved for more than 15 years. Evidence supporting this lack of success was provided by Papakonstantinou, Rackoff and Vahlis (ePrint '12), who ruled out the existence of generic-group identity-based encryption schemes supporting an identity space of sufficiently large polynomial size. Nevertheless, the breakthrough result of Döttling and Garg (CRYPTO '17) settled this long-standing challenge via a non-generic construction.

We prove a tight impossibility result for generic-group identity-based encryption, ruling out the existence of any non-trivial construction: We show that any scheme whose public parameters include n_{pp} group elements may support at most n_{pp} identities. This threshold is trivially met by any generic-group public-key encryption scheme whose public keys consist of a single group element (e.g., ElGamal encryption).

In the context of algebraic constructions, generic realizations are often both conceptually simpler and more efficient than non-generic ones. Thus, identifying exact thresholds for the limitations of generic groups is not only of theoretical significance but may in fact have practical implications when considering concrete security parameters.

2012 ACM Subject Classification Security and privacy → Cryptography; Theory of computation → Cryptographic primitives

Keywords and phrases Identity-based encryption, generic-group model

Digital Object Identifier 10.4230/LIPIcs.ITC.2021.26

Funding Supported by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253).

1 Introduction

Identity-based encryption [16, 5, 9] is one of the key pillars underlying modern cryptography, enabling a variety of access-control applications and paving a path towards more expressive forms of encryption schemes. Starting with the first realizations of identity-based encryption schemes by Boneh and Franklin [5] (based on the bilinear Diffie-Hellman assumption) and Cocks [9] (based on the quadratic residuosity assumption) in the random-oracle model [2], extensive research has been devoted to constructing such schemes in the standard model (e.g., [7, 3, 4, 18]) and based on other cryptographic assumptions (e.g., [12, 8, 1]).

Despite the significant progress, a substantial gap remained for nearly two decades between the cryptographic assumptions that are known to imply public-key encryption and those that are known to imply identity-based encryption. This gap was first studied by Boneh, Papakonstantinou, Rackoff, Vahlis, and Waters [6] who showed that identity-based encryption cannot be realized in a black-box manner based on trapdoor permutations or CCA-secure public-key encryption. Then, Papakonstantinou, Rackoff and Vahlis [15] studied the possibility of constructing generic-group identity-based encryption schemes



© Gili Schul-Ganz and Gil Segev;
licensed under Creative Commons License CC-BY 4.0
2nd Conference on Information-Theoretic Cryptography (ITC 2021).
Editor: Stefano Tessaro; Article No. 26; pp. 26:1–26:23



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

(i.e., identity-based encryption schemes that do not exploit any particular property of the representation of the underlying group [17, 14]). They showed that there are no generic-group constructions of identity-based encryption schemes supporting an identity space of sufficiently large polynomial size. The result of Papakonstantinou, Rackoff and Vahlis explained, in particular, the lack of success in resolving the long-standing open problem of constructing an identity-based encryption scheme based on the Diffie-Hellman assumption. Nevertheless, the recent breakthrough of Döttling and Garg [11, 10] settled this open problem via a non-generic construction.

Our contribution: A tight impossibility result for generic-group IBE. In the context of algebraic constructions, generic realizations are often both conceptually simpler and more efficient than non-generic ones. Thus, identifying exact thresholds for the limitations of generic groups is not only of theoretical significance but may in fact have practical implications when considering concrete security parameters.

For identity-based encryption schemes, such a potential threshold naturally arises by comparing the size of the scheme’s identity space to the number of group elements that are included in the scheme’s public parameters. Specifically, for any $n_{pp} \geq 1$, already ElGamal encryption yields a generic-group identity-based encryption scheme that supports n_{pp} identities and whose public parameters consist of n_{pp} group elements (not including the group’s generator). However, the work of Papakonstantinou, Rackoff and Vahlis [15] only ruled out the existence of generic-group identity-based encryption schemes over an identity space of sufficiently large polynomial size¹.

We prove a tight impossibility result for constructing generic-group identity-based encryption schemes, showing that any such scheme whose public parameters consist of n_{pp} group elements may support up to n_{pp} identities. This matches the above-mentioned naive threshold that is obtained via ElGamal encryption, and more generally via any generic-group public-key encryption scheme whose public keys consist of a single group element. We prove the following theorem:

► **Theorem 1 (Simplified).** *Let IBE be a secure generic-group identity-based encryption scheme over an identity space $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$ whose public parameters consist of $n_{pp}(\lambda)$ group elements, where $\lambda \in \mathbb{N}$ is the security parameter. Then, $|\mathcal{ID}_\lambda| \leq n_{pp}(\lambda)$ for all sufficiently large $\lambda \in \mathbb{N}$.*

We prove our result by presenting a generic-group adversary that breaks the security of any identity-based encryption scheme whose public parameters consist of n_{pp} group elements and supports more than n_{pp} identities. Our result applies to schemes satisfying a rather weak (non-adaptive) notion of security (thus ruling out, in particular, schemes that satisfy more standard notions of security), and to schemes with imperfect correctness.

Compared to the work of Papakonstantinou, Rackoff and Vahlis [15], on the one hand our proof follows a similar two-step structure: We first show that any generic-group identity-based encryption scheme can be transformed into one in which secret keys do not contain group elements, and then we present an attack on any such scheme that supports more identities than the number of group elements included in its public parameters. On the other hand, however, our result does not only provide a tight impossibility result, but in fact provides a somewhat more direct technical description of our attack and of its analysis. Such a description is enabled partially due to the fact that we prove our result within Maurer’s

¹ Papakonstantinou et al. proved their result for an identity space of exponential size, but their proof seems to hold for an identity space of sufficiently large polynomial size.

generic-group model [14], whereas Papakonstantinou et al. proved their result within Shoup’s incomparable generic-group model [17], as discussed in Section 1.1 (e.g., in Maurer’s model we do not have to take into account the additional randomness that is somewhat artificially “injected” into cryptographic computations in Shoup’s model due to its random injective encoding of group elements).

Specifically, for our first step, our transformation for eliminating group elements from secret keys is essentially identical to the corresponding transformation of Papakonstantinou et al. and is provided together with a significantly more direct analysis. For our second step, our attack is based on that of Papakonstantinou et al. which relies on the common technique of attacking the security of an idealized-model scheme relative to a partly-simulated view of the model. Unlike our first step, in this step our attack and its analysis simultaneously refine and simplify those of Papakonstantinou et al. for obtaining a tight bound.

1.1 Overview of Our Approach

The framework. We prove our result within the generic-group model introduced by Maurer [14], which together with the incomparable model introduced by Shoup [17], seem to be the most commonly used approaches for capturing generic-group computations. At a high level, in both models algorithms have access to an oracle \mathcal{O} for performing the group operation and for testing whether two group elements are equal. The difference between the two models is in the way that algorithms specify their queries to the oracle. In Maurer’s model algorithms specify their queries by pointing to two group elements that have appeared in the computation so far (e.g., the 4th and the 7th group elements), whereas in Shoup’s model group elements have an explicit representation (sampled uniformly at random from the set of all injective mappings from the group to sufficiently long strings) and algorithms specify their queries by providing two strings that have appeared in the computation so far as encodings of group elements.

Jager and Schwenk [13] proved that the complexity of any computational problem that is defined in a manner that is independent of the representation of the underlying group (e.g., computing discrete logarithms) in one model is essentially equivalent to its complexity in the other model. More generally, however, these two models are rather incomparable. On one hand, the class of cryptographic schemes that are captured by Maurer’s model is a subclass of that of Shoup’s model – although as demonstrated by Maurer his model still captures all schemes that only use the abstract group operation and test whether two group elements are equal. On the other hand, the same holds also for the class of adversaries, and thus in Maurer’s model we have to break the security of a given scheme using an adversary that is more restricted when compared to adversaries in Shoup’s model. We refer the reader to Section 2.1 for a formal description of Maurer’s generic-group model.

Generic-group identity-based encryption. A generic-group identity-based encryption scheme IBE over an identity space \mathcal{ID} consists of four algorithms, denoted Setup , KG , Enc and Dec . Informally (and quite briefly), the algorithm Setup produces a master secret key $\text{msk} \in \{0,1\}^*$ and public parameters pp , and the algorithm KG on input the master secret msk and an identity $id \in \mathcal{ID}$ produces a secret key sk_{id} . Next, the algorithm Enc on input public parameters pp , an identity $id \in \mathcal{ID}$ and a message $b \in \{0,1\}$, produces a ciphertext c , which should be correctly decrypted (allowing decryption error) by the decryption algorithm Dec using the secret key sk_{id} . The outputs of these four algorithms may consist of a combination of group elements and an explicit string, with the exception of assuming without loss of generality that the master secret key msk is always an explicit string (e.g., the internal randomness on which Setup is invoked).

The structure of our proof. We prove our result by presenting a generic-group adversary that breaks the security of any identity-based encryption scheme whose public parameters pp consist of n_{pp} group elements (and, possibly, an additional explicit string) and supports more than n_{pp} identities. As mentioned above, at a high level, we follow a two-step structure similar to that introduced in the work of Papakonstantinou et al. [15]: We first show that any generic-group identity-based encryption scheme can be transformed into one in which secret keys do not contain group elements (while modifying only its key-generation and decryption algorithms), and then we present an attack on any such scheme that supports more identities than the number of group elements included in its public parameters. The remainder of this section consists of a high-level informal description of these two steps (we note that the following description omits crucial technical details, and we refer the reader to the relevant sections for formal descriptions and proofs).

In what follows, given a generic-group identity-based encryption scheme we let $pp_1, \dots, pp_{n_{\text{pp}}}, sk_{id,1}, \dots, sk_{id,n_{\text{sk}}}$ and $c_1, \dots, c_{n_{\text{ct}}}$ denote the group elements included in its public parameters pp and in each of its secret keys sk_{id} and ciphertexts c , respectively (for simplicity, we assume throughout this informal overview that public parameters, secret keys and ciphertexts do not additionally contain explicit strings).

Step I: Eliminating group elements from secret keys. Given a generic-group identity-based encryption scheme $\mathcal{IBE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$, we modify its key-generation algorithm KG and decryption algorithm Dec as follows:

- The modified key-generation algorithm $\widetilde{\text{KG}}$ on input the public parameters pp , the master secret key $\text{msk} \in \{0, 1\}^*$ and an identity $id \in \mathcal{ID}$, first produces a secret key sk_{id} by invoking the underlying key-generation algorithm KG . Then, for each message $b \in \{0, 1\}$, it repeatedly computes $\text{Dec}^{\mathcal{O}}(\text{pp}, sk_{id}, \text{Enc}^{\mathcal{O}}(\text{pp}, id, b))$ using fresh randomness for Enc and Dec , and collects into a set \mathcal{L}_{id} all linear equations that result from the positively-answered equality queries in these computations. Note that since the group elements that are given as input to these computations are those included in pp and sk_{id} (as well as the generator $1 \in \mathbb{Z}_N$ that is given as input to all computations), then each such equation is of the form $\alpha_0 \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} \alpha_{\ell} \cdot pp_{\ell} + \sum_{\ell=1}^{n_{\text{sk}}} \beta_{\ell} \cdot sk_{id,\ell} = 0$ for some coefficients $\alpha_0, \dots, \alpha_{n_{\text{pp}}}, \beta_1, \dots, \beta_{n_{\text{sk}}} \in \mathbb{Z}_N$. The algorithm then outputs the modified secret key $\widetilde{sk}_{id} = \mathcal{L}_{id}$ which consists of $(n_{\text{pp}} + n_{\text{sk}} + 1)$ -dimensional vectors of coefficients over \mathbb{Z}_N (and does not contain group elements).
- The modified decryption algorithm $\widetilde{\text{Dec}}$ on input the public parameters pp , a modified secret key $\widetilde{sk}_{id} = \mathcal{L}_{id}$ and a ciphertext c , emulates the computation $\text{Dec}^{\mathcal{O}}(\text{pp}, sk_{id}, c)$ using symbolic variables instead of the group elements $sk_{id,1}, \dots, sk_{id,n_{\text{sk}}}$ included in the secret key sk_{id} . As long as it is able to obtain and to respond with the correct answer to all emulated equality queries, then the emulation will be identical to the actual computation $\text{Dec}^{\mathcal{O}}(\text{pp}, sk_{id}, c)$.

