4th Conference on Information-Theoretic Cryptography

ITC 2023, June 6–8, 2023, Aarhus University, Aarhus, Denmark

Edited by

Kai-Min Chung

Editors

Kai-Min Chung
Academia Sinica, Taipei City, Taiwan
kmchung@iis.sinica.edu.tw

ACM Classification 2012
Mathematics of computing → Information theory; Theory of computation → Computational complexity and cryptography; Security and privacy → Cryptography

ISBN 978-3-95977-271-6

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/978-3-95977-271-6.

Publication date
July, 2023

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.2023.0

ISBN 978-3-95977-271-6 **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 and Nanyang Technological University, SG)
- 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

Papers

4th Conference on Information-Theoretic Cryptography (ITC 2023).
Editor: Kai-Min Chung
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

The fourth Conference on Information-Theoretic Cryptography (ITC 2023) took place from June 6–8, 2023, at the Department of Computer Science, Aarhus University, Aarhus, Denmark. For the first time since the COVID-19 pandemic, we were thrilled to hold a fully in-person conference. This year's conference was co-located with the TPMPC 2023 workshop. The general chair was Ivan Damgård, and the program chair was Kai-Min Chung. As with the previous editions, the conference was held in cooperation with the International Association for Cryptologic Research (IACR).

In its fourth year, ITC continued its mission of bringing together the cryptography and information theory communities, and advancing research in all aspects of information-theoretic techniques for cryptography and security. In pursuit of this mission, we invited multiple Program Committee members from the information theory community, and broadened the Call for Papers to encompass emerging topics such as adversarial and robust learning and algorithmic fairness. Although we didn't see a substantial increase in submissions from these areas this year, we remain hopeful that this will be a valuable initiative for future years.

We received a total of 29 submissions which overall were of high quality. As in the last year, we leveraged the small conference size to have interactive and anonymous discussions with the authors to clarify technical issues. With the help of external reviewers, the program committee selected 18 papers. One was conditionally accepted at first but eventually accepted after shepherding. The proceedings contain the revised versions of these 18 papers. The revisions were not reviewed, and the authors bear full responsibility for the content.

Continuing the tradition, the conference featured six "spotlight talks," highlighting the exciting development of information theoretical techniques in the cryptography and information theory community. This year, the selection of spotlight talks was carried out by the steering committee and the program chair. It was our great pleasure this year to feature a historical talk by Ivan Damgård on information-theoretic MPC to celebrate the 35th anniversary of its invention.

We are grateful to everyone who made the 4th ITC conference a success. Our heartfelt thanks go out to the authors who submitted their papers. We extend our sincere thanks to the PC members and external reviewers for their dedicated efforts in providing thorough reviews, insightful discussions, and expert opinions. We are deeply indebted to the steering committee, particularly Benny Applebaum and Hoeteck Wee, for their invaluable guidance. Special thanks are also due to the previous PC chairs, especially Dana Dachman-Soled, for sharing their experience and providing answers to numerous questions. Lastly, we extend our gratitude to all the invited speakers, presenting authors, and participants who devoted their time and energy to ensuring the success of this conference.

Kai-Min Chung

4th Conference on Information-Theoretic Cryptography (ITC 2023).
Editor: Kai-Min Chung
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 (MIT and Microsoft Research New England)
- Stefano Tessaro (University of Washington)
- Vinod Vaikuntanathan (MIT)
- Hoeteck Wee (ENS Paris)
- Daniel Wichs (Northeastern University and NTT Research)
- Mary Wootters (Stanford)
- Chaoping Xing (Nanyang Technological University)
- Moti Yung (Google)

4th Conference on Information-Theoretic Cryptography (ITC 2023).
Editor: Kai-Min Chung
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Organization

General chairs

- Ivan Damgård (Aarhus University)

Program chair

- Kai-Min Chung (Academia Sinica)

Program Committee

- Divesh Aggarwal (National University of Singapore)
- Prabhanjan Ananth (University of California, Santa Barbara)
- Jeremiah Blocki (Purdue University)
- Amos Beimel (Ben Gurion University)
- Rawad Bitar (Technical University of Munich)
- Mahdi Cheraghchi (University of Michigan Ann Arbor)
- Albert Cheu (Georgetown University)
- Suhas Diggavi (University of California, Los Angeles)
- Wei-Kai Lin (Northeastern University)
- Qipeng Liu (Simons Institute)
- Xiao Liang (Rice University)
- Russell W. F. Lai (Aalto University)
- Saeed Mahloujifar (Princeton University)
- Maciej Obremski (National University of Singapore)
- Vinod M. Prabhakaran (Tata Institute of Fundamental Research)
- Anat Paskin-Cherniavsky (Ariel University)
- Lior Rotem (Stanford University)
- João Ribeiro (New University of Lisbon)
- Noga Ron-Zewi (University of Haifa)
- Salim El Rouayheb (Rutgers University)
- Sruthi Sekar (University of California, Berkeley)
- Kevin Yeo (Google NYC and Columbia University)
- Takashi Yamakawa (NTT)
- Yihan Zhang (Institute of Science and Technology Austria)

Local Organization Committee

- Yashvanth Kondi (Aarhus University)
- Divya Ravi (Aarhus University)
- Malene Bisgaard (Aarhus University)

4th Conference on Information-Theoretic Cryptography (ITC 2023).
Editor: Kai-Min Chung
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

External Reviewers

Gilles Zemor, Rafael D'Oliveira, Aditya Gulati, Laszlo Csirmaz, Oded Nir, Elisaweta Masserova, Rahul Rachuri, Obbattu Sai Lakshmi Bhavana, Varun Narayanan, Nathan Manohar, Aarushi Goel, Rohit Chatterjee, Justin Raizes, Ohad Klein, Fatih Kaleoglu, Bar Alon, Alexander Poremba, Chen-Da Liu Zhang, Siyao Guo, Ethan Mook, Sebastian Bitzer

Two-Round Perfectly Secure Message Transmission with Optimal Transmission Rate

Nicolas Resch
Informatics' Institute, University of Amsterdam, The Netherlands

Chen Yuan
School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China

Abstract

In the model of *Perfectly Secure Message Transmission (PSMT)*, a sender Alice is connected to a receiver Bob via n parallel two-way channels, and Alice holds an ℓ symbol secret that she wishes to communicate to Bob. There is an unbounded adversary Eve that controls t of the channels, where $n = 2t + 1$. Eve is able to corrupt any symbol sent through the channels she controls, and furthermore may attempt to infer Alice's secret by observing the symbols sent through the channels she controls. The transmission is required to be (a) *reliable*, i.e., Bob must always be able to recover Alice's secret, regardless of Eve's corruptions; and (b) *private*, i.e., Eve may not learn anything about Alice's secret. We focus on the two-round model, where Bob is permitted to first transmit to Alice, and then Alice responds to Bob.

In this work we provide upper and lower bounds for the PSMT model when the length of the communicated secret ℓ is asymptotically large. Specifically, we first construct a protocol that allows Alice to communicate an ℓ symbol secret to Bob by transmitting at most $2(1 + o_{\ell\to\infty}(1))n\ell$ symbols. Under a reasonable assumption (which is satisfied by all known efficient two-round PSMT protocols), we complement this with a lower bound showing that $2n\ell$ symbols are necessary for Alice to privately and reliably communicate her secret. This provides strong evidence that our construction is optimal (even up to the leading constant).

2012 ACM Subject Classification Security and privacy → Mathematical foundations of cryptography

Keywords and phrases Secure transmission, Information theoretical secure, MDS codes

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.1

Related Version *Full Version*: `https://eprint.iacr.org/2021/158`

Funding *Nicolas Resch*: Research supported in part by ERC H2020 grant No.74079 (ALGSTRONG-CRYPTO).
Chen Yuan: Research supported in part by the National Key Research and Development Projects under Grant 2022YFA1004900 and Grant 2021YFE0109900, the National Natural Science Foundation of China under Grant 12101403 and Grant 12031011.

Acknowledgements CY would also like to thank Serge Fehr for introducing him to this problem.

1 Introduction

Perfectly secure message transmission (PSMT) was first introduced by Dolev et al. in [2]. This problem involves two parties, the sender Alice and the receiver Bob. Alice wishes to communicate a secret to Bob over n parallel channels in the presence of a computationally unbounded adversary Eve. Eve is able to take control of up to t channels in such a way that she can listen to and/or overwrite the message passing through these t corrupted channels. Here, we assume Eve is *static*, i.e., she chooses up to t channels to corrupt before the protocol and will not change corrupted channels during the protocol. The goal of PSMT is to devise a procedure permitting Alice and Bob to communicate the secret reliably and privately. More


© Nicolas Resch and Chen Yuan;
licensed under Creative Commons License CC-BY 4.0

precisely, it is guaranteed that Bob always completely recovers the secret (*reliability*) and Eve learns absolutely nothing about the secret (*privacy*).[1] PSMT can be done in multiple communication rounds. During each round, one party acts as the sender and the other acts as the receiver. They are not permitted to change their roles in one round.

It is clear that for $t > n/2$, PSMT is not possible, regardless of how many rounds the protocol uses. One can treat all the message transmitted over these n channels as a codeword of length n. Assume $\mathbf{c}_1$ represents the secret 1 and $\mathbf{c}_0$ represents the secret 0 that Alice wants to communicate to Bob. Since the distance of these two codewords is at most n and the number of errors t is more than the half the distance between $\mathbf{c}_1$ and $\mathbf{c}_0$, unique decoding is not possible.

The original paper in [2] showed that one-round PSMT is possible if $n \geq 3t + 1$. The same paper also showed that PSMT is possible when $n \geq 2t + 1$ if two or more rounds are performed. There have since been a number of efforts to devise improved PSMT protocols in various settings. The most challenging case is two-round PSMT with $n = 2t + 1$ channels. To measure the performance of a PSMT protocol in this case, we use the metric of *transmission rate*, which is the total number of bits transmitted divided by the length (in bits) of the secret communicated.

Prior Work. In what follows, we focus on the case that $n = 2t + 1$. Sayeed and Abu-Amara [5] first presented a two-round PSMT achieving transmission rate $O(n^3)$. Agarwal et al. [1] further improved it to $O(n)$ which is asymptotically optimal as a lower bound of n was proved in [7]. However, implementing this protocol requires an inefficient exponential-time algorithm. A breakthrough was achieved by Kurosawa and Suzuki [4] whose protocol achieves transmission rate $6n$, and can be run in polynomial time. Inspired by this protocol, Spini and Zémor [6] further reduced the transmission rate to $5n$, and moreover their protocol is arguably simpler than those that preceded it. Our protocol builds off of their ideas, as we discuss at the end of this introduction. Their work also answers in the affirmative an open problem posed in [4] of whether it is possible to achieve $O(n)$ transmission rate for a secret of size at most $O(n^2 \log n)$.

Hence, in reviewing the literature on PSMT, we note that the only known lower bound on the transmission rate for two-round PSMT is n, while the current state-of-the-art construction in [6] achieves transmission rate $5n$. While both bounds are $\Theta(n)$, there is still a gap of $4n$ between the lower bound and the upper bound.

Our Results. Our results are two-fold. Our first contribution is a two-round PSMT protocol communicating a length ℓ secret with transmission rate $2(1 + o_{\ell\to\infty}(1))n$.[2] This protocol improves over the state-of-the-art protocol in [6] by $3n$. Furthermore, our protocol reaches this transmission rate when Alice and Bob merely communicate an $\omega(n \log n)$-bit secret, and moreover achieves transmission rate $O(n)$ when they communicate an $\Omega(n \log n)$-bit secret as in [6].

Our second contribution is a lower bound on two-round PSMT protocols. Specifically, under a reasonable assumption, we show that Alice and Bob have to transmit at least $2n\ell$ bits so as to securely communicate an ℓ-bit secret. Our assumption comes from the observation

[1] One can also consider the model of secure message transmission where privacy and/or reliability is only guaranteed to hold with high probability [3]. However, in this work, we focus exclusively on the case of *perfect* privacy and reliability.

[2] Here and throughout, $o_{\ell\to\infty}(1)$ denotes a quantity which tends to 0 as $\ell \to \infty$, holding n fixed.

that all known efficient constructions such as [1, 4, 6] allow the adversary to learn the whole transmission in the second round of communication. This means the adversary can recover the transmission of *all* n channels by only listening to t of them. The reason is that in the second round, Alice encodes the message via an error correcting code which ensures the correctness of the transmission but sacrifices privacy. Therefore, in the security analysis of their protocols, they assume that the adversary could learn the whole transmission in the second round. Under this assumption, our two-round PSMT protocol actually achieves the optimal transmission rate. In this sense, our lower bound argument reveals an inherent limit for optimizing two-round PSMT: to beat our protocol, one must design a two-round PSMT protocol bypassing this assumption.

Our Techniques. As mentioned above, we obtain tight upper and lower bounds for communicating an ℓ-bit secret in the model of two-round PSMT. We start by outlining the upper bound proof.

Upper Bound. For the upper bound, we construct a two-round PSMT protocol achieving transmission rate $\sim 2n$. Instead of presenting our optimal protocol immediately, we first present a simplified protocol which allows for communicating a $\log n$ bit secret securely, which we view as a symbol $m \in \mathbb{F}_q$ with $q \geq n$.

Bob first sends $t+1$ codewords $\mathbf{c}_1, \ldots, \mathbf{c}_{t+1}$ which are picked independently and uniformly at random from a $[n, t+1, n-t]_q$ Reed-Solomon code[3] over $\mathbb{F}_q$. Alice receives the corrupted codewords $\tilde{\mathbf{c}}_i = \mathbf{c}_i + \mathbf{e}_i$. She uses the parity check matrix of this Reed-Solomon code to calculate the syndrome vectors $\mathbf{H}\tilde{\mathbf{c}}_i = \mathbf{s}_i$. Since Eve can corrupt at most t channels, there exist coefficients $\lambda_1, \ldots, \lambda_{t+1} \in \mathbb{F}_q$, not all zero, such that $\sum_{i=1}^{t+1} \lambda_i \mathbf{s}_i = \mathbf{0}$. From this one can show $\sum_{i=1}^{t+1} \lambda_i \mathbf{e}_i = \mathbf{0}$ and thus $\sum_{i=1}^{t+1} \lambda_i \mathbf{c}_i = \sum_{i=1}^{t+1} \lambda_i \tilde{\mathbf{c}}_i$. To simplify the following expressions, denote $\bar{\mathbf{c}} := \sum_{i=1}^{t+1} \lambda_i \mathbf{c}_i = \sum_{i=1}^{t+1} \lambda_i \tilde{\mathbf{c}}_i$.

Let $\mathbf{h} \in \mathbb{F}_q^n$ be a vector of weight n that is not orthogonal to the $[n, t+1, n-t]$ Reed-Solomon code. Alice broadcasts[4] $\lambda_1, \ldots, \lambda_{t+1}$ together with $\langle \mathbf{h}, \bar{\mathbf{c}} \rangle + m$ to Bob where m is the secret; $\langle \mathbf{h}, \bar{\mathbf{c}} \rangle$ is a mask for the secret. Bob first uses $\lambda_1, \ldots, \lambda_{t+1}$ to recover $\bar{\mathbf{c}}$ and then obtains m by removing the mask $\langle \mathbf{h}, \bar{\mathbf{c}} \rangle$ from the last broadcasted message.

The privacy analysis is quite straightforward. First, Eve can calculate $\lambda_1, \ldots, \lambda_{t+1}$ by herself since each $\mathbf{s}_i = \mathbf{H}\mathbf{e}_i$ is available to her. This means we can reduce the privacy argument to the last message $\langle \mathbf{h}, \bar{\mathbf{c}} \rangle + m$ which is an immediate consequence of the $[n, t+1, n-t]$ Reed-Solomon code we use. This protocol allows Alice and Bob to securely communicate the secret $m \in \mathbb{F}_q$ at the cost of $n^2 \log n$ communication complexity (measured in bits).

Observe that if the syndrome space spanned by $\mathbf{s}_1, \ldots, \mathbf{s}_{t+1}$ has dimension r, Alice only needs to send $r+1$ coefficients instead of $t+1$ so as to share a common codeword with Bob. This observation leads to our most efficient two-round PSMT.

We now present the general protocol. Assume Alice and Bob want to communicate an $\ell \log n$-bit secret securely. We first split it into ℓ secrets $m_1, \ldots, m_\ell$, each of size $\log n$, which we think of as lying in $\mathbb{F}_q$ with $q \geq n$. Bob first sends $t+\ell$ codewords $\mathbf{c}_1, \ldots, \mathbf{c}_{t+\ell}$ which are picked independently and uniformly at random from a $[n, t+1, n-t]$ Reed-Solomon code over $\mathbb{F}_q$. Alice receives the corrupted codewords $\tilde{\mathbf{c}}_i = \mathbf{c}_i + \mathbf{e}_i$ for $i \in [t+\ell]$. She uses the parity-check matrix of this Reed-Solomon code to calculate the syndrome vectors $\mathbf{H}\tilde{\mathbf{c}}_i = \mathbf{s}_i$.

[3] A $[n, k, d]_q$ Reed-Solomon code has block-length n, dimension k and distance $d = n - k + 1$.
[4] To broadcast $\lambda \in \mathbb{F}_q$, Alice sends λ through every channel; note that Bob can easily recover λ by choosing the majority symbol.

Assume that the space spanned by $\mathbf{s}_1, \ldots, \mathbf{s}_{t+\ell}$ has dimension r. Let $S \subset [t+\ell]$ be the index set of $\mathbf{s}_i$ that form the basis of this syndrome space. Without loss of generality, let us assume $S = \{t+\ell-r+1, t+\ell-r+2, \ldots, t+\ell\}$, the last r elements of $[t+\ell]$. For each $i \in [\ell]$, there exist not all zero coefficients λ_{ij} for $j \in S$ such that $\mathbf{s}_i = \sum_{j\in S} \lambda_{ij}\mathbf{s}_j$. In analogy to what we did in the simpler protocol, we let $\bar{\mathbf{c}}_i := \mathbf{c}_i - \sum_{j\in S} \lambda_{ij}\mathbf{c}_j = \tilde{\mathbf{c}}_i - \sum_{j\in S} \lambda_{ij}\tilde{\mathbf{c}}_j$.

Before entering into the second round, we do the same thing as [6] so as to reduce the communication complexity: we spot a corrupted codeword with error weight at least r by applying linear operations to the $\tilde{\mathbf{c}}_j$'s.[5] We take a different approach which simplifies the argument; for details, please see Algorithm 4. Let's suppose Alice has managed to spot a corrupted codeword $\tilde{\mathbf{c}} = \sum_{j\in S} \lambda_j\tilde{\mathbf{c}}_j$ with error weight at least r. Alice first broadcasts the index set S together with λ_j for $j \in S$ and $\tilde{\mathbf{c}}$ to Bob. Then, Alice uses an $[n, r+1, n-r]$ Reed-Solomon code to encode the message data $\lambda_{ij}, j \in S$ and $\langle \mathbf{h}, \bar{\mathbf{c}}_i\rangle + m_i$ for $i \in [\ell]$.

Once Bob receives the messages, he can correctly recover the index set S and λ_j for $j \in S$ and $\tilde{\mathbf{c}}$ as these messages are broadcasted. By applying the same linear operation on the codewords in S, Bob will obtain $\mathbf{c} = \sum_{j\in S} \lambda_j\mathbf{c}_j$ which is at least distance r away from $\tilde{\mathbf{c}}$. Bob then ignores the r channels that cause the inconsistency between $\mathbf{c}$ and $\tilde{\mathbf{c}}$. Bob can decode the rest of Alice's messages correctly which were encoded by the $[n, r+1, n-r]$ Reed-Solomon code since Eve can only cause r erasures and $t-r$ errors now. The recovery procedure is exactly the same as in the first protocol. The privacy argument is also quite straightforward. First of all, the coefficients λ_{ij} can be computed by Eve on her own. Then, the privacy of the secret m_i can be reduced to the privacy of $\mathbf{c}_i$ for $i \in [r]$ which is guaranteed by the $[n, t+1, n-t]$ Reed-Solomon code.

It remains to bound the communication complexity. The first-round communication complexity is $(\ell+t)n\log n$. The second-round communication complexity is $nr\log(t+\ell) + (r+n)n\log n + \frac{n}{r+1}(r+1)\ell\log n$. Thus, the transmission rate is $2n + O(\frac{n^2}{\ell})$ which becomes $2(1+o_{\ell\to\infty}(1))n$ if Alice communicates to Bob an $\ell\log n = \omega(n\log n)$-bit secret.

Lower Bound. Let us first formalize PSMT by defining Alice and Bob's moves. Assume that Alice wants to communicate an ℓ-bit secret s securely to Bob via a two-round PSMT. In the first round, Bob sends a vector $\mathbf{a} = (a_1, \ldots, a_n)$ to Alice, and Alice receives a corrupted vector $\tilde{\mathbf{a}}$. Based on $\tilde{\mathbf{a}}$ and the secret $s \in [2^\ell]$, Alice sends back a vector $\mathbf{b} = (b_1, \ldots, b_n)$ to Bob. On receiving the corrupted vector $\tilde{\mathbf{b}}$, Bob tries to decode the correct secret s with the help of $\mathbf{a}$.

Next, we justify our assumption that Eve learn the whole transmission in the second round of communication. We design an adversary Eve to force Alice and Bob to transmit at least $2\ell n$ bits so as to securely send the ℓ-bit secret. In the first round, Eve does nothing. That means Alice will receive a correct vector $\mathbf{a}$. Moreover, she has no idea which channels are corrupted. She must therefore assume that any subset of t channels are *equally likely* to be corrupted in the second round. Given $\mathbf{a}$, Alice has to use a code of distance $n = 2t+1$ to encode the secret $s \in [2^\ell]$ so as to achieve reliability. This gives a lower bound ℓn on the second round communication complexity. In the meanwhile, if the code of distance $n = 2t+1$ used by Alice and Bob in the second round is known to Eve, Eve will learn $\mathbf{a}$. In fact, all

[5] Note that Eve has to corrupt at least r channels so as to make the syndrome space have dimension r. To simplify our discussion here, we assume $r \le \frac{t}{3}$; otherwise the protocol will be little more complicated. Specifically, Alice first broadcasts a corrupted codeword with error weight $\frac{t}{3}$ and then sends all corrupted codewords in S to Bob via a $[n, \frac{t}{3}, n-\frac{t}{3}+1]$ Reed-Solomon code. This extra cost will not affect transmission rate as we can amortize it out by communicating $\ell\log n = \omega(n\log n)$-bit secret. The interested reader can find the details in our proof.

known efficient constructions use the same code book in this situation. Their protocol only protects the correctness of the transmission in the second round not the privacy.[6] In the following argument, we assume that Eve knows $\mathbf{b}$ if there is no corruption in the first round. Therefore, to achieve perfect security, Alice and Bob must share a private key of size ℓ in the first round. We also notice that the message sent by Bob in the first round is *independent of* Eve's strategy, which means that the lower bound on the communication complexity of the first round can be applied to the case Eve does nothing in the first round. We construct a secret sharing scheme by treating $\mathbf{a} = (a_1, \ldots, a_n)$ as n shares and this private key as a secret. Since Eve can listen to t channels, this means any t shares should learn nothing of this secret. This implies that such a secret sharing scheme has t-privacy. We next show that such secret sharing scheme must have $t+1$-reconstruction.

Let $\mathbf{a}_1$ be any share vector of secret s_1 and $\mathbf{a}_2$ be any share vector of secret s_2. If $\mathbf{a}_1$ and $\mathbf{a}_2$ are within distance t, Eve may inject t errors to change $\mathbf{a}_1$ to $\mathbf{a}_2$. Then, Alice can not detect any corruption and take the move as if no corruption happens. However, this will lead to the situation that Alice and Bob share a wrong key and thus Alice fails to recover the correct secret. This implies the share vectors associated with different secrets must have distance $t+1$ and thus any $n-(t+1)+1 = t+1$ shares can reconstruct the secret. As we have t-privacy and $t+1$-reconstruction, our secret sharing scheme is threshold, which implies that the number of bits communicated in the first round is also at least ℓn. Putting it all together, we obtain the desired $2\ell n$ lower bound on the communication of the two-round PSMT. Although we do not pin down the actual value of optimal two-round PSMT, our lower bound shows that any two-round PSMT beating our lower bound must bypass this assumption. We leave this as a future direction.

Comparison to Previous Version. Our previous version does not include this assumption and prove the same lower bound. However, one of the conference referees points out that Eve may not learn the whole transmission in the second round if the code used by Alice and Bob are not fixed in this situation. We thank for his valuable comment which helps us to fix this bug. We also emphasize that in all known efficient PSMT protocols, Eve can predict the code used by Alice and Bob. This means our new assumption holds for these constructions. To beat our construction, one has to design a PSMT protocol bypassing this assumption.

Technical Comparison to Previous Works. Our protocol achieving transmission rate $2n$ utilizes ideas from prior works, and we would like to take a moment here to properly acknowledge them. The idea of leveraging the syndrome space and pseudobasis to correct errors was first introduced by Kurosawa and Suzuki in [4]. They also proposed the idea of generalized broadcast to decrease the communication cost of the second round. Spini and Zémor [6] further developed this idea by showing how to spot a codeword with large error. They also abandon the dependency on the codeword communicated in the first round in [4] which greatly simplified the technique. These ideas also appear in our protocol; in particular, the first round of our protocol matches that of [6].

To obtain a more efficient PSMT protocol, we observe that the protocol in [6] divided the size of the global support of the errors into two cases: the small and the big one. In the second round, Alice transmits information for both of the potential cases. Thus, in some

[6] It might be possible that Alice and Bob use different codes with same minimum distance $n = 2t+1$ in the second round. In this case, Bob and Alice must share the code information which is kept secret from Eve. We are not aware of any construction with this property and can not be sure that such strategy will gain them any advantage.

sense, half of her communication is wasted. Dealing with both cases simultaneously required a more careful analysis of the syndrome space to generate the required masks: we exploit linear dependencies amongst the syndromes, unlike [6] that used a decoding algorithm, which itself was already a key improvement over the protocol in [4]. Furthermore, the approach in [6] sends back syndrome vectors whose lengths are always $t+1$. In our protocol, we exploit the codewords in the pseudobasis S to correct the error, allowing us to only send back $|S|$ symbols to identify the vector. The bigger $|S|$ is, the more errors can be detected, permitting the use of more efficient generalized broadcast.

On the other hand, the lower bound argument is new, except that the need for broadcast in the second round is also mentioned in the $O(n)$ lower bound argument [7].

2 Preliminaries

Notations. For an integer $n \geq 1$, we denote $[n] := \{1, 2, \ldots, n\}$. By default, log denotes the base-2 logarithm.

Throughout, $\mathbb{F}_q$ denotes the finite field with q elements, for q a prime power. We let n denote the number of channels through which Alice and Bob may communicate and t the number of channels Eve may corrupt; we focus exclusively on the $n = 2t + 1$ case. The complexity measure of a protocol that concerns us is its *transmission rate*, defined as the total number of symbols communicated divided by the number of symbols of the transmitted secret. The length of the transmitted secret is denoted by ℓ. By $o_{\ell\to\infty}(1)$ we refer to a quantity which tends to 0 as $\ell \to \infty$ (fixing all other parameters, including n), and we write $f(\ell) \sim g(\ell)$ if $\lim_{\ell\to\infty} \frac{f(\ell)}{g(\ell)} = 1$ (again, fixing all other parameters).

▶ Remark 1. As usual, a *bit* refers to an element of $\{0, 1\}$, while in this work, a *symbol* refers to an element from the field $\mathbb{F}_q$, and we will need $q \geq n$. While it is most natural to measure the total communication in bits, as our protocols will involve transmitting elements of $\mathbb{F}_q$ it is more convenient for us to talk about the number of symbols transmitted. Note that when we compute the transmission rate and we assume the length of the secret is a growing parameter, whether we measure the communication in bits or symbols does not matter. However, when we present our lower bound proof in Section 4 it will be most convenient for us to talk about bits.

Codes. As in previous works, our protocols rely crucially on linear codes with desirable properties. For two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{F}_q^n$, the *(Hamming) distance* between them is $d(\mathbf{x}, \mathbf{y}) := |\{i \in [n] : x_i \neq y_i\}|$. Given a vector $\mathbf{x}$ and a subset $\mathcal{Y} \subseteq \mathbb{F}_q^n$ we denote $d(\mathbf{x}, \mathcal{Y}) := \min\{d(\mathbf{x}, \mathbf{y}) : \mathbf{y} \in \mathcal{Y}\}$. The *(Hamming) weight* of a vector is $\mathrm{wt}(\mathbf{x}) := d(\mathbf{x}, \mathbf{0})$. The *support* of $\mathbf{x}$ is $\mathrm{supp}(\mathbf{x}) := \{i \in [n] : x_i \neq 0\}$. Note that $\mathrm{wt}(\mathbf{x}) = |\mathrm{supp}(\mathbf{x})|$ and $d(\mathbf{x}, \mathbf{y}) = |\mathrm{supp}(\mathbf{x} - \mathbf{y})|$. For a vector $\mathbf{x} \in \mathbb{F}_q^n$ and a subset $S \subseteq [n]$, $\mathbf{x}|_S := (x_i)_{i\in S}$ denotes the length $|S|$ vector obtained by projecting on the coordinates indexed by S. By a *(linear) code*, we refer to a linear subspace $\mathcal{C} \leq \mathbb{F}_q^n$; n is the *block-length*, $k = \dim(\mathcal{C})$ is the *dimension* and $d = \min\{\mathrm{wt}(\mathbf{c}) : \mathbf{c} \in \mathcal{C} \setminus \{\mathbf{0}\}\}$ is the *(minimum) distance*. We refer to such a code as an $[n, k, d]_q$ code.

A code is called *maximum distance separable (MDS)* if $d = n - k + 1$. Such codes exist whenever $q \geq n$ and are furnished by the well-known Reed-Solomon (RS) codes defined via the evaluations of degree $\leq k - 1$ polynomials. However, in this work, we will not directly use the specific structure of RS codes,[7] so we will state our results for arbitrary linear MDS codes.

[7] Although in order to implement the protocol efficiently we will use the existence of efficient encoding and decoding algorithms for RS codes.

Any linear code $\mathcal{C}$ may be described as the kernel of a matrix, i.e., $\mathcal{C} = \{\mathbf{x} \in \mathbb{F}_q^n : \mathbf{H}\mathbf{x} = \mathbf{0}\}$. Such a matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$ is called a *parity-check matrix*.

Given two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$ we define their *inner product* via $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i$. We will need the following lemma from [6]. It states that there exists an MDS code $\mathcal{C} \leq \mathbb{F}_q^n$ of dimension t for $n = 2t + 1$ for which one can find a vector $\mathbf{h} \in \mathbb{F}_q^n$ such that, even once t coordinates are revealed from a codeword $\mathbf{c} \in \mathcal{C}$, the inner-product $\langle \mathbf{h}, \mathbf{c} \rangle \in \mathbb{F}_q$ is completely unconstrained.

▶ **Lemma 2** (Lemma 1 from [6]). *For any n and any $t < n$ there exists a linear MDS code $\mathcal{C}$ of parameters $[n, t+1, n-t]$ and a vector $\mathbf{h} \in \mathbb{F}_q^n$ such that given a uniformly random codeword $\mathbf{c} \in \mathcal{C}$, the scalar product $\langle \mathbf{h}, \mathbf{c} \rangle$ is a uniformly random element of $\mathbb{F}_q$, even when conditioned on any t symbols of $\mathbf{c}$. Moreover, $\mathbf{h}$ can be found efficiently.*

Formally, for any $1 \leq i_1 < i_2 < \cdots < i_t \leq n$ and $\alpha_1, \alpha_2, \ldots, \alpha_t, \beta \in \mathbb{F}_q$, we have

$$\Pr[\langle \mathbf{h}, \mathbf{c} \rangle = \beta | \mathbf{c}_{i_1} = \alpha_1, \mathbf{c}_{i_2} = \alpha_2, \ldots, \mathbf{c}_{i_t} = \alpha_t] = \frac{1}{q} \ ,$$

where the randomness is over the uniformly random $\mathbf{c} \in \mathcal{C}$.

▶ Remark 3. We note that any such vector $\mathbf{h}$ must not lie in the dual of $\mathcal{C}$, and moreover that it must have weight at least $t + 1$.

Broadcast. Next, observe that since Eve controls at most $t < n/2$ of the channels, if Alice transmits the same symbol through all n channels, then Bob can always recover Alice's intended symbol by choosing the majority symbol. Of course, such a procedure does not guarantee any privacy, i.e., Eve will always learn the symbol Alice transmits to Bob.

Pseudobases

An important technical tool in our protocols are *pseudobases*, as introduced in the work of Kurosawa and Suzuki [4]. Before providing the definition, we explain their utility. (A similar discussion of the utility of pseudobases is available in Section 3.2 of [6].) Consider the scenario where Bob has sent a codeword $\mathbf{c} \in \mathcal{C}$ to Alice by sending the i-th coordinate c_i through the i-th channel. In order to guarantee privacy, as Eve can observe t of the channels, it must be that $\dim \mathcal{C} \geq t + 1$. However, by the Singleton bound, that forces the distance of $\mathcal{C}$ to be at most $n - (t + 1) + 1 = n - t = t + 1$, which means that Bob can uniquely decode Alice's transmission only if Eve introduces $\leq t/2$ errors. However, as Eve can introduce up to t errors, it appears that we do not have an effective means of enforcing reliability.

However, consider the following scenario: instead of sending a single codeword through the channel in this way, Bob sends many codewords $\mathbf{c}_1, \ldots, \mathbf{c}_r$. Privacy is preserved so long as the transmissions are not correlated in any way (say, each one is sampled independently and uniformly at random). However, Alice now has an advantage in decoding: all of the corruptions introduced by Eve are confined to the same set of t coordinates. The idea is to exploit this fact to allow Alice and Bob to agree on some codeword $\bar{\mathbf{c}}$ of which Eve knows at most t coordinates (which in turn means that $\langle \mathbf{h}, \bar{\mathbf{c}} \rangle$ can effectively mask the secret m). Using the concept of pseudobases, it turns out that this is possible (so long as the distance of $\mathcal{C}$ is at least $t + 1$, as is the case when $\mathcal{C}$ is MDS).

We now provide the formal definition of a pseudobasis.

▶ **Definition 4** (Pseudobasis [4]). *Let $\mathbf{y}_1, \ldots, \mathbf{y}_s \in \mathbb{F}_q^n$ be vectors. A* pseudobasis *for $\mathbf{y}_1, \ldots, \mathbf{y}_s$ is a subcollection $\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_r}$ with $1 \leq i_1 < \cdots < i_r \leq s$ such that $\mathbf{H}\mathbf{y}_{i_1}, \ldots, \mathbf{H}\mathbf{y}_{i_r} \in \mathbb{F}_q^{n-k}$ is a basis for the linear space* $\operatorname{span}\{\mathbf{H}\mathbf{y}_1, \ldots, \mathbf{H}\mathbf{y}_s\}$.

In other words, one computes a basis for the space spanned by $\mathbf{H}\mathbf{y}_1, \ldots, \mathbf{H}\mathbf{y}_s \in \mathbb{F}_q^{n-k}$, and then the preimage of the basis vectors in $\mathbb{F}_q^n$ provides a pseudobasis. Observe that, given access to $\mathbf{H}$, such a pseudobasis can be found in time polynomial in n, and furthermore that it consists of at most $n-k$ vectors.

▶ Remark 5. Note that if we have a code $\mathcal{C} \leq \mathbb{F}_q^n$ with parity-check matrix $\mathbf{H}$ and we write $\mathbf{y}_i = \mathbf{c}_i + \mathbf{e}_i$ for each $i \in [s]$ with $\mathbf{c}_i \in \mathcal{C}$, then as

$$\mathbf{H}\mathbf{y}_i = \mathbf{H}(\mathbf{c}_i + \mathbf{e}_i) = \mathbf{H}\mathbf{c}_i + \mathbf{H}\mathbf{e}_i = \mathbf{H}\mathbf{e}_i \ ,$$

we conclude that $\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_r}$ forms a pseudobasis for $\mathbf{y}_1, \ldots, \mathbf{y}_s$ if and only if $\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_r}$ forms a pseudobasis for $\mathbf{e}_1, \ldots, \mathbf{e}_s$.

This observation will be crucial for us in our privacy analysis. We will be in the scenario that Alice has received potentially corrupted codewords from Bob, which we write as $\tilde{\mathbf{c}}_i = \mathbf{c}_i + \mathbf{e}_i$, where $\mathbf{e}_i$ denotes the errors introduced by Eve. Alice will then broadcast some information about a pseudobasis for her received vectors to Bob. This does not leak any information to Eve, as she could have computed the same pseudobasis from the error vectors $\mathbf{e}_i$ that she knows.

3 The Protocol

In this section, we present our protocol which allows Alice to privately and reliably transmit an ℓ symbol secret $(m_1, \ldots, m_\ell) \in \mathbb{F}_q^\ell$ to Bob. In order to ease readability, we present two simplifications of our full protocol first before presenting the full construction. The first construction, presented in Section 3.1, allows Alice to transmit a one symbol secret $m \in \mathbb{F}_q$. Despite being fairly simple, it already introduces a crucial idea, which is a method for Alice and Bob to agree on a random codeword that is not completely revealed to Eve. As we elaborate upon further in Remark 8, this means of extracting this secret codewords represents our core improvement over [6].

Next, in Section 3.2, we show how to generalize the protocol to the case of $\ell \geq 1$, and achieve communication rate $(4 + o_{\ell\to\infty}(1))n$. Intuitively, this requires Alice and Bob to agree on ℓ random codewords that are not completely known to Eve. In order to guarantee small transmission rate, we need a few more tricks. As in [6], one useful technique we employ is a method for Alice to find a vector which indicates many of the channels that Eve is corrupting, allowing Bob to safely ignore those channels.[8] Informally, this transforms symbol corruptions into erasures, and erasures are easier to recover from. In particular, Alice can encode her data with a code of higher rate and Bob will still be able to uniquely-decode. To get our final protocol achieving transmission rate $(2 + o_{\ell\to\infty}(1))n$, we note that we only need to do something different if Eve invests many corruptions in the first round.[9] In order to handle this, we ask Alice to send a bit more information to Bob to indicate a larger number of corrupted channels, which transforms more of the symbol corruptions into erasures in the subsequent transmissions, and hence allows Alice to use an error-correcting code of higher rate. We describe the necessary modifications in Section 3.3.

[8] There is a procedure with the same guarantee in [6]; however, we believe our procedure is simpler, and moreover does not use the specific structure of RS codes.

[9] More precisely, if the dimension of the syndrome space exceeds $t/3$.

Notations for this section. Throughout, $\mathcal{C} \leq \mathbb{F}_q^n$ denotes an MDS code of dimension $t+1$ and $\mathbf{h} \in \mathbb{F}_q^n$ a vector satisfying the conclusion of Lemma 2. Also, $\mathbf{H} \in \mathbb{F}_q^{t \times n}$ denotes a parity-check matrix for $\mathcal{C}$. The datum $(\mathcal{C}, \mathbf{h}, \mathbf{H})$ is *public*, fixed prior to the execution of the protocol and available to Alice, Bob and Eve throughout the execution. Lastly, we denote by $E \subseteq [n]$ the set of t channels that Eve controls. Of course, this set is unknown to Alice and Bob; we introduce this notation exclusively for the analysis.

3.1 A Simple Protocol for $\ell = 1$

We begin by providing a simple protocol which allows Alice to transmit one secret symbol $m \in \mathbb{F}_q$ to Bob. While this does not achieve our main goal, we find that it clarifies our means of extracting a codeword known to both Alice and Bob but secret from Eve, which we call $\bar{\mathbf{c}}$ and $\mathbf{c}'$. As we discuss further in Remark 8, this idea is the core of what allows us to go beyond the protocol of [6] and eventually compress Alice's communication to just $\sim n\ell$ symbols. The details of the protocol are provided in Algorithm 1.

We now sketch why the protocol indeed yields a PSMT.

Reliability. First, we argue that Lines 8 and 9 from Algorithm 1 are justified, i.e., that Alice can indeed find $p \in [t+1]$ and $\lambda_j \in \mathbb{F}_q$ for $j \in [t+1] \setminus \{p\}$ such that $\mathbf{s}_p = \sum_{j \neq p} \mathbf{s}_j$. As $\mathbf{s}_1, \ldots, \mathbf{s}_{t+1} \in \mathbb{F}_q^t$ are $t+1$ vectors in a t-dimensional space, they must satisfy a nontrivial linear dependence $\sum_{j=1}^{t+1} \lambda'_j \mathbf{s}_j = \mathbf{0}$. Alice can thus pick any $p \in [t+1]$ for which $\lambda'_p \neq 0$, and then set $\lambda_j = -\lambda'_j/\lambda'_p$ for $j \in [t+1] \setminus \{p\}$.

Now, the important observation is that since the code $\mathcal{C}$ has distance $t+1$, we have $\mathbf{c}' = \bar{\mathbf{c}}$. Indeed, first note that $\bar{\mathbf{c}} \in \mathcal{C}$, as

$$\mathbf{H}\bar{\mathbf{c}} = \mathbf{H}\left(\tilde{\mathbf{c}}_p - \sum_{j \neq p} \lambda_j \tilde{\mathbf{c}}_j\right) = \mathbf{H}\tilde{\mathbf{c}}_p - \sum_{j \neq p} \lambda_j \mathbf{H}\tilde{\mathbf{c}}_j = \mathbf{s}_p - \sum_{j \neq p} \lambda_j \mathbf{s}_j = \mathbf{0} \ .$$

Now, recalling that $E \subseteq [n]$ denotes the channels that the adversary controls, the coordinates on which each $\mathbf{c}_j$ can disagree with $\tilde{\mathbf{c}}_j$ are confined to the set E. Thus, the support of $\left(\mathbf{c}_p - \sum_{j \neq p} \lambda_j \mathbf{c}_j\right) - \left(\tilde{\mathbf{c}}_p - \sum_{j \neq p} \lambda_j \tilde{\mathbf{c}}_j\right)$ is also contained in the set E. As $|E| \leq t$, we conclude that the codewords $\mathbf{c}' = \mathbf{c}_p - \sum_{j \neq p} \lambda_j \mathbf{c}_j$ and $\tilde{\mathbf{c}} = \tilde{\mathbf{c}}_p - \sum_{j \neq p} \lambda_j \tilde{\mathbf{c}}_j$ are distance at most t from one another; as $\mathcal{C}$ has distance $t+1$, they must be the same vector.

Thus, in particular, $\langle \mathbf{h}, \mathbf{c}' \rangle = \langle \mathbf{h}, \bar{\mathbf{c}} \rangle$, so $m' - \langle \mathbf{h}, \mathbf{c}' \rangle = m + \langle \mathbf{h}, \bar{\mathbf{c}} \rangle - \langle \mathbf{h}, \mathbf{c}' \rangle = m$, i.e., Bob returns Alice's intended secret m.

Privacy. In the first round of the protocol, Eve can only see $|E| \leq t$ symbols from each transmitted codeword. As the code $\mathcal{C}$ has dimension $t+1$ and is MDS, Eve learns only learns these $|E|$ symbols from $\mathbf{c}_1, \ldots, \mathbf{c}_{t+1}$.

In the second round, Eve sees $(p, \lambda_j : j \neq p)$. However, she already knows $\mathbf{e}_1, \ldots, \mathbf{e}_{t+1}$ and $\mathbf{H}$ and, using the fact that $\mathbf{s}_j = \mathbf{H}\tilde{\mathbf{c}}_j = \mathbf{H}\mathbf{e}_j$ for $j \in [t+1]$, $(p, \lambda_j : j \neq p)$ can be computed from $\mathbf{e}_1, \ldots, \mathbf{e}_{t+1}$ and $\mathbf{H}$. Thus, she does not learn anything from the second transmission.

We conclude that after the protocol, Eve has only learned the symbols indexed by the corrupted channels E from $\mathbf{c}_1, \ldots, \mathbf{c}_{t+1}$. In particular, Eve only knows t symbols of $\mathbf{c}' = \bar{\mathbf{c}} = \tilde{\mathbf{c}}_p - \sum_{j \neq p} \lambda_j \tilde{\mathbf{c}}_j$ which is a codeword distributed uniformly at random in $\mathcal{C}$, and so Lemma 2 guarantees that Eve has no information on $\langle \mathbf{h}, \bar{\mathbf{c}} \rangle$. Thus, even after observing $m + \langle \mathbf{h}, \bar{\mathbf{c}} \rangle$, she has no information on m, as desired.

Communication Cost. In the first round, Bob transmits $(t+1)n \sim n^2/2$ symbols. In the second round, Alice transmits $\log_q(t+1) + tn + n \sim n^2/2$ symbols. Hence, to communicate a single symbol, the total communication requirement of Algorithm 1 is $\sim n^2$. In terms of bits, as we require $q \geq n$, we conclude that Alice and Bob must transmit $\sim n^2 \log n$ bits.

3.2 A Protocol with $(4 + o_{\ell\to\infty}(1))n$ Transmission Rate

In this subsection, we provide a protocol that will allow Alice to transmit an ℓ symbol secret to Bob requiring only $\sim 4n\ell$ symbols to be communicated. We begin by outlining some of the new ingredients we need.

Generalized Broadcast. One technique that we will use in our protocol is *generalized broadcast*, as introduced in previous works [4, 6]. The situation that motivates the idea of generalized broadcast is the following: imagine that in some way, Bob has become aware that Eve is controlling some set $R \subseteq [n]$ of the channels. Then, when decoding a transmission from Alice, he can replace the symbols he receives through the channels in R by an erasure symbol. Thus, instead of decoding from t symbol corruptions, he only has to perform the easier task of decoding from $t - r$ symbol corruptions and r erasures, where $r = |R|$.

In particular, to uniquely decode from t errors where $n = 2t + 1$, if Alice wants to guarantee that the codeword she transmits can be uniquely-decoded by Bob, then she must use a code with distance $2t + 1 = n$: by the Singleton bound, she must use an MDS code of dimension 1, i.e., she can only send a single symbol. A natural example of a dimension 1 MDS code is the repetition code: this precisely recovers broadcast as introduced earlier.

However, if Bob knows a subset R as above, then he can uniquely decode so long as the code has distance at least $2(t - r) + r + 1 = n - r$. Thus, if Alice uses an MDS code of dimension $r + 1$, Bob can recover her intended transmission. We refer to this as r-generalized broadcast, which we now formally define.

▶ **Definition 6** (Generalized Broadcast). *For an integer $r \geq 0$, r-*generalized broadcast *refers to the procedure where Alice uses an $[n, r + 1, n - r]_q$ code $\mathcal{C}_r$ to transmit $r + 1$ symbols $(x_1, \ldots, x_{r+1}) \in \mathbb{F}_q^{r+1}$ by encoding the message $(x_1, \ldots, x_{r+1})$ into a codeword $\mathbf{c} \in \mathcal{C}_r$, and sending the i-th symbol of $\mathbf{c}$ through the i-th channel for each $i \in [n]$.*

For succinctness, we write Alice r-broadcasts $(x_1, \ldots, x_{r+1})$ *to indicate that Alice uses the r-generalized broadcast to transmit the data $(x_1, \ldots, x_{r+1})$ to Bob.*

▶ Remark 7. Assuming Alice and Bob communicate with a dimension $r + 1$ Reed-Solomon code, then both encoding the message and decoding from r erasures and $t - r$ symbol corruptions can be done in polynomial time [8].

Thus, r-generalized broadcast allows Alice to reliably transmit $r+1$ times more information to Bob than standard (i.e., 0-)broadcast, which can greatly improve the transmission rate of the protocol if r is sufficiently large.

Finding a Set of Corrupted Channels. In light of the above discussion, we would like to allow Bob to find a large set of corrupted channels. For general ℓ, we will have Bob transmit $t+\ell$ uniformly random codewords in the first round, and Alice receives the corrupted codewords $\tilde{\mathbf{c}}_j = \mathbf{c}_j + \mathbf{e}_j$, where the support of each $\mathbf{e}_j$ is contained in the t channels Eve controls, E.

Now, if Alice were aware that $\mathbf{e}_j$ has large weight for some j, then she could just broadcast $\tilde{\mathbf{c}}_j$ and the index j to Bob. Bob could then compute the set $\mathrm{supp}(\tilde{\mathbf{c}}_j - \mathbf{c}_j)$ and subsequently ignore the transmissions sent through those channels. However, one problem is that there might not be an $\mathbf{e}_j$ that has sufficiently large weight. More concerningly, Alice does not actually know $\mathbf{e}_1, \dots, \mathbf{e}_{t+\ell}$!

Dealing with the first issue, note that it actually suffices to find multipliers λ_j such that $\sum_j \lambda_j \mathbf{e}_j$ has large weight: then Alice can broadcast the λ_j's and $\mathbf{y} := \sum_j \lambda_j \tilde{\mathbf{c}}_j$, and then Bob can compute $\mathrm{supp}\left(\mathbf{y} - \sum_j \lambda_j \mathbf{c}_j\right)$ and ignore the subsequent transmissions sent through those channels.

Actually, in order to ensure a good transmission rate it will be important that the linear dependency is chosen to be relatively short; in particular, it should be independent of ℓ. It will turn out that we can find such a vector $\mathbf{y}$ which is a linear combination of a pseudobasis for the vectors $\tilde{\mathbf{c}}_1, \dots, \tilde{\mathbf{c}}_{t+\ell}$. Recalling that the dimension of the syndrome space is at most t, this guarantees that we don't need to transmit too many multipliers λ_j.

However, we still haven't addressed the issue that Alice does not have direct access to the $\mathbf{e}_j$'s. But it turns out that this is not an problem: given a set of vectors with linearly independent syndromes, we will be able to find a linear combination $\sum_j \lambda_j \tilde{\mathbf{c}}_j$ that is far from *every* codeword. So, in particular, it will be far from $\sum_j \lambda_j \mathbf{c}_j$, as required.

Specifically, if $r \le t/3$ and $\mathbf{y}_1, \dots, \mathbf{y}_r \in \mathbb{F}_q^r$ are vectors such that the syndromes $\mathbf{H}\mathbf{y}_1, \dots, \mathbf{H}\mathbf{y}_r \in \mathbb{F}_q^t$ are linearly independent, then Algorithm 4 finds a vector $\mathbf{y}$ in the span of $\mathbf{y}_1, \dots, \mathbf{y}_r$ that satisfies $d(\mathbf{y}, \mathcal{C}) \ge r$. This procedure and its analysis are presented in Appendix D.

▶ Remark 8. There is a procedure in [6] with the same guarantee; however, we believe our algorithm is a bit simpler, so we have chosen to present it. In particular, we do not need to apply a unique-decoding algorithm as is required by the procedure in [6]; we just use simple linear-algebraic operations.

A more significant difference between our protocols concerns the communication of the masked secrets. For each of the message symbols $m_1, \dots, m_\ell$, the most efficient protocol of [6] requires Alice to broadcast *two* symbols $z_1^{(i)}, z_2^{(i)} \in \mathbb{F}_q$ which each mask the message symbol m_i in a different way. The symbol $z_1^{(i)}$ uses the mask $\langle \mathbf{h}, \mathbf{y}_{p_i} \rangle$; $z_2^{(i)}$ uses the mask $\langle \mathbf{h}, \tilde{\mathbf{c}}_{p_i} \rangle$ where $\tilde{\mathbf{c}}_{p_i}$ is the decoding of $\mathbf{y}_{p_i}$, or $z_2^{(i)}$ is just set to 0 if the decoding failed. Bob then chooses which mask to open, depending on the size of the pseudobasis. The authors comment they could use generalized broadcast for these symbols (as we do) to somewhat decrease the communication cost; however, even this change would not bring the second round communication down to $\sim n\ell$. Thus, a key difference between our protocols can be observed: by more carefully exploiting the structure of the pseudobasis, our extraction of the codewords $\bar{\mathbf{c}}_{p_i} = \mathbf{c}'_{p_i}$ to yield the masks $\langle \mathbf{h}, \bar{\mathbf{c}}_i \rangle$ prevents us from needing to use two different masks to guarantee that Bob can reliably recover the message symbols.

The Protocol. We are now in position to give our PSMT for transmitting an ℓ symbol secret: the details are in Algorithm 2.

▶ **Theorem 9.** *Algorithm 2 is a PSMT with transmission rate* $(4 + o_{\ell \to \infty}(1))n$.

Proof. We first verify that the protocol is reliable. After, we show that it is private. Lastly, we compute its transmission rate. Throughout the proof, we let $E \subseteq [n]$ denote the set of t channels that Eve is corrupting.

Reliability. We first make a few observations to justify the algorithm. First, we note that the definition of T on Appendix B is valid: indeed, $r = |S| \leq t$ since a pseudobasis has size at most t, so there are at least ℓ elements in $[t+\ell] \setminus S$. Also, we note that $\mathbf{z} = \sum_{j \in S} \lambda_j \mathbf{c}_j \in \mathcal{C}$, so since $\mathbf{y}$ is at distance at least r' from $\mathcal{C}$, we have $|\mathrm{supp}(\mathbf{z}-\mathbf{y})| = d(\mathbf{z},\mathbf{y}) \geq r'$, as stated in Appendix B. Furthermore, as $\mathbf{y} = \sum_{j \in S} \lambda_j \tilde{\mathbf{c}}_j$, if $E \subseteq [n]$ denotes the set of channels that Eve controls, then $\mathrm{supp}(\mathbf{y}-\mathbf{z}) \subseteq E$. Hence, for each $i \in [\ell]$, the transmission from Alice to Bob of $(\lambda_{ij} : j \in S)$ and $\langle \mathbf{h}, \bar{\mathbf{c}}_{p_i} \rangle + m_i$ via r'-generalized broadcast is reliable.

As in the analysis in Section 3.1, the reliability of Algorithm 2 follows from the fact that for $i = 1, \ldots, \ell$, we have $\bar{\mathbf{c}}_{p_i} = \mathbf{c}'_{p_i}$. And once again, the argument proceeds by demonstrating that both $\bar{\mathbf{c}}_{p_i}$ and $\mathbf{c}'_{p_i}$ are elements of $\mathcal{C}$. This is clear for $\mathbf{c}'_{p_i}$; for $\bar{\mathbf{c}}_{p_i}$, we use the parity-check matrix $\mathbf{H}$:

$$\mathbf{H}\bar{\mathbf{c}}_{p_i} = \mathbf{H}\left(\tilde{\mathbf{c}}_{p_i} - \sum_{j \in S} \lambda_{ij} \tilde{\mathbf{c}}_j\right) = \mathbf{s}_{p_i} - \sum_{j \in S} \lambda_{ij} \mathbf{s}_j = \mathbf{0} \ .$$

Now, since $\mathrm{supp}(\mathbf{c}_j - \tilde{\mathbf{c}}_j) \subseteq E$ for each $j \in [t+\ell]$, we also have

$$\mathrm{supp}(\mathbf{c}'_{p_i} - \bar{\mathbf{c}}_{p_i}) = \mathrm{supp}\left(\left(\mathbf{c}_{p_i} - \sum_{j \in S} \lambda_{ij} \mathbf{c}_j\right) - \left(\tilde{\mathbf{c}}_{p_i} - \sum_{j \in S} \lambda_{ij} \tilde{\mathbf{c}}_j\right)\right) \subseteq E \ ,$$

which implies $d(\mathbf{c}'_{p_i}, \bar{\mathbf{c}}_{p_i}) \leq |E| \leq t$. As $\mathcal{C}$ has distance $t+1$, it follows that $\mathbf{c}'_{p_i} = \bar{\mathbf{c}}_{p_i}$. In particular, we have $\langle \mathbf{h}, \mathbf{c}'_{p_i} \rangle = \langle \mathbf{h}, \bar{\mathbf{c}}_{p_i} \rangle$.