Note that since the group elements that are given as input to the actual computation are those included in pp , sk_{id} and c (as well as the generator $1 \in \mathbb{Z}_N$), then each emulated equality query corresponds to a linear equation of the form $\alpha_0 \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} \alpha_{\ell} \cdot pp_{\ell} + \sum_{\ell=1}^{n_{\text{sk}}} \beta_{\ell} \cdot sk_{id,\ell} + \sum_{\ell=1}^{n_{\text{ct}}} \gamma_{\ell} \cdot c_{\ell} = 0$, for coefficients $\alpha_0, \dots, \alpha_{n_{\text{pp}}}, \beta_1, \dots, \beta_{n_{\text{sk}}}, \gamma_1, \dots, \gamma_{n_{\text{ct}}} \in \mathbb{Z}_N$. Now, the algorithm $\widetilde{\text{Dec}}$ uses the set \mathcal{L}_{id} and the actual oracle \mathcal{O} for responding to this query as follows. If there exist $\alpha'_0, \dots, \alpha'_{n_{\text{pp}}} \in \mathbb{Z}_N$ such that $(\alpha'_0, \dots, \alpha'_{n_{\text{pp}}}, \beta_1, \dots, \beta_{n_{\text{sk}}}) \in \text{span}(\mathcal{L}_{id})$, then $\widetilde{\text{Dec}}$ issues to the actual oracle \mathcal{O} an equality queries corresponding to the linear equation $(\alpha_0 - \alpha'_0) \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} (\alpha_{\ell} - \alpha'_{\ell}) \cdot pp_{\ell} + \sum_{\ell=1}^{n_{\text{ct}}} \gamma_{\ell} \cdot c_{\ell} = 0$, and return its output as the response. If there do not exist such $\alpha'_0, \dots, \alpha'_{n_{\text{pp}}} \in \mathbb{Z}_N$, then $\widetilde{\text{Dec}}$ responds negatively. In other words, the algorithm $\widetilde{\text{Dec}}$ uses the knowledge provided by the set \mathcal{L}_{id} in order to

translate each equality query involving the group elements of pp , sk_{id} and c into an equality queries that involves the group elements of only pp and c . A simple probabilistic argument (see Claim 6) shows that this translation introduces only an arbitrary polynomially-small decryption error $1/p(\lambda)$ when setting the number of iterations performed by the modified key-generation algorithm to $p(\lambda) \cdot (n_{\text{pp}}(\lambda) + n_{\text{sk}}(\lambda))$.

Step II: Our attack. Let $\mathcal{IBE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$ be a generic-group identity-based encryption scheme over an identity space \mathcal{ID} whose public parameters consist of n_{pp} group elements, whose secret keys do not contain group elements, and that supports at least $n_{\text{pp}} + 1$ identities. For simplicity and without loss of generality we assume that $\{1, \dots, n_{\text{pp}} + 1\} \subseteq \mathcal{ID}$.

The key observation underlying our attack is based on considering the set of linear equations that result from the positively-answered equality queries in the computations $\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_{id}, \text{Enc}^{\mathcal{O}}(\text{pp}, id, b))$ for each message $b \in \{0, 1\}$ and identity $id \in \{1, \dots, n_{\text{pp}} + 1\}$. Given that the secret keys sk_{id} do not contain any group elements, then the group elements that are given as input to these computations are only those that are included in the public parameters pp (as well as the generator $1 \in \mathbb{Z}_N$ that is given as input to all computations). Thus, each such equation is of the form $\alpha_0 \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} \alpha_{\ell} \cdot \text{pp}_{\ell} = 0$ for some coefficients $\alpha_0, \dots, \alpha_{n_{\text{pp}}} \in \mathbb{Z}_N$. Given that $(1, \text{pp}_1, \dots, \text{pp}_{n_{\text{pp}}})$ is a non-zero vector, then the vectors of coefficients of these sets of equations span a linear subspace of dimension at most n_{pp} .

Therefore, for at least one identity $id \in \{1, \dots, n_{\text{pp}} + 1\}$, it must be the case that the set of linear equations that result from the positively-answered equality queries in the computation $\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_{id}, \text{Enc}^{\mathcal{O}}(\text{pp}, id, b))$ is contained in the linear subspace spanned by the sets of linear equations that result from the positively-answered equality queries in the computations

$$\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_1, \text{Enc}^{\mathcal{O}}(\text{pp}, 1, b)), \dots, \text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_{id-1}, \text{Enc}^{\mathcal{O}}(\text{pp}, id-1, b)).$$

Moreover, once our adversary discovers this subspace by using the secret keys $\text{sk}_1, \dots, \text{sk}_{id-1}$, then it can intuitively generate alternative public parameters pp^* that are consistent with this subspace, together with a matching alternative master secret key msk^* . Then, it uses the alternative public parameters pp^* and master secret key msk^* for generating an alternative secret key sk_{id}^* for decrypting the challenge ciphertext. The correctness of the scheme guarantees that, with high probability, sk_{id}^* will decrypt correctly a ciphertext that is encrypted and decrypted relative to pp^* , and we show that sk_{id}^* is in fact useful also when encrypting and decrypting relative to pp .

1.2 Paper Organization

The remainder of this paper is organized as follows. First, in Section 2 we present the basic notation used throughout the paper, and formally describe the framework of generic-group identity-based encryption. Then, in Section 3 we show that any generic-group identity-based encryption scheme can be transformed into one in which secret keys do not contain group elements. Finally, in Section 4 we present an attack on any generic-group identity-based encryption scheme whose secret keys do not contain group elements, and that supports more identities than the number of group elements included in its public parameters.

2 Preliminaries

In this section we present the basic notions and standard cryptographic tools that are used in this work. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X . Similarly, for a set \mathcal{X} we denote by $x \leftarrow \mathcal{X}$ the process of sampling

a value x from the uniform distribution over \mathcal{X} . For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \dots, n\}$. A function $\nu : \mathbb{N} \rightarrow \mathbb{R}^+$ is *negligible* if for any polynomial $p(\cdot)$ there exists an integer N such that for all $n > N$ it holds that $\nu(n) \leq 1/p(n)$.

2.1 Generic Groups and Algorithms

We prove our results within the generic-group model introduced by Maurer [14]. We consider computations in cyclic groups of order N (all of which are isomorphic to \mathbb{Z}_N with respect to addition modulo N), for a λ -bit prime N that is generated by an order-generation algorithm $\text{PrimeGen}(1^\lambda)$, where $\lambda \in \mathbb{N}$ is the security parameter.

When considering such groups, each computation in Maurer’s model is associated with a table \mathbf{B} . Each entry of this table stores an element of \mathbb{Z}_N , and we denote by V_i the group element that is stored in the i th entry. Generic algorithms access this table via an oracle \mathcal{O} , providing black-box access to \mathbf{B} as follows. A generic algorithm \mathcal{A} that takes d group elements as input (along with an optional bit-string) does not receive an explicit representation of these group elements, but instead, has oracle access to the table \mathbf{B} , whose first d entries store the \mathbb{Z}_N elements corresponding to the d group elements in \mathcal{A} ’s input. That is, if the input of an algorithm \mathcal{A} is a tuple (g_1, \dots, g_d, x) , where g_1, \dots, g_d are group elements and x is an arbitrary string, then from \mathcal{A} ’s point of view the input is the tuple $(\widehat{g}_1, \dots, \widehat{g}_d, x)$, where $\widehat{g}_1, \dots, \widehat{g}_d$ are pointers to the group elements g_1, \dots, g_d (these group elements are stored in the table \mathbf{B}), and x is given explicitly.

All generic algorithms in this paper receive as input the order N and a generator of the group (we capture this fact by always assuming that the first entry of \mathbf{B} is occupied by $1 \in \mathbb{Z}_N$). The oracle \mathcal{O} allows for two types of queries:

- **Group-operation queries:** On input (i, j, \circ) for $i, j \in \mathbb{N}$ and $\circ \in \{+, -\}$, the oracle checks that the i th and j th entries of the table \mathbf{B} are not empty, computes $V_i \circ V_j \bmod N$ and stores the result in the next available entry. If either the i th or the j th entries are empty, the oracle ignores the query.
- **Equality queries:** On input $(i, j, =)$ for $i, j \in \mathbb{N}$, the oracle checks that the i th and j th entries of the table \mathbf{B} are not empty, and then returns 1 if $V_i = V_j$ and 0 otherwise. If either the i th or the j th entries are empty, the oracle ignores the query.

In this paper we consider interactive computations in which multiple algorithms pass group elements (as well as non-group elements) as inputs to one another. This is naturally supported by the model as follows: When a generic algorithm \mathcal{A} outputs k group elements (along with a potential bit-string σ), it outputs the indices of k (non-empty) entries in the table \mathbf{B} (together with σ). When these outputs (or some of them) are passed on as inputs to a generic algorithm \mathcal{C} , the table \mathbf{B} is re-initialized, and these values (and possibly additional group elements that \mathcal{C} receives as input) are placed in the first entries of the table. Additionally, we rely on the following conventions:

1. Throughout the paper we refer to values as either “explicit” ones or “implicit” ones. Explicit values are all values whose representation (e.g., binary strings of a certain length) is explicitly provided to the generic algorithms under consideration. Implicit values are all values that correspond to group elements and that are stored in the table \mathbf{B} – thus generic algorithms can access them only via oracle queries. We will sometimes interchange between providing group elements as input to generic algorithms implicitly, and providing them explicitly. Note that moving from the former to the latter is well defined, since a generic algorithm \mathcal{A} that receives some of its input group elements explicitly can always simulate the computation as if they were received as part of the table \mathbf{B} .

2. For a group element g , we will differentiate between the case where g is provided explicitly and the case where it is provided implicitly via the table \mathbf{B} , using the notation g in the former case, and the notation \hat{g} in the latter. Additionally, we extend this notation to a vector v of group elements, which may be provided either explicitly (denoted v) or implicitly via the table \mathbf{B} (denoted \hat{v}).

2.2 Generic-Group Identity-Based Encryption

The following definition adapts the standard notion of an identity-based encryption scheme to the generic-group model.

► **Definition 2.** A generic-group identity-based encryption scheme over an identity space $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$ is a quadruple $\mathcal{IBE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$ of generic algorithms defined as follows:

- The algorithm **Setup** is a probabilistic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$ and the group order N , and outputs a master secret key $\text{msk} \in \{0, 1\}^*$ and public parameters $\text{pp} = (\text{pp}_G, \text{pp}_{\text{str}})$, where pp_G is a tuple of n_{pp} group elements and pp_{str} is a binary string.
- The algorithm **KG** is a (potentially) probabilistic algorithm that receives as input public parameters pp , a master secret key msk and an identity id . It outputs an identity secret key $\text{sk}_{id} = (\text{sk}_{id,G}, \text{sk}_{id,\text{str}})$, where $\text{sk}_{id,G}$ is a tuple of group elements and $\text{sk}_{id,\text{str}}$ is a binary string.
- The algorithm **Enc** is a probabilistic algorithm that receives as input public parameters pp , an identity id , and a bit $b \in \{0, 1\}$. It outputs a ciphertext $c = (c_G, c_{\text{str}})$, where c_G is a tuple of group elements and c_{str} is a binary string.
- The algorithm **Dec** is a (potentially) probabilistic algorithm that receives as input public parameters pp , an identity secret key sk_{id} , and a ciphertext c . It outputs either a bit $b \in \{0, 1\}$ or the special rejection symbol \perp .

We consider the standard correctness and security requirements of identity-based encryption schemes. In fact, we consider a rather weak notion of non-adaptive security asking the attacker to choose both the challenge identity and the identities for which secret keys are provided ahead of time (since we prove an impossibility result then this can only strengthen our result).

► **Definition 3.** A generic-group identity-based encryption scheme $\mathcal{IBE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$ over an identity space $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$ has decryption error $\epsilon = \epsilon(\lambda)$ if for any security parameter $\lambda \in \mathbb{N}$, for any N produced by $\text{PrimeGen}(1^\lambda)$, for any (msk, pp) produced by $\text{Setup}^\circ(1^\lambda)$, for any $id \in \mathcal{ID}_\lambda$, and for any $b \in \{0, 1\}$ it holds that

$$\Pr \left[\text{Dec}^\circ(\text{pp}, \text{sk}_{id}, \text{Enc}^\circ(\text{pp}, id, b)) = b \right] \geq 1 - \epsilon$$

where $\text{sk}_{id} \leftarrow \text{KG}^\circ(\text{pp}, \text{msk}, id)$, and the probability is taken over the internal randomness of the algorithms **KG**, **Enc** and **Dec**.