Hence, for each $i \in [\ell]$, $m'_i - \langle \mathbf{h}, \mathbf{c}'_{p_i} \rangle = m_i + \langle \mathbf{h}, \bar{\mathbf{c}}_{p_i} \rangle - \langle \mathbf{h}, \mathbf{c}'_{p_i} \rangle = m_i$, demonstrating reliability.

Privacy. First, we describe Eve's view of the protocol. In the first round, she observes $(\mathbf{c}_1)|_E, \ldots, (\mathbf{c}_{t+\ell})|_E$. In the second round, she first observes $(S, (\lambda_j : j \in S), \mathbf{y})$. Then, for each $i \in [\ell]$, she observes $(\lambda_{ij} : j \in S)$ and $m'_i = \langle \mathbf{h}, \bar{\mathbf{c}}_{p_i} \rangle + m_i$.

We wish to establish that Eve learns nothing about the symbols m_i for each $i \in [\ell]$. To establish this, it suffices to show that, conditioned on Eve's view, $\langle \mathbf{h}, \bar{\mathbf{c}}_{p_i} \rangle$ is a uniformly random element of $\mathbb{F}_q$. And to do this, according to Lemma 2, it suffices to show that from Eve's perspective, $\bar{\mathbf{c}}_{p_i}$ is a uniformly random codeword from which Eve has observed only t coordinates.

First of all, as $\mathbf{c}_1, \ldots, \mathbf{c}_{t+\ell}$ are sampled independently and uniformly from $\mathcal{C}$ and $\mathcal{C}$ has dimension $t+1$ and is MDS, after the first round Eve only learns $(\mathbf{c}_j)|_E$ for each $j \in [t+\ell]$.

Next, we consider the second round. We begin by noting that Eve can compute S from $\mathbf{H}$ and $\mathbf{e}_1, \ldots, \mathbf{e}_{t+\ell}$, which she knows. Indeed, as $\mathbf{s}_j = \mathbf{H}\check{\mathbf{c}}_j = \mathbf{H}\mathbf{e}_j$, Eve can also compute the pseudobasis S. So she learns nothing from this transmission. Once she has computed S Eve can then compute the set T and subsequently $(\lambda_{ij} : j \in S)$ for each $i \in [\ell]$, as the λ_{ij}'s are a function of the sets S and T and the syndromes $\mathbf{s}_1, \ldots, \mathbf{s}_{t+\ell}$, to which she has access.

Next, consider revealing to Eve the codewords $(\mathbf{c}_j : j \in S)$. Then, she can compute the corrupted codeword $\tilde{\mathbf{c}}_j = \mathbf{c}_j + \mathbf{e}_j$ for $j \in S$, so she can then compute the vector $\mathbf{y}$ and the multipliers $(\lambda_j : j \in S)$. Hence, what Eve sees in the second round is at most as informative as $(\mathbf{c}_j : j \in S)$.

Hence, at the termination of the protocol, what Eve can infer from her view about the masks $\langle \mathbf{h}, \bar{\mathbf{c}}_{p_i} \rangle$ for $i \in [\ell]$ is no more than what she can infer about them from the following data:

- The codewords $(\mathbf{c}_j : j \in S)$;
- The coordinates of all the codewords indexed by E, i.e., $(\mathbf{c}_j)|_E$ for $j \in [t+\ell]$.

Recall that, for each $i \in [\ell]$, $\bar{\mathbf{c}}_{p_i} = \mathbf{c}'_{p_i} = \mathbf{c}_{p_i} - \sum_{j \in S} \lambda_{ij} \mathbf{c}_j$. On the one hand, from the two pieces of data above, we have shown that Eve can compute exactly $\sum_{j \in S} \lambda_{ij} \mathbf{c}_j$. On the other hand, as the $\mathbf{c}_j$'s are sampled independently, the above data reveals nothing about $\mathbf{c}_{p_i}$ other than the coordinates indexed by E. Thus, from Eve's perspective, $\bar{\mathbf{c}}_{p_i} = \mathbf{c}_{p_i} - \sum_{j \in S} \lambda_{ij} \mathbf{c}_j$ is a uniformly random codeword from which she has only observed the coordinates indexed by E. Therefore the messages $m'_i = m_i + \langle \mathbf{h}, \bar{\mathbf{c}}_{p_i} \rangle$ reveal nothing about the secret vector $(m_1, \ldots, m_\ell)$. This concludes the proof of the assertion that the protocol is private.

Transmission Rate. In the first round, Bob sends $(t+\ell)n$ symbols. In the second round, Alice first broadcasts $\frac{r \log(t+\ell)}{\log q} + r + n$ symbols and then r'-broadcasts $\ell(r+1)$ symbols, where we recall that r denotes the size of the pseudobasis and $r' = \min\{r, \lfloor t/e \rfloor\}$. This requires her to send

$$\frac{nr \log(t+\ell)}{\log q} + (r+n)n + (r+1)\ell \frac{n}{r'+1}$$

elements from $\mathbb{F}_q$. Thus, if N is the total number of symbols transmitted, then $\frac{N}{\ell}$ is

$$\frac{tn}{\ell} + n + \frac{nr \log(t+\ell)}{\ell \log q} + \frac{n^2 + rn}{\ell} + \frac{(r+1)n}{r'+1} \leq 4n + O\left(\frac{n^2}{\ell} + \frac{n^2 \log(n+\ell)}{\ell \log n}\right), \tag{1}$$

where the inequality uses $q \geq n$, $r \leq t \leq n$ and $\frac{r+1}{r'+1} \leq 3$. Hence, assuming $\ell = \omega(n)$ we have $\frac{N}{\ell} \sim 4n$, as promised. ◀

▶ Remark 10. Note that if we had been in the case that $r = r'$, i.e., $r \leq \frac{t}{3}$, then the transmission rate of Algorithm 2 would have been $\sim 2n$. Hence, in order to get our desired transmission rate of $2n$, we will only have to amend the protocol in the case that $r > \frac{t}{3}$. This is what we do in the following subsection.

3.3 Protocol with $(2 + o_{\ell \to \infty}(1))n$ Transmission Rate

In order to decrease the transmission rate to $\sim 2n$, we look more carefully at the transmission rate as computed in (1). We have a factor of $\sim n$ from the first round when Bob communicates to Alice, and then a factor of $\sim 3n$ when Alice replies to Bob in the second round. In our lower bound argument, we will show that both parties will have to communicate $n\ell$ symbols in each round; hence, our only hope of getting a $\sim 2n$ transmission rate will be to decrease the communication of Alice in the second round.

Now, we note that the dominant term in Alice's communication is the $\frac{(r+1)n}{r'+1}\ell$ term which comes from the ℓ r'-generalized broadcasts from Appendix B; as $r' \leq \frac{t}{3}$ and r can be as large as t, this term could be as large as $3n\ell$. If Alice used r-generalized broadcast for each of these transmissions, then this communication would cost only $\sim n\ell$ symbols, and we would get the $\sim 2n$ transmission rate we desire. However, as $\mathbf{y}$ only informs Bob of r' corrupted channels, if $r > r' = \min\{r, \lfloor t/3 \rfloor\}$ then Alice will have to communicate some more information for Bob to learn of r corrupted channels, which will guarantee the reliability of the transmission.

The solution for this is rather simple. We assume from now on that $r > r'$, which is the same as saying $r > \frac{t}{3}$. First, Alice broadcasts $(\mathbf{y}, S, \lambda_j : j \in S)$ as before (see Appendix B); thus, $t/3$-generalized broadcast is now reliable. Next, we have Alice $t/3$-generalized broadcast the entire pseudobasis to Bob, i.e., all the vectors $\tilde{\mathbf{c}}_j$ for $j \in S$. We claim that this implies that r-generalized broadcast will now be reliable. Indeed, this follows from the following simple lemma.

▶ **Lemma 11.** *Let* $\tilde{\mathbf{c}}_j = \mathbf{c}_j + \mathbf{e}_j$ *for* $j \in S$ *with* $\mathbf{c}_j \in \mathcal{C}$ *and put* $\mathbf{s}_j = \mathbf{H}\tilde{\mathbf{c}}_j = \mathbf{H}\mathbf{e}_j$*. Assume that* $\dim(\mathrm{span}\{\mathbf{s}_j : j \in S\}) = r$*. Then* $\left|\bigcup_{j\in S} \mathrm{supp}(\mathbf{e}_j)\right| \geq r$*.*

Proof. Let $\mathbf{d}_i \in \mathbb{F}_q^n$ denote the vector whose i-th coordinate is 1 and the remaining coordinates are 0. Let $R = \bigcup_{j\in S} \mathrm{supp}(\mathbf{e}_j)$; then clearly $\mathrm{span}\{\mathbf{d}_i : i \in R\} \supseteq \mathrm{span}\{\mathbf{e}_j : j \in S\}$, so also

$$\mathrm{span}\{\mathbf{H}\mathbf{d}_i : i \in R\} \supseteq \mathrm{span}\{\mathbf{H}\mathbf{e}_j : j \in S\} = \mathrm{span}\{\mathbf{s}_j : j \in S\}.$$

As $\dim(\mathrm{span}\{\mathbf{H}\mathbf{d}_i : i \in R\}) \leq |R|$, we conclude $|R| \geq \dim(\mathrm{span}\{\mathbf{s}_j : j \in S\}) = r$, as desired. ◀

Thus, suppose Alice reliably transmits to Bob the vectors $\tilde{\mathbf{c}}_j$ for $j \in S$. From this, Bob can compute the set $\bigcup_{j\in S} \mathrm{supp}(\mathbf{c}_j - \tilde{\mathbf{c}}_j) = \bigcup_{j\in S} \mathrm{supp}(\mathbf{e}_j)$; this set has cardinality at least r, and moreover it is contained in E (where, as usual, E denotes the set of channels Eve controls). Hence, there are now r channels that Bob can safely ignore, so Alice may reliably r-broadcast the ℓ transmissions $(\lambda_{ij} : j \in S)$ and $\langle \mathbf{h}, \bar{\mathbf{c}}_{p_i} \rangle + m_i$, as in Appendix B.

It is reasonable now to wonder if this will negatively impact the privacy of the protocol, as more information is revealed to Eve. However, by observing the proof of Theorem 9, one can see that even if Eve learns of $\tilde{\mathbf{c}}_j$ for $j \in S$, the inner-product $\langle \mathbf{h}, \bar{\mathbf{c}}_{p_i} \rangle$ is still wholly unknown to her, implying that they yield an effective mask for the secrets m_i.

Instead of completely rewriting the protocol, we just indicate in Algorithm 3 the changes that need to be made to Algorithm 2 to obtain the $\sim 2n$ transmission rate.

▶ **Theorem 12.** *Algorithm 3 is a PSMT with transmission rate* $(2 + o_{\ell\to\infty}(1))n$*.*

Proof. The proof is omitted due to page limit. ◀

4 Lower Bound

In this section, we prove a lower bound on the transmission rate of any two-round PSMT under an assumption about the protocol which we now formally introduce.

Our starting point is the observation that in our two-round PSMTs from Section 3, we always have Alice broadcast her desired transmission to Bob which completely sacrifices the privacy of her transmission. That is, the adversary completely learns the transmission from the second round. And this is not unique to our protocols: all of the efficient two-round PSMT protocols from the literature [1, 4, 6] sacrifice the privacy of Alice's transmission.

Therefore, we make the assumption that the adversary learns the entire transmission of the second round and prove a $2n$ lower bound on the transmission rate under this assumption. This argument shows that among all two-round PSMTs satisfying this assumption, the one guaranteed by Theorem 12 is actually optimal. In other words, if one want to design a more efficient PSMT, the second round of this protocol must somehow bypass this assumption and keep something hidden from Eve. In this sense, we prove an inherent limitation for the line of optimizing two-round PSMT protocols [1, 4, 6].

▶ **Assumption 1.** *The adversary learns the whole transmission of the second round. More precisely, there is a function mapping the symbols Alice transmits through* t *of the channels to the symbols she sends through the other channels.*

▶ **Theorem 13.** *Under Assumption 1, any two-round perfectly secure message transmission of an* ℓ*-bit secret requires communicating* $2n\ell$ *bits.*

Proof. First of all, we formalize the behaviours of the sender Alice and the receiver Bob in a two-round PSMT.

1. In the first round, Bob runs a randomized algorithm $A(\ell)$ to generate a message $\mathbf{a} = (a_1, \ldots, a_n) \in \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ where the randomness is only available to Bob. Bob sends $\mathbf{a}$ to Alice such that a_i is sent through the i-th channel.
2. Alice receives the corrupted vector $\tilde{\mathbf{a}}$ and runs the algorithm $B(\tilde{\mathbf{a}}, s)$ to generate the message $\mathbf{b} = (b_1, \ldots, b_n) \in \mathcal{B}_1 \times \cdots \times \mathcal{B}_n$ where $s \in [2^\ell]$ is the secret. Then Alice sends $\mathbf{b}$ to Bob such that b_i is sent through the i-th channel.
3. Bob receives the corrupted vector $\tilde{\mathbf{b}}$ and runs the algorithm $C(\tilde{\mathbf{b}}, \mathbf{a})$ to recover the secret. The protocol succeeds if C outputs s and Eve learns nothing about the secret.

Note that if $B(\mathbf{a}, s) = \mathbf{b}$ then we must have $C(\mathbf{b}, \mathbf{a}) = s$, i.e., the protocol must succeed if the adversary Eve injects no errors. We defer the formal proof to the full version. ◀

References

1 Saurabh Agarwal, Ronald Cramer, and Robbert de Haan. Asymptotically optimal two-round perfectly secure message transmission. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 394–408. Springer, 2006. doi:10.1007/11818175_24.

2 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.

3 Matthew Franklin and Rebecca N Wright. Secure communication in minimal connectivity models. *Journal of Cryptology*, 13(1):9–30, 2000.

4 Kaoru Kurosawa and Kazuhiro Suzuki. Truly efficient 2-round perfectly secure message transmission scheme. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 324–340. Springer, 2008.

5 Hasan Md. Sayeed and Hosame Abu-Amara. Efficient perfectly secure message transmission in synchronous networks. *Inf. Comput.*, 126(1):53–61, 1996. doi:10.1006/inco.1996.0033.

6 Gabriele Spini and Gilles Zémor. Perfectly secure message transmission in two rounds. In *Theory of Cryptography Conference*, pages 286–304. Springer, 2016.

7 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 CryptologyConference, 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.

8 Lloyd R Welch and Elwyn R Berlekamp. Error correction for algebraic block codes, December 1986. US Patent 4,633,470.

A Algorithm 1

Algorithm 1 A first protocol for transmitting a one symbol secret $m \in \mathbb{F}_q$.

procedure ROUND 1: BOB TRANSMITS
    Bob samples $\mathbf{c}_1, \ldots, \mathbf{c}_{t+1} \in \mathcal{C}$ independently and uniformly at random.
    For $j = 1, \ldots, t+1$, Bob transmits the i-th coordinate of $\mathbf{c}_j$ through the i-th channel.
end procedure
procedure ROUND 2: ALICE TRANSMITS
    For $j = 1, \ldots, t+1$, Alice receives the vectors $\tilde{\mathbf{c}}_j$ where $d(\mathbf{c}_j, \tilde{\mathbf{c}}_j) \leq t$.
    For $j = 1, \ldots, t+1$, Alice computes $\mathbf{s}_j = \mathbf{H}\tilde{\mathbf{c}}_j \in \mathbb{F}_q^t$.
    Alice finds a coordinate $p \in [t+1]$ such that $\mathbf{s}_p \in \text{span}\{\mathbf{s}_j : j \neq p\}$.
    Alice finds $\lambda_j \in \mathbb{F}_q$ for $j \in [t+1] \setminus \{p\}$ such that $\mathbf{s}_p = \sum_{j \neq p} \lambda_j \mathbf{s}_j$.
    $\bar{\mathbf{c}} \leftarrow \tilde{\mathbf{c}}_p - \sum_{j \neq p} \lambda_j \tilde{\mathbf{c}}_j$
    Alice broadcasts p, $(\lambda_j : j \neq p)$ and the symbol $m' \leftarrow m + \langle \mathbf{h}, \bar{\mathbf{c}} \rangle$.
end procedure
procedure OUTPUT PHASE
    Bob receives p, $(\lambda_j : j \neq p)$ and the symbol m'.
    $\mathbf{c}' \leftarrow \mathbf{c}_p - \sum_{j \neq p} \lambda_j \mathbf{c}_j$
    return $m' - \langle \mathbf{h}, \mathbf{c}' \rangle$.
end procedure

B Algorithm 2

Algorithm 2 A protocol for transmitting an ℓ-symbol secret $(m_1, \ldots, m_\ell) \in \mathbb{F}_q^\ell$, which achieves transmission rate $(4 + o_{\ell\to\infty}(1))n$.

procedure ROUND 1: BOB TRANSMITS
  Bob samples $\mathbf{c}_1, \ldots, \mathbf{c}_{t+\ell} \in \mathcal{C}$ independently and uniformly at random.
  For $j = 1, \ldots, t+\ell$, Bob transmits the i-th symbol of $\mathbf{c}_j$ through the i-th channel.
end procedure
procedure ROUND 2: ALICE TRANSMITS
  For $j = 1, \ldots, t+\ell$, Alice receives the vectors $\tilde{\mathbf{c}}_j$ where $d(\mathbf{c}_j, \tilde{\mathbf{c}}_j) \le t$.
  For $j = 1, \ldots, t+\ell$, Alice computes $\mathbf{s}_j = \mathbf{H}\tilde{\mathbf{c}}_j \in \mathbb{F}_q^t$.
  Alice computes a pseudobasis for $\tilde{\mathbf{c}}_1, \ldots, \tilde{\mathbf{c}}_{t+\ell}$. Let $S \subseteq [t+\ell]$ index the elements of the pseudobasis.
  $r \leftarrow |S|$ and $r' \leftarrow \min\{r, \lfloor t/3 \rfloor\}$.
  Let $S' \subseteq S$ denote a subset of size r'.
  Let $\mathbf{y} \leftarrow (\tilde{\mathbf{c}}_j : j \in S')$; write $\mathbf{y} = \sum_{j\in S} \lambda_j \tilde{\mathbf{c}}_j$. ▷ Of course, for $j \in S \setminus S'$, we may put $\lambda_j = 0$.
  Let $T \leftarrow \{p_1, \ldots, p_\ell\}$ denote the ℓ smallest elements of $[t+\ell] \setminus S$.
  For $i \in [\ell]$, choose coefficients $\lambda_{ij} \in \mathbb{F}_q$ such that $\mathbf{s}_{p_i} = \sum_{j\in S} \lambda_{ij}\mathbf{s}_j$, and define $\bar{\mathbf{c}}_{p_i} \leftarrow \tilde{\mathbf{c}}_{p_i} - \sum_{j\in S} \lambda_{ij}\tilde{\mathbf{c}}_j$.
  Alice broadcasts the information $(S, (\lambda_j : j \in S), \mathbf{y})$.
  For each $i \in [\ell]$, Alice r'-broadcasts the data $(\lambda_{ij} : j \in S)$ and $m_i' \leftarrow m_i + \langle \mathbf{h}, \bar{\mathbf{c}}_{p_i} \rangle$.
end procedure
procedure OUTPUT PHASE
  Bob recovers $(S, (\lambda_j : j \in S), \mathbf{y})$ and defines $\mathbf{z} \leftarrow \sum_{j\in S} \lambda_j \mathbf{c}_j$. He also lets $T = \{p_1, \ldots, p_\ell\}$ denote the ℓ smallest elements of $[t+\ell] \setminus S$.
  Bob ignores the channels in the set $\mathrm{supp}(\mathbf{y} - \mathbf{z})$, a set of cardinality at least r'.
  For each $i \in [\ell]$, Bob recovers the information $(\lambda_{ij} : j \in S)$ and m_i', defines $\mathbf{c}'_{p_i} \leftarrow \mathbf{c}_{p_i} - \sum_{j\in S} \lambda_{ij}\mathbf{c}_j$, and then defines $m_i \leftarrow m_i' - \langle \mathbf{h}, \mathbf{c}'_{p_i} \rangle$.
  return $(m_1, \ldots, m_\ell)$.
end procedure

C Algorithm 3

Algorithm 3 Our final protocol for transmitting an ℓ-symbol secret $(m_1, \ldots, m_\ell) \in \mathbb{F}_q^\ell$, which achieves transmission rate $(2 + o_{\ell\to\infty}(1))n$. We just indicate what needs to be changed from Algorithm 2 when $r > r' = \min\{r, \lfloor t/3 \rfloor\}$.

procedure ROUND 1: BOB TRANSMITS
  Bob performs lines 2-3 from Algorithm 2.
end procedure
procedure ROUND 2: ALICE TRANSMITS
  Alice performs lines 6-14 from Algorithm 2.
  if $r = r'$ **then**
    Alice performs Appendix B from Algorithm 2.
  else
    Alice r'-broadcasts $\tilde{\mathbf{c}}_j$ for each $j \in S$.
    For each $i \in [\ell]$, Alice r-broadcasts the data $(\lambda_{ij} : j \in S)$ and $\langle \mathbf{h}, \bar{\mathbf{c}}_{p_i} \rangle + m_i$.
  end if
end procedure
procedure OUTPUT PHASE
  Bob performs lines 18-19 from Algorithm 2.
  Let $r \leftarrow |S|$.
  if $r \le t/3$ **then** Bob performs line 20
  else
    Bob recovers $\tilde{\mathbf{c}}_j$ for each $j \in S$.
    Bob ignores the channels in the set $\bigcup_{j\in S} \mathrm{supp}(\tilde{\mathbf{c}}_j - \mathbf{c}_j)$, which has cardinality at least r.
    For each $i \in [\ell]$, Bob recovers the information $(\lambda_{ij} : j \in S)$ and m'_i, defines $\mathbf{c}'_{p_i} \leftarrow \mathbf{c}_{p_i} - \sum_{j\in S} \lambda_{ij}\mathbf{c}_j$, and then defines $m_i \leftarrow m'_i - \langle \mathbf{h}, \mathbf{c}'_{p_i} \rangle$.
  end if
  return $(m_1, \ldots, m_\ell)$.
end procedure

D Procedure for Finding a Vector Far from Code

In this section, we present our algorithm for finding a vector that is far from the code.

▶ **Lemma 14.** *Let $\mathbf{y}_1, \ldots, \mathbf{y}_r$ have linearly independent syndromes and assume $r \le \frac{t}{3}$. Then the vector $\mathbf{y}$ returned by Algorithm 4 has distance at least r from $\mathcal{C}$.*

Proof. By assumption, we have that the syndromes $\mathbf{s}_i = \mathbf{H}\mathbf{y}_i \in \mathbb{F}_q^t$ for $i = 1, \ldots, r$ are linearly independent. We claim that the vectors $\mathbf{e}_1, \ldots, \mathbf{e}_r \in \mathbb{F}_q^n$ are linearly independent. Suppose $\lambda_1, \ldots, \lambda_r \in \mathbb{F}_q$ are such that $\sum_{i=1}^r \lambda_i \mathbf{e}_i = \mathbf{0}$. Then

$$\mathbf{0} = \sum_{i=1}^r \lambda_i \mathbf{H}\mathbf{e}_i = \sum_{i=1}^r \lambda_i \mathbf{H}(\mathbf{y}_i - \mathbf{x}_i) = \sum_{i=1}^r \lambda_i \mathbf{s}_i \ .$$

As $\mathbf{s}_1, \ldots, \mathbf{s}_r$ are linearly independent, this implies $\lambda_1 = \cdots = \lambda_r = 0$, as desired.

Algorithm 4 A procedure for Alice to find a vector whose distance from $\mathcal{C}$ is at least r for $r \leq \frac{t}{3}$.

procedure MANY-ERRORS($\mathbf{y}_1, \ldots, \mathbf{y}_r$)
  For $i = 1, \ldots, r$, let $\mathbf{x}_i \in \mathcal{C}$ denote the codeword agreeing with $\mathbf{y}_i$ on the last $t+1$ coordinates. ▷ This is possible, as every subset of $t+1$ coordinates forms an information set for $\mathcal{C}$.
  For $i = 1, \ldots, r$, $\mathbf{e}_i \leftarrow \mathbf{y}_i - \mathbf{x}_i$.
  Let M denote the matrix in $\mathbb{F}_q^{r \times n}$ whose rows are $\mathbf{e}_1, \ldots, \mathbf{e}_r$.
  Using Gaussian elimination, put M in reduced row echelon form; let $\mathbf{e}_1^*, \ldots, \mathbf{e}_r^*$ denote the rows.
  if $\exists i \in [r]$ s.t. $\mathrm{wt}(\mathbf{e}_i^*) \geq r$ **then** $\mathbf{e} \leftarrow \mathbf{e}_i^*$
  else
    for $j = 2, 3, \ldots, r$ **do**
      if $\mathrm{wt}\left(\sum_{i=1}^{j} \mathbf{e}_i^*\right) \geq r$ **then** $\mathbf{e} \leftarrow \sum_{i=1}^{j} \mathbf{e}_i^*$
      end if
    end for
  end if
  Choose $\lambda_1, \ldots, \lambda_r \in \mathbb{F}_q$ such that $\mathbf{e} = \sum_{i=1}^{r} \lambda_i \mathbf{e}_i$.
  $\mathbf{y} \leftarrow \sum_{i=1}^{r} \lambda_i \mathbf{y}_i$
  return $\mathbf{y}$
end procedure

Now, we note that if $\mathbf{e} = \sum_{i=1}^{r} \lambda_i \mathbf{e}_i$ is found such that $d(\mathbf{e}, \mathcal{C}) \geq r$, then it also follows that $\mathbf{y} = \sum_{i=1}^{r} \lambda_i \mathbf{y}_i$ satisfies $d(\mathbf{y}, \mathcal{C}) \geq r$. Indeed,

$$d(\mathbf{y}, \mathcal{C}) = d\left(\mathbf{e} + \sum_{i=1}^{r} \lambda_i \mathbf{x}_i, \mathcal{C}\right) = d\left(\mathbf{e}, \mathcal{C} + \sum_{i=1}^{r} \lambda_i \mathbf{x}_i\right) = d(\mathbf{e}, \mathcal{C}) \geq r$$

as $\sum_{i=1}^{r} \lambda_i \mathbf{x}_i \in \mathcal{C}$.

Now, for $\mathbf{e} \in \mathrm{span}\{\mathbf{e}_1, \ldots, \mathbf{e}_r\}$, to ensure $d(\mathbf{e}, \mathcal{C}) \geq r$, note that it is sufficient to show that $r \leq \mathrm{wt}(\mathbf{e}) \leq t - r + 1$. Indeed, as we have $d(\mathbf{0}, \mathbf{e}) = \mathrm{wt}(\mathbf{e}) \geq r$, it suffices to verify that for all nonzero codewords $\mathbf{c} \in \mathcal{C} \setminus \{\mathbf{0}\}$ we have $d(\mathbf{e}, \mathbf{c}) \geq r$. And indeed, this follows as

$$t + 1 \leq d(\mathbf{0}, \mathbf{c}) \leq d(\mathbf{0}, \mathbf{e}) + d(\mathbf{e}, \mathbf{c}) \leq t - r + 1 + d(\mathbf{e}, \mathbf{c}) \ ,$$

and so $d(\mathbf{e}, \mathbf{c}) \geq r$.