We note that our results can be easily adapted to a more relaxed notion of correctness, asking that the above holds for almost all (msk, pp) produced by $\text{Setup}^\circ(1^\lambda)$ instead of for all such (msk, pp) .

► **Definition 4.** A generic-group identity-based encryption scheme $\mathcal{IBE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$ over an identity space $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$ is non-adaptively secure if for any generic-group algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that issues a polynomial number of queries there exists a negligible function $\nu(\lambda)$ such that

$$\left| \Pr [\text{Expt}_{\mathcal{IBE}, \mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the experiment $\text{Expt}_{\mathcal{IBE}, \mathcal{A}}(\lambda)$ is defined as follows:

1. $N \leftarrow \text{PrimeGen}(1^\lambda)$.
2. $(id^*, id_1, \dots, id_k, \text{state}) \leftarrow \mathcal{A}_1^{\mathcal{O}}(1^\lambda, N)$, for a polynomial $k = k(\lambda)$, where $id^*, id_1, \dots, id_k \in \mathcal{ID}_\lambda$ and $id^* \notin \{id_1, \dots, id_k\}$.
3. $(\text{msk}, \text{pp}) \leftarrow \text{Setup}^{\mathcal{O}}(1^\lambda, N)$.
4. $\text{sk}_{id_i} \leftarrow \text{KG}^{\mathcal{O}}(\text{pp}, \text{msk}, id_i)$ for $i \in [k]$.
5. $c^* \leftarrow \text{Enc}^{\mathcal{O}}(\text{pp}, id^*, b)$ for $b \leftarrow \{0, 1\}$.
6. $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(\text{state}, \text{pp}, c^*, \text{sk}_{id_1}, \dots, \text{sk}_{id_k})$.
7. If $b' = b$ then output 1, and otherwise output 0.

3 Eliminating Group Elements From Secret Keys

In this section we show that any generic-group identity-based encryption scheme can be transformed into one in which secret keys do not contain group elements. The transformation supports the same identity space, and does not modify the scheme's setup and encryption procedures (in particular, it does not increase the number of group elements that are contained in the scheme's public parameters). The transformation does modify the scheme's key-generation and decryption algorithms, leading to an arbitrary polynomially-small increase in the scheme's decryption error. We prove the following theorem:

► **Theorem 5.** Let \mathcal{IBE} be a generic-group identity-based encryption scheme over an identity space $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$ with decryption error $\epsilon(\lambda)$ and whose public parameters consist of $n_{\text{pp}}(\lambda)$ group elements. Then, for any polynomial $p(\lambda)$, there exists a generic-group identity-based encryption scheme $\widetilde{\mathcal{IBE}}$ over the identity space \mathcal{ID} with decryption error $\epsilon(\lambda) + 1/p(\lambda)$, whose public parameters consist of $n_{\text{pp}}(\lambda)$ group elements, and whose secret keys do not contain group elements.

Preliminaries. Let A be a generic-group algorithm that receives as input group elements g_1, \dots, g_k (in addition to the group element $1 \in \mathbb{Z}_N$ that is always provided as the first input to all algorithms) and an explicit string str . We let $\mathcal{EQ}(A^{\mathcal{O}}(\widehat{g}_1, \dots, \widehat{g}_k, \text{str}))$ denote the random variable corresponding to the set of all $(k+1)$ -dimensional vectors over \mathbb{Z}_N resulting from the positively-answered equality queries in the (possibly randomized) computation $A^{\mathcal{O}}(\widehat{g}_1, \dots, \widehat{g}_k, \text{str})$.

Formally, for each equality query (i, j) that is positively answered during this computation, let V_i and V_j denote the group elements that are located in the i th and j th entries of the table \mathbf{B} associated with oracle \mathcal{O} in this computation (i.e., V_i and V_j are the two group elements for which A issues this equality query). Then, V_i and V_j are linear combinations of the group elements $1, g_1, \dots, g_k$ that are provided as input to the computation, and the coefficients of these linear combinations can be determined by keeping track of the computation's group-operation queries. Let $V_i - V_j = \alpha_0 \cdot 1 + \sum_{\ell=1}^k \alpha_\ell \cdot g_\ell$ for $\alpha_0, \dots, \alpha_k \in \mathbb{Z}_N$. The set $\mathcal{EQ}(A^{\mathcal{O}}(\widehat{g}_1, \dots, \widehat{g}_k, \text{str}))$ consists of all such vectors $(\alpha_0, \dots, \alpha_k) \in \mathbb{Z}_N^{k+1}$.

In addition, for a generic-group identity-based encryption $\mathcal{IBE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$, and for any public parameters pp produced by Setup we let $\text{pp} = (pp_1, \dots, pp_{n_{\text{pp}}}, \text{pp}_{\text{str}})$, where $pp_1, \dots, pp_{n_{\text{pp}}}$ are group elements and pp_{str} is an explicit string (recall that any msk produced by Setup is an explicit string). Similarly, for any secret key sk_{id} produced by KG we let $\text{sk}_{id} = (sk_{id,1}, \dots, sk_{id,n_{\text{sk}}}, \text{sk}_{id,\text{str}})$ where $sk_{id,1}, \dots, sk_{id,n_{\text{sk}}}$ are group elements and $\text{sk}_{id,\text{str}}$ is an explicit string, and for any ciphertext c produced by Enc we let $c = (c_1, \dots, c_{n_{\text{ct}}}, c_{\text{str}})$, where $c_1, \dots, c_{n_{\text{ct}}}$ are group elements and c_{str} is an explicit string.

Finally, our proof relies on the following lemma (which is proved in Appendix A):

► **Lemma 6.** *Let $k \geq 1$, and let X_1, \dots, X_k be independent and identically distributed random variables over subsets of a linear vector space V of dimension $\dim(V)$. Then,*

$$\Pr[X_k \notin \text{span}(X_1 \cup \dots \cup X_{k-1})] \leq \frac{\dim(V)}{k}.$$

The remainder of this section consists of the proof of Theorem 5.

Proof of Theorem 5. Let $\mathcal{IBE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$, and let $p = p(\lambda)$ be a polynomial. We construct a generic-group identity-based encryption scheme $\widetilde{\mathcal{IBE}} = (\widetilde{\text{Setup}}, \widetilde{\text{KG}}, \widetilde{\text{Enc}}, \widetilde{\text{Dec}})$ by letting $\widetilde{\text{Setup}} = \text{Setup}$ and $\widetilde{\text{Enc}} = \text{Enc}$, and by defining the algorithms $\widetilde{\text{KG}}$ and $\widetilde{\text{Dec}}$ as follows.

The key-generation algorithm $\widetilde{\text{KG}}^{\circ}(\text{pp}, \text{msk}, id)$:

1. Generate $\text{sk}_{id} = (\widehat{sk_{id,1}}, \dots, \widehat{sk_{id,n_{\text{sk}}}}, \text{sk}_{id,\text{str}}) \leftarrow \text{KG}^{\circ}(\text{pp}, \text{msk}, id)$.
2. For every $b \in \{0, 1\}$ and $i \in [M - 1]$, where $M = p \cdot (n_{\text{pp}} + n_{\text{sk}})$, compute $\text{Dec}^{\circ}(\text{pp}, \text{sk}_{id}, \text{Enc}^{\circ}(\text{pp}, id, b))$ using fresh randomness for Enc and Dec , and let

$$\mathcal{L}_{id,b,i} = \mathcal{EQ}(\text{Dec}^{\circ}(\text{pp}, \text{sk}_{id}, \text{Enc}^{\circ}(\text{pp}, id, b))) \subseteq \mathbb{Z}_N^{n_{\text{pp}} + n_{\text{sk}} + 1}.$$

3. Output $\widetilde{\text{sk}}_{id} = (\mathcal{L}_{id}, \text{sk}_{id,\text{str}})$, where $\mathcal{L}_{id} = \bigcup_{b \in \{0,1\}, i \in [M-1]} \mathcal{L}_{id,b,i}$.

The decryption algorithm $\widetilde{\text{Dec}}^{\circ}(\text{pp}, \widetilde{\text{sk}}_{id}, c)$:

1. Let $\text{pp} = (\widehat{pp}_1, \dots, \widehat{pp}_{n_{\text{pp}}}, \text{pp}_{\text{str}})$, $\widetilde{\text{sk}}_{id} = (\mathcal{L}_{id}, \text{sk}_{id,\text{str}})$, and $c = (\widehat{c}_1, \dots, \widehat{c}_{n_{\text{ct}}}, c_{\text{str}})$.
2. Emulate the computation $\text{Dec}^{\circ}(\text{pp}, \text{sk}_{id}, c)$ using symbolic variables instead of $sk_{id,1}, \dots, sk_{id,n_{\text{sk}}}$ (recall that $\text{sk}_{id} = (\widehat{sk_{id,1}}, \dots, \widehat{sk_{id,n_{\text{sk}}}}, \text{sk}_{id,\text{str}})$) by responding to each equality query (i, j) as follows:

- a. Let V_i and V_j denote the corresponding group elements, and let

$$V_i - V_j = \alpha_0 \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} \alpha_{\ell} \cdot pp_{\ell} + \sum_{\ell=1}^{n_{\text{sk}}} \beta_{\ell} \cdot sk_{id,\ell} + \sum_{\ell=1}^{n_{\text{ct}}} \gamma_{\ell} \cdot c_{\ell},$$

where $\alpha_0, \dots, \alpha_{n_{\text{pp}}}, \beta_1, \dots, \beta_{n_{\text{sk}}}, \gamma_1, \dots, \gamma_{n_{\text{ct}}} \in \mathbb{Z}_N$ (as explained above, these coefficients can be found by keeping track of the emulated computation's group-operation queries).

- b. If there exist $\alpha'_0, \dots, \alpha'_{n_{\text{pp}}} \in \mathbb{Z}_N$ such that $(\alpha'_0, \dots, \alpha'_{n_{\text{pp}}}, \beta_1, \dots, \beta_{n_{\text{sk}}}) \in \text{span}(\mathcal{L}_{id})$, then issue group-operation queries for positioning the group element $W_{i,j} = (\alpha_0 - \alpha'_0) \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} (\alpha_{\ell} - \alpha'_{\ell}) \cdot pp_{\ell} + \sum_{\ell=1}^{n_{\text{ct}}} \gamma_{\ell} \cdot c_{\ell}$ in the table **B**. If $W_{i,j} = 0$ (this can be determined by issuing a single equality query), then answer the equality query (i, j) positively, and otherwise answer it negatively.

26:10 Generic-Group Identity-Based Encryption: A Tight Impossibility Result

c. If there do not exist such $\alpha'_0, \dots, \alpha'_{n_{pp}} \in \mathbb{Z}_N$, then answer the equality query (i, j) negatively.

3. Output the result of the emulated computation.

First, in terms of efficiency, note that if the algorithms $(\text{Setup}, \text{KG}, \text{Enc}, \text{Dec})$ issue at most a polynomial number of queries, then so do the algorithms $(\widetilde{\text{Setup}}, \widetilde{\text{KG}}, \widetilde{\text{Enc}}, \widetilde{\text{Dec}})$. Second, in terms of security, note that the scheme $\widetilde{\mathcal{IBE}}$ is at least as secure as the scheme \mathcal{IBE} : The schemes have the same setup and encryption algorithms, and the modified key-generation algorithm $\widetilde{\text{KG}}$ is defined by applying a poly-query procedure to the output of the underlying key-generation algorithm KG . Therefore, any adversary attacking the scheme $\widetilde{\mathcal{IBE}}$ while issuing a polynomial number of queries (recall Definition 4) can be efficiently transformed into an adversary attacking the scheme \mathcal{IBE} while issuing a polynomial number of queries and with (at least) the same advantage.

We are thus left with bounding the decryption error of the scheme $\widetilde{\mathcal{IBE}}$ (recall Definition 3). Fix a security parameter $\lambda \in \mathbb{N}$, an integer N that is produced by $\text{PrimeGen}(1^\lambda)$, a pair (msk, pp) that is produced by $\text{Setup}^\mathcal{O}(1^\lambda)$, an identity $id \in \mathcal{ID}_\lambda$, and a message $b \in \{0, 1\}$. The scheme \mathcal{IBE} has decryption error at most $\epsilon(\lambda)$, and therefore

$$\begin{aligned} & \Pr \left[\widetilde{\text{Dec}}^\mathcal{O}(\text{pp}, \widetilde{\text{sk}}_{id}, \text{Enc}^\mathcal{O}(\text{pp}, id, b)) \neq b \right] \\ & \leq \epsilon(\lambda) + \Pr \left[\widetilde{\text{Dec}}^\mathcal{O}(\text{pp}, \widetilde{\text{sk}}_{id}, \text{Enc}^\mathcal{O}(\text{pp}, id, b)) \neq \text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_{id}, \text{Enc}^\mathcal{O}(\text{pp}, id, b)) \right] \end{aligned} \quad (1)$$

In order to bound the probability in which the computations $\widetilde{\text{Dec}}^\mathcal{O}(\text{pp}, \widetilde{\text{sk}}_{id}, \text{Enc}^\mathcal{O}(\text{pp}, id, b))$ and $\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_{id}, \text{Enc}^\mathcal{O}(\text{pp}, id, b))$ do not produce the same output, it suffices to bound the probability in which an equality query is answered positively in one computation but negatively in the other computation (as long as the responses to all equality queries are consistent then the emulated computation carried out by $\widetilde{\text{Dec}}$ perfectly simulates Dec 's computation).

Assuming that the responses to equality queries are consistent among the two computations up to a certain point, then both computations issue the exact same next equality query (i, j) . Following the description of $\widetilde{\text{Dec}}$, let V_i and V_j denote the group elements in the i th and j th entries of the emulated table $\widetilde{\mathbf{B}}$, and let $V_i - V_j = \alpha_0^{(t)} \cdot 1 + \sum_{\ell=1}^{n_{pp}} \alpha_\ell^{(t)} \cdot \text{pp}_\ell + \sum_{\ell=1}^{n_{sk}} \beta_\ell^{(t)} \cdot \text{sk}_{id,\ell} + \sum_{\ell=1}^{n_{ct}} \gamma_\ell^{(t)} \cdot c_\ell$. There are three cases to consider:

Case I: If there exist $\alpha'_0, \dots, \alpha'_{n_{pp}} \in \mathbb{Z}_N$ such that $(\alpha'_0, \dots, \alpha'_{n_{pp}}, \beta_1^{(t)}, \dots, \beta_{n_{sk}}^{(t)}) \in \text{span}(\mathcal{L}_{id})$, then $\alpha'_0 \cdot 1 + \sum_{\ell=1}^{n_{pp}} \alpha'_\ell \cdot \text{pp}_\ell + \sum_{\ell=1}^{n_{sk}} \beta_\ell^{(t)} \cdot \text{sk}_{id,\ell} = 0$. Therefore, $\alpha_0^{(t)} \cdot 1 + \sum_{\ell=1}^{n_{pp}} \alpha_\ell^{(t)} \cdot \text{pp}_\ell + \sum_{\ell=1}^{n_{sk}} \beta_\ell^{(t)} \cdot \text{sk}_{id,\ell} + \sum_{\ell=1}^{n_{ct}} \gamma_\ell^{(t)} \cdot c_\ell = 0$ if and only if $(\alpha_0^{(t)} - \alpha'_0) \cdot 1 + \sum_{\ell=1}^{n_{pp}} (\alpha_\ell^{(t)} - \alpha'_\ell) \cdot \text{pp}_\ell + \sum_{\ell=1}^{n_{ct}} \gamma_\ell^{(t)} \cdot c_\ell = 0$, and thus the emulation obtains the correct answer to the equality query (i, j) .

Case II: If the equality query (i, j) is negatively answered in Dec 's computation, and there do not exist $\alpha'_0, \dots, \alpha'_{n_{pp}} \in \mathbb{Z}_N$ such that $(\alpha'_0, \dots, \alpha'_{n_{pp}}, \beta_1^{(t)}, \dots, \beta_{n_{sk}}^{(t)}) \in \text{span}(\mathcal{L}_{id})$, then it is also answered negatively in $\widetilde{\text{Dec}}$'s computation.

Case III: If the equality query (i, j) is positively answered in Dec 's computation, and there do not exist $\alpha'_0, \dots, \alpha'_{n_{pp}} \in \mathbb{Z}_N$ such that $(\alpha'_0, \dots, \alpha'_{n_{pp}}, \beta_1^{(t)}, \dots, \beta_{n_{sk}}^{(t)}) \in \text{span}(\mathcal{L}_{id})$, then the

equality query (i, j) is negatively answered in $\widetilde{\text{Dec}}$'s computation. However, we show that this case occurs with probability at most $1/p(\lambda)$.

Recall that a ciphertext $c \leftarrow \text{Enc}^{\mathcal{O}}(\text{pp}, \text{id}, b)$ is of the form $c = (c_1, \dots, c_{n_{\text{ct}}}, c_{\text{str}})$, where $c_1, \dots, c_{n_{\text{ct}}}$ are group elements and c_{str} is an explicit string. Since the only group elements that are given as input to the computation $\text{Enc}^{\mathcal{O}}(\text{pp}, \text{id}, b)$ are $1, \text{pp}_1, \dots, \text{pp}_{n_{\text{pp}}}$, then each c_v is of the form $c_v = \delta_{v,0} \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} \delta_{v,\ell} \cdot \text{pp}_\ell$, for coefficients $\delta_{v,0}, \dots, \delta_{v,n_{\text{pp}}} \in \mathbb{Z}_N$ that are determined by the group-operation queries issued during the computation $\text{Enc}^{\mathcal{O}}(\text{pp}, \text{id}, b)$. Therefore,

$$\begin{aligned} V_i - V_j &= \alpha_0^{(t)} \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} \alpha_\ell^{(t)} \cdot \text{pp}_\ell + \sum_{\ell=1}^{n_{\text{sk}}} \beta_\ell^{(t)} \cdot \text{sk}_{\text{id},\ell} + \sum_{\ell=1}^{n_{\text{ct}}} \gamma_\ell^{(t)} \cdot c_\ell \\ &= \left(\alpha_0^{(t)} + \sum_{v=1}^{n_{\text{ct}}} \delta_{v,0} \right) \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} \left(\alpha_\ell^{(t)} + \sum_{v=1}^{n_{\text{ct}}} \delta_{v,\ell} \right) \cdot \text{pp}_\ell + \sum_{\ell=1}^{n_{\text{sk}}} \beta_\ell^{(t)} \cdot \text{sk}_{\text{id},\ell}. \end{aligned}$$

Now, in this case there do not exist $\alpha'_0, \dots, \alpha'_{n_{\text{pp}}} \in \mathbb{Z}_N$ such that $(\alpha'_0, \dots, \alpha'_{n_{\text{pp}}}, \beta_1^{(t)}, \dots, \beta_{n_{\text{sk}}}^{(t)}) \in \text{span}(\mathcal{L}_{\text{id}})$, and therefore in particular $(\alpha'_0, \dots, \alpha'_{n_{\text{pp}}}, \beta_1^{(t)}, \dots, \beta_{n_{\text{sk}}}^{(t)}) \notin \text{span}(\bigcup_{i \in [M-1]} \mathcal{L}_{\text{id},b,i})$ for the specific choice of $\alpha'_\ell = (\alpha_\ell^{(t)} + \sum_{v=1}^{n_{\text{ct}}} \delta_{v,\ell})$ for every $\ell \in \{0, \dots, n_{\text{pp}}\}$. That is, this implies that for the computation $\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_{\text{id}}, \text{Enc}^{\mathcal{O}}(\text{pp}, \text{id}, b))$ it holds that

$$\mathcal{EQ} \left(\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_{\text{id}}, \text{Enc}^{\mathcal{O}}(\text{pp}, \text{id}, b)) \right) \not\subseteq \text{span} \left(\bigcup_{i \in [M-1]} \mathcal{L}_{\text{id},b,i} \right).$$

Applying Lemma 6 with the linear subspace $V \subseteq \mathbb{Z}_N^{n_{\text{pp}}+n_{\text{sk}}+1}$ defined as

$$V = \left\{ (\alpha_0, \dots, \alpha_{n_{\text{pp}}}, \beta_1, \dots, \beta_{n_{\text{sk}}}) \in \mathbb{Z}_N^{n_{\text{pp}}+n_{\text{sk}}+1} \mid \alpha_0 \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} \alpha_\ell \cdot \text{pp}_\ell + \sum_{\ell=1}^{n_{\text{sk}}} \beta_\ell \cdot \text{sk}_{\text{id},\ell} = 0 \right\},$$

which is of dimension at most $n_{\text{pp}} + n_{\text{sk}}$ since $(1, \text{pp}_1, \dots, \text{pp}_{n_{\text{pp}}}, \text{sk}_{\text{id},1}, \dots, \text{sk}_{\text{id},n_{\text{sk}}})$ is a non-zero vector, and with random variables X_1, \dots, X_M that are independently sampled from the distribution $\mathcal{EQ} \left(\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_{\text{id}}, \text{Enc}^{\mathcal{O}}(\text{pp}, \text{id}, b)) \right)$, we obtain from Eq. (1) that

$$\begin{aligned} &\Pr \left[\widetilde{\text{Dec}}^{\mathcal{O}}(\text{pp}, \widetilde{\text{sk}}_{\text{id}}, \text{Enc}^{\mathcal{O}}(\text{pp}, \text{id}, b)) \neq b \right] \\ &\leq \epsilon(\lambda) + \Pr \left[\mathcal{EQ} \left(\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_{\text{id}}, \text{Enc}^{\mathcal{O}}(\text{pp}, \text{id}, b)) \right) \not\subseteq \text{span} \left(\bigcup_{i \in [M-1]} \mathcal{L}_{\text{id},b,i} \right) \right] \\ &= \epsilon(\lambda) + \Pr [X_M \not\subseteq \text{span}(X_1 \cup \dots \cup X_{M-1})] \\ &\leq \epsilon(\lambda) + \frac{\dim V}{M(\lambda)} \\ &\leq \epsilon(\lambda) + \frac{n_{\text{pp}}(\lambda) + n_{\text{sk}}(\lambda)}{M(\lambda)} \\ &= \epsilon(\lambda) + \frac{1}{p(\lambda)}. \end{aligned}$$

4 Attacking Generic-Group IBE Schemes

In this section we present a generic-group adversary that breaks the security of any generic-group identity-based encryption scheme whose secret keys do not contain group elements, and that supports more identities than the number of group elements included in its public parameters. We prove the following theorem:

► **Theorem 7.** *Let $n_{\text{pp}}(\lambda)$ be a function of the security parameter $\lambda \in \mathbb{N}$. Let IBE be a secure generic-group identity-based encryption scheme over an identity space $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$ with decryption error $\epsilon(\lambda) \leq 1/160(n_{\text{pp}}(\lambda) + 1)$, whose public parameters consist of $n_{\text{pp}}(\lambda)$ group elements, and whose secret keys do not contain group elements. Then, $|\mathcal{ID}_\lambda| \leq n_{\text{pp}}(\lambda)$ for all sufficiently large $\lambda \in \mathbb{N}$.*

Regarding the decryption error $\epsilon(\lambda) \leq 1/160(n_{\text{pp}}(\lambda) + 1)$ considered in the above theorem, recall that our transformation from Section 3 leads to an arbitrary polynomially-small increase in the scheme's decryption error.