Hence, we now show how the algorithm finds a vector $\mathbf{e} \in \mathrm{span}\{\mathbf{e}_1, \ldots, \mathbf{e}_r\}$ which satisfies $r \leq \mathrm{wt}(\mathbf{e}) \leq t - r + 1$. Consider the matrix

$$M = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \vdots \\ \mathbf{e}_r \end{bmatrix} \in \mathbb{F}_q^{r \times n}$$

whose rows are given by vectors $\mathbf{e}_1, \ldots, \mathbf{e}_r$.

Consider putting the matrix M into reduced row echelon form; denote the resulting rows $\mathbf{e}_1^*, \ldots, \mathbf{e}_r^*$. By the definition of row operations, $\mathrm{span}\{\mathbf{e}_1, \ldots, \mathbf{e}_r\} = \mathrm{span}\{\mathbf{e}_1^*, \ldots, \mathbf{e}_r^*\}$, so it suffices to find a vector $\mathbf{e}^* \in \mathrm{span}\{\mathbf{e}_1^*, \ldots, \mathbf{e}_r^*\}$ satisfying $r \leq \mathrm{wt}(\mathbf{e}^*) \leq t - r + 1$.

As the vectors $\mathbf{e}_1, \ldots, \mathbf{e}_r$ are linearly independent, there is a set $R \subseteq [n]$ of r pivot points: that is, we have indices $1 \leq j_1 < j_2 < \cdots < j_r \leq n$ such that for each $i, p \in [r]$:

$$(\mathbf{e}_i)_{j_p} = \begin{cases} 1 & \text{if } i = p \\ 0 & \text{otherwise} \end{cases} .$$

Therefore, for each $i \in [r]$ we have $\mathrm{supp}(\mathbf{e}_i^*) \subseteq ([t] \setminus R) \cup \{j_i\}$, so $\mathrm{wt}(\mathbf{e}_i^*) \leq t - r + 1$. Thus, if we are in the case that for some $i \in [r]$ we have $r \leq \mathrm{wt}(\mathbf{e}_i^*)$, we can just return the vector $\mathbf{e}_i^*$.

Assume now that for each i we have $\mathrm{wt}(\mathbf{e}_i^*) < r$. Consider the sequence of vectors $\sum_{i=1}^{j} \mathbf{e}_i^*$ for $j = 2, \ldots, r$. Note that $\mathrm{supp}\left(\sum_{i=1}^{r} \mathbf{e}_i^*\right) \supseteq R$, so $\mathrm{wt}\left(\sum_{i=1}^{r} \mathbf{e}_i^*\right) \geq |R| = r$. Hence, there exists $2 \leq j \leq r$ such that:

- $\mathrm{wt}\left(\sum_{i=1}^{j} \mathbf{e}_i^*\right) \geq r$;
- for all $1 \leq j' \leq j$, $\mathrm{wt}\left(\sum_{i=1}^{j'} \mathbf{e}_i^*\right) < r$.

We claim that $\mathbf{e}^* := \sum_{i=1}^{j} \mathbf{e}_i^*$ satisfies $r \leq \mathrm{wt}(\mathbf{e}^*) \leq t + 1 - r$. The lower bound is obvious by the definition of j. For the upper bound, we note that

$$\mathrm{wt}\left(\sum_{i=1}^{j} \mathbf{e}_i^*\right) \leq \mathrm{wt}\left(\sum_{i=1}^{j-1} \mathbf{e}_i^*\right) + \mathrm{wt}(\mathbf{e}_j^*) < r + r \leq t + 1 - r \ ,$$

where the upper bound on the weight of $\sum_{i=1}^{j-1} \mathbf{e}_i^*$ is again by the definition of j and the upper bound on $\mathrm{wt}(\mathbf{e}_j^*)$ follows from our earlier assumption. That $2r \leq t + 1 - r$ follows from $r \leq t/3$. ◀

A Lower Bound on the Share Size in Evolving Secret Sharing

Noam Mazor ✉
The Blavatnik School of Computer Science, Tel Aviv University, Israel

Abstract

Secret sharing schemes allow sharing a secret between a set of parties in a way that ensures that only authorized subsets of the parties learn the secret. *Evolving secret sharing* schemes (Komargodski, Naor, and Yogev [TCC '16]) allow achieving this end in a scenario where the parties arrive in an online fashion, and there is no a-priory bound on the number of parties.

An important complexity measure of a secret sharing scheme is the share size, which is the maximum number of bits that a party may receive as a share. While there has been a significant progress in recent years, the best constructions for both secret sharing and evolving secret sharing schemes have a share size that is exponential in the number of parties. On the other hand, the best lower bound, by Csirmaz [Eurocrypt '95], is sub-linear.

In this work, we give a tight lower bound on the share size of evolving secret sharing schemes. Specifically, we show that the sub-linear lower bound of Csirmaz implies an exponential lower bound on evolving secret sharing.

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

Keywords and phrases Secret sharing, Evolving secret sharing

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.2

Related Version *Full Version*: `https://eprint.iacr.org/2023/129.pdf`

Funding Research supported by Israel Science Foundation grant 666/19.

Acknowledgements We thank Iftach Haitner and Ilan Komargodski for useful discussions.

1 Introduction

Secret sharing is a fundamental concept in cryptography, that allows a dealer to distribute a secret among a set of parties in a way that ensures that only *authorized* subsets of parties learn the secret. Such schemes are used in secure multi-party computation, amplification schemes for cryptographic primitives, Byzantine agreement protocols, and more (see [5]). *Evolving secret sharing* (Komargodski, Naor, and Yogev [10]) is a variant of secret sharing, that can be used in evolving systems, for which there is no a-priory bound on the number of parties. In such schemes, the dealer distributes the secret to an infinite number of parties in an online fashion: the parties arrive one by one, and each party receives its share of the secret as it arrives. The correctness guarantee promises that by the time the n-th party receives their share, all the authorized subsets among the first n parties can reconstruct the secret. Such a scheme is *adaptive* if the dealer does not need to know the entire access structure to give a share to a party. Rather, it is sufficient to know the list of authorized sets containing only parties that already arrived.

The main complexity measure of a secret sharing scheme is its share size: the maximal number of bits a party might receive as a share. While there have been significant advancements in the area in recent years ([13, 12, 1, 2]), the best known constructions for (classical) secret sharing have exponential share size in the number of parties (Applebaum and Nir [4]). For the harder task of evolving secret sharing, the best construction for arbitrary access structure gives the i-th party share of size 2^{i-1} ([10]).


© Noam Mazor;
licensed under Creative Commons License CC-BY 4.0

Somewhat surprisingly, we do not know if exponential share size is the best possible, or even if the share size must be *super linear* in the number of parties. Indeed, the best known lower bound on (classical) secret sharing is due to Csirmaz [8], which showed a specific access structure for which every scheme must give some party a share of size $\Omega(n/\log n)$. Thus, the optimal share size for arbitrary access structures is an important open question. Prior to this paper, this question was open also for the case of evolving secret sharing.

1.1 Our Result

In this work, we resolve the above question for the case of evolving secret sharing. We show that the linear lower bound of Csirmaz [8] implies a tight exponential lower bound on evolving secret sharing. This is stated in the following two theorems. The first is for adaptive evolving secret sharing schemes.

▶ **Theorem 1** (Lower bound for adaptive schemes, informal)**.** *There exists an access structure $\mathcal{A}$ such that for every adaptive evolving secret sharing scheme and for every n, the total share size of the first n parties in $\mathcal{A}$ is at least 2^n. In particular, the share size of the i-th party is at least 2^{i-1} for infinitely many i's.*

As stated before, this lower bound is tight with the scheme of [10] which gives the i-th party share of size 2^{i-1}. Interestingly, the access structure for which we prove this lower bound does not contain a single authorized set. We also prove the following slightly weaker lower bound, for a larger class of schemes, namely, non-adaptive schemes.

▶ **Theorem 2** (Lower bound for non-adaptive schemes, informal)**.** *There exists an access structure $\mathcal{A}$ such that the following holds. For every evolving secret sharing scheme for $\mathcal{A}$ and for every n, the total share size of the first n parties is at least $2^{n-o(n)}$. Moreover, the share size of the i-th party is at least $2^{i-o(i)}$ for infinitely many i's.*

The formal bound we prove (Theorem 14) is somewhat stronger, as we can choose the $o(n)$ term to be any super-constant. For example, Theorem 14 implies that the total share size of the first n parties is at least $2^{n-\log n}$. The proof of both theorems follows from an observation on [8]'s lower bound. In his work, Csirmaz [8] shows that in some access structure over n parties, there is a specific set of $t = \log n$ parties that must hold together at least n bits. We observe that if these t parties are the first to arrive, by [8]'s lower bound they must hold exponential (in t) share size. See more details in Section 3.[1]

1.2 Additional Related Work

Lower bounds on secret sharing schemes

Besides the aforementioned lower bound of [8], Csirmaz [7] showed an access structure for which, the *total share size* must be quadratic. The construction is simply duplicating the parties with large shares in [8]'s construction. Csirmaz [8] also shows that a better lower bound on the share size cannot be proven using Shannon information inequalities. Beimel and Orlov [6] showed the same result for a larger set of information inequalities. Recently, Applebaum, Beimel, Nir, Peter, and Pitassi [3] showed a connection between the known

[1] We remark that, as in [8], both of our bounds generalize to the *information-ratio* of the scheme. That is, the ratio between the total share size of the first n parties to the length of the secret must be exponential in n.

constructions of secret sharing and monotone real circuits, and used this connection to give a lower bound on a family of constructions. For evolving schemes, [10] gave a tight lower bound for the special case of the 2-threshold access structure.

Constructions of evolving secret sharing schemes

Following Komargodski et al. [10], Paskin-Cherniavsky [14] showed a more efficient construction for some classes of access structures. In this scheme, the dealer needs to know the access structure in advance. More efficient schemes are known for specific types of access structures ([9, 10, 11]).

Paper Organization

Basic definitions and notations are given in Section 2, and the proofs of the lower bounds are given in Section 3.

2 Preliminaries

2.1 Notations

All logarithms are taken in base 2. We use calligraphic letters to denote sets and distributions, uppercase for random variables, and lowercase for values and functions. We use $[n]$ to denote the set $\{1,\ldots,n\}$. Given a vector $v \in \Sigma^n$, let v_i denote its i-th entry, let $v_{<i} = (v_1,\ldots,v_{i-1})$ and $v_{\leq i} = (v_1,\ldots,v_i)$. Similarly, for a set $\mathcal{I} \subseteq [n]$, let $v_{\mathcal{I}}$ be the ordered sequence $(v_i)_{i\in\mathcal{I}}$.

When unambiguous, we will naturally view a random variable as its marginal distribution. For a (discrete) distribution $\mathcal{D}$, let $x \leftarrow \mathcal{D}$ denote that x was sampled according to $\mathcal{D}$. Let $\mathrm{Supp}(\mathcal{D}) = \{p\colon \Pr_{\mathcal{D}}[p] > 0\}$, and define $|\mathcal{D}| = \log(|\mathrm{Supp}(\mathcal{D})|)$.

2.1.1 Entropy and Mutual Information

The *Shannon entropy* of a distribution $\mathcal{P}$ is defined by $H(\mathcal{P}) = \sum_{p\in\mathrm{Supp}(\mathcal{P})} \Pr_{\mathcal{P}}[p]\cdot\log\frac{1}{\Pr_{\mathcal{P}}[p]}$. The conditional entropy of a random variable A given B, is defined as $H(A \mid B) = \mathrm{E}_{b\leftarrow B}[H(A|_{B=b})]$. The mutual information between two random variables A and B is defined by

$$I(A;B) = H(A) - H(A \mid B) = H(B) - H(B \mid A)$$

and the conditional mutual information given a random variable C is defined similarly

$$I(A;B \mid C) = H(A \mid C) - H(A \mid B,C).$$

We will use the following well known facts:

▶ **Fact 3** (Chain rule for mutual information). *For two random variables A and $B = (B_1,\ldots,B_n)$, it holds that $I(A;B) = \sum_{i=1}^{n} I(A;B_i \mid B_{<i})$.*

▶ **Fact 4** (Upper bound on mutual information). *For two random variables A and B, it holds that $I(A;B) \leq |A|$.*

2.2 Secret Sharing Schemes

We now formally define secret sharing schemes. Let $\mathcal{P}$ be a set of parties. An *access structure* is a monotone collection of subsets of $\mathcal{P}$.

▶ **Definition 5** (Access structure). *A collection of sets $\mathcal{A} \subseteq 2^{\mathcal{P}}$ is an* access structure *if it is monotone: for every set $\mathcal{B} \in \mathcal{A}$ and for every $\mathcal{B}'$ such that $\mathcal{B} \subseteq \mathcal{B}' \subseteq \mathcal{P}$, it holds that $\mathcal{B}' \in \mathcal{A}$. A set $\mathcal{B}$ is* authorized *if $\mathcal{B} \in \mathcal{A}$, and* unauthorized *otherwise.*

An access structure can be defined by a set of minimal authorized sets. Given a (non-monotone) set $\mathcal{M}$ of subsets of parties, the induced access structure $\mathcal{A}_{\mathcal{M}}$ is received by adding to $\mathcal{A}_{\mathcal{M}}$ all the subsets containing a set in $\mathcal{M}$. That is, $\mathcal{A}_{\mathcal{M}} := \{\mathcal{B} \subseteq \mathcal{P} \colon \exists \mathcal{C} \in \mathcal{M} \text{ s.t. } \mathcal{C} \subseteq \mathcal{B}\}$. We are now ready to define secret sharing schemes.

▶ **Definition 6** (Secret sharing scheme). *A* secret sharing scheme *for an access structure $\mathcal{A}$ is a pair of algorithms* $(\mathsf{SHARE}, \mathsf{RECON})$ *such that* SHARE *is a randomized algorithm and the following holds:*

1. *Given a secret $s \in \{0,1\}$, $\mathsf{SHARE}(s)$ returns shares $\pi = \{\pi_p\}_{p \in \mathcal{P}}$. π_p is called the share of party p.*
2. *Correctness: For every secret $s \in \{0,1\}$, $\pi \leftarrow \mathsf{SHARE}(s)$ and an authorized set $\mathcal{B} \in \mathcal{A}$, $\mathsf{RECON}(\mathcal{B}, \pi_{\mathcal{B}}) = s$.*
3. *Perfect Privacy: For every unauthorized set $\mathcal{B} \notin \mathcal{A}$, it holds that*

$$\mathsf{SHARE}(0)_{\mathcal{B}} \equiv \mathsf{SHARE}(1)_{\mathcal{B}}.$$

2.3 Evolving Secret Sharing

We now formally define evolving secret sharing schemes, introduces by Komargodski et al. [10].

▶ **Definition 7** (Restriction). *Given an access structure $\mathcal{A}$ over $\mathcal{P}$, and a subset of parties $\mathcal{P}' \subseteq \mathcal{P}$, let $\mathcal{A}|_{\mathcal{P}'} := \{\mathcal{B} \in \mathcal{A} \colon \mathcal{B} \subseteq \mathcal{P}'\}$.*

[10] showed that $\mathcal{A}|_{\mathcal{P}'}$ is an access structure for every $\mathcal{A}$ and $\mathcal{P}'$.

▶ **Definition 8** (Evolving access structure). *Let $\mathcal{P} = \mathbb{N}$ be an infinite set of parties. An* evolving access structure over $\mathcal{P}$ *is a set of access structures $\{\mathcal{A}_n\}_{n \in \mathbb{N}}$ such that for every n, $\mathcal{A}_n$ is an access structure over $[n]$ and $\mathcal{A}_{n+1}|_{[n]} = \mathcal{A}_n$.*

For an evolving access structure $\mathcal{A}$ and a finite set of parties $\mathcal{I} \subseteq \mathcal{P}$, we use $\mathcal{A}|_{\mathcal{I}}$ to denote the access structure $\mathcal{A}_n|_{\mathcal{I}}$ for some n with $\mathcal{I} \subseteq [n]$. Notice that the set $\mathcal{A}_n|_{\mathcal{I}}$ is independent from the choice of such n (That is, $\mathcal{A}_n|_{\mathcal{I}} = \mathcal{A}_{n'}|_{\mathcal{I}}$ for every n and n' such that $\mathcal{I} \subseteq [n]$ and $\mathcal{I} \subseteq [n']$).

▶ **Definition 9** (Evolving secret sharing scheme). *An* evolving secret sharing scheme *for an evolving access structure $\{\mathcal{A}_n\}_{n \in N}$ is a pair of algorithms* $(\mathsf{SHARE}, \mathsf{RECON})$ *such that the following holds for every n:*

1. *Given a secret $s \in \{0,1\}$ and sequence of shares $\pi_1, \ldots, \pi_{n-1}$, $\mathsf{SHARE}(s, \pi_1, \ldots, \pi_{n-1})$ returns a share π_n for party n. Denote by $\Pi^s = (\Pi^s_1, \Pi^s_2, \ldots)$ the distribution of the shares of the parties on secret s. That is, $\Pi^s_i = \mathsf{SHARE}(s, \Pi^s_1, \ldots, \Pi^s_{i-1})$.*
2. *Correctness: For every secret $s \in \{0,1\}$, shares $\pi = (\pi_1, \ldots, \pi_n) \leftarrow \Pi^s_{\leq n}$ and an authorized set $\mathcal{B} \in \mathcal{A}_n$, $\mathsf{RECON}(\mathcal{B}, \pi_{\mathcal{B}}) = s$.*
3. *Perfect Privacy: For every set $\mathcal{B} \subseteq [n]$ of parties with $\mathcal{B} \notin \mathcal{A}_n$, it holds that $\Pi^0_{\mathcal{B}} \equiv \Pi^1_{\mathcal{B}}$.*

Note that for every set $\mathcal{B} \subseteq [n]$ of parties with $\mathcal{B} \notin \mathcal{A}_n$, it holds that $\mathcal{B} \notin \mathcal{A}_k$ for every $k \in \mathbb{N}$.

An adaptive evolving secret sharing scheme is a secret sharing scheme that doesn't know the access structure in advance. In this definition, the algorithms SHARE and RECON get a description of the access structure.

▶ **Definition 10** (Adaptive evolving secret sharing scheme)**.** *An* adaptive evolving secret sharing scheme *is a pair of algorithms* $(\mathsf{SHARE}, \mathsf{RECON})$ *such that the following hold for every evolving access structure* $\{\mathcal{A}_n\}_{n\in N}$ *and for every* n*:*

1. *Given a secret* $s \in \{0,1\}$*,* $\mathcal{A}_n$ *and sequence of shares* $\pi_1, \ldots, \pi_{n-1}$*,* $\mathsf{SHARE}(s, \mathcal{A}_n, \pi_1, \ldots, \pi_{n-1})$ *returns a share* π_n *for party* n*. Denote by* $\Pi^s = (\Pi^s_1, \Pi^s_2, \ldots)$ *the distribution of the shares of the first* n *parties on secret* s*. That is,*

$$\Pi^s_i = \mathsf{SHARE}(s, \mathcal{A}_i, \Pi^s_1, \ldots, \Pi^s_{i-1}).$$

2. *Correctness: For every secret* $s \in \{0,1\}$*, shares* $\pi = (\pi_1, \ldots, \pi_n) \leftarrow \Pi^s_{\leq n}$ *and an authorized set* $\mathcal{B} \in \mathcal{A}_n$*,* $\mathsf{RECON}(\mathcal{B}, \mathcal{A}_n, \pi_B) = s$*.*
3. *Perfect Privacy: For every set* $\mathcal{B} \subseteq [n]$ *of parties with* $\mathcal{B} \notin \mathcal{A}_n$*, it holds that* $\Pi^0_{\mathcal{B}} \equiv \Pi^1_{\mathcal{B}}$*.*

We now formally define the share size of a set of parties.

▶ **Definition 11** (Share size)**.** *For an evolving access structure* $\mathcal{A} = \{\mathcal{A}_n\}_{n\in N}$*, an adaptive scheme* $(\mathsf{SHARE}, \mathsf{RECON})$*, and* $S \leftarrow \{0,1\}$*, let* $\Pi_i := \mathsf{SHARE}(S, \mathcal{A}_i, \Pi_1, \ldots, \Pi_{i-1})$ *for every* $i \in \mathbb{N}$*. Then the* share size for $\mathcal{A}$ *of a party* $p \in \mathbb{N}$ *is simply* $|\Pi_p|$*. The* total share size *of a set of parties* $\mathcal{B}$ *is* $|\Pi_{\mathcal{B}}| \leq \sum_{p\in\mathcal{B}} |\Pi_p|$*.* [2]

We define share size and total share size for non-adaptive/non-evolving secret sharing schemes similarly.

2.4 Csirmaz's lower bound

Csirmaz [8] proved a lower bound on the share size of a (classic) secret sharing scheme for a specific access structure. We exploit the properties of this access structure in our proof. The following is the formal statement we need.

▶ **Theorem 12** ([8])**.** *For every* $t \in \mathbb{N}$*, there exists an access structure* $\mathcal{Z}_t$ *over* $t + 2^t$ *parties, such that the following holds: The set of players is composed of two disjoint sets,* $\mathcal{B}$ *and* $\mathcal{C}$*, such that* $|\mathcal{C}| = t$*,* $|\mathcal{B}| = 2^t$*, and:*

1. $\mathcal{C}$ *is an unauthorized set, and,*
2. *the total share size of players in* $\mathcal{C}$ *is at least* $2^t - 1$*.*

For completeness, we give here the proof.

Proof. Fix $t \in \mathbb{N}$ and let $n = 2^t$. We start with describing the access structure $\mathcal{Z}_t$. Let $\mathcal{B} = \{P_1, \ldots, P_n\}$ be a set of n parties, and let $\mathcal{C}$ be a disjoint set of parties of size t. Let $\mathcal{C}_1, \ldots, \mathcal{C}_n$ be an ordering of all the subsets of $\mathcal{C}$, such that for every $i < j$ it holds that $\mathcal{C}_i \not\subseteq \mathcal{C}_j$.[3] Define the set of minimal authorized sets of $\mathcal{Z}_t$ to be the set

$$\mathcal{M} = \{\mathcal{C}_i \cup \{P_1, \ldots, P_i\} : i \in [n]\},$$

[2] Recall that $|\Pi_b| := \log(|\mathrm{Supp}(\Pi_p)|)$ is a lower bound on the maximal representation size of a sample from Π_p.

[3] For example, order the sets according to there size in reverse order, with arbitrary order between sets of equal size.

and let $\mathcal{Z}_t = \mathcal{A}_{\mathcal{M}}$ be the induced access structure. Item 1 holds by construction. Moreover, by the definition of $\mathcal{C}_1, \ldots, \mathcal{C}_n$ and $\mathcal{M}$, for every i the set $\mathcal{C}_i \cup \{P_1, \ldots, P_{i-1}\}$ is unauthorized. We now use this to prove the lower bound on the share size. Let $S \leftarrow \{0,1\}$ be a uniformly chosen secret, and Π be a random sharing of S. We want to lower bound the size of $\Pi_{\mathcal{C}}$. It holds that,

$$\begin{aligned}
|\Pi_{\mathcal{C}}| + |S| &\geq I(\Pi_{\mathcal{C}}, S; \Pi_{\mathcal{B}}) \\
&= \sum_i I(\Pi_{\mathcal{C}}, S; \Pi_{P_i} \mid \Pi_{P_{<i}}) \\
&\leq \sum_i I(\Pi_{\mathcal{C}_i}, S; \Pi_{P_i} \mid \Pi_{P_{<i}}) \\
&\leq \sum_i I(S; \Pi_{P_i} \mid \Pi_{\mathcal{C}_i}, \Pi_{P_{<i}}) \\
&= \sum_i H(S \mid \Pi_{\mathcal{C}_i}, \Pi_{P_{<i}}) - H(S \mid \Pi_{P_i}, \Pi_{\mathcal{C}_i}, \Pi_{P_{<i}}) \\
&= \sum_i 1 - 0 \\
&= n
\end{aligned}$$

where the first inequality holds by Fact 4. The first equality, the second inequality, and the third inequality hold by the chain rule of mutual information. The last inequality holds since $\mathcal{C}_i \cup P_{<i}$ is an unauthorized set, but $\mathcal{C}_i \cup P_{\leq i}$ is authorized. Item 2 now follows from the above since $n = 2^t$ and $|S| = 1$. ◀

3 The Lower Bound on the Share Size

In this section, we formally prove our lower bound. We start with a lower bound on adaptive evolving secret sharing, and then show how to generalize the bound to hold for non-adaptive schemes.

3.1 The Adaptive Case

We start by formally stating our main result.

▶ **Theorem 13.** *Let $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ be the access structure for which $\mathcal{A}_n = \emptyset$ for every n. Then for every adaptive evolving secret sharing scheme and every t, the total share size for $\mathcal{A}$ of the first t parties is at least $2^t - 1$. In particular, there are infinitely many parties i with share size at least $2^{i-1} - 1$.*

The proof of the lower bound is by showing that for every t, after the first t parties arrived, it is possible to add 2^t parties such that the resulting access structure will be Csirmaz's structure. Thus, by Csirmaz's lower bound, the t parties must hold long shares.

Proof. Let $(\mathsf{SHARE}, \mathsf{RECON})$ be an adaptive secret sharing scheme, and fix $t \in \mathbb{N}$. We start by defining an evolving access structure $\mathcal{A}'$, and bounding its share size. Later, we relate the share size of $\mathcal{A}$ and $\mathcal{A}'$.

Let $\mathcal{C} = [t]$, and let $n = 2^t$. Let $\mathcal{B} = \{P_1, \ldots, P_n\}$ for $P_i = i + t$. Define the evolving access structure $\mathcal{A}' = \{\mathcal{A}'_m\}_{m\in\mathbb{N}}$ as follows: for every $i \in [t]$, let $\mathcal{A}'_i = \mathcal{A}_i = \emptyset$. Let $\mathcal{A}'_{t+n} = \mathcal{Z}_t$ be the access structure over the set $\mathcal{B} \cup \mathcal{C}$ promised by Theorem 12. For every $j \in [n]$, define $\mathcal{A}'_{t+j} = \mathcal{A}'_{t+n}|_{[t+j]}$. Finally, for every $i > t + n$, let $\mathcal{A}'_i = \mathcal{A}'_{t+n}$. Notice that $\mathcal{A}'$ is indeed an evolving access structure as $\mathcal{A}'_{t+n}|_{[t]} = \mathcal{A}_t$.

Let $S \leftarrow \{0,1\}$ be an uniformly random secret, and let $\Pi = (\Pi_1, \ldots, \Pi_{t+n})$ be the distribution of the shares of the first $t+n$ parties on $\mathcal{A}$. That is, $\Pi_i = \mathsf{SHARE}(S, \mathcal{A}_i, \Pi_1, \ldots, \Pi_{i-1})$. Similarly, let $\Pi' = (\Pi'_1, \ldots, \Pi'_{t+n})$ be the distribution of the shares of the first $t+n$ parties on $\mathcal{A}'$ (using SHARE and the secret S).

Notice that by definition of evolving secret sharing scheme, the pair $(\widehat{\mathsf{SHARE}}, \mathsf{RECON})$ is a secret sharing scheme for the access structure $\mathcal{A}'_{t+n}$, for $\widehat{\mathsf{SHARE}}(s) := \Pi'|_{S=s}$. Thus, it must hold by Theorem 12 that $|\Pi'_{\mathcal{C}}| = \left|\Pi'_{\leq t}\right| \geq 2^t - 1$. However, since $\mathcal{A}'_i = \mathcal{A}_i$ for every $i \leq t$, it holds that $\Pi'_{\leq t} = \Pi_{\leq t}$. Therefore, $|\Pi_{\leq t}| \geq 2^t - 1$, and the first part of the theorem follows.