Preliminaries. Recall that for any generic-group algorithm A that receives as input group elements g_1, \dots, g_k and an explicit string str we let $\mathcal{EQ}(A^\mathcal{O}(\widehat{g}_1, \dots, \widehat{g}_k, \text{str}))$ denote the random variable corresponding to the set of all $(k+1)$ -dimensional vectors over \mathbb{Z}_N resulting from the positively-answered equality queries in the computation $A^\mathcal{O}(\widehat{g}_1, \dots, \widehat{g}_k, \text{str})$ (see Section 3 for the more formal definition). Our proof relies on the following lemma (which is proved in Appendix B):

► **Lemma 8.** *Let $k \geq 1$, and let X_1, \dots, X_k be random variables over subsets of a linear vector space V of dimension $\dim V$. Let Y be distributed uniformly over $\{1, \dots, k\}$ and independent of X_1, \dots, X_k . Denote by GoodSpan the set of all $(i, U_1, \dots, U_k) \subseteq [k] \times (2^V)^k$ for which*

$$\Pr_{X_1, \dots, X_k, Y} [X_Y \subseteq \text{span}(X_1 \cup \dots \cup X_{Y-1}) \mid Y = i, X_1 = U_1, \dots, X_{i-1} = U_{i-1}] \geq \frac{k - \dim V}{2k}.$$

Then,

$$\Pr_{X_1, \dots, X_k, Y} [(Y, X_1, \dots, X_k) \in \text{GoodSpan}] \geq \frac{k - \dim V}{2k}.$$

The remainder of this section consists of the proof of Theorem 7.

Proof of Theorem 7. Let IBE be a generic-group identity-based encryption scheme over an identity space $\mathcal{ID} = \{\mathcal{ID}_\lambda\}_{\lambda \in \mathbb{N}}$ with decryption error $\epsilon(\lambda) \leq 1/160(n_{\text{pp}} + 1)$, whose public parameters consist of $n_{\text{pp}} = n_{\text{pp}}(\lambda)$ group elements, and whose secret keys do not contain group elements. Assume toward a contradiction that $|\mathcal{ID}_\lambda| \geq n_{\text{pp}} + 1$ for infinitely many values of $\lambda \in \mathbb{N}$, and assume without loss of generality that $\{1, \dots, n_{\text{pp}} + 1\} \subseteq \mathcal{ID}_\lambda$ for any such $\lambda \in \mathbb{N}$. We present a generic-group adversary \mathcal{A} that issues a number of queries which is polynomial in λ , n_{pp} , and in the number of queries issued by Enc and Dec , and for which $|\Pr[\text{Expt}_{\text{IBE}, \mathcal{A}}(\lambda) = 1] - 1/2|$ is non-negligible for any such $\lambda \in \mathbb{N}$ (recall Definition 4 describing the experiment $\text{Expt}_{\text{IBE}, \mathcal{A}}$). The adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is defined as follows:

Our adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$

The algorithm $\mathcal{A}_1^{\mathcal{O}}(1^\lambda, N)$:

1. Sample $i \leftarrow \{1, \dots, n_{\text{pp}} + 1\}$, and output the challenge identity $id^* = i$, the identities $(1, \dots, i - 1)$ for which secret keys will be provided to \mathcal{A}_2 , and the state $\text{state} = (1^\lambda, N, i)$.

The algorithm $\mathcal{A}_2^{\mathcal{O}}(\text{state}, \text{pp}, c^*, \text{sk}_1, \dots, \text{sk}_{i-1})$:

1. Let $\text{pp} = (pp_1, \dots, pp_{n_{\text{pp}}}, \text{pp}_{\text{str}})$ for group elements $pp_1, \dots, pp_{n_{\text{pp}}}$ and an explicit string pp_{str} (and recall that $\text{sk}_1, \dots, \text{sk}_{i-1}$ are explicit strings).

[Part I: Using $\text{sk}_1, \dots, \text{sk}_{i-1}$ for learning information on pp]

2. For each $j \in \{1, \dots, i - 1\}$ perform the following steps:
 - a. Initialize a set $\mathcal{E}_j = \emptyset$ of $(n_{\text{pp}} + 1)$ -dimensional vectors over \mathbb{Z}_N .
 - b. For each message $b \in \{0, 1\}$ repeat the following step for $8(n_{\text{pp}} + 1)$ iterations: Compute $\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_j, \text{Enc}^{\mathcal{O}}(\text{pp}, j, b))$ using fresh randomness for Enc and Dec , and update $\mathcal{E}_j \leftarrow \mathcal{E}_j \cup \mathcal{E}\mathcal{Q}(\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_j, \text{Enc}^{\mathcal{O}}(\text{pp}, j, b)))$.
 - c. Emulate a fresh oracle $\widehat{\mathcal{O}}$ in order to find $(\text{pp}^*, \text{msk}^*, \text{sk}_j^*)$, where $\text{pp}^* = (pp_1^*, \dots, pp_{n_{\text{pp}}}^*, \text{pp}_{\text{str}}^*)$ for group elements $pp_1^*, \dots, pp_{n_{\text{pp}}}^*$ and explicit strings pp_{str}^* , msk^* and sk_j^* , subject to the following requirements:
 - i. $(\text{pp}^*, \text{msk}^*)$ and sk_j^* are in the supports of $\text{Setup}^{\widehat{\mathcal{O}}}(1^\lambda, N)$ and $\text{KG}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{msk}^*, j)$, respectively.
 - ii. $\text{pp}_{\text{str}}^* = \text{pp}_{\text{str}}$.
 - iii. For every $(\alpha_0, \dots, \alpha_{n_{\text{pp}}}) \in \mathcal{E}_1 \cup \dots \cup \mathcal{E}_{j-1}$ it holds that $\alpha_0 \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} \alpha_\ell \cdot pp_\ell^* = 0$ (i.e., pp^* satisfies the constraints induced by $\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{j-1}$).
 - iv. For each $b \in \{0, 1\}$ it holds that $\Pr \left[\text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_j^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, j, b)) = b \right] \geq 19/20$, where the probability is taken over the internal randomness of Enc and Dec (i.e., the decryption error of sk_j^* is at most $1/20$).
 - v. For each $b \in \{0, 1\}$ it holds that

$$\Pr \left[\mathcal{E}\mathcal{Q} \left(\text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_j^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, j, b)) \right) \notin \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{j-1}) \right] \leq \frac{1}{5},$$

where the probability is taken over the internal randomness of Enc and Dec .

- d. If such $(\text{msk}^*, \text{pp}^*, \text{sk}_j^*)$ are found then for each message $b \in \{0, 1\}$ repeat the following step for $8(n_{\text{pp}} + 1)$ iterations: Compute $\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_j^*, \text{Enc}^{\mathcal{O}}(\text{pp}, j, b))$ using fresh randomness for Enc and Dec , and update $\mathcal{E}_j \leftarrow \mathcal{E}_j \cup \mathcal{E}\mathcal{Q}(\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_j^*, \text{Enc}^{\mathcal{O}}(\text{pp}, j, b)))$.

[Part II: Constructing an alternative sk_i^* for decrypting the challenge ciphertext]

3. Emulate a fresh oracle $\widehat{\mathcal{O}}$ in order to find $(\text{pp}^*, \text{msk}^*, \text{sk}_i^*)$, where $\text{pp}^* = (pp_1^*, \dots, pp_{n_{\text{pp}}}^*, \text{pp}_{\text{str}}^*)$ for group elements $pp_1^*, \dots, pp_{n_{\text{pp}}}^*$ and explicit strings pp_{str}^* , msk^* and sk_i^* , subject to the following requirements:
 - a. $(\text{pp}^*, \text{msk}^*)$ and sk_i^* are in the supports of $\text{Setup}^{\widehat{\mathcal{O}}}(1^\lambda, N)$ and $\text{KG}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{msk}^*, i)$, respectively.
 - b. $\text{pp}_{\text{str}}^* = \text{pp}_{\text{str}}$.
 - c. For every $(\alpha_0, \dots, \alpha_{n_{\text{pp}}}) \in \mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1}$ it holds that $\alpha_0 \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} \alpha_\ell \cdot pp_\ell^* = 0$ (i.e., pp^* satisfies the constraints induced by $\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1}$).
 - d. For each $b \in \{0, 1\}$ it holds that $\Pr \left[\text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_i^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, i, b)) = b \right] \geq 19/20$, where the probability is taken over the internal randomness of Enc and Dec (i.e., the decryption error of sk_i^* is at most $1/20$).

26:14 Generic-Group Identity-Based Encryption: A Tight Impossibility Result

e. For each $b \in \{0, 1\}$ it holds that

$$\Pr \left[\mathcal{E} \mathcal{Q} \left(\text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, sk_i^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, i, b)) \right) \not\subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1}) \right] \leq \frac{1}{5},$$

where the probability is taken over the internal randomness of Enc and Dec .

4. If such $(\text{msk}^*, \text{pp}^*, sk_i^*)$ are not found then sample and output $b' \leftarrow \{0, 1\}$.
5. If such $(\text{msk}^*, \text{pp}^*, sk_i^*)$ are found then compute $\text{Dec}^{\mathcal{O}}(\text{pp}, sk_i^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, b))$ for λ times, where each computation uses fresh randomness for Enc , Dec and b , and count the number of times in which decryption was correct. If decryption was correct less than $\lambda \cdot \frac{11}{20} \cdot \left(1 - \frac{1}{20}\right)$ times, then sample and output $b' \leftarrow \{0, 1\}$.
6. Compute and output $b' \leftarrow \text{Dec}^{\mathcal{O}}(\text{pp}, sk_i^*, c^*)$.

In what follows we first analyze the number of queries issued by \mathcal{A} , and then analyze its success probability. In terms of oracle queries (i.e., group-operations queries and equality queries), note that \mathcal{A}_1 does not issue any queries, and that \mathcal{A}_2 issues queries only in Steps 2(b), 2(d), 5 and 6. These queries result from invoking the algorithms Enc and Dec , where Steps 2(b) and 2(d) consist of at most $O((n_{\text{pp}})^2)$ such invocations, Step 5 consists of λ such invocations, and Step 6 consists of one such invocation.

For analyzing \mathcal{A} 's success probability, fix a security parameter $\lambda \in \mathbb{N}$, a prime integer N that is produced by $\text{PrimeGen}(1^\lambda)$, and a pair (msk, pp) that is produced by $\text{Setup}^{\mathcal{O}}(1^\lambda)$. Our proof relies on the following notation:

- The experiment $\text{Expt}_{\text{IBE}, \mathcal{A}}$ and the description of our adversary define the random variables sk_1, \dots, sk_{i-1} corresponding to the secret keys that \mathcal{A}_2 is given as input. For our analysis, we additionally consider the random variables $sk_i, \dots, sk_{n_{\text{pp}}+1}$ that are independently sampled by computing $sk_j \leftarrow \text{KG}^{\mathcal{O}}(\text{pp}, \text{msk}, j)$ for each $j \in \{i, \dots, n_{\text{pp}}+1\}$.
- We denote by Y the random variable corresponding to the choice of the challenge identity $i \leftarrow \{1, \dots, n_{\text{pp}}+1\}$ by \mathcal{A}_1 .
- For each $j \in \{1, \dots, n_{\text{pp}}+1\}$ we let $\mathcal{E}_j = \cup_{v=1}^{8(n_{\text{pp}}+1)} \mathcal{E}_{j,v}$, where each $\mathcal{E}_{j,v}$ denotes the random variable corresponding to the set of vectors of coefficients of the equations found in one iteration of Step 2(b) and of Step 2(d). More specifically, each $\mathcal{E}_{j,v}$ is sampled from the distribution

$$\begin{aligned} & \mathcal{E} \mathcal{Q} \left(\text{Dec}^{\mathcal{O}}(\text{pp}, sk_j, \text{Enc}^{\mathcal{O}}(\text{pp}, j, 0)) \right) \cup \mathcal{E} \mathcal{Q} \left(\text{Dec}^{\mathcal{O}}(\text{pp}, sk_j, \text{Enc}^{\mathcal{O}}(\text{pp}, j, 1)) \right) \\ & \cup \mathcal{E} \mathcal{Q} \left(\text{Dec}^{\mathcal{O}}(\text{pp}, sk_j^*, \text{Enc}^{\mathcal{O}}(\text{pp}, j, 0)) \right) \cup \mathcal{E} \mathcal{Q} \left(\text{Dec}^{\mathcal{O}}(\text{pp}, sk_j^*, \text{Enc}^{\mathcal{O}}(\text{pp}, j, 0)) \right), \end{aligned}$$

where if \mathcal{A}_2 does not find a suitable sk_j^* in Step 2(c), then we define

$$\mathcal{E} \mathcal{Q} \left(\text{Dec}^{\mathcal{O}}(\text{pp}, sk_j^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, 0)) \right) = \mathcal{E} \mathcal{Q} \left(\text{Dec}^{\mathcal{O}}(\text{pp}, sk_j^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, 0)) \right) = \emptyset.$$

- We denote by GoodSpan the set of all $(i, U_1, \dots, U_{n_{\text{pp}}+1}) \in \{1, \dots, n_{\text{pp}}+1\} \times \left(2^{\mathbb{Z}_N^{n_{\text{pp}}+1}}\right)^{n_{\text{pp}}+1}$ for which

$$\Pr[\mathcal{E}_Y \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1}) \mid Y = i, \mathcal{E}_1 = U_1, \dots, \mathcal{E}_{i-1} = U_{i-1}] \geq \frac{1}{2(n_{\text{pp}}+1)}.$$

For avoiding additional notation, we abuse notation and denote by GoodSpan the event in which $(Y, \mathcal{E}_1, \dots, \mathcal{E}_{n_{\text{pp}}+1}) \in \text{GoodSpan}$.

▷ Claim 9. $\Pr[\text{GoodSpan}] \geq \frac{1}{2(n_{\text{pp}}+1)}$.

Proof. This is direct application of Lemma 8 with $k = n_{\text{pp}} + 1$, $(X_1, \dots, X_k) = (\mathcal{E}_1, \dots, \mathcal{E}_k)$, Y as defined above, and

$$V = \left\{ (\alpha_0, \dots, \alpha_{n_{\text{pp}}}) \in \mathbb{Z}_N^{n_{\text{pp}}+1} \mid \alpha_0 \cdot 1 + \sum_{\ell=1}^{n_{\text{pp}}} \alpha_\ell \cdot \text{pp}_\ell = 0 \right\}.$$

Note that since $(1, \text{pp}_1, \dots, \text{pp}_{n_{\text{pp}}})$ is a non-zero vector then $\dim V \leq n_{\text{pp}}$. \triangleleft

For the next claim, we denote by FindKey the event that \mathcal{A}_2 finds $(\text{msk}^*, \text{pp}^*, \text{sk}_i^*)$ in Step 3 that satisfies the required properties.

▷ Claim 10. $\text{GoodSpan} \subseteq \text{FindKey}$.

Proof. We show that whenever the event GoodSpan occurs, then msk , pp , and at least one sk_Y in the support of $\text{KG}^\mathcal{O}(\text{pp}, \text{msk}, Y)$ already satisfy the the required properties. Therefore, in particular, \mathcal{A}_2 finds some $(\text{msk}^*, \text{pp}^*, \text{sk}_i^*)$ in Step 3 that satisfies the required properties. Properties (a), (b) and (c) are trivially satisfied by $(\text{msk}, \text{pp}, \text{sk}_Y)$ for any sk_Y in the support of $\text{KG}^\mathcal{O}(\text{pp}, \text{msk}, Y)$. In what follows we show that properties (d) and (e) are satisfied by at least one sk_Y in the support of $\text{KG}^\mathcal{O}(\text{pp}, \text{msk}, Y)$.

The decryption error of the scheme is at most $\frac{1}{160(n_{\text{pp}}+1)}$. Therefore, for any value of Y it holds that

$$\Pr_{\text{KG, Enc, Dec}} \left[\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_Y, \text{Enc}^\mathcal{O}(\text{pp}, Y, b)) = b \right] \geq 1 - \frac{1}{160(n_{\text{pp}} + 1)}$$

where $\text{sk}_Y \leftarrow \text{KG}^\mathcal{O}(\text{pp}, \text{msk}, Y)$, and the probability is taken over the internal randomness of the algorithms KG, Enc and Dec. The above holds for any value of Y and independently of $\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1}$, and therefore

$$\Pr_{\text{KG, Enc, Dec}} \left[\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_Y, \text{Enc}^\mathcal{O}(\text{pp}, Y, b)) = b \mid \text{GoodSpan} \right] \geq 1 - \frac{1}{160(n_{\text{pp}} + 1)},$$

where $\text{sk}_Y \leftarrow \text{KG}^\mathcal{O}(\text{pp}, \text{msk}, Y)$, and the probability is taken over the internal randomness of the algorithms KG, Enc and Dec. Denote by SkSmallError_Y the set of all outputs sk_Y of $\text{KG}^\mathcal{O}(\text{pp}, \text{msk}, Y)$ for which

$$\Pr_{\text{Enc, Dec}} \left[\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_Y, \text{Enc}^\mathcal{O}(\text{pp}, Y, b)) = b \mid \text{GoodSpan} \right] \geq \frac{19}{20},$$

where now the probability is taken only over the internal randomness of the algorithms Enc and Dec. Then,

$$\begin{aligned} 1 - \frac{1}{160(n_{\text{pp}} + 1)} &\leq \Pr \left[\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_Y, \text{Enc}^\mathcal{O}(\text{pp}, Y, b)) = b \mid \text{GoodSpan} \right] \\ &\leq \Pr [\text{sk}_Y \in \text{SkSmallError}_Y \mid \text{GoodSpan}] \cdot 1 \\ &\quad + (1 - \Pr [\text{sk}_Y \in \text{SkSmallError}_Y \mid \text{GoodSpan}]) \cdot \frac{19}{20}. \end{aligned}$$

Therefore,

$$\Pr[\text{sk}_Y \in \text{SkSmallError}_Y \mid \text{GoodSpan}] \geq 1 - \frac{1}{8(n_{\text{pp}} + 1)}. \quad (2)$$

26:16 Generic-Group Identity-Based Encryption: A Tight Impossibility Result

Similarly, denote by SkGood_Y the set of all outputs sk_Y of $\text{KG}^\mathcal{O}(\text{pp}, \text{msk}, Y)$ for which

$$\Pr[\mathcal{E}_Y \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1}) \mid (Y, \mathcal{E}_1, \dots, \mathcal{E}_{n_{\text{pp}}+1}) \in \text{GoodSpan}, \text{sk}_Y] \geq \frac{1}{4(n_{\text{pp}}+1)}.$$

Then, from the definition of GoodSpan we obtain

$$\begin{aligned} \frac{1}{2(n_{\text{pp}}+1)} &\leq \Pr[\mathcal{E}_Y \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1}) \mid (Y, \mathcal{E}_1, \dots, \mathcal{E}_{n_{\text{pp}}+1}) \in \text{GoodSpan}] \\ &\leq \Pr[\text{SkGood}_Y \mid \text{GoodSpan}] \cdot 1 + (1 - \Pr[\text{SkGood}_Y \mid \text{GoodSpan}]) \cdot \frac{1}{4(n_{\text{pp}}+1)} \end{aligned}$$

and therefore

$$\Pr[\text{sk}_Y \in \text{SkGood}_Y \mid (Y, \mathcal{E}_1, \dots, \mathcal{E}_{n_{\text{pp}}+1}) \in \text{GoodSpan}] \geq \frac{1}{4(n_{\text{pp}}+1)}. \quad (3)$$

Therefore, combining Eq. (2) and (3) we obtain

$$\begin{aligned} \Pr[\text{sk}_Y \in \text{SkGood}_Y \cap \text{SkSmallError}_Y \mid (Y, \mathcal{E}_1, \dots, \mathcal{E}_{n_{\text{pp}}+1}) \in \text{GoodSpan}] \\ \geq \frac{1}{4(n_{\text{pp}}+1)} - \frac{1}{8(n_{\text{pp}}+1)} \\ = \frac{1}{8(n_{\text{pp}}+1)}. \end{aligned} \quad (4)$$

Note that property (d) is satisfied by any $\text{sk}_Y \in \text{SkSmallError}_Y$. We will now show that property (e) is satisfied by any $\text{sk}_Y \in \text{SkGood}_Y$ conditioned on GoodSpan , which together with Eq. (4) (and the fact that $\Pr[\text{GoodSpan}] > 0$ as shown in Claim 9) settles the proof.

Recall that $\mathcal{E}_Y = \bigcup_{v=1}^{8(n_{\text{pp}}+1)} \mathcal{E}_{Y,v}$ where $\{\mathcal{E}_{Y,v}\}_{v=1}^{8(n_{\text{pp}}+1)}$ are identically distributed and independent given $\mathcal{E}_1, \dots, \mathcal{E}_{Y-1}$ and sk_Y . Therefore, the definition of SkGood_Y implies that

$$\begin{aligned} \frac{1}{4(n_{\text{pp}}+1)} &\leq \Pr \left[\mathcal{E}_Y \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1}) \mid \begin{array}{l} (Y, \mathcal{E}_1, \dots, \mathcal{E}_{n_{\text{pp}}+1}) \in \text{GoodSpan} \\ \text{sk}_Y \in \text{SkGood}_Y \end{array} \right] \\ &= \Pr \left[\bigwedge_{v=1}^{8(n_{\text{pp}}+1)} (\mathcal{E}_{Y,v} \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1})) \mid \begin{array}{l} (Y, \mathcal{E}_1, \dots, \mathcal{E}_{n_{\text{pp}}+1}) \in \text{GoodSpan} \\ \text{sk}_Y \in \text{SkGood}_Y \end{array} \right] \\ &= \prod_{v=1}^{8(n_{\text{pp}}+1)} \Pr \left[(\mathcal{E}_{Y,v} \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1})) \mid \begin{array}{l} (Y, \mathcal{E}_1, \dots, \mathcal{E}_{n_{\text{pp}}+1}) \in \text{GoodSpan} \\ \text{sk}_Y \in \text{SkGood}_Y \end{array} \right] \\ &= \left(\Pr \left[(\mathcal{E}_{Y,1} \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1})) \mid \begin{array}{l} (Y, \mathcal{E}_1, \dots, \mathcal{E}_{n_{\text{pp}}+1}) \in \text{GoodSpan} \\ \text{sk}_Y \in \text{SkGood}_Y \end{array} \right] \right)^{8(n_{\text{pp}}+1)}. \end{aligned}$$

Therefore,

$$\begin{aligned} \Pr [(\mathcal{E}_{Y,1} \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1})) \mid (Y, \mathcal{E}_1, \dots, \mathcal{E}_k) \in \text{GoodSpan}, \text{sk}_Y \in \text{SkGood}_Y] \\ \geq \left(\frac{1}{4(n_{\text{pp}}+1)} \right)^{\frac{1}{8(n_{\text{pp}}+1)}} \geq \frac{4}{5}. \end{aligned}$$

In addition, recall that,

$$\begin{aligned} \mathcal{E}_{Y,v} &= \mathcal{E}\mathcal{Q} \left(\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_Y, \text{Enc}^\mathcal{O}(\text{pp}, Y, 0)) \right) \cup \mathcal{E}\mathcal{Q} \left(\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_Y, \text{Enc}^\mathcal{O}(\text{pp}, Y, 1)) \right) \\ &\quad \cup \mathcal{E}\mathcal{Q} \left(\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_Y^*, \text{Enc}^\mathcal{O}(\text{pp}, Y, 0)) \right) \cup \mathcal{E}\mathcal{Q} \left(\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_Y^*, \text{Enc}^\mathcal{O}(\text{pp}, Y, 0)) \right), \end{aligned}$$

Now, since for each $b \in \{0, 1\}$, $\mathcal{EQ}(\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_Y, \text{Enc}^\mathcal{O}(\text{pp}, Y, b))) \subseteq \mathcal{E}_{Y,1}$, then for each $b \in \{0, 1\}$ it holds that

$$\Pr \left[\begin{array}{l} \mathcal{EQ}(\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_Y, \text{Enc}^\mathcal{O}(\text{pp}, Y, b))) \\ \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1}) \end{array} \middle| \begin{array}{l} (Y, \mathcal{E}_1, \dots, \mathcal{E}_{n_{\text{pp}}+1}) \in \text{GoodSpan} \\ \text{sk}_Y \in \text{SkGood}_Y \end{array} \right] \geq \frac{4}{5},$$

as required. \triangleleft

For the next claim, note that if the event **GoodSpan** occurs, then by Claim 10 the event **FindKey** occurs as well, and therefore pp^* , msk^* and sk_i^* are well defined.