To see the second part, assume towards a contradiction that there is only a finite number of parties i for which the share size is at least $2^{i-1} - 1$, and let i^* be the maximal such i (or $i^* = 1$ if no such exists). Let ℓ be the total share size of the first i^* parties. Consider the $i^* + \ell$ first parties of $\mathcal{A}$. By the assumption, their total share size is at most

$$\ell + \sum_{j=i^*+1}^{i^*+\ell} (2^{j-1} - 1) = \sum_{j=i^*+1}^{i^*+\ell} 2^{j-1} < \sum_{j=1}^{i^*+\ell} 2^{j-1} = 2^{i^*+\ell} - 1.$$

On the other hand, by the first part of the theorem, the total share size of the first $i^* + \ell$ parties is at least $2^{i^*+\ell} - 1$ which is a contradiction to the above. ◀

3.2 The Non-Adaptive Case

We now prove our main result for non-adaptive schemes. We start with formally stating the result.

▶ **Theorem 14.** *For every function $f : \mathbb{N} \to \mathbb{N}$ with $f \in \omega(1)$, there exists an access structure $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathbb{N}}$ such that the following holds for any evolving secret sharing scheme for $\mathcal{A}$. For every t, the total share size of the first t parties is at least $2^{t-f(t)} - 1$. Moreover, there are infinitely many parties i with share size at least $2^{i-f(i)-1} - 1$.*

The proof of the above theorem is similar to the proof of Theorem 13. However, since the access structure is fixed, we cannot argue that the security and correctness hold if we change the access structure on parties that did not arrive yet. To overcome this, we need to embed inside $\mathcal{A}$ all the access structures $\mathcal{Z}_t$ for every value of $t \in \mathbb{N}$. Recall that Csirmaz's structure $\mathcal{Z}_t$ is over two sets of parties, $\mathcal{C}$ and $\mathcal{B}$, such that the set $\mathcal{C}$ is of size t and has total share size 2^t. To get the stated lower bound, we need to embed in $\mathcal{A}$ the structure $\mathcal{Z}_t$ in such a way that the parties that hold long shares (that is, the parties in the set $\mathcal{C}$) will arrive early enough. This is done by associating only a sparse fraction (determined by the function f) of the parties in $\mathcal{A}$ with the set $\mathcal{B}$.

Proof. Fix a function $f \in \omega(1)$. We start by describing the access structure $\mathcal{A}$. Assume without loss of generality that $f(0) = 0$ and $0 \leq f(n+1) - f(n) \leq 1/2$,[4] and for every n let x_n be a number such that $f(x_n) \geq n$ and $f(x_n - 1) < n$. Let $\mathcal{X} = \{x_1, x_2, \ldots\}$. We divide $\mathcal{X}$ into disjoint segments $\{\mathcal{I}_j\}_{j\in\mathbb{N}}$ as follows, such that the size of the j-th segment is 2^j. Namely, for every $j \in \mathbb{N}$ let $\mathcal{I}_j = \{x_{2^j}, \ldots, x_{2^{j+1}-1}\}$. For every $t \in \mathbb{N}$, let $[t]_{\overline{\mathcal{X}}} = [t] \setminus \mathcal{X}$, and let $t' = \left|[t]_{\overline{\mathcal{X}}}\right|$ be the size of $[t]_{\overline{\mathcal{X}}}$. Observe that $t' \geq t - f(t)$.

[4] Otherwise, define $f'(n) = \min\left\{f'(n-1) + 1/2, \min_{n'>n}\left\{f(n')\right\}\right\}$. Clearly f' has the assumed property, and for every n, $f'(n) \leq f(n)$.

We next define the evolving access structure $\mathcal{A}$ such that $\mathcal{A}|_{[t]_{\overline{\mathcal{X}}} \cup \mathcal{I}_{t'}} = \mathcal{Z}_{t'}$ where $\mathcal{Z}_{t'}$ is the access structure promised by Theorem 12. Moreover, $[t]_{\overline{\mathcal{X}}}$ will match the set $\mathcal{C}$ in Theorem 12. This concludes the proof of the theorem similarly to the proof of Theorem 13, as it follows that the total share size of the parties in $[t]_{\overline{\mathcal{X}}}$ (and therefore also in $[t]$) is at least $2^{t'} - 1 \geq 2^{t - f(t)} - 1$.

To define $\mathcal{A}$ as stated above, for every $t' \in \mathbb{N}$ let $\mathcal{Z}_{t'}$ be the access structure promised by Theorem 12, over the sets of parties $\mathcal{C} = [t]_{\overline{\mathcal{X}}}$ and $\mathcal{B} = \mathcal{I}_{t'}$. For every $n \in \mathbb{N}$ define

$$\mathcal{A}_n := \bigcup_{t'=1}^{\infty} \{\mathcal{D} \in \mathcal{Z}_{t'} : \mathcal{D} \subseteq [n]\}.$$

By definition the sequence $\mathcal{A} = \{\mathcal{A}_n\}_{n \in \mathbb{N}}$ is an evolving access structure. Moreover, by construction it holds that for every t' and for every large enough n (with $f(n) > 2^{t'+1}$), it holds that $\mathcal{A}_n|_{[t]_{\overline{\mathcal{X}}} \cup \mathcal{I}_{t'}}$ is equal to $\mathcal{Z}_{t'}$, as stated above. Indeed, to make sure that we didn't add additional authorized subsets, observe that every authorized set of any structure $\mathcal{Z}_j$ for $j \neq t'$ contains at least one party from $\mathcal{I}_j$. Since $[t]_{\overline{\mathcal{X}}} \cup \mathcal{I}_{t'}$ and $\mathcal{I}_j$ are disjoint, all the authorized sets in $\mathcal{A}_n|_{[t]_{\overline{\mathcal{X}}} \cup \mathcal{I}_{t'}}$ are authorized in $\mathcal{Z}_{t'}$. ◀

3.3 Evolving Secret Sharing Over a Fixed Number of Parties

Our technique also implies a (weaker) lower bound on the share size of adaptive evolving secret sharing, when the number of parties is known from advanced (but the access structure is unknown).[5] For example, one can prove that for the empty access structure, every scheme that supports an arbitrary structure over $2n$ parties, must give a share of length $n/\log n$ to at least $n - \log n$ of the first n parties. Otherwise, there are $\log n$ such parties with total share size less than n. We thus can use the remaining n parties to complete Csirmaz's structure, with these $\log n$ parties being the set $\mathcal{C}$. This is of course a contradiction to Theorem 12.

We also observe that the share size in this model, of adaptive evolving secret sharing when the number of parties is known, is related to the share size in classical secret sharing, up to a linear factor in the number of parties n. Indeed, assume that for *every* access structure over n parties there exists a (classical) secret sharing scheme with maximal share size ℓ. The following shows that the optimal share size of evolving secret sharing over n players is at most $2n \times \ell$ (the other direction - that the share size in evolving secret sharing is not smaller than the share size in classical secret sharing - is trivial). Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of parties, and first assume for simplicity that *every* authorized set contains the last party P_n. Let $\mathcal{A}_n$ be the final access structure, and let $(SHARE, REC)$ be a (classic) secret sharing scheme for $\mathcal{A}_n$. We can construct an evolving secret sharing scheme as follows: when the i-th party arrives, for every $i \in [n-1]$, the scheme gives it random (uniformly and independently chosen) ℓ bits as the share π_i. The share of P_n is $\{\pi_i \oplus SHARE(s)_i\}_{i \in [n]}$ (letting $\pi_n = 0^\ell$). Clearly, the share size of the P_n in this scheme is $n \cdot \ell$, and all other parties get a share of size ℓ.

To get rid of the assumption that all the authorized sets contain the last player, we can simply share the secret independently to n access structures, when the i-th access structure contains all the authorized sets in which P_i is the last party. This will yield a share size of length at most $(n-1) \cdot \ell + n \cdot \ell$ (as every party is the last in exactly one such access structure).

[5] Non-adaptive evolving secret sharing with finite number of parties is equivalent to classical secret sharing.

References

1 Benny Applebaum, Amos Beimel, Oriol Farràs, Oded Nir, and Naty Peter. Secret-sharing schemes for general and uniform access structures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 441–471. Springer, 2019.

2 Benny Applebaum, Amos Beimel, Oded Nir, and Naty Peter. Better secret sharing via robust conditional disclosure of secrets. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 280–293, 2020.

3 Benny Applebaum, Amos Beimel, Oded Nir, Naty Peter, and Toniann Pitassi. Secret sharing, slice formulas, and monotone real circuits. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

4 Benny Applebaum and Oded Nir. Upslices, downslices, and secret-sharing with complexity of 1. 5 n. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part III*, pages 627–655, 2021.

5 Amos Beimel. Secret-sharing schemes: A survey. In *International conference on coding and cryptology*, pages 11–46. Springer, 2011.

6 Amos Beimel and Ilan Orlov. Secret sharing and non-shannon information inequalities. *IEEE Transactions on Information Theory*, 57(9):5634–5649, 2011.

7 László Csirmaz. The dealer's random bits in perfect secret sharing schemes. *Studia Scientiarum Mathematicarum Hungarica*, 32(3):429–438, 1996.

8 László Csirmaz. The size of a share must be large. *Journal of cryptology*, 10(4):223–231, 1997.

9 László Csirmaz and Gábor Tardos. On-line secret sharing. *Designs, Codes and Cryptography*, 63(1):127–147, 2012.

10 Ilan Komargodski, Moni Naor, and Eylon Yogev. How to share a secret, infinitely. *IEEE Transactions on Information Theory*, 64(6):4179–4190, 2017.

11 Ilan Komargodski and Anat Paskin-Cherniavsky. Evolving secret sharing: dynamic thresholds and robustness. In *Theory of Cryptography Conference*, pages 379–393. Springer, 2017.

12 Tianren Liu and Vinod Vaikuntanathan. Breaking the circuit-size barrier in secret sharing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 699–708, 2018.

13 Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Towards breaking the exponential barrier for general secret sharing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 567–596. Springer, 2018.

14 Anat Paskin-Cherniavsky. How to infinitely share a secret more efficiently. *Cryptology ePrint Archive*, 2016.

Csirmaz's Duality Conjecture and Threshold Secret Sharing

Andrej Bogdanov
University of Ottawa, Canada

Abstract

We conjecture that the smallest possible share size for binary secrets for the t-out-of-n and $(n-t+1)$-out-of-n access structures is the same for all $1 \le t \le n$. This is a strenghtening of a recent conjecture by Csirmaz (*J. Math. Cryptol.*, 2020). We prove the conjecture for $t = 2$ and all n. Our proof gives a new $(n-1)$-out-of-n secret sharing scheme for binary secrets with share alphabet size n.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures; Theory of computation → Cryptographic primitives; Mathematics of computing → Information theory; Security and privacy → Mathematical foundations of cryptography

Keywords and phrases Threshold secret sharing, Fourier analysis

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.3

Funding *Andrej Bogdanov*: This work was supported by RGC GRF grant CUHK 14301519 and NSERC grant RGPIN-2023-05006.

Acknowledgements Part of the research was carried out while the author was with the Chinese University of Hong Kong. I thank the anonymous ITC 2023 reviewers for helpful suggestions.

An *access structure* $\mathcal{A}$ over n parties is a nonempty monotone set system over ground set $\{1, \dots, n\}$. A *secret sharing scheme* [7, 1] for $\mathcal{A}$ with secret alphabet Σ is a collection of joint distributions $(X_1(\sigma), \dots, X_n(\sigma))$ with $\sigma \in \Sigma$ taking values in Γ^n such that

Secrecy: If $S \notin \mathcal{A}$ then $(X_i(\sigma) : i \in S)$ are identically distributed for all $\sigma \in \Sigma$.
Reconstruction: If $R \in \mathcal{A}$ then $(X_i(\sigma) : i \in R)$ determine σ with probability 1.

The *information rate* of the scheme is the ratio $\log|\Sigma| / \log|\Gamma|$ of the secret size and the share size. The *dual* of $\mathcal{A}$ is the access structure $\mathcal{A}^* = \{\overline{S} \colon S \notin \mathcal{A}\}$. Csirmaz [4] asks whether the following duality conjecture holds:

▶ **Conjecture 1.** *If $\mathcal{A}$ has a secret sharing scheme of information rate ρ for some secret alphabet size $|\Sigma|$, then $\mathcal{A}^*$ has a secret sharing scheme of information rate at least ρ for some secret alphabet size $|\Sigma'|$.*

As supporting evidence, Csirmaz shows that duality holds for the polymatroid relaxation of $\mathcal{A}$. This is a relaxation whose variables are the joint entropies of subsets of shares and whose constraints consist of a (in general incomplete) set of linear inequalities. On the other hand, he proves that duality fails for a relaxed asymptotic notion of secrecy. It is natural to consider the following even stronger conjecture:

▶ **Conjecture 2.** *For every Σ, if $\mathcal{A}$ has a secret sharing scheme of information rate ρ for secret alphabet Σ, then so does $\mathcal{A}^*$.*

In the case when Σ is the order of a finite field and the scheme is restricted to be linear, Conjecture 2 is known to hold (see Lemma 7.2 in [5]).


© Andrej Bogdanov;
licensed under Creative Commons License CC-BY 4.0

My motivation for Conjecture 2 is that it can be tested on threshold schemes. Such schemes have asymptotic information rate 1 as $|\Sigma|$ grows with the number of parties so Conjecture 1 does not say anything new about them. In contrast, when $|\Sigma| < n$, Conjecture 2 appears to be open for threshold schemes.

Here I study Conjecture 2 for threshold schemes and binary secrets, i.e., $|\Sigma| = 2$. This specialization is formulated as Conjecture 3. The *t-out-of-n* access structure consists of all t-element subsets of $\{1, \dots, n\}$.

▶ **Conjecture 3.** *If there exists a t-out-of-n scheme for binary secrets and share alphabet size γ then there also exists a $(n-t+1)$-out-of-n scheme for binary secrets and share alphabet size γ.*

The conjecture is true for every $n \geq 2$ when $t \in \{1, n\}$. Let $\gamma_2(\mathcal{A})$ denote the smallest possible share alphabet size for binary secrets and access structure $\mathcal{A}$. When $t = 1$ and $t = n$ one-bit secrets are possible and clearly optimal, so $\gamma_2(1\text{-out-of-}n) = \gamma_2(n\text{-out-of-}n) = 2$. A more interesting case is $t \in \{2, n-1\}$.

▶ **Proposition 4.** *For all $n \geq 2$, $\gamma_2(2\text{-out-of-}n) = n$.*

The lower bound $\gamma_2(2\text{-out-of-}n) \geq n$ was proved by Kilian and Nisan (see [2]). When n is a power of a prime (i.e., a finite field order) the upper bound can be obtained from Shamir's secret sharing with "infinity" as one of the evaluation points (see e.g. [3]). An alternative construction, which was communicated to me by Ilan Komargodski around 2016, works for all n. A variant of it is shown in the proof of Proposition 4 below.

If duality were to hold the same bound should be expected for the $(n-1)$-out-of-n access structure. The required lower bound was shown by Bogdanov, Guo, and Komargodski [2]. When n is a power of a prime the upper bound can also be derived from Shamir's scheme. The main result here is that this bound can be matched for non-prime powers n:

▶ **Theorem 5.** *For all $n \geq 2$, $\gamma_2((n-1)\text{-out-of-}n) = n$.*

The smallest example for which Theorem 5 is new is $n = 6$. This is a good example to keep in mind for the rest of the discussion.

Perspective: Lower bounds on alphabet size

There are two methods for lower bounding $\gamma_2(t\text{-out-of-}n)$ that give incomparable results. The analysis of Kilian and Nisan (KN) shows $\gamma_2(t\text{-out-of-}n) \geq n - t + 2$ for all $t \geq 2$. The analysis of Bogdanov, Guo, and Komargodski (BGK) shows the same lower bound for $\gamma_2((n-t+1)\text{-out-of-}n)$. Among the two, KN is more intuitive. They reduce their statement to the special case $t = 2$. When $t = 2$ let X_i and Y_i denote the i-th party's share of zero and one, respectively. Assuming the shares of zero and one are sampled independently, the KN bound follows from the two inequalities

$$1 = \mathbb{E}[1] \geq \mathbb{E}\big[|\{i\colon X_i = Y_i\}|\big] = \sum_{i=1}^{n} \Pr[X_i = Y_i] \geq \sum_{i=1}^{n} \frac{1}{|\Gamma|} = \frac{n}{|\Gamma|}.$$

The first inequality is by correctness of reconstruction (if $X_i = Y_i$ and $X_j = Y_j$ is possible the corresponding values would reconstruct to both zero and one) and the second one is by secrecy (X_i and Y_i are identically distributed, so $\Pr[X_i = Y_i]$ is a collision probability). The middle equality is linearity of expectation.

In contrast, BGK work directly with the probability mass functions p_0, p_1 of the shares of zero and one. They derive two types of constraints on the Fourier transform $\hat{f}$ of the real-valued function $f = p_1 - p_0$ over Γ^n. The first type is a reformulation of secrecy in the Fourier domain:

$$|\hat{f}(\chi)|^2 = 0 \qquad \text{for every } \chi \text{ such that } \operatorname{Supp}\chi \notin \mathcal{A}, \tag{BGK1}$$

where $\operatorname{Supp}\chi = \{i\colon \chi_i \neq 0\}$ is the support of the character χ viewed as an element of $\mathbb{Z}_q^n$ where $q = |\Gamma|$. The second type of constraint is the following (somewhat mysterious) relaxation of reconstruction:

$$\sum_A \Big(\sum_{\chi\colon \operatorname{Supp}\chi = A} |\hat{f}(\chi)|^2 \Big) \Big(-\frac{1}{q-1} \Big)^{|A \setminus B|} \geq 0 \qquad \text{for all } B \in \mathcal{A}. \tag{BGK2}$$

This system of constraints is a linear program in the variables $|\hat{f}(\chi)|^2, \chi \in \mathbb{Z}_q^n$. The BGK lower bound follows from its infeasibility when $q < n$ and $\mathcal{A}$ is the $(n-1)$-out-of-n access structure.

If Conjecture 3 were true, BGK would be a direct consequence of it and KN. Thus a natural first step towards Conjecture 3 would be to seek an alternative proof of BGK. The Conjecture itself suggests a route for such a proof: Assume that a $(n-t+1)$-out-of-n scheme with impossibly good share alphabet size γ_2 exists. Use this scheme to construct a t-out-of-n scheme with the same parameters. BGK offers a possible clue about this transformation: A feasible solution to the linear program (BGK1-BGK2) for access structure $\mathcal{A}$ should correspond to a secret sharing scheme for $\mathcal{A}^*$.

I do not know how to construct this transformation. For the purposes of investigating this potential "duality" between a secret sharing scheme and its Fourier transform it should be instructive to compare known secret sharing schemes for $\mathcal{A}$ and $\mathcal{A}^*$ and their Fourier transforms. I discovered the proof of Theorem 5 by working backwards from this correspondence. In the case of $(n-1)$-out-of-n schemes, the constraints (BGK1-BGK2) provide substantial information about what a scheme for this access structure should look like, if one exists at all. The scheme itself was obtained by reverse engineering f (and the distributions p_0 and p_1) from its Fourier transform. It would be interesting if the same result can be obtained by direct construction.

Concrete challenges

Figure 1 shows the best currently known lower and upper bounds on $\gamma_2(t\text{-out-of-}n)$ for small values of t and n. Except for the entries in bold, the upper bounds follow from Shamir's scheme, while the lower bounds are from KN or BGK. The upper bound for γ_2(3-out-of-5) can be obtained from a $(6, 4, 3)$ MDS code over $\mathbb{F}_4$ (see e.g. [6, Chapter 11]). The upper bound for γ_2(2-out-of-6) is from Proposition 4. The upper bound for γ_2(5-out-of-6) is from Theorem 5. The obvious next challenges are to calculate γ_2(3-out-of-6), and γ_2(4-out-of-6), or for those who prefer prime n, γ_2(3-out-of-7) and γ_2(5-out-of-7).

Constructions

Proof of the upper bound in Proposition 4. Shares of zero are n random symbols in $\Gamma = \{0, \ldots, n-1\}$ all equal to one another, while shares of one are a random cyclic permutation of the sequence $(0, 1, \ldots, n-1)$. Reconstruct to zero if the shares are equal and to one if they are different. The scheme is secret because the marginal distribution of every share is uniform (and therefore identical) in both cases. ◀

n \ t	1	2	3	4	5	6	7
2	2	2					
3	2	3	2				
4	2	4	4	2			
5	2	5	**4**	5	2		
6	2	**6**	5-7	5-7	**6**	2	
7	2	7	6-7	5-7	6-7	7	2

Figure 1 Upper and lower bounds on $\gamma_2(t\text{-out-of-}n)$.

The proof of Theorem 5 uses Fourier analysis of functions $f\colon \mathbb{Z}_q^n \to \mathbb{C}$. The q^n character functions

$$\chi(x) = \chi(x_0, \dots, x_{n-1}) = \exp\Big(\frac{2\pi i}{n} \cdot (x_0\chi_0 + \dots + x_{n-1}\chi_{n-1})\Big)$$

with $(\chi_0, \dots, \chi_{n-1}) \in \mathbb{Z}_q^n$ (also denoted by χ) form an orthonormal basis of the linear space of such functions with respect to the inner product $\langle f, g\rangle = \mathbb{E}[f(x)\overline{g(x)}]$ for x chosen uniformly at random from $\mathbb{Z}_q^n$. The Fourier transform of f is the unique function $\hat{f}\colon \mathbb{Z}_q^n \to \mathbb{C}$ for which $f = \sum_{\chi\in\mathbb{Z}_q^n} \hat{f}(\chi)\cdot\chi$. The Fourier coefficients $\hat{f}(\chi)$ are given by $\langle f, \chi\rangle$. Parseval's identity states that $\langle f, g\rangle = \sum_{\chi\in\mathbb{Z}_q^n} \hat{f}(\chi)\cdot\overline{\hat{g}(\chi)}$.

Proof of the upper bound in Theorem 5. Let $f\colon \mathbb{Z}_n^n \to \mathbb{C}$ be the function whose Fourier transform is

$$\hat{f}(\chi) = \begin{cases} 1, & \text{if } \chi \text{ is a cyclic shift of } (0, 1, \dots, n-1) \text{ or } (n-1, n-2, \dots, 0), \\ 0, & \text{if not.} \end{cases}$$

As will be shown shortly (or derived from symmetry of $\hat{f}$ under negation) f is real-valued. Shares of zero and one are sampled from the disjoint distributions p_0 and p_1 obtained by writing $f = C(p_0 - p_1)$ for a suitable normalizing constant $C > 0$. In more detail, let

$$p_0(x) = \begin{cases} C\cdot f(x), & \text{if } f(x) \ge 0 \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \quad p_1(x) = \begin{cases} -C\cdot f(x), & \text{if } f(x) \le 0 \\ 0, & \text{otherwise,} \end{cases}$$

where C is the factor that scales p_0 and p_1 to probability mass functions. The scaling factor is the same because $\hat{f}(0) = 0$.

Security follows from the fact that $\hat{f}(\chi)$ vanishes on all characters χ of Hamming weight at most $n-2$. In more detail, the advantage of any distinguisher D is

$$C\sum_{x\in\mathbb{Z}_n^n} D(x)f(x) = \frac{C}{n^n}\,\mathbb{E}\big[D(x)\overline{f(x)}\big] = \frac{C}{n^n}\sum_{\chi\in\mathbb{Z}_n^n} \hat{D}(\chi)\overline{\hat{f}(\chi)}$$

by Parseval's identity. If D depends on at most $n-2$ variables then $\hat{D}(\chi) = 0$ unless $|\chi| \le n-2$. As $\hat{f}(\chi) = 0$ for all χ of size at most $n-2$ the advantage of D must be zero.

To show reconstruction, f is calculated using the inverse Fourier formula. Letting $x = (x_0, \ldots, x_{n-1})$,

$$\begin{aligned}
f(x) &= \sum_{\chi \in \mathbb{Z}_n^n} \hat{f}(\chi)\chi(x) \\
&= \sum_{t \in \mathbb{Z}_n} \exp\Big(\frac{2\pi i}{n} \cdot \sum_{k=0}^{n-1} (k+t)x_k\Big) + \sum_{t \in \mathbb{Z}_n} \exp\Big(\frac{2\pi i}{n} \cdot \sum_{k=0}^{n-1} (-k+t)x_k\Big) \\
&= \sum_{t \in \mathbb{Z}_n} \exp\Big(\frac{2\pi i t}{n} \cdot \sum_{k=0}^{n-1} x_k\Big)\Big(\exp\Big(\frac{2\pi i}{n} \cdot \sum_{k=0}^{n-1} kx_k\Big) + \exp\Big(-\frac{2\pi i}{n} \cdot \sum_{k=0}^{n-1} kx_k\Big)\Big) \\
&= \Big(\sum_{t \in \mathbb{Z}_n} \exp\Big(\frac{2\pi i t}{n} \sum_{k=0}^{n-1} x_k\Big)\Big) \cdot 2\cos\Big(\frac{2\pi}{n} \sum_{k=0}^{n-1} kx_k\Big) \\
&= n \cdot \mathbf{1}\big(x_0 + \cdots + x_{n-1} = 0\big) \cdot 2\cos\Big(\frac{2\pi}{n} \cdot (x_1 + 2x_2 + \cdots + (n-1)x_{n-1})\Big).
\end{aligned}$$

Any $n-1$ of the n values $x_0, \ldots, x_{n-1}$ determine the remaining one on the set of inputs where f does not vanish. These values will satisfy the constraint $x_0 + \cdots + x_{n-1} = 0$ from which the missing x_i can be determined. This in turn determines the value of f and therefore the secret, which equals $\operatorname{sign} f(x)$ up to a change in representation. ◀

In more detail, the reconstruction procedure is this: Given shares $x_0, \ldots, x_{n-1}$ except for x_i, first compute $x_i = -\sum_{j \neq i} x_j \bmod n$, then output the sign of $\cos(2\pi(\sum kx_k)/n)$. (The cosine will never evaluate to zero because p_0 and p_1 assign zero probability to those shares.) Two alternative descriptions of $\operatorname{sign}\cos(2\pi(\sum kx_k)/n)$ are

- the parity of $\lfloor(\sum kx_k)/n\rceil$, where $\lfloor\cdot\rceil$ is the closest integer,
- the indicator of $|\lfloor\sum kx_k\rceil_n| < n/4$, where $\lfloor\cdot\rceil_n$ is the unique integer in the set $(-n/2, n/2]$ congruent modulo n.

The reconstruction procedure is clearly efficient. Its running time is quasilinear in n. How about sharing? Perfect sampling of the shares is not even possible in a model where the random seed is uniform over some finite domain! The reason is that some of the probabilities are irrational numbers. The scheme has perfect secrecy and reconstruction, but any realistic implementation of it must be imperfect.

It is possible to deduce from general considerations that if there exists a bit secret sharing scheme, then there exists one over the same share alphabet in which all probabilities are rational. The reason is that once the sign-pattern of f is fixed (i.e., once it is determined which shares reconstruct to zero and which reconstruct to one), finding the share probabilities that satisfy the secrecy constraints amounts to solving a linear program with rational coefficients. If this linear program is feasible then a rational solution must exist.