\triangleright **Claim 11.** For each $b \in \{0, 1\}$ it holds that

$$\Pr \left[\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_i^*, \text{Enc}^\mathcal{O}(\text{pp}, i, b; r); r') = \text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_i^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, i, b; r); r') \middle| \text{GoodSpan} \right] \geq \frac{3}{5},$$

where the probability is taken over the internal randomness $r \in \{0, 1\}^*$ and $r' \in \{0, 1\}^*$ of **Enc** and **Dec**, respectively.

Proof. Fix $b \in \{0, 1\}$. The definition of the set **GoodSpan**, together with the fact that $\mathcal{E}_Y = \bigcup_{v=1}^{8(n_{\text{pp}}+1)} \mathcal{E}_{Y,v}$ where $\{\mathcal{E}_{Y,v}\}_{v=1}^{8(n_{\text{pp}}+1)}$ are identically distributed and independent given $\mathcal{E}_1, \dots, \mathcal{E}_{Y-1}$ and sk_Y , imply that

$$\begin{aligned} & \frac{1}{2(n_{\text{pp}}+1)} \\ & \leq \Pr[\mathcal{E}_Y \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1}) \mid (Y, \mathcal{E}_1, \dots, \mathcal{E}_k) \in \text{GoodSpan}, \text{sk}_Y] \\ & = \Pr[\bigwedge_{v=1}^{8(n_{\text{pp}}+1)} (\mathcal{E}_{Y,v} \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1})) \mid (Y, \mathcal{E}_1, \dots, \mathcal{E}_k) \in \text{GoodSpan}, \text{sk}_Y] \\ & = \prod_{v=1}^{8(n_{\text{pp}}+1)} \Pr[(\mathcal{E}_{Y,v} \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1})) \mid (Y, \mathcal{E}_1, \dots, \mathcal{E}_k) \in \text{GoodSpan}, \text{sk}_Y] \\ & = \left(\Pr[(\mathcal{E}_{Y,1} \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1})) \mid (Y, \mathcal{E}_1, \dots, \mathcal{E}_k) \in \text{GoodSpan}, \text{sk}_Y] \right)^{8(n_{\text{pp}}+1)}. \end{aligned}$$

Therefore,

$$\begin{aligned} & \Pr[(\mathcal{E}_{Y,1} \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{Y-1})) \mid (Y, \mathcal{E}_1, \dots, \mathcal{E}_k) \in \text{GoodSpan}, \text{sk}_Y] \\ & \geq \left(\frac{1}{2(n_{\text{pp}}+1)} \right)^{\frac{1}{8(n_{\text{pp}}+1)}} \\ & \geq \frac{4}{5}. \end{aligned}$$

Since

$$\begin{aligned} \mathcal{E}_{i,1} = & \mathcal{EQ}(\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_i, \text{Enc}^\mathcal{O}(\text{pp}, i, 0))) \cup \mathcal{EQ}(\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_i, \text{Enc}^\mathcal{O}(\text{pp}, i, 1))) \\ & \cup \mathcal{EQ}(\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_i^*, \text{Enc}^\mathcal{O}(\text{pp}, i, 0))) \cup \mathcal{EQ}(\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_i^*, \text{Enc}^\mathcal{O}(\text{pp}, i, 0))), \end{aligned}$$

then, in particular, it holds that

$$\Pr[\mathcal{EQ}(\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_i^*, \text{Enc}^\mathcal{O}(\text{pp}, i, b))) \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1}) \mid \text{GoodSpan}, \text{sk}_i] \geq \frac{4}{5}.$$

Since sk_i^* and the randomness of **Enc** and **Dec** are independent of sk_i , then also

$$\Pr \left[\mathcal{EQ}(\text{Dec}^\mathcal{O}(\text{pp}, \text{sk}_i^*, \text{Enc}^\mathcal{O}(\text{pp}, i, b))) \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1}) \mid \text{GoodSpan} \right] \geq \frac{4}{5}.$$

26:18 Generic-Group Identity-Based Encryption: A Tight Impossibility Result

One of the requirements of $(\text{msk}^*, \text{pp}^*, \text{sk}_i^*)$ is that

$$\Pr \left[\mathcal{E}\mathcal{Q} \left(\text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_i^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, i, b)) \right) \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1}) \right] \geq \frac{4}{5},$$

where the probability is taken over the internal randomness of Enc and Dec , and therefore

$$\Pr \left[\begin{array}{c} \mathcal{E}\mathcal{Q} \left(\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_i^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, b; r); r') \right) \cup \\ \mathcal{E}\mathcal{Q} \left(\text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_i^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, i, b; r); r') \right) \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1}) \end{array} \middle| \text{GoodSpan} \right] \geq \frac{4}{5} + \frac{4}{5} - 1 = \frac{3}{5},$$

where the probability is taken over the internal randomness $r \in \{0, 1\}^*$ and $r' \in \{0, 1\}^*$ of Enc and Dec , respectively. Now, for each such r and r' that satisfy

$$\begin{aligned} \mathcal{E}\mathcal{Q} \left(\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_i^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, b; r); r') \right) \cup \mathcal{E}\mathcal{Q} \left(\text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_i^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, i, b; r); r') \right) \\ \subseteq \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1}) \end{aligned}$$

we claim that

$$\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_i^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, b; r); r') = \text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_i^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, i, b; r); r').$$

The two computations have the same explicit inputs since $\text{pp}_{\text{str}} = \text{pp}_{\text{str}}^*$ and sk_i^* does not contain group elements. Assuming that the responses to the equality queries are consistent among the two computations up to a certain point, then both computations issue the exact same next equality query (i_1, i_2) . Let V_{i_1} and V_{i_2} denote the group elements that are located in the corresponding entries of the table \mathbf{B} associated with oracle \mathcal{O} . Let $V_{i_1}^*$ and $V_{i_2}^*$ denote the group elements that are located in the corresponding entries of the table $\widehat{\mathbf{B}}$ associated with oracle $\widehat{\mathcal{O}}$. Let $V_{i_1} - V_{i_2} = \alpha_0 \cdot 1 + \sum_{r=1}^{n_{\text{pp}}} \alpha_r \cdot \text{pp}_r$ for $\alpha_0, \dots, \alpha_r \in \mathbb{Z}_N$. Since the two computations are the same up to this point, $V_{i_1}^* - V_{i_2}^* = \alpha_0 \cdot 1 + \sum_{r=1}^{n_{\text{pp}}} \alpha_r \cdot \text{pp}_r^*$.

On the one hand, if the equality query (i_1, i_2) is answered positively in the computation $\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_i^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, b; r); r')$, then

$$(\alpha_0, \dots, \alpha_{n_{\text{pp}}}) \in \mathcal{E}\mathcal{Q} \left(\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_i^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, b; r); r') \right)$$

and therefore $(\alpha_0, \dots, \alpha_{n_{\text{pp}}}) \in \text{span}(\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1})$. But pp^* is chosen to satisfy $\mathcal{E}_1 \cup \dots \cup \mathcal{E}_{i-1}$, and so $\alpha_0 \cdot 1 + \sum_{r=1}^{n_{\text{pp}}} \alpha_r \cdot \text{pp}_r^* = 0$, and this equality query is also answered positively by the computation $\text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_i^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, i, b; r); r')$.

On the other hand, if the equality query (i_1, i_2) is answered positively by the computation $\text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_i^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, i, b; r); r')$, then a symmetric argument shows that this equality query is also answered positively by the computation $\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_i^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, b; r); r')$. \triangleleft

\triangleright Claim 12. For each $b \in \{0, 1\}$ it holds that

$$\Pr \left[\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_i^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, b)) = b \mid \text{GoodSpan} \right] \geq \frac{11}{20}.$$

Proof. The event GoodSpan implies the event FindKey , and therefore the secret key sk_i^* chosen in Step 3 satisfies

$$\Pr \left[\text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_i^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, i, b)) = b \right] \geq \frac{19}{20},$$

where the probability is taken over the internal randomness of the algorithms Enc and Dec. Combining this with Claim 11 we obtain

$$\begin{aligned}
& \Pr \left[\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_i^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, b)) \neq b \mid \text{GoodSpan} \right] \\
& \leq \Pr \left[\begin{array}{l} \text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_i^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, b; r)); r' \\ \neq \text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_i^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, i, b; r)); r' \end{array} \mid \text{GoodSpan} \right] \\
& \quad + \Pr \left[\text{Dec}^{\widehat{\mathcal{O}}}(\text{pp}^*, \text{sk}_i^*, \text{Enc}^{\widehat{\mathcal{O}}}(\text{pp}^*, i, b)) \neq b \right] \\
& \leq \frac{2}{5} + \frac{1}{20} \\
& = \frac{9}{20}. \quad \triangleleft
\end{aligned}$$

For the following claims, we denote by Pass the event that sk_i^* passes the test in Step 5, and denote by Win the event in which $\text{Expt}_{\text{IBE}, \mathcal{A}}(\lambda) = 1$.

▷ Claim 13. $\Pr[\text{Win} \mid \text{GoodSpan}] \geq \frac{11}{20} \cdot (1 - e^{-\Omega(\lambda)})$.

Proof. Claim 12 and Chernoff's bound imply that

$$\begin{aligned}
\Pr[\text{Win} \mid \text{GoodSpan}] & \geq \Pr[\text{Win} \mid \text{Pass} \cap \text{GoodSpan}] \cdot \Pr[\text{Pass} \mid \text{GoodSpan}] \\
& \geq \frac{11}{20} \cdot \left(1 - e^{-\frac{(0.05)^2}{2} \cdot \frac{11}{20} \cdot \lambda} \right). \quad \triangleleft
\end{aligned}$$

▷ Claim 14. $\Pr[\text{Win} \mid \overline{\text{GoodSpan}}] \geq \frac{1}{2} \cdot (1 - e^{-\Omega(\lambda)})$.

Proof. Recall that FindKey denotes the event in which \mathcal{A} finds $(\text{msk}^*, \text{pp}^*, \text{sk}_i^*)$ in Step 3 that satisfies the required properties. Therefore,

$$\begin{aligned}
\Pr[\text{Win} \mid \overline{\text{GoodSpan}}] & = \Pr[\text{Win} \mid \text{FindKey} \cap \overline{\text{GoodSpan}}] \cdot \Pr[\text{FindKey} \mid \overline{\text{GoodSpan}}] \\
& \quad + \Pr[\text{Win} \mid \overline{\text{FindKey}} \cap \overline{\text{GoodSpan}}] \cdot \Pr[\overline{\text{FindKey}} \mid \overline{\text{GoodSpan}}] \\
& = \Pr[\text{Win} \mid \text{FindKey} \cap \overline{\text{GoodSpan}}] \cdot \Pr[\text{FindKey} \mid \overline{\text{GoodSpan}}] \\
& \quad + \frac{1}{2} \cdot \Pr[\overline{\text{FindKey}} \mid \overline{\text{GoodSpan}}] \tag{5}
\end{aligned}$$

Denote by FoundUseful the event in which \mathcal{A} finds sk_i^* in Step 3 and the success probability of sk_i^* at decrypting correctly is at least $1/2$. That is, FoundUseful is the event in which for each $b \in \{0, 1\}$ it holds that $\Pr[\text{Dec}^{\mathcal{O}}(\text{pp}, \text{sk}_i^*, \text{Enc}^{\mathcal{O}}(\text{pp}, i, b)) = b] \geq 1/2$, where the probability is taken over the internal randomness of the algorithms Enc and Dec. Then, FoundUseful \subseteq FindKey, and therefore,

$$\begin{aligned}
& \Pr[\text{Win} \mid \text{FindKey} \cap \overline{\text{GoodSpan}}] \\
& = \Pr[\text{Win} \mid \text{FoundUseful} \cap \overline{\text{GoodSpan}}] \cdot \Pr[\text{FoundUseful} \mid \text{FindKey} \cap \overline{\text{GoodSpan}}] \\
& \quad + \Pr[\text{Win} \mid \text{FindKey} \cap \overline{\text{FoundUseful}} \cap \overline{\text{GoodSpan}}] \cdot \Pr[\overline{\text{FoundUseful}} \mid \text{FindKey} \cap \overline{\text{GoodSpan}}] \tag{6}
\end{aligned}$$