Nevertheless, even if imperfections in sampling are allowed, it is unclear how efficient a $(n-1)$-out-of-n scheme with share alphabet size n can be. Is it possible to sample an ϵ-approximation to the shares in time polynomial in n and $1/\epsilon$ for all n and ϵ?

To summarize, the crucial property of f is that its *weak sign* can be determined from any subset of shares that allow reconstruction. By weak sign I mean that one of the non-exclusive conclusions $f(x) \leq 0$ or $f(x) \geq 0$ can be reached only from knowledge of those coordinates of x that fall inside the reconstruction set. If an f with this property can be constructed under the constraints (BGK1) then reconstruction is possible. In the proof of Theorem 5 the cyclic structure of the nonvanishing Fourier coefficient plays a useful role. If, for example, $\hat{f}(\chi)$ was chosen to equal 1 on all characters of weight $n-1$ it appears that reconstruction wouldn't be possible.

Finally, notice the symmetry between the secret sharing scheme in the proof of Proposition 4 and the construction of $\hat{f}$ in the proof of Theorem 5. Is this a coincidence or an instance of duality?

References

1 G.R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the 1979 AFIPS National Computer Conference*, pages 313–317, Monval, NJ, USA, 1979. AFIPS Press.

2 Andrej Bogdanov, Siyao Guo, and Ilan Komargodski. Threshold secret sharing requires a linear-size alphabet. *Theory of Computing*, 16(2):1–18, 2020. doi:10.4086/toc.2020.v016a002.

3 Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015. URL: http://www.cambridge.org/de/academic/subjects/computer-science/cryptography-cryptology-and-coding/secure-multiparty-computation-and-secret-sharing?format=HB&isbn=9781107043053.

4 László Csirmaz. Secret sharing and duality. *J. Math. Cryptol.*, 15(1):157–173, 2020. doi:10.1515/jmc-2019-0045.

5 Satyanarayana V. Lokam. *Complexity Lower Bounds Using Linear Algebra*. Now Publishers Inc., Hanover, MA, USA, 2009.

6 Ron Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006. doi:10.1017/CBO9780511808968.

7 Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979. doi:10.1145/359168.359176.

The Cost of Statistical Security in Proofs for Repeated Squaring

Cody Freitag
Cornell Tech, New York, NY, USA

Ilan Komargodski
The Hebrew University, Jerusalem, Israel
NTT Research, Sunnyvale, CA, USA

Abstract

In recent years, the number of applications of the repeated squaring assumption has been growing rapidly. The assumption states that, given a group element x, an integer T, and an RSA modulus N, it is hard to compute $x^{2^T} \bmod N$ – or even decide whether $y \stackrel{?}{=} x^{2^T} \bmod N$ – in parallel time less than the trivial approach of simply computing T squares. This rise has been driven by efficient proof systems for repeated squaring, opening the door to more efficient constructions of verifiable delay functions, various secure computation primitives, and proof systems for more general languages.

In this work, we study the complexity of *statistically sound* proofs for the repeated squaring relation. Technically, we consider proofs where the prover sends at most $k \geq 0$ elements and the (probabilistic) verifier performs generic group operations over the group $\mathbb{Z}_N^\star$. As our main contribution, we show that for any (one-round) proof with a randomized verifier (i.e., an MA proof) the verifier either runs in parallel time $\Omega(T/(k+1))$ with high probability, or is able to factor N given the proof provided by the prover. This shows that either the prover essentially sends p, q such that $N = p \cdot q$ (which is infeasible or undesirable in most applications), or a variant of Pietrzak's proof of repeated squaring (ITCS 2019) has optimal verifier complexity $O(T/(k+1))$. In particular, it is impossible to obtain a statistically sound one-round proof of repeated squaring with efficiency on par with the computationally-sound protocol of Wesolowski (EUROCRYPT 2019), with a generic group verifier.

We further extend our one-round lower bound to a natural class of recursive interactive proofs for repeated squaring. For r-round recursive proofs where the prover is allowed to send k group elements per round, we show that the verifier either runs in parallel time $\Omega(T/(k+1)^r)$ with high probability, or is able to factor N given the proof transcript.

2012 ACM Subject Classification Theory of computation → Proof complexity

Keywords and phrases Cryptographic Proofs, Repeated Squaring, Lower Bounds

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.4

Related Version *Full Version*: `https://eprint.iacr.org/2022/766`

Funding *Cody Freitag*: Cody Freitag's work was partially done during an internship at NTT Research. He is also supported in part by the National Science Foundation Graduate Research Fellowship under Grant No. DGE–2139899, DARPA Award HR00110C0086, and AFOSR Award FA9550-18-1-0267. *Ilan Komargodski*: Ilan Komargodski is the incumbent of the Harry & Abe Sherman Senior Lectureship at the School of Computer Science and Engineering at the Hebrew University, supported in part by an Alon Young Faculty Fellowship, by a JPM Faculty Research Award, by a grant from the Israel Science Foundation (ISF Grant No. 1774/20), and by a grant from the US-Israel Binational Science Foundation and the US National Science Foundation (BSF-NSF Grant No. 2020643).

© Cody Freitag and Ilan Komargodski;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

The repeated squaring (RS) assumption (first introduced by Rivest, Shamir, and Wagner [33]) states that for an RSA modulus N, a group element x, and a time bound T, it is hard to compute $x^{2^T} \bmod N$ – or even decide whether $y \stackrel{?}{=} x^{2^T} \bmod N$ – in parallel time less than the trivial approach of simply computing T squares. Under this assumption, the RS function is a candidate *sequential* function, meaning it cannot be sped up using parallel processors. This gives the ability to tune the time bound T so that computing the RS function requires a specified amount of wall-clock time, e.g. you can set T so that computing $x^{2^T} \bmod N$ takes at least 1 hour. This property, combined with the algebraic structure of the repeated squaring function, has led to many exciting applications.

Originally, Rivest, Shamir, and Wagner [33] used the RS function to construct time-lock puzzles. Time-lock puzzles provide a mechanism to send a message "to the future", by allowing a sender to quickly generate a puzzle with an underlying message that remains hidden for a specified amount of wall-clock time. These are possible because repeated squaring in RSA group has a natural trapdoor that allows the puzzle generator to evaluate the function quickly. Namely, given the factorization of N, one can reduce 2^T mod the order of the group to compute x^{2^T} efficiently. In this sense, time-lock puzzles effectively give a "fine-grained" variant of standard cryptographic commitments, where the hiding property only holds for some fixed amount of time specified by T.

More recently, Pietrzak [30] and Wesolowski [41] showed how to construct efficient (non-interactive) proofs for the RS relation. (In both works, this was obtained by first designing an interactive protocol for RS and then making it non-interactive via the Fiat-Shamir transform [12].) Such proofs for the RS relation have been the main driving force behind various applications of the repeated squaring function. For instance, they are used to construct *verifiable delay functions* (VDFs), first proposed by Boneh, Bonneau, Bünz, and Fisch [5], where the output of the function serves as a unique "proof-of-sequential-work" that can be efficiently verified with an associated non-interactive proof. Among other applications, Boneh et al [5] propose VDFs as a way to generate randomness for a trusted lottery or to construct resource-efficient blockchains (which has since been adopted by Chia [1]).

Since the initial works of [6, 30, 41], there have been many new proposed applications for proofs of repeated squaring: [7] construct accumulators, [11] construct randomness beacons, [10] construct polynomial commitments for succinct arguments, and [4] build off of [10] to construct time and space-efficient arguments. Furthermore, there has been much focus on understanding the efficiency and security of such proofs (see e.g. [30, 41, 6, 11, 34, 4, 19]) as well as the security of the sequentiality assumption underlying RS (see e.g. [36, 23, 37, 35]).

Proofs of repeated squaring. In this work, we are interested in proofs for the repeated squaring language, defined with respect to a multiplicative group of integers modulo N:

$$\mathcal{RS}_N = \left\{ (x, y, T) \mid y = x^{2^T} \bmod N \right\},$$

where x and y are two group elements and T is an integer.[1] A proof system for this language consists of a prover P and a (probabilistic) verifier V. The goal of the verifier is to decide, given an instance (x, y, T), whether it is in $\mathcal{RS}_N$ or not, and the prover sends the verifier a

[1] More generally, our results hold for the repeated squaring relation in any multiplicative group of unknown order. We focus on RSA groups for this introduction for simplicity of presentation.

proof string π to help in this task. We emphasize that we are mostly interested in the setting of non-interactive proofs in this work as they are most useful for applications, but later we also generalize the above and consider interactive protocols.

For a proof system to be meaningful, it must satisfy completeness and soundness. Completeness stipulates that an honestly generated proof will convince the verifier whenever $(x, y, T) \in \mathcal{RS}_N$, and soundness requires that, whenever $(x, y, T) \notin \mathcal{RS}_N$, there is no cheating proof π that will convince a verifier with noticeable probability. As stated, this is a statistical notion of soundness, asking whether there exist cheating proofs at all. We also consider proof systems that are only computationally sound – commonly known as *arguments* – where there may exist such cheating proofs, but we assume they are hard to find. Furthermore, computationally sound proofs come in many forms depending on the kind of assumptions they rely on.

There are several known proofs for $\mathcal{RS}_N$ (see Appendix B for an overview). For all of them, at least one of the following hold:

(A) The proof is only computationally sound. This is undesirable or even insufficient for several important applications; see below.

(B) The prover leaks the factorization to the verifier. This only allows N to be used once and is infeasible unless the prover can factor N.

(C) There is a tradeoff between proof size, $|\pi|$, and (parallel) running time of V, Time_V, where

$$|\pi| \cdot \mathsf{Time}_V \geq T.$$

That is, inefficiency is somewhat necessary: any improvement in communication complexity must necessarily cause an increase in computational complexity and vice versa.

All three of the above properties are undesirable for various reasons as we discussed above. So, in this work, we aim to understand whether having one of the above drawbacks is necessary. We do this by studying the cost of statistical soundness in $\mathcal{RS}_N$:

Can we construct a (statistically sound) non-interactive proof system for $\mathcal{RS}_N$ with low communication, an efficient verifier, and that doesn't leak the factorization of N?

On statistical soundness and tradeoffs between efficiency and security. Purely based on security and ignoring efficiency, it is clear that proofs with statistical soundness are strictly better than ones with only computational soundness. So perhaps in high-stakes applications (e.g. for blockchains where lots of money is at stake), having soundness rely on newer and untested mathematical/ computational assumptions may not be worth it. It's worth emphasizing, however, that the computationally sound, non-interactive proofs for $\mathcal{RS}_N$ rely not only on well-formulated computational assumptions, but potentially also on assumptions regarding the setup used to generate the RSA modulus N.

Wesolowski's argument [41], for instance, is completely broken if the prover knows the factorization of N. This is not the case for Pietrzak's non-interactive proof [30], but this already suffers an $O(\log T)$ multiplicative communication overhead in efficiency. Still, Pietrzak's protocol is potentially broken if N is not a product of safe primes. To fix this, Block et al. [4] give a non-interactive proof that works for any group and hence value of N, but results in an additional $O(\lambda)$ blowup in efficiency over Pietrzak's proof.[2] This protocol

[2] We mention that Hoffmann et al. [19] give a similar result to [4] that works in any group with improved efficiency by considering repeated qth powers for structured $q \gg 2$. Still, their protocol inherently cannot be made more efficient than the protocol of [30].

still relies on a random oracle to securely instantiate the FS heuristic though. Technically, the FS heuristic can be instantiated for [4] assuming LWE [3], but only at a further cost in efficiency for both the prover and the verifier with its own additional trusted setup. So it seems, no matter which computationally sound proof you choose, there is a complex combination of both computational and setup assumptions, and if one piece fails, the security of the entire system may be completely compromised. However, if the underlying proof is statistically sound, then this problem does not exist as it is *impossible* to generate accepting proofs for false statements (this is true even if $P = NP$ and factoring is easy).

Even if one is extremely confident in their computational assumptions, there are protocols based on proofs for $\mathcal{RS}_N$ that are completely broken if the wrong underlying protocol is used. Specifically, in the recent work of Freitag, Komargodski, Pass, and Sirkin [13], they use proofs for $\mathcal{RS}_N$ on top of a time-lock puzzle based on the RS function in order to construct publicly verifiable and non-malleable time-lock puzzles (on their way to building fair multi-party coin-flipping and auction protocols without trusted setup). In the context of time-lock puzzles, the party who generates the puzzle needs to know the factorization of N; this is actually a *feature* of time-lock puzzles, not a bug. However, this implies that they need the corresponding proof in their construction to be sound even if some party may know the factorization of N, so as pointed out above, Wesolowki's protocol will not suffice. They instead rely on Pietrzak's protocol, which is still plausibly secure when the factorization of N is leaked. But again, even though Wesolowski and Pietrzak's protocols are both "computationally secure", you cannot simply default to using the more practically efficient protocol of Wesolowski.

We highlight two important takeaways from the example of [13]:

- If using computationally sound rather than statistically sound proofs, protocol designers need to be very careful about the specific assumptions that the soundness of this proof relies on. This crucially includes the interplay between the setup assumptions and mathematical/ computational assumptions that are needed in the case of [13].
- In settings where security is required (or simply desired) even when the factorization of N may be known, current protocols start with a statistically sound interactive proof, and then compile it to a non-interactive argument using the FS heuristic. As such, we see it as an important goal to characterize the efficiency of general, statistically sound, interactive proofs for $\mathcal{RS}_N$, which first requires understanding the setting of statistically sound, non-interactive proofs.

Still, statistically secure protocols tend to be much less efficient than their computationally secure counterparts. As such, our overall goal is to try to formally characterize the exact tradeoffs between efficiency and security for proofs of $\mathcal{RS}_N$, which has led to many exciting practical and theoretical applications in recent years.

The complexity of RS (without proofs). Even ignoring the potential help from a prover, the complexity of the RS function – or deciding the $\mathcal{RS}_N$ language – was not well understood until the very recent works of [36, 23], even in generic models. Specifically, Rotem and Segev [36] show that computing the RS function or deciding $\mathcal{RS}_N$ in less than T parallel time implies a factoring algorithm for N, at least when restricted to generic-ring algorithms. Katz, Loss, and Xu [23] show a similar result for computing the RS function in the strong algebraic group model.[3]

[3] These are incomparable models. The generic-ring model allows for multiplication/ division/ addition/ subtraction/ equality queries, but require that queries are independent of the group elements representations. The strong algebraic group model only allows multiplication/ division queries, but allows these queries to be made in a way that depends on the group elements explicit bit representations.

1.1 Our Results

We make progress towards resolving the above-mentioned questions. Within a certain restricted model (the generic-group model relative to a hidden order group; see below), we prove results on the tradeoffs between the communication complexity and the verifier's complexity in a large class of proof systems for $\mathcal{RS}$. In particular, for the class of proof systems that we consider, any improvement over known ones would lead to a non-trivial factoring algorithm. Thus, assuming that factoring is hard, any improvement must either be outside of the restricted model or relax soundness to computational.

A bound for MA proof systems. We consider proof systems where the prover sends the verifier a single possibly long message, and then the verifier decides whether to accept or not by running a probabilistic polynomial time computation. This corresponds to the class MA (which generalizes NP by allowing the verifier to be probabilistic).

We briefly mention two statistically sound proofs for $\mathcal{RS}$. First, the prover can just send the factorization (p, q) where $N = p \cdot q$. The verifier can check that $N = p \cdot q$, compute the order of the group $\varphi(N)$, and then efficiently check that y equals $x^{2^T \bmod \varphi(N)} \bmod N$. The second is a sumcheck-style proof [27] that is a generalization of Pietrzak's protocol [30] due to [11]. Here, the prover sends k evenly spaced "midpoints" between x and T, which results in $k+1$ statements corresponding to $T/(k+1)$ squares. The verifier uses random exponents to combines these statements into a new statement $(x', y', T/(k+1))$ that it can check itself in time $T/(k+1)$.

We show that the above two protocols are essentially the best possible among all generic-group proofs. Specifically, we show that *either* we can factor composite numbers (matching the first protocol), or otherwise in any MA proof that includes $k \geq 0$ group elements, the verifier must run in parallel time at least $\Omega(T/(k+1))$ (matching the second protocol). Additionally, if neither of these hold, then the protocol must not be statistically sound – there must exist proofs for false statements, even if they may be computationally hard to find.

We prove our result by presenting an algorithm that uses any "too-good-to-be-true" generic-group MA proof to solve factoring in the plain model. To this end, we use Maurer's [28] generic-group algorithms abstraction and extend it to capture MA proofs. In our model, we restrict the verifier to be a generic-group algorithm (in Maurer's sense) that makes a bounded number of group multiplication and division queries[4], and we say that it accepts if it outputs the group's identity 1. Notice, for example, that this allows the verifier to compute two element g, h and accept if they are equal by outputting $g \cdot h^{-1}$. Furthermore, the verifier can perform ANDs of equality checks and accept if many pairs $(g_1, h_1), \ldots, (g_n, h_n)$ are equal (allowing parallel repetition). This can be done by sampling random exponents $r_1, \ldots, r_n \in [2^\lambda]$ and outputting $\prod_{i=1}^{n} (g_i \cdot h_i^{-1})^{r_i}$, a la the sumcheck-style technique used in [30]. Finally, we note that all efficient proofs specifically designed for $\mathcal{RS}_N$ fall into this generic model.

The prover, on the other hand, may still be an unbounded (not necessarily generic) algorithm whose proof consists of a bit string and a sequence of group elements. Note that *not* restricting the prover to be generic only makes our result applicable to larger classes of constructions, thereby making it stronger. Refer to Section 2 for the precise model

[4] Such algorithms are sometimes referred to as straight-line programs.

definition. We emphasize that even in this simplified one-round setting, it turns out to be highly non-trivial to prove our result in a way that captures the behaviors of arbitrary provers and verifiers; see Section 1.3 for an overview.

▶ **Theorem 1** (Simplified and Informal; see Theorem 4)**.** *For any generic-group MA proof system for $\mathcal{RS}_N$, if the prover sends $k \geq 0$ group elements and a string* st*, the verifier either runs in parallel time $\Omega(T/(k+1))$, or is able to factor N given* st*.*

In fact, we prove in Corollary 9 that the above holds for any hidden order group. In addition to RSA groups, this notably includes class groups of unknown order, which was suggested in the context of repeated squaring by Wesolowski [41] (see [9] for a more general survey on the use of class groups in cryptography). In the general case, we show that either the verifier runs in parallel time $\Omega(T/(k+1))$, or is able to compute (a non-zero multiple of) the order of the group given the string st output by the prover. However, by a variant of the Miller-Rabin primality test [29, 31], it is well known that this implies a factoring algorithm for N when working over the multiplicative group $\mathbb{Z}_N^\star$.

We note that if the prover is efficient, we can compute st ourselves. So, the existence of a verifier with $o(T/(k+1))$ parallel runtime implies a standard model factoring algorithm.

A bound for recursive interactive proofs. We extend our lower bound for MA proofs to a certain natural class of general (multi-round) interactive proofs (IPs). Specifically, we consider a class of *recursive IPs*, where in every round of communication, the prover attempts to prove a new instance of $\mathcal{RS}_N$, although with a different starting point x, a different endpoint y, and a different delay parameter T. This class of IPs captures many sumcheck-style proofs for $\mathcal{RS}_N$; see Appendix B for an overview. In particular, for a bound on the round complexity r and a communication bound k, the adaptation of Pietrzak's [30] protocol results with a recursive IP with total communication $k \cdot r$ and verifier running time $O(T/(k+1)^r)$. Here, we obtain an optimal tradeoff between the message complexity, the round complexity, and the verifier's parallel running time, at least when restricted to generic group verifiers.

▶ **Theorem 2** (Simplified and Informal)**.** *For any generic-group r-round recursive interactive proof system for $\mathcal{RS}_N$, if the prover sends k group elements per round and results in a transcript* tr*, the verifier either runs in parallel time $\Omega(T/(k+1)^r)$, or is able to factor N given* tr*.*

Future Directions and Open Problems

Our work leaves many exciting open problems. We mention some of them next:

1. We prove our result in the generic-group model where we only allow multiplication and division queries. It would be interesting to extend this to handle general equality queries or addition/ subtraction queries in the the generic-ring model [2, 21, 36].
2. Can we get a similar result to Theorem 2 for general (public-coin) IPs rather than just for "recursive" IPs?
3. In general, for what other languages can we say that sumcheck-style (e.g. see [8] and references therein) proofs are optimal (at least among a reasonable but restricted class of verifiers)?

Paper Organization

In Section 1.2, we give an overview of related work, and then in Section 1.3, we give a detailed overview of the main techniques in this work. In Section 2, we define the generic group model we use in this work in the context of proofs. Then in Section 3, we give our main result for MA proofs. We provide standard notation and preliminaries in Appendix A and a detailed overview of existing non-interactive proofs for the repeated squaring relation in Appendix B.

Due to space constraints, we refer the reader to the full version of the paper for more details regarding our result on recursive interactive proofs and for all proofs.

1.2 Related Work

Complexity of interactive proofs. Goldreich and Håstad [15] initiated the investigation of interactive proofs with bounded communication. They showed that if a language L has an interactive proof in which the total communication is bounded by $c(n)$ bits then $L \in \mathsf{BPTime}(2^{c(n)} \cdot \mathsf{poly}(\mathsf{n}))$. Further relations between the communication complexity of interactive proof for a language and its complement were shown by Goldreich, Vadhan, and Widgerson [16].

The IP=PSPACE result [27, 38] says that languages that can be verified in polynomial time are exactly those proofs that can be generated with polynomial space. In this interactive proof system, the honest prover runs in super-polynomial time (even for log-space languages); this is true even for the scaled down version which captures polynomially recognizable languages. Nevertheless, the "easy" side of this result says that every language with an interactive proof of c bits is decidable with c space [27, 38]. Therefore, languages that require a lot of space to decide cannot have super efficient interactive proof systems.

Computationally sound proof systems can recognize any language in NP while using only poly-logarithmic message complexity (assuming collision resistant hash functions) [24].

In the statistical setting, the first interactive proofs with an *efficient* prover were given by Goldwasser, Kalai, and Rothblumn [17]. They designed an interactive proof system where the honest prover is efficient and run in polynomial time. In their proof system the language is given by a log-space uniform Boolean circuit with depth d and input length n. Their verifier runs in time $n \cdot \mathsf{poly}(d, \log n)$, the communication complexity is $\mathsf{poly}(d, \log n)$, and the prover runs in time $\mathsf{poly}(n)$. This protocol is very useful for low-depth computations.

Reingold, Rothblum, and Rothblum [32] showed a different protocol which suits polynomial time and bounded-polynomial space computations. They give a constant round protocol for polynomial time and space $S = S(n)$ languages such that: the honest prover runs in polynomial time, the verifier is almost linear time, and the communication complexity is $O(S \cdot n^{\delta})$ for $\delta \in (0, 1)$. Applied on the repeated squaring language, (where $S = \mathsf{poly} \log n$) this protocol's communication roughly matches Pietrzak's [30] when adapted to run in constant rounds (in which case it also requires the transmission of n^{δ} group elements).

Generic models. The problem we consider can be placed in a long line of research on proving efficiency trade-offs for various primitives, in some restricted class of constructions usually termed "black-box" or "generic". Generic or black-box constructions have the benefit of being applicable to every instantiation of the underlying structure, irrespectively of the exact details of its description. For specific instances, this usually allows for cleaner and more efficient constructions. The interactive proofs for $\mathcal{RS}$ of Pietrzak [30] and Wesolowski [41] are generic.

Our work is the first to study the complexity of proofs for $\mathcal{RS}$ from a foundational perspective. The most relevant previous works study the ("generic") complexity of related cryptographic primitives or assumptions. Rotem and Segev [36] and Katz et al. [23] showed that any generic algorithm for repeated squaring which is faster-than-trivial can be used to solve factoring. The result of [36] rules out generic constructions in the generic-ring model introduced by Aggarwal and Maurer [2] (see also Jager and Schwenk [21]). The result of [23] rules out constructions in the strong algebraic group model (extending [14]) wherein the adversary may use the concrete representation of group elements to make its group queries. In another work, Rotem, Segev, and Shahaf [37] showed that hidden order groups are necessary for achieving "delay" functions, at least generically. The result of [37] rules out generic-group constructions in Maurer's model [28] (same as our proof).

On class groups. It is worth noting that class groups are an alternative candidate for a group of hidden order. In contrast to RSA groups, they only require a trusted setup consisting of an honestly generated random string. Since this setup is simpler and easy to generate, it is presumably less likely that someone may know a trapdoor (the order of the group) for class groups. However, while they can be used to construct VDFs, it is not known how to use them to get TLPs. See [9] for a general survey of the use of class groups in cryptography.

1.3 Technical Overview

Throughout this overview, we use $\lambda \in \mathbb{N}$ to refer to the security parameter and let N denote the RSA modulus, where N is a product of two random λ-bit primes. We use $\mathbb{Z}_N^\star$ to denote the multiplicative group of integers mod N. We consider interactive proof systems for the repeated squaring relation $\mathcal{RS}_N$, which we represent via the function $f_{N,T}(x) = x^{2^T} \bmod N$ for any time bound $T \in \mathbb{N}$. As a warm up, we will start by considering single-round, NP-style, proof systems where the verifier is a *deterministic*, generic group algorithm. We will later show how to deal with randomized verifiers, and additionally extend to the class of *recursive* interactive proofs.

Overview of generic group proof systems. A (non-interactive) proof system consists of two parties, the prover P and the verifier V. On input a group element $x \in \mathbb{Z}_N^\star$, P's goal is to convince V that another group element y is equal to $f_{N,T}(x) = x^{2^T} \bmod N$. P is allowed to send V up to k group elements $\pi_1, \ldots, \pi_k \in \mathbb{Z}_N^\star$ as well as a bit string $\mathsf{st} \in \{0,1\}^*$. Throughout the overview, we will always assume that P sends exactly k group elements as part of its proof. V processes this information and outputs 1 to accept that $y = x^{2^T} \bmod N$ or rejects otherwise. We require that the proof system satisfies the standard notions of completeness and soundness. Completeness says that if $y = x^{2^T} \bmod N$, then an honest prover P causes V to accept. We parameterize soundness by a parameter δ, which says that if $y \neq x^{2^T} \bmod N$, then no (potentially unbounded) cheating prover $P^\star$ can cause V to accept with probability more than δ.

We restrict the above model by requiring that V is a (straight-line) *generic group* verifier, whereas we still allow the prover to be unbounded and behave arbitrarily. Specifically, V takes as input the modulus N, the time bound T, the prover's string st as explicit inputs. However, V only has implicit access to the input group element x, the purported output y, and the proof elements $\pi_1, \ldots, \pi_k$ sent by P. Intuitively, this means that V is allowed to multiply and divide these elements arbitrarily, as long as it does so in a way that independent of their representation. We formalize this following Maurer's generic group model [28], which we outline in Section 2.

At the end of the day, we leverage the fact that V uses its explicit inputs[5] to effectively generate various exponents α, β, $\gamma_1, \ldots, \gamma_k$ such that its output is given by the group element corresponding to

$$V(N, T, \mathsf{st}, x, y, \pi_1, \ldots, \pi_k) = x^{\alpha} \cdot y^{\beta} \cdot \prod_{i=1}^{k} \pi_i^{\gamma_i} = g.$$

Furthermore, we can always run V with dummy elements $x, y, \pi_1, \ldots, \pi_k$ and compute the exponents $(\alpha, \beta, \gamma_1, \ldots, \gamma_k)$ by observing its group operations. We say that V accepts if the output group element g is equal to the multiplicative identity $1 \in \mathbb{Z}_N^\star$, and V rejects otherwise. While this convention may seem restrictive, as V doesn't even know whether it is accepting or rejecting, we claim that this is still very expressive as V can compute two different group elements g, h and then output $g \cdot h^{-1}$, which is 1 if and only if $g = h$. Most natural protocol for repeated squaring including [30, 41] fall into this category. Furthermore, the verifier can perform ANDs of equality checks and accept if many pairs $(g_1, h_1), \ldots, (g_n, h_n)$ are equal (allowing parallel repetition). This can be done by sampling random exponents $r_1, \ldots, r_n \in [2^\lambda]$ and outputting $\prod_{i=1}^{n} (g_i \cdot h_i^{-1})^{r_i}$, a la the sumcheck-style technique used in [30].

The complexity of deterministic (NP) proofs. As a warm-up, suppose that the verifier V is deterministic. This means that for every set of explicit inputs N, T, st that V receives, it generates the same exponents $(\alpha, \beta, \gamma_1, \ldots, \gamma_k)$. Given this knowledge, we want to characterize all possible strategies a cheating prover may use. So, say a cheating prover $P^\star$ wants to fool V on any $y = x^d \neq x^{2^T} \bmod N$. Effectively, $P^\star$ can only set each group element π_i to be equal to x^{z_i} for some value z_i.[6] Then, it follows that V accepts if

$$x^{\alpha} \cdot x^{d \cdot \beta} \cdot \prod_{i=1}^{k} x^{z_i \cdot \gamma_i} = 1.$$

However, since the base x is shared by all of the group elements, the above holds if

$$\alpha + d \cdot \beta + \sum_{i=1}^{k} z_i \cdot \gamma_i = 0 \bmod \mathsf{Carm}(N),$$

where $\mathsf{Carm}(N)$ is Carmichael totient function, which is defined as the minimal value c such that $g^c = 1 \in \mathbb{Z}_N^\star$ for all $g \in \mathbb{Z}_N^\star$.[7] But, as long as $\vec{\gamma} = (\gamma_1, \ldots, \gamma_k) \neq \vec{0} \bmod \mathsf{Carm}(N)$, it follows that $P^\star$ can simply solve for a solution to $z_1, \ldots, z_k$ in the equation above to generate a proof that will falsely convince V that $x^d = x^{2^T}$.[8]

[5] If we allowed V to also use the representation of the input group elements, this would correspond to the strong algebraic group model of [23].

[6] Note that this is not true in general since $\mathbb{Z}_N^\star$ is not cyclic and hence there are group elements not represented as x^c for some $c \in \mathbb{Z}$. However, we assume this in the overview for simplicity as it captures the main idea of the proof.

[7] We note that we can simply choose x to be a group element whose order attains the maximal value $\mathsf{Carm}(N)$. This is what allows us to switch to working over the exponent without loss of generality.

[8] We note that this style of attack works for Wesolowski's (computationally sound) proof of repeated squaring [41], which is an AM protocol. The adaptive root assumption essentially states that it is computationally infeasible to perform such an attack, leveraging the randomness sampled by the verifier before the prover sends its message.

Still, it may be the case that V simply ignores the proof elements $\pi_1, \ldots, \pi_k$ by setting $\gamma_1, \ldots, \gamma_k = 0$. In this case, we leverage the completeness of the proof system to conclude that either V is inefficient and runs in parallel time T, or V must be able to factor N. If $y = x^{2^T} \bmod N$ and $\gamma_1, \ldots, \gamma_k = 0$, then we know, by the above equation, that V accepts if

$$\alpha + 2^T \cdot \beta = 0 \bmod \mathsf{Carm}(N).$$

We consider two different cases, either (1) $\alpha + 2^T \cdot \beta = 0 \in \mathbb{Z}$ or (2) $\alpha + 2^T \cdot \beta = c \cdot \mathsf{Carm}(N)$ for some $c \neq 0 \in \mathbb{Z}$.

In case (2), this actually immediately implies a probabilistic factoring algorithm for N via a well known adaptation of the Miller-Rabin primality test (formally stated in Lemma 6). Since we can compute α and β, given the code of V and the prover's string st, and hence $\alpha + 2^T \cdot \beta = c \cdot \mathsf{Carm}(N)$, this implies a factoring algorithm in the standard model given st. If the prover P is efficient, then we can compute st by ourselves, so it implies a factoring algorithm for any N, without any auxiliary advice. We emphasize, however, that it may be the case that the explicit string st sent by P helps V to compute some value $\alpha = 2^T \bmod \mathsf{Carm}(N)$. For example, P could have just set st to be a representation of $\mathsf{Carm}(N)$, and V simply set $\alpha = 2^T \bmod \mathsf{Carm}(N)$ and $\beta = -1$. This is why the factoring algorithm must receive the proof string st as input in general.

We split case (1) into two further subcases, either (1A) $\beta = 0$ or (1B) $\beta \neq 0$. In case (1B) where $\beta \neq 0$, this implies that

$$2^T \leq 2^T \cdot |\beta| \leq |\alpha|.$$

But that implies that V must run in parallel time T to compute x^α since $|\alpha| \geq 2^T$.

In case (1A) where $\beta = 0$ and $\alpha + 2^T \cdot \beta = 0$, it must also be the case that $\alpha = 0$. However, we've already assumed that $\gamma_1, \ldots, \gamma_k = 0$, so this means that V just always outputs 1 and accepts! So clearly, (P, V) cannot be a valid proof system as V accepts any $y \neq x^{2^T} \bmod N$ with probability 1 in this case.

In summary, if (P, V) is a sound proof system where V is a *deterministic* generic group verifier, then either:

1. V must run in parallel time at least T, or
2. there is a standard model factoring algorithm for N given the code of V and the string st output by P.

Stated another way, if V runs in parallel time less than T, then V must be able to factor N (with the help of the prover via st).

Extending to randomized verifiers. The high level outline of the lower bound for randomized verifiers is actually very similar to the case of deterministic verifiers. However, allowing the verifier to use randomness to determine its exponents introduces many highly non-trivial challenges. The key distinction between deterministic and randomized verifiers is that randomized verifiers are allowed to choose their exponents as a function of their randomness, so the attack where a cheating prover simply solves a single equation to fool the verifier no longer works. Instead, the cheating prover needs to satisfy a random equation with better than δ probability in order to violate soundness. Still, we will show how we can use the verifier's exponents to factor, or argue that the verifier must have parallel running time greater than $T/(k+1)$ with high probability.

Throughout, we will consider a fixed set of explicit inputs N, T, and st received by the verifier. Then, for any random string $\rho \in \{0,1\}^\lambda$ sampled by the verifier, we use $\mathsf{coef}(\rho)$ to denote the exponents that V uses to compute its output. So, if

$$V(N, T, \mathsf{st}, x, y, \pi_1, \ldots, \pi_k; \rho) = x^\alpha \cdot y^\beta \cdot \prod_{i=1}^{k} \pi_i^{\gamma_i},$$

then we say that $\mathsf{coef}(\rho) = (\alpha, \beta, \gamma_1, \ldots, \gamma_k)$. We note that we refer to these exponents as "coefficients" as they will correspond to coefficients in a system of equations over the exponent, hence the notation $\mathsf{coef}(\rho)$.

Our main strategy is to sample many different values $\rho_1, \ldots, \rho_n$ such that $||\mathsf{coef}(\rho_i)||_{\max} \ll 2^{T/(k+1)}$ for each $i \in [n]$, where $||\cdot||_{\max}$ indicates the maximum absolute value in the coefficient vector. If this isn't possible, then that means that the verifier must run in parallel time at least $T/(k+1)$, and we are done. Otherwise, it remains to show that we can either use these coefficients to factor or show that (P, V) is not a valid proof system. For each randomness value ρ_i, let $\mathsf{coef}(\rho_i) = (\alpha_i, \beta_i, \gamma_{i,1}, \ldots, \gamma_{i,k})$ denote the corresponding coefficient vector for ρ_i. We combine all of these coefficients together in the following way. Let $\Gamma \in \mathbb{Z}^{n \times k}$ be the matrix consisting of all of the $\gamma_{i,j}$ values, and let $\vec{\alpha}, \vec{\beta} \in \mathbb{Z}^n$ be vectors of the α_i and β_i values. A key property we will leverage is that the system of equations $\Gamma \cdot \vec{z} = -\vec{\alpha} - d \cdot \vec{\beta} \bmod \mathsf{Carm}(N)$ has a solution for $d = 2^T$ by completeness, but does not have a solution for any $d \neq 2^T \bmod \mathsf{Carm}(N)$ by soundness (with high probability), which we explain next.

For simplicity, we will assume throughout this overview that the proof elements π_j potentially output by the prover are all equal to x^{z_j} for some $z_j \in \mathbb{Z}$. Then, for $y = x^{2^T}$ and all $i \in [n]$, completeness tells us that there must be a solution for $z_1, \ldots, z_k$ to the equation

$$\alpha_i + 2^T \cdot \beta_i + \sum_{j=1}^{k} \gamma_{i,j} \cdot z_j = 0 \bmod \mathsf{Carm}(N).$$

Since the prover's proof must work for all randomness values by completeness, we know that the prover's vector $\vec{z} = (z_1, \ldots, z_k)^\top$ actually satisfies

$$\Gamma \cdot \vec{z} = -\vec{\alpha} - 2^T \cdot \vec{\beta} \bmod \mathsf{Carm}(N).$$

However, for any $d \neq 2^T \bmod \mathsf{Carm}(N)$ corresponding to $x^d \neq x^{2^T}$, we use soundness to show that

$$\nexists \vec{z},\ \Gamma \cdot \vec{z} = -\vec{\alpha} - d \cdot \vec{\beta} \bmod \mathsf{Carm}(N),$$

as long as we sample enough vectors n. At a very high level, this follows since each newly sampled coefficient vector must restrict the space of solutions in a non-trivial way, since otherwise the same solution will work with good probability for many different choices of exponents. So we set n large enough such that, with high probability, the space of possible solutions for any $d \neq 2^T \bmod \mathsf{Carm}(N)$ is empty. The details of this argument are given in the full version of the paper.

Next, we prove a key technical lemma that allows us to relate whether or not a system of equations mod $\mathsf{Carm}(N)$ has a solution. Specifically, we show that there exists an efficiently computable matrix M that satisfies the following two properties:

1. If there exists a solution $\vec{z}$ such that $\Gamma \cdot \vec{z} = -\vec{\alpha} - d \cdot \vec{\beta} \bmod \mathsf{Carm}(N)$, then $M \cdot (-\vec{\alpha} - d \cdot \vec{\beta}) = \vec{0} \bmod \mathsf{Carm}(N)$.
2. If $M \cdot (-\vec{\alpha} - d \cdot \vec{\beta}) = \vec{0}$ over $\mathbb{Z}$, then there exists a solution $\vec{z}$ such that $\Gamma \cdot \vec{z} = (-\vec{\alpha} - d \cdot \vec{\beta})$ over $\mathbb{Z}$ (and hence $\bmod \mathsf{Carm}(N)$).

Furthermore, we show that $||M \cdot \vec{v}||_{\max} < 2^T$ when $||\vec{v}||_{\max}, ||\Gamma||_{\max} << 2^{T/(k+1)}$. When working over a field, such a result is well known by simply converting Γ into reduced row echelon form and the linear function M is closely related to the determinant of Γ. However, working over the integers mod $\mathsf{Carm}(N)$, this becomes much messier to work with. At a very high level, we show the lemma by first converting Γ to its Hermite normal form H, which is the integer counterpart to reduced row echelon form. We then augment the matrix H with the column $(-\vec{\alpha} - d \cdot \vec{\beta})$ and apply linear operations to zero out the last column to construct the matrix M. However, working over the integers, we must be careful to make sure that the values don't blow up in order to get our desired bound on $||M \cdot \vec{v}||_{\max}$. The full details for the proof of this technical lemma are provided in the full version of the paper.

Armed with our key technical lemma and the observations above, we are ready to complete the logic of our result, which follows the same high level structure as the deterministic case. Given M, we compute $\vec{v} = M \cdot (-\vec{\alpha} - 2^T \cdot \vec{\beta})$. By completeness, we know that there exists a vector $\vec{z}$ such that $\Gamma \cdot \vec{z} = (-\vec{\alpha} - d \cdot \vec{\beta}) \bmod \mathsf{Carm}(N)$, so by the technical lemma, we know that $\vec{v} = \vec{0} \bmod \mathsf{Carm}(N)$. We consider two different cases, either (1) $\vec{v} = \vec{0}$ over $\mathbb{Z}$ or (2) there exists an index i such that $\vec{v}_i = c \cdot \mathsf{Carm}(N)$ for $c \in \mathbb{Z}$. In case (2), we can factor given $\vec{v}_i$ using the variant of the Miller-Rabin primality test, so we are done.

For case (1), we use the fact that M is linear, so

$$\vec{v} = M \cdot (-\vec{\alpha} - 2^T \cdot \vec{\beta}) = -M \cdot \vec{\alpha} - 2^T \cdot M \cdot \vec{\beta} \bmod \mathsf{Carm}(N).$$

We consider two further subcases, either (1A) $M \cdot \vec{\beta} = \vec{0}$ over $\mathbb{Z}$ or (1B) there exists an index i such that $M_i \cdot \vec{\beta} \neq 0$. In case (1B), this implies that

$$2^T \leq 2^T \cdot |M_i \cdot \vec{\beta}| \leq |M_i \cdot \vec{\alpha}|,$$

but we show in our key technical lemma that $|M_i \cdot \vec{\alpha}| < 2^T$. So case (1B) cannot happen.

In case (1A) where $M \cdot \vec{\beta} = \vec{0}$, this actually implies that $M \cdot \vec{\alpha} = \vec{0}$ since we have already assumed that $\vec{v} = M \cdot (-\vec{\alpha} - 2^T \cdot \vec{\beta}) = \vec{0}$. But, this implies that $M \cdot (-\vec{\alpha} - d \cdot \vec{\beta}) = \vec{0}$ over $\mathbb{Z}$ for any $d \neq 2^T \bmod \mathsf{Carm}(N)$! So, by our key technical lemma, we conclude that there exists a solution over $\mathbb{Z}$, and hence mod $\mathsf{Carm}(N)$ for some $d \neq 2^T \bmod \mathsf{Carm}(N)$. However, we argued above that this cannot be the case by soundness (with high probability).

Combining the above, we've ruled out the possibility of case (1), so case (2) must hold, which implies we can factor with high probability. So, in summary, if (P, V) is a sound proof system where V is now a *randomized* generic group verifier and P sends at most k group elements in its proof, then either:

1. V runs in parallel time at least $T/(k+1)$ with high probability, or
2. there is a standard model factoring algorithm for N given the code of V and the string st output by P.

An alternative way to view this result is as follows. If V runs in parallel time less than $T/(k+1)$ with good probability, then either it must "know" a factorization of N to be able to reduce its exponents mod $\mathsf{Carm}(N)$, or there must be a cheating strategy that falsely convinces V on such randomness values. Hence, if you want both statistical security and an efficient verifier V, it must be the case that V can factor N.

Recursive interactive proofs. We next discuss how our result for one-round, MA-style, proofs extends to the class of recursive interactive proofs. First, we define what we mean by a r-round recursive interactive proof for the function $f_{N,T}(x) = x^{2^T} \bmod N$. In each round i, there is an input statement (x, y, T) claiming that $y = x^{2^T} \bmod N$. P starts the round by sending a string $\mathsf{st} \in \{0,1\}^*$ and up to k group elements $\pi_1, \ldots, \pi_k$. V then responds with a random string $\rho \leftarrow \{0,1\}^\lambda$. If i is the last round, V uses its randomness ρ and the message from P to decide whether or not $y = x^{2^T} \bmod N$. Otherwise, P and V both use a generic group algorithm A_i to compute a new statement (x', y', T') given the prover's message and the verifier's random coins, and they start a new independent (recursive) proof for this statement with one fewer round.

The overall running time of V is simply the running time of A_i in each round i, plus its final running time to compute its output at the end of the protocol. In addition to standard notions of completeness and soundness, we require that if (x, y, T) is valid at the beginning of the round, then (x', y', T') is also valid for the start of the next round. However, if (x, y, T) starts as invalid, so $y \neq x^{2^T} \bmod N$, then we require that (x', y', T') is invalid with probability at least $1 - \delta$.

Due to the recursive nature of this interactive proof, we are able to reduce to the one-round case to show that in each round T' cannot shrink too much relative to T, assuming A_i (and hence V) runs in low parallel time. If there exists a round i such that T' is much smaller than T, then we could construct a proof system $(\widehat{P}, \widehat{V})$ for $y = x^{2^T} \bmod N$ as follows. The prover $\widehat{P}$ sends whatever P would have sent in round i. Then, $\widehat{V}$ runs A_i to compute (x', y', T') and outputs $(x')^{2^{T'}} \cdot (y')^{-1}$. It follows that $\widehat{V}$ runs in time corresponding to the running time of A_i plus T', which is dominated by T'. By our result for one-round proofs, this means that T' must be at least $T/(k+1)$ with high probability, otherwise we can construct a factoring algorithm given the proof string st from P in round i. Hence, after $r-1$ rounds, the final time bound T' must be at least $T/(k+1)^{r-1}$ and V must run in parallel time at least $T/(k+1)^r$ to be a valid proof system.

In summary, if (P, V) is a *recursive*, generic group, r-round interactive proof for $f_{N,T}(x) = x^{2^T} \bmod N$, where the prover sends at most k group elements per round, then either:

1. V runs in parallel time at least $T/(k+1)^r$ with high probability, or
2. there is a standard model factoring algorithm for N given the code of V and the transcript generated by an honest prover P.

2 Generic Group Proof Systems

We next give the details for the generic group model we use in this work. Then we define proof systems where the verifier is restricted to generic group operations.

2.1 The Generic Group Model

In this work, we use Maurer's generic group model abstraction [28], following the related works of Aggarwal and Maurer [2] and Rotem and Segev [36]. We note that this is not the same as Shoup's random representation model [39]. See the work of Zhandry [42] for a detailed comparison between these two models.

Informally, a generic group algorithm is one that can perform arbitrary group operations as long as the operations performed are independent of the representation of the group elements. At a high level, we model this by giving the algorithm indirect access to its input group elements via pointers into a table, and each new multiplication or division adds a new element to the table and returns the corresponding pointer.

Formally, we consider the multiplicative group $\mathbb{Z}_N^\star$ in this work, where N is an RSA modulus in $\mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$ for some security parameter $\lambda \in \mathbb{N}$. A generic group algorithm A receives N as input as an explicit bit string and also receives access to a table Table via an oracle $\mathcal{O}$ that stores the group elements computed so far. Initially, Table contains the identity $v_0 = 1 \in \mathbb{Z}_N^\star$ at index 0, and all of the group elements $x_1, \ldots, x_k \in \mathbb{Z}_N^\star$ provided as input to A in indices $1, \ldots, k$. A can make queries to the oracle $\mathcal{O}$ via the following syntax:

- Mutliplication: On input $(i_1, i_2, j, \times)$, the oracle $\mathcal{O}$ checks that the values v_{i_1} and v_{i_2} at indices i_1 and i_2 in Table are non-empty and not $\perp$. If so, $\mathcal{O}$ computes $v_{i_1} \circ v_{i_2}$ and stores the result at index j in Table. Otherwise, $\mathcal{O}$ stores $\perp$ at index j.
- Division: On input $(i_1, i_2, j, \div)$, the oracle $\mathcal{O}$ additionally checks v_{i_2} is invertible. If so, $\mathcal{O}$ computes $v_{i_2}^{-1}$ and stores $v_i \times v_{i_2}^{-1}$ at index j in Table, if applicable. Otherwise, $\mathcal{O}$ stores $\perp$ at index j.

We note that Maurer's generic group model usually includes equality queries, which we do not handle in this work. An algorithm A that does not issue any equality queries is known as a *straight-line* algorithm, so for this reason, we state our formal results for straight-line generic group algorithms to avoid confusion. We note that generic-ring algorithms are defined similarly as above, but they also include addition and subtraction queries with essentially the same syntax.

For a group element g computed by A, we use $\widehat{g}$ to denote the pointer to the corresponding element g in the table Table. We abuse notation slightly and whenever we write that A receives a group element g as input, we mean that it receives a pointer $\widehat{g}$ to the element in the corresponding table Table.

We allow generic group algorithms to receive and output both "explicit" values, represented by bit strings, and "implicit" values indicating group elements, represented by pointers into Table. We can think of all of the explicit values as helping the generic algorithm decide how to invoke the oracle $\mathcal{O}$ to perform generic operations.

A randomized generic group algorithm also receives as input a string $\rho \in \{0,1\}^\lambda$ (we assume λ bits of randomness for simplicity, however this could be extended arbitrarily). For any input inp, We denote $A(1^\lambda, N, \mathsf{inp}; \rho)$ the randomized generic group algorithm with random tape ρ.

Measuring complexity. Let A be a generic group algorithm. We denote by $\mathsf{Time}_A(1^\lambda, N, \mathsf{inp}; \rho)$ the total running time of A on the given inputs with random tape ρ, where each oracle query costs a single unit of time. Additionally, we allow A to be a parallel algorithm. Following Rotem and Segev [36], we model parallel generic group algorithms A by allowing A to issue oracle queries in "rounds". In each round, A can issue any number of oracle queries to $\mathcal{O}$ in a single time step via multiple processors. We use $\mathsf{Width}_A(1^\lambda, N, \mathsf{inp}; \rho)$ to denote the maximum number of processors used by A at any time step and $\mathsf{ParTime}_A(1^\lambda, N, \vec{x}; \rho)$ to denote the number of sequential time steps that it takes for A to compute its output. Whenever we omit input/ randomness parameters from Time_A, Width_A, or $\mathsf{ParTime}_A$, we mean the worst case running time over an arbitrary choice of input parameters.

The behavior of generic group algorithms. Let $\lambda \in \mathbb{N}$ and $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$. Let A be a straight-line generic group algorithm such that $A(1^\lambda, N, \mathsf{st}, \vec{x}; \rho)$ takes as input an explicit string $\mathsf{st} \in \{0,1\}^*$ and group elements $\vec{x} = x_1, \ldots, x_k \in \mathbb{Z}_N^\star$ and outputs a group element g. As A is only allowed to perform generic operations, it follows that A's output is of the form $\prod_{i=1}^k x_i^{\gamma_i}$ for $\gamma_1, \ldots, \gamma_k \in \mathbb{Z}$. Furthermore, by running A, we can compute these coefficients by providing arbitrary pointers as input to A in place of $\vec{x}$. We use the notation

$\mathsf{coef}_{V,\lambda,N,\mathsf{st}}(\rho) = (\gamma_1, \ldots, \gamma_k)^\top$ to denote the coefficient vector of V on input ρ for security parameter λ, modulus N, and explicit string st. We note that the main distinction between our model and the strong algebraic group model of [23] is that they allow the coefficient vector to additionally depend on the bit representations of the input group elements.

Relating parallel running time to degree. Its easy to see that a straight-line generic group algorithm that computes $A(1^\lambda, N, \mathsf{st}, \vec{x}; \rho) = \prod_{i=1}^{k} x_i^{\gamma_i}$, where $\gamma_1, \ldots, \gamma_k$ are given by $\vec{\gamma} = \mathsf{coef}_{A,\lambda,N,\mathsf{st}}(\rho)$, must run in depth at least $\log ||\vec{\gamma}||_{\max}$. This can be shown by induction for $||\vec{\gamma}||_{\max}$ equal to 2^i for $i \geq 0$. If $||\vec{\gamma}||_{\max} = 2^0 = 1$, then it may be the case that A just immediately outputs a group element in 0 steps, satisfying the base case. Suppose that $||\vec{\gamma}||_{\max} = 2^i$. After $i-1$ steps, the maximum exponent in absolute value of any group element in Table is 2^{i-1} by assumption. So, in the next time step, A can issue a multiplication query multiplying two such elements together. However, this will result in an element with depth at most 2^i, as required. It follows that

$$\mathsf{ParTime}_A(1^\lambda, N, \mathsf{st}, \vec{x}; \rho) \geq \log ||\mathsf{coef}_{A,\lambda,N,\mathsf{st}}(\rho)||_{\max}$$

for all $\lambda \in \mathbb{N}$, $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$, string $\mathsf{st} \in \{0,1\}^*$, input elements $\vec{x}$, and random string $\rho \in \{0,1\}^\lambda$.

We additionally note that, even if we only require A to compute a high degree function with high probability and with pre-processing over a random input, then the same lower bound holds by the work of Rotem and Segev [36].

2.2 Proof Systems in the Generic Group Model

A proof system consists of two algorithms: the prover P and the verifier V. For a language L, P and V interact on common input x over potentially many rounds until V either accepts or rejects. In order to be non-trivial, the prover P must have some additional capabilities compared to the verifier V. For classical proof systems, the prover P is an unbounded algorithm while V is polynomially bounded. The two main properties of a proof system are completeness and soundness. Completeness stipulates that P convinces V on $x \in L$, and δ-soundness stipulates no cheating prover $P^\star$ can convince V on $x \notin L$ with probability better than δ.

We consider *generic group* proof systems for languages defined by a function f defined over a group $\mathbb{Z}_N^\star$ for $\lambda \in \mathbb{N}$ and $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$. For such proof systems, we restrict V to be a generic group algorithm that makes a bounded number of group multiplication and division queries, whereas P may still be an unbounded (not necessarily generic) algorithm that sends a bit string and group elements to V. So, for a function f, P and V receive an input a security parameter 1^λ, the group description N, an input group element x, and the output of the function $f(x)$ as common input. P sends a bit string $\mathsf{st} \in \{0,1\}^*$ and sequence of group elements $\pi_1, \ldots, \pi_k$ to V, which V receives access to via pointers into a table as a generic group algorithm. V then performs generic computations and outputs a pointer to a group element $\widehat{g}$ and "accepts" if the corresponding group element $g = 1$.

▶ **Definition 3** (Generic Group Proof Systems). *Let $\delta\colon \mathbb{N} \to [0,1]$ and $k\colon \mathbb{N} \to \mathbb{N}$. For any $\lambda \in \mathbb{N}$, $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$, let $f\colon \mathbb{Z}_N^\star \to \mathbb{Z}_N^\star$ be a function. We say that the pair (P, V) is a k-element generic group proof system for f with δ-soundness if V is a generic group algorithm, and for all $\lambda \in \mathbb{N}$, $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$, and $k = k(\lambda)$, the following hold:*

- *Completeness: For all* $x \in \mathbb{Z}_N^\star$*, let* $\mathsf{st}, \pi_1, \ldots, \pi_k$ *be the output of* $P(1^\lambda, N, x, f(x))$*, then it holds that*

$$V(1^\lambda, N, \mathsf{st}, x, f(x), \pi_1, \ldots, \pi_k) = 1.$$

- *Soundness: For all* $x \in \mathbb{Z}_N^\star$*,* $y \neq f(x)$*, and algorithms* $P^\star$ *such that* $P^\star(1^\lambda, N, x, y)$ *outputs a string* st *and group elements* $z_1, \ldots, z_k$*, it holds that*

$$\Pr_{\rho \leftarrow \{0,1\}^\lambda}\left[V(1^\lambda, N, \mathsf{st}, x, y, z_1, \ldots, z_k) = 1\right] \leq \delta(\lambda).$$

If the verifier V *is a straight-line algorithm, we say that* (P, V) *is a straight-line generic group proof system.*

3 One Round Proofs

In this section, we provide our main theorem. Let $\lambda \in \mathbb{N}$ and $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$. We show that if there is a k-element generic group proof system with a straight-line verifier that runs in parallel time less than $T/2(k+1)$ with probability ϵ, then there is a $\mathsf{poly}(1/\epsilon) \cdot \mathsf{Time}_V$ algorithm that factors N. We define some useful notation for the theorem first, and then provide a high level outline of the proof structure.