Now, it holds that

$$\Pr[\text{Win} \mid \text{FoundUseful} \cap \overline{\text{GoodSpan}}] \geq \frac{1}{2} \tag{7}$$

and that

$$\begin{aligned}
& \Pr[\text{Win} \mid \text{FindKey} \cap \overline{\text{FoundUseful}} \cap \overline{\text{GoodSpan}}] \\
& \geq \Pr[\text{Win} \mid \overline{\text{Pass}} \cap \text{FindKey} \cap \overline{\text{FoundUseful}} \cap \overline{\text{GoodSpan}}] \\
& \quad \cdot \Pr[\overline{\text{Pass}} \mid \text{FindKey} \cap \overline{\text{FoundUseful}} \cap \overline{\text{GoodSpan}}] \tag{8}
\end{aligned}$$

26:20 Generic-Group Identity-Based Encryption: A Tight Impossibility Result

Recall that in Step 5 sk_i^* passes the test when decryption is correct for more than $\lambda \cdot \frac{11}{20} \cdot (1 - \frac{1}{20}) = 0.5225 \cdot \lambda$ times out of λ times. Therefore, by Chernoff's bound,

$$\Pr[\text{Pass} \mid \text{FindKey} \cap \overline{\text{FoundUseful}} \cap \overline{\text{GoodSpan}}] \leq e^{-\frac{(0.0225)^2}{3} \cdot \frac{1}{2} \cdot \lambda} = e^{-\Omega(\lambda)}. \quad (9)$$

In addition,

$$\Pr[\text{Win} \mid \overline{\text{Pass}} \cap \text{FindKey} \cap \overline{\text{FoundUseful}} \cap \overline{\text{GoodSpan}}] = \frac{1}{2} \quad (10)$$

Thus, combining Eq. (8), (9) and (10) we obtain

$$\Pr[\text{Win} \mid \text{FindKey} \cap \overline{\text{FoundUseful}} \cap \overline{\text{GoodSpan}}] \geq \frac{1}{2} \cdot (1 - e^{-\Omega(\lambda)}) \quad (11)$$

and combining Eq. (6), (7) and (11) we obtain

$$\Pr[\text{Win} \mid \text{FindKey} \cap \overline{\text{GoodSpan}}] \geq \frac{1}{2} \cdot (1 - e^{-\Omega(\lambda)}). \quad (12)$$

Finally, combining Eq. (5) and (12) we obtain

$$\Pr[\text{Win} \mid \overline{\text{GoodSpan}}] \geq \frac{1}{2} \cdot (1 - e^{-\Omega(\lambda)}). \quad \triangleleft$$

▷ **Claim 15.** $\Pr[\text{Expt}_{\mathcal{IBE}, \mathcal{A}}(\lambda) = 1] \geq \frac{1}{2} + \frac{1}{40(n_{\text{pp}}+1)} - e^{-\Omega(\lambda)}$

Proof. From Claim 13 and Claim 14 we obtain

$$\begin{aligned} \Pr[\text{Win}] &= \Pr[\text{Win} \mid \text{GoodSpan}] \cdot \Pr[\text{GoodSpan}] + \Pr[\text{Win} \mid \overline{\text{GoodSpan}}] \cdot \Pr[\overline{\text{GoodSpan}}] \\ &\geq \frac{11}{20} \cdot (1 - e^{-\Omega(\lambda)}) \cdot \Pr[\text{GoodSpan}] + \frac{1}{2} \cdot (1 - e^{-\Omega(\lambda)}) \cdot (1 - \Pr[\text{GoodSpan}]) \\ &= \frac{1}{2} + \left(\frac{11}{20} - \frac{1}{2} \right) \cdot \Pr[\text{GoodSpan}] - e^{-\Omega(\lambda)} \\ &= \frac{1}{2} + \frac{1}{20} \cdot \Pr[\text{GoodSpan}] - e^{-\Omega(\lambda)}. \end{aligned}$$

Lemma 9 now implies that

$$\begin{aligned} \Pr[\text{Win}] &\geq \frac{1}{2} + \frac{1}{20} \cdot \frac{1}{2(n_{\text{pp}}+1)} - e^{-\Omega(\lambda)} \\ &= \frac{1}{2} + \frac{1}{40(n_{\text{pp}}+1)} - e^{-\Omega(\lambda)} \quad \triangleleft \end{aligned}$$

This settles the proof of Theorem 7. ◀

References

- 1 Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *Advances in Cryptology – EUROCRYPT '10*, pages 553–572, 2010.
- 2 Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
- 3 Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT '04*, pages 223–238, 2004.
- 4 Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology – CRYPTO '04*, pages 443–459, 2004.

- 5 Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology – CRYPTO ’01*, pages 213–229, 2001.
- 6 Dan Boneh, Periklis A. Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. On the impossibility of basing identity based encryption on trapdoor permutations. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 283–292, 2008.
- 7 Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology – EUROCRYPT ’03*, pages 255–271, 2003.
- 8 David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *Advances in Cryptology – EUROCRYPT ’10*, pages 523–552, 2010.
- 9 Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363, 2001.
- 10 Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In *Proceedings of the 15th Theory of Cryptography Conference*, pages 372–408, 2017.
- 11 Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In *Advances in Cryptology – CRYPTO ’17*, pages 537–569, 2017.
- 12 Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of computing*, pages 197–206, 2008.
- 13 Tibor Jager and Jörg Schwenk. On the equivalence of generic group models. In *Proceedings of the 2nd International Conference on Provable Security*, pages 200–209, 2008.
- 14 Ueli Maurer. Abstract models of computation in cryptography. In *Proceedings of the 10th IMA International Conference on Cryptography and Coding*, pages 1–12, 2005.
- 15 Periklis A. Papakonstantinou, Charles W. Rackoff, and Yevgeniy Vahlis. How powerful are the DDH hard groups? Cryptology ePrint Archive, Report 2012/653, 2012.
- 16 Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – CRYPTO ’84*, pages 47–53, 1984.
- 17 Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology – EUROCRYPT ’97*, pages 256–266, 1997.
- 18 Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT ’05*, pages 114–127, 2005.

A Proof of Lemma 6

The random variables X_1, \dots, X_k are independent and identically distributed, and therefore for every $i \in [k]$ it holds that

$$\Pr[X_k \notin \text{span}(X_1 \cup \dots \cup X_{k-1})] = \Pr[X_i \notin \text{span}(X_1 \cup \dots \cup X_{i-1} \cup X_{i+1} \cup \dots \cup X_k)].$$

Thus,

$$\begin{aligned} & \Pr[X_k \notin \text{span}(X_1 \cup \dots \cup X_{k-1})] \\ &= \frac{1}{k} \cdot \sum_{i=1}^k \Pr[X_i \notin \text{span}(X_1 \cup \dots \cup X_{i-1} \cup X_{i+1} \cup \dots \cup X_k)] \\ &= \frac{1}{k} \cdot \sum_{i=1}^k \sum_{U_1, \dots, U_k \subseteq V} \Pr[(X_1, \dots, X_k) = (U_1, \dots, U_k)] \cdot \mathbf{1}_{\{U_i \notin \text{span}(U_1 \cup \dots \cup U_{i-1} \cup U_{i+1} \cup \dots \cup U_k)\}} \\ &= \frac{1}{k} \cdot \sum_{U_1, \dots, U_k \subseteq V} \Pr[(X_1, \dots, X_k) = (U_1, \dots, U_k)] \cdot \sum_{i=1}^k \mathbf{1}_{\{U_i \notin \text{span}(U_1 \cup \dots \cup U_{i-1} \cup U_{i+1} \cup \dots \cup U_k)\}}, \end{aligned}$$

26:22 Generic-Group Identity-Based Encryption: A Tight Impossibility Result

where for any event \mathcal{E} we denote by $\mathbf{1}_{\mathcal{E}}$ its indicator. Since the vector space V is of dimension $\dim(V)$, then for any $U_1, \dots, U_k \subseteq V$ there are at most $\dim(V)$ indices $i \in [k]$ for which $U_i \not\subseteq \text{span}(U_1 \cup \dots \cup U_{i-1} \cup U_{i+1} \cup \dots \cup U_k)$. Therefore,

$$\begin{aligned} \Pr[X_k \not\subseteq \text{span}(X_1 \cup \dots \cup X_{k-1})] \\ \leq \frac{1}{k} \cdot \sum_{U_1, \dots, U_k \subseteq V} \Pr[(X_1, \dots, X_k) = (U_1, \dots, U_k)] \cdot \dim(V) = \frac{\dim(V)}{k}. \quad \blacktriangleleft \end{aligned}$$

B Proof of Lemma 8

Our proof of Lemma 8 relies on the following lemma (note that, unlike in the statement of Lemma 6, here the random variables X_1, \dots, X_k are not assumed to be independent or identically distributed):

► **Lemma 16.** *Let $k \geq 1$, and let X_1, \dots, X_k be random variables over subsets of a linear vector space V of dimension $\dim(V)$. Let Y be distributed uniformly over $\{1, \dots, k\}$ and independent of X_1, \dots, X_k . Then,*

$$\Pr_{X_1, \dots, X_k, Y}[X_Y \not\subseteq \text{span}(X_1 \cup \dots \cup X_{Y-1})] \leq \frac{\dim(V)}{k}.$$

Proof of Lemma 16. Observe that

$$\begin{aligned} & \Pr_{X_1, \dots, X_k, Y}[X_Y \not\subseteq \text{span}(X_1 \cup \dots \cup X_{Y-1})] \\ &= \sum_{i=1}^k \sum_{U_1, \dots, U_k \subseteq V} \Pr_{X_1, \dots, X_k, Y}[Y = i \wedge (X_1, \dots, X_k) = (U_1, \dots, U_k)] \cdot \mathbf{1}_{\{U_i \not\subseteq \text{span}(U_1 \cup \dots \cup U_{i-1})\}} \\ &= \sum_{i=1}^k \sum_{U_1, \dots, U_k \subseteq V} \Pr_Y[Y = i] \cdot \Pr_{X_1, \dots, X_k}[(X_1, \dots, X_k) = (U_1, \dots, U_k)] \cdot \mathbf{1}_{\{U_i \not\subseteq \text{span}(U_1 \cup \dots \cup U_{i-1})\}} \quad (13) \end{aligned}$$

$$= \frac{1}{k} \cdot \sum_{U_1, \dots, U_k \subseteq V} \Pr_{X_1, \dots, X_k}[(X_1, \dots, X_k) = (U_1, \dots, U_k)] \cdot \left(\sum_{i=1}^k \mathbf{1}_{\{U_i \not\subseteq \text{span}(U_1 \cup \dots \cup U_{i-1})\}} \right) \quad (14)$$

$$\leq \frac{1}{k} \cdot \sum_{U_1, \dots, U_k \subseteq V} \Pr_{X_1, \dots, X_k}[(X_1, \dots, X_k) = (U_1, \dots, U_k)] \cdot \dim(V) \quad (15)$$

$$= \frac{\dim(V)}{k}$$

where Eq. (13) follows from the fact that Y is independent of X_1, \dots, X_k , Eq. (14) follows from the fact that Y is uniformly distributed, and Eq. (15) follows from the fact that V is of dimension $\dim V$. ◀

Equipped with Lemma 16, we now prove Lemma 8.

Proof of Lemma 8. On the one hand, Lemma 16 implies that

$$\frac{k - \dim V}{k} \leq \Pr[X_Y \subseteq \text{span}(X_1 \cup \dots \cup X_{Y-1})].$$

On the other hand,

$$\begin{aligned} & \Pr [X_Y \subseteq \text{span} (X_1 \cup \dots \cup X_{Y-1})] \\ &= \Pr [X_Y \subseteq \text{span} (X_1 \cup \dots \cup X_{Y-1}) \mid (Y, X_1, \dots, X_k) \in \text{GoodSpan}] \cdot \Pr[\text{GoodSpan}] \\ & \quad + \Pr [X_Y \subseteq \text{span} (X_1 \cup \dots \cup X_{Y-1}) \mid (Y, X_1, \dots, X_k) \in \overline{\text{GoodSpan}}] \cdot \Pr[\overline{\text{GoodSpan}}] \\ & \leq \Pr[\text{GoodSpan}] + \frac{k - \dim V}{2k}. \end{aligned}$$

Therefore,

$$\Pr[\text{GoodSpan}] \geq \frac{k - \dim V}{2k}. \quad \blacktriangleleft$$