For each randomness ρ, let $\mathsf{coef}_{V,\lambda,N,\mathsf{st}}(\rho) = (\gamma_1, \ldots, \gamma_k, \alpha, \beta)^\top$ be the coefficients such that $V(1^\lambda, N, \mathsf{st}, x, y, z_1, \ldots, z_k)$ outputs $x^\alpha \cdot y^\beta \cdot \prod_{i=1}^k z_i^{\gamma_i}$. As V is a generic group algorithm, we can compute $\mathsf{coef}_{V,\lambda,N,\mathsf{st}}(\rho)$ by simply running $V(1^\lambda, N, \mathsf{st}, x, y, z_1, \ldots, z_k)$ for generic elements $x, y, z_1, \ldots, z_k$ and keep track of the operations of V. For notational convenience, when $V, \lambda, N, \mathsf{st}$ are clear from context, we simply write $\mathsf{coef}(\rho)$. We also define $\mathsf{dcoef}_{V,\lambda,N,\mathsf{st}}(\rho, d)$ to denote the vector $(\gamma_1, \ldots, \gamma_k, \alpha + d \cdot \beta)^\top$, where $(\gamma_1, \ldots, \gamma_k, \alpha, \beta)$ are given by $\mathsf{coef}(\rho)$, which will be useful in our analysis.

▶ **Theorem 4.** *Let* $\lambda \geq 2, T \in \mathbb{N}$*,* $k \colon \mathbb{N} \to \mathbb{N}$*,* $\delta, \epsilon \colon \mathbb{N} \to [0,1]$*,* $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$*, and* (P, V) *be a* k*-element straight-line generic-group proof system for the function* $f_{N,T}(x) = x^{2^T} \bmod N$ *with soundness error* δ*.*

Let $x \in \mathbb{Z}_N^\star$ *and* $(\mathsf{st}, \pi_1, \ldots, \pi_{k(\lambda)}) \in \mathrm{Supp}\left(P(1^\lambda, N, T, x, f_{N,T}(x))\right)$*. If*

$$\Pr_\rho\left[\mathsf{ParTime}_V(1^\lambda, N, T, \mathsf{st}) < \frac{T}{k(\lambda)+1} - \log(k(\lambda))\right] \geq \max(2\delta(\lambda), \epsilon(\lambda)),$$

then there exists a standard model probabilistic $\mathsf{poly}(\lambda, k(\lambda), T, 1/\epsilon(\lambda)) \cdot \mathsf{Time}_V(1^\lambda, N, \mathsf{st})$ *time algorithm* A *such that*

$$\Pr\left[p, q \leftarrow A\left(1^\lambda, N, k, T, \mathsf{st}, 1/\epsilon(\lambda)\right) : N = p \cdot q\right] \geq 1 - 2^{-\lambda}.$$

We refer the reader to the full version for the proof of the theorem.

References

1 Chia network. `https://chia.net/`. Accessed: 2022-10-05.

2 Divesh Aggarwal and Ueli Maurer. Breaking RSA generically is equivalent to factoring. *IEEE Trans. Inf. Theory*, 62(11):6251–6259, 2016.

3 Nir Bitansky, Arka Rai Choudhuri, Justin Holmgren, Chethan Kamath, Alex Lombardi, Omer Paneth, and Ron D. Rothblum. Ppad is as hard as lwe and iterated squaring. In *TCC*, 2022.

4 Alexander R. Block, Justin Holmgren, Alon Rosen, Ron D. Rothblum, and Pratik Soni. Time- and space-efficient arguments from groups of unknown order. In *Advances in Cryptology - CRYPTO*, pages 123–152, 2021.

5 Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Advances in Cryptology - CRYPTO*, pages 757–788, 2018.

6 Dan Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. *IACR Cryptol. ePrint Arch.*, page 712, 2018.

7 Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to iops and stateless blockchains. In *CRYPTO (1)*, volume 11692 of *Lecture Notes in Computer Science*, pages 561–586. Springer, 2019.

8 Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Sumcheck arguments and their applications. In *CRYPTO (1)*, volume 12825 of *Lecture Notes in Computer Science*, pages 742–773. Springer, 2021.

9 Johannes Buchmann and Safuat Hamdy. A survey on iq cryptography. In *Public-Key Cryptography and Computational Number Theory*, pages 1–15, 2001.

10 Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from DARK compilers. In *Advances in Cryptology - EUROCRYPT*, pages 677–706, 2020.

11 Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous verifiable delay functions. In *Advances in Cryptology - EUROCRYPT*, pages 125–154, 2020.

12 Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO*, pages 186–194, 1986.

13 Cody Freitag, Ilan Komargodski, Rafael Pass, and Naomi Sirkin. Non-malleable time-lock puzzles and applications. In *Theory of Cryptography - 19th International Conference, TCC*, pages 447–479, 2021.

14 Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In *Advances in Cryptology - CRYPTO*, pages 33–62, 2018.

15 Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4):205–214, 1998.

16 Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Comput. Complex.*, 11(1-2):1–53, 2002.

17 Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4):27:1–27:64, 2015.

18 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. URL: `http://www.jstor.org/stable/2282952`.

19 Charlotte Hoffmann, Pavel Hubácek, Chethan Kamath, Karen Klein, and Krzysztof Pietrzak. Practical statistically-sound proofs of exponentiation in any group. In *CRYPTO (2)*, 2022.

20 Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. Fiat-shamir via list-recoverable codes (or: parallel repetition of GMW is not zero-knowledge). In *STOC*, pages 750–760. ACM, 2021.

21 Tibor Jager and Jörg Schwenk. On the analysis of cryptographic assumptions in the generic ring model. *J. Cryptol.*, 26(2):225–245, 2013.

22 Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Yun Zhang. Snargs for bounded depth computations and PPAD hardness from sub-exponential LWE. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 708–721, 2021.

23 Jonathan Katz, Julian Loss, and Jiayu Xu. On the security of time-lock puzzles and timed commitments. In *Theory of Cryptography – TCC*, pages 390–413, 2020.

24 Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, STOC*, pages 723–732, 1992.

25 Swastik Kopparty and Abhishek Bhrushundi. Lecture 3: Finding integer solutions to systems of linear equations, fall 2014. URL: `https://sites.math.rutgers.edu/~sk1233/courses/ANT-F14/lec3.pdf`.

26 Alex Lombardi and Vinod Vaikuntanathan. Fiat-shamir for repeated squaring with applications to ppad-hardness and vdfs. In *Advances in Cryptology - CRYPTO*, pages 632–651, 2020.

27 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

28 Ueli M. Maurer. Abstract models of computation in cryptography. In *IMACC*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.

29 Gary L Miller. Riemann's hypothesis and tests for primality. *Journal of computer and system sciences*, 13(3):300–317, 1976.

30 Krzysztof Pietrzak. Simple verifiable delay functions. In *10th Innovations in Theoretical Computer Science Conference, ITCS*, pages 60:1–60:15, 2019.

31 Michael O Rabin. Probabilistic algorithm for testing primality. *Journal of number theory*, 12(1):128–138, 1980.

32 Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. *SIAM J. Comput.*, 50(3), 2021.

33 Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. Technical report, Massachusetts Institute of Technology. Laboratory for Computer Science, 1996.

34 Lior Rotem. Simple and efficient batch verification techniques for verifiable delay functions. In *TCC (3)*, volume 13044 of *Lecture Notes in Computer Science*, pages 382–414. Springer, 2021.

35 Lior Rotem. Revisiting the uber assumption in the algebraic group model: Fine-grained bounds in hidden-order groups and improved reductions in bilinear groups. In *ITC*, volume 230 of *LIPIcs*, pages 13:1–13:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

36 Lior Rotem and Gil Segev. Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. In *Advances in Cryptology - CRYPTO*, pages 481–509, 2020.

37 Lior Rotem, Gil Segev, and Ido Shahaf. Generic-group delay functions require hidden-order groups. In *Advances in Cryptology - EUROCRYPT*, pages 155–180, 2020.

38 Adi Shamir. Ip=pspace. In *31st Annual Symposium on Foundations of Computer Science, FOCS*, pages 11–15, 1990.

39 Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.

40 Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge University Press, 2006.

41 Benjamin Wesolowski. Efficient verifiable delay functions. *J. Cryptol.*, 33(4):2113–2147, 2020.

42 Mark Zhandry. To label, or not to label (in generic groups). *IACR Cryptol. ePrint Arch.*, page 226, 2022.

A Preliminaries

For any $n \in \mathbb{N}$, we use $[n] = \{1, \ldots, n\}$ to denote the set from 1 to n. For a distribution X, we denote by $x \leftarrow X$ the process of sampling a value x from the distribtion X. For a set $\mathcal{X}$, we use $x \leftarrow \mathcal{X}$ to denote the process of sampling a value x from the uniform distribution over $\mathcal{X}$. For a bit string $\mathsf{st} \in \{0,1\}^*$, we use $|\mathsf{st}|$ to denote the length of st. Throughout, we use $\lambda \in \mathbb{N}$ to denote the security parameter.

A.1 Number Theory

In this work, we consider the multiplicative group of integers mod N, denoted by $\mathbb{Z}_N^\star$, where N is a product of two primes. Specifically, for any $\lambda \in \mathbb{N}$, we let $\mathsf{ModGen}(1^\lambda)$ denote the

algorithm that samples two random primes p, q in the interval $[2^\lambda, 2^{\lambda+1})$ and outputs $N = p \cdot q$. The group is given by $\mathbb{Z}_N^\star = \{x \in [1, N) : \gcd(x, N) = 1\}$, and multiplication in the group corresponds to multiplication over $\mathbb{Z}$ mod N. When it is clear from context we are working in the group $\mathbb{Z}_N^\star$, we will omit mod N when discussing multiplication of group elements.

The main language we consider in this work is the repeated squaring relation, $\mathcal{RS}_N$, defined as follows

$$\mathcal{RS}_N = \left\{(x, y, T) \mid y = x^{2^T} \bmod N\right\}.$$

For a particular value of N and T, we represent this relation by the function $f_{N,T}(x) = x^{2^T} \bmod N$. It is widely believed that $f_{N,T}$ cannot be computed and $\mathcal{RS}_N$ cannot be decided in depth less than T even with $\mathsf{poly}(\lambda, T)$ parallel processors. We focus on the proof complexity of this language in this work.

For any $a, b \in \mathbb{Z}$, we use $\gcd(a, b)$ and $\mathrm{lcm}(a, b)$ to denote the greatest common divisor and least common multiple of a and b, respectively. Specifically, $\gcd(a, b)$ is the maximal $c \in \mathbb{N}$ such that c divides a and b, and lcm is the minimal $c \in \mathbb{N}$ such that a and b both divide c. Let $a, b \in \mathbb{Z}$, then there always exist integers c, d such that $c \cdot a + d \cdot b = \gcd(a, b)$. c and d are known as Bezout coefficients for a and b. While Bezout coefficients may not be unique, we note that there always exist bezout coefficients such that $|c|, |d| \leq \max(|a|, |b|)$, and these are the coefficients given by the standard euclidean algorithm.

We denote by $\varphi(N) = |\mathbb{Z}_N^\star|$, known as the Euler totient function of N, and $\mathsf{Carm}(N) = \min\{a \in \mathbb{N} : \forall g \in \mathbb{Z}_N^\star, g^a = 1\}$, known as the Carmichael totient function. For $\lambda \in \mathbb{N}$ and $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$ such that $N = p \cdot q$, it holds that

$$\varphi(N) = (p-1) \cdot (q-1), \text{ and } \mathsf{Carm}(N) = \mathrm{lcm}(p-1, q-1).$$

For a specific element $g \in \mathbb{Z}_N^\star$, we define the order of g, $\mathsf{ord}(g)$, to be the minimum $c \in \mathbb{N}$ such that $g^c = 1 \in \mathbb{Z}_N^\star$.

In this work, we use the fact that for $N = p \cdot q$, $\mathbb{Z}_N^\star \cong \mathbb{Z}_p^\star \times \mathbb{Z}_q^\star$, where $\mathbb{Z}_p^\star$ and $\mathbb{Z}_q^\star$ are each cyclic groups of order $\varphi(p) = p - 1$ and $\varphi(q) = q - 1$, respectively. Let g_p and g_q be generators for the corresponding subgroups. Then, we can write any group element $h \in \mathbb{Z}_N^\star$ in the form $h = g_p^a \cdot g_q^b$ for some $a, b \in \mathbb{N}$. For convenience of notation, we will use $h|_p$ to denote the p "component" of h and $h|_q$ to denote the q component, so $a = h|_p$ and $b = h|_q$ above.

In order to translate between results mod a composite number Φ and its solutions mod its prime power divisors, we make use of the Chinese remainder theorem (CRT). We use the following version of CRT.

▶ **Lemma 5.** *Let $k \in \mathbb{N}$, $n_1, \ldots, n_k, a_1, \ldots, a_k \in \mathbb{N}$. Then, the set of equations*

$$x = a_i \bmod n_i$$

has a solution over $\mathbb{Z}$ if and only if for all $i, j \in [k]$, $a_i = a_j \bmod \gcd(n_i, n_j)$. Moreover, any two solutions x_1, x_2 satisfy $x_1 = x_2 \bmod \mathrm{lcm}(n_1, \ldots, n_k)$.

The following lemma, based on the Miller-Rabin primality test [29, 31], gives a probabilistic factoring algorithm given any non-zero multiple of $\mathsf{Carm}(N)$. For the proof of the lemma and further discussion, we refer the reader to Section 10.4 of Shoup [40].

▶ **Lemma 6** (Factoring Lemma)**.** *Let $\lambda \in \mathbb{N}$, $N \in \mathrm{Supp}\left(\mathsf{ModGen}(1^\lambda)\right)$, and $m = c \cdot \mathsf{Carm}(N)$ for $c \in \mathbb{Z}$ such that $c \neq 0$. For any $\delta \colon \mathbb{N} \to [0, 1]$, there exists a probabilistic algorithm A that runs in $\mathsf{poly}(\lambda, \log(1/\delta(\lambda)))$ time such that*

$$\Pr\left[p, q \leftarrow A(1^\lambda, N, m) : N = p \cdot q\right] \geq 1 - \delta(\lambda).$$

A.2 Linear Algebra

Let M be a matrix in $\mathbb{Z}^{m\times\ell}$. For $i\in[m]$, $j\in[\ell]$, we use M_i to denote the ith row and $M_{i,j}$ to denote the element in the ith row and jth column. We use $M^\top$ to denote the transpose of a matrix. We treat vectors $\vec{v}\in\mathbb{Z}^n$ as column vectors, so implicitly of the form $\vec{v}\in\mathbb{Z}^{n\times 1}$. To take the dot product of two vectors $\vec{v},\vec{u}$, we write $\vec{v}^\top\cdot\vec{u}$. If $v\in\mathbb{Z}^{m\times 1}$ is a vector, we simply write v_i to denote the ith component. We write $||M||_{\max}=\max_{i\in[m],j\in[\ell]}|M_{i,j}|$ to denote the largest element in absolute value in the matrix M. For a matrix $M^{(1)}\in\mathbb{Z}^{m\times\ell_1}$ and a matrix $M^{(2)}\in\mathbb{Z}^{m\times\ell_2}$, we write $M'=(M^{(1)}|M^{(2)})$ to denote the augmented matrix which appends $M^{(2)}$ to the right of $M^{(1)}$ to get the matrix $M'\in\mathbb{Z}^{m\times(\ell_1+\ell_2)}$.

For any composite Φ, let $\mathbb{Z}_\Phi$ be the ring of integers mod Φ. We say that a function $f\colon\mathbb{Z}_\Phi^n\to\mathbb{Z}_\Phi^n$ is linear if for any vectors $\vec{g},\vec{h}\in\mathbb{Z}_\Phi$ and $a,b\in\mathbb{Z}$, it satisfies $f(a\cdot\vec{g}+b\cdot\vec{h})=a\cdot f(\vec{g})+b\cdot f(\vec{h})$. We say that a function f is affine if there exists some matrix M such that $f(\vec{g})=M\cdot\vec{g}'$ where $\vec{g}'$ is equal to $\vec{g}$ appended by 1. In particular, this means that f is a linear function shifted by a constant.

Let $\mathsf{Perm}(n)$ denote the set of all permutations over $[n]$. For a permutation $\sigma\in\mathsf{Perm}(n)$, we write $\mathsf{sign}(\sigma)$ to denote the sign of σ, i.e. 1 if there are an even number transpositions from the identity to σ, and -1 otherwise. For a square matrix M, the determinant of M is given by $\det(M)=\sum_{\sigma\in\mathsf{Perm}(n)}\mathsf{sign}(\sigma)\cdot\prod_{i=1}^n M_{i,\sigma(i)}$. It follows by definition of the determinant that $\det(M)\le n!\cdot||M||_{\max}^n$. We say that an integer matrix $U\in\mathbb{Z}^{m\times m}$ is unimodular if $\det(U)\in\{+1,-1\}$.

Let $\vec{v}^{(1)},\ldots,\vec{v}^{(n)}\in\mathbb{Z}^m$ be a set of vectors. This determines a lattice

$$\mathcal{L}=\mathcal{L}\left(\vec{v}^{(1)},\ldots,\vec{v}^{(n)}\right)=\left\{\sum_{i=1}^m c_i\cdot\vec{v}^{(i)}:c_1,\ldots,c_m\in\mathbb{Z}\right\}$$

of points spanned by these vectors. For a lattice $\mathcal{L}$, we refer to a basis of the lattice as a set of vectors $\vec{b}^{(1)},\ldots,\vec{b}^{(m)}$, often written in matrix matrix $B=(\vec{b}^{(1)}|\ldots|\vec{b}^{(m)})$, that are linearly independent over $\mathbb{R}$ and $\mathcal{L}=\mathcal{L}(B)$. A lattice is unique up to multiplication of B by a unimodular matrix U, so when the basis is clear from context, we refer simply to the lattice $\mathcal{L}$. The determinant of a lattice $\det(\mathcal{L})$ is defined to be the volume of the parallelepiped formed by a set of basis vectors over $\mathbb{R}^m$.

We next define the Hermite normal form (HNF) of an integer matrix $M\in\mathbb{Z}^{m\times n}$. We use the notion of column-style HNF, defined via right multiplication by a unimodular matrix, in contrast to row-style HNF.

▶ **Definition 7** (Hermite Normal Form). *A matrix $H\in\mathbb{Z}^{m\times n}$ is in Hermite normal form if the following hold:*

1. *Lower triangular: For some $h\le n$, there exists a sequence $1\le i_1<i_2<\ldots<i_h\le n$ such that $H_{i,j}\neq 0\Rightarrow i>i_j$.*
2. *Row-reduced: For all $k\le j\le n$, $0\le H_{i_j,k}\le H_{i_j,j}$.*

We additionally use the fact that the HNF of a matrix $M\in\mathbb{Z}^{m\times n}$ has entries bounded by $||M||_{\max}^n$. See [25] for a proof of this claim.

When working over a field $\mathbb{F}$, such as the integers mod a prime p or the rationals $\mathbb{Q}$, we can define standard notions like span and rank. The span of a set over vectors over an n dimensional vector space over a field $\mathbb{F}$ is defined as the set of all linear combinations of the vectors, with coefficients from the field $\mathbb{F}$. When clear from context, we use span in the context of integers to refer to the set of linear combinations with coefficients from $\mathbb{Z}$, as in the definition of a lattice. The rank of a matrix or vector space over a field $\mathbb{F}$ is the size of the minimal set of vectors that spans the space over $\mathbb{F}$.

A.3 Concentration Inequalities

Concentration inequalities allow us to bound the probability that certain random variables take values too far away from their mean. In this work, we use the following version of the well known Chernoff-Hoeffding bound [18].

▶ **Lemma 8** (Chernoff-Hoeffding Bound [18]). *Let $X = \sum_{i=1}^{m} X_i$ such that $X_i \in [0,1]$ are independent random variables. Let $\mu = \mathrm{E}[X]$. Then, for all t,*

$$\Pr[|X - \mu| > t] \leq 2e^{-2t^2/m}.$$

B Existing Proofs for RS

We give a brief overview of the currently known proof systems for $\mathcal{RS}_N$, focusing on the practical setting of non-interactive proofs. When discussing the proofs below, we use λ to denote the security parameter. Informally, we say that a verifier is efficient if it runs in time $\mathrm{poly}(\lambda, \log T)$, essentially independent of the time bound T.

- **The empty proof.** The prover can always do nothing and let the verifier check the relation $y = x^{2^T} \bmod N$ itself.
  This is a valid, albeit not very helpful, proof system that is perfectly complete and sound. In terms of efficiency, the verifier runs in time T to compute T squares, so nothing has been gained.
- **The factoring proof.** The prover can factor N to get primes p, q where $N = p \cdot q$ and send (p, q) to the verifier. The verifier can check that indeed $N = p \cdot q$, compute the order of the group $\varphi(N)$, and check if $y = x^{2^T \bmod \phi(N)} \bmod N$.
  This protocol is extremely efficient for the verifier, and is perfectly complete and sound. However, such a proof disallows N to be reused again since RS is not a sequential function whenever p, q are known. Furthermore, unless P generated N itself, it requires an inefficient prover.
- **Sumcheck-style proofs.** This is a general proof style that follows the structure of the sumcheck protocol of Lund, Fortnow, Karloff, and Nisan [27]. The main idea is that the prover first splits the statement (x, y, T) into $k \geq 2$ sub-statements (x_i, y_i, T') for $i \in [k]$ for $T' < T$. Then, the verifier uses its randomness to merge these sub-statements into a single statement (x', y', T') which is hopefully easier to handle. Such protocols naturally lend themselves to recursive interactive proofs. We note that the proofs of [30, 11, 4, 19] as well as a generic proof for space-bounded computation [32] generally fall into this framework. We focus on Pietrzak's protocol [30] as it is the simplest and is specifically tailored for $\mathcal{RS}_N$.
  In the proof of [30], the prover sends a midpoint $\mu = x^{2^{T/2}}$, which induces two sub-statements $(x, \mu, T/2)$ and $(\mu, y, T/2)$. The verifier samples a random exponent $r \leftarrow [2^\lambda]$, and computes a new statement $(x', y', T/2)$ where $x' = x^r \cdot \mu$, $y' = \mu^r \cdot y$, and $T' = T/2$. In the non-interactive setting, the verifier can then simply check $(x')^{2^{T/2}} = y'$ itself.
  In terms of efficiency, this protocol only cuts down the running time of the verifier by a factor of 2. [11] show how to reduce this to an $T/k + 1$-time verifier for any $k \geq 0$ by having the prover sending k evenly spaced midpoints.
  It's easy to see that if (x, y, T) is valid, then so is (x', y', T'). Soundness follows since if (x, y, T) is invalid, then (x', y', T') becomes valid only with probability at most $O(1/s)$, where s is the size of the smallest subgroup of $\mathbb{Z}_N^\star$. If N is a product of safe primes, then $s = 2^\lambda$, and the protocol is statistically sound. Block et al. [4] show how to adapt this protocol, at the cost of $O(\lambda)$ multiplicative overhead in communication, to be statistically sound for *any* multiplicative group.

- **FS-style arguments.** We can get non-interactive proofs by applying the Fiat-Shamir (FS) heuristic [12] to the public-coin, interactive variants of the sumcheck-style proofs above. Again, we focus on the protocol of Pietrzak [30] for sake of comparison.
  The FS heuristic generates the verifier's randomness in each round by applying a (sufficiently random) hash function on the transcript of the protocol so far. Hence, the prover can generate all of its messages without needing to interact with the verifier, resulting in a non-interactive proof.
  In the case of [30], the prover generates an initial midpoint $\mu_1 = x^{2^{T/2}}$, then hashes μ_1 (along with the statement) to get a random value $r_1 \in [2^\lambda]$. The prover can then compute $(x', y', T/2)$ itself as above. At this point, P compute a second midpoint $\mu_2 = (x')^{2^{T/4}}$, generate randomness r_2 using the hash function, and continue this process r times until it generates a statement $(\hat{x}, \hat{y}, T')$ where $T' = T/2^r$ that the verifier can check directly. If $r = \log T$, then $T' = O(1)$, resulting in an efficient verifier. The prover needs to send r group elements in this protocol, so this requires $\Omega(\log T \cdot \lambda)$-bits of communication in total.
  In terms of security, we note that, even when modeling the hash function h as a random function, the resulting protocols are only computationally sound. An unbounded prover that can query the random oracle arbitrarily can generate cheating proofs for false statements. However, there is a recent line of work (see e.g. [26, 22, 3]) showing how to securely instantiate hash functions for different sumcheck-style protocols from more standard assumptions. Most relevant to us is the work of Bitansky et al. [3] that instantiates the FS-heuristic for the interactive proof of [4] for $\mathcal{RS}_N$ assuming only (polynomially hardness) LWE using the hash function of [20].
- **Wesolowski's argument.** Wesolowski [41] gave an extremely efficient non-interactive proof for $\mathcal{RS}_N$ where the prover sends a single group element and the verifier computes only $O(\lambda)$ squares. In this protocol, the verifier first samples a random λ-bit prime ℓ (or is sampled using a random function as in the FS-heuristic), and the prover sends an ℓth root of y, $\pi = x^{\lfloor 2^T/\ell \rfloor}$. The verifier then accepts iff $y = \pi^\ell \cdot x^c$ for $c = 2^T \bmod \ell$.
  The computational soundness of this protocol relies on a new "adaptive root assumption", which says that the prover cannot compute an ℓth root of a group element for a random prime ℓ. Aside from this assumption being relatively new, this protocol is *broken* if the prover knows the factorization of N. Namely, given the order of the group, the prover can break the adaptive root assumption. This means that the protocol additionally requires a strong assumption on the setup used to generate N as well. We note that this style of assumption is not required for the computational soundness for the FS-style arguments mentioned above.

C Extension to General Hidden Order Groups

Let $\mathbb{G}$ be any finite, abelian, multiplicative group. For any $\lambda \in \mathbb{N}$, we let $\mathsf{GroupGen}(1^\lambda)$ be an algorithm that outputs some group of size $[2^\lambda, 2^\lambda + 1)$ such that it is believe that it is hard to compute the order of a random group $\mathbb{G} \leftarrow \mathsf{GroupGen}(1^\lambda)$. Any such group $\mathbb{G}$ must be finitely generated, so there exist elements $g_1, \ldots, g_s$ such that every $h \in \mathbb{Z}_N^\star$ is equal to $\prod_{i=1}^s g_i^{h|_i}$, where $h|_i \in \mathbb{Z}$ is the ith component of h. We use $\mathsf{ord}(\mathbb{G})$ to denote the size of the group, and $\mathsf{ord}(g)$ to denote the minimum c such that $g^c = 1$. Borrowing notation from $\mathbb{Z}_N^\star$, we use $\mathsf{Carm}(\mathbb{G})$ to denote the maximum value of $\mathsf{ord}(g)$ for any $g \in \mathbb{G}$. In particular, there must exist some $g \in \mathbb{G}$ such that $\mathsf{ord}(g) = \mathsf{Carm}(\mathbb{G})$.

The proof of Theorem 4 goes through by considering an arbitrary group $\mathbb{G}$. We refer to the full version of the paper for more details.

▶ **Corollary 9.** *Let* $\lambda \geq 2, T \in \mathbb{N}$, $k\colon \mathbb{N} \to \mathbb{N}$, $\delta, \epsilon\colon \mathbb{N} \to [0,1]$, $\mathbb{G} \in \mathrm{Supp}\left(\mathsf{GroupGen}(1^\lambda)\right)$, *and* (P,V) *be a* k*-element straight-line generic-group proof system for the function* $f_{\mathbb{G},T}(x) = x^{2^T} \in \mathbb{G}$ *with soundness error* δ. *For any* $(\mathsf{st}, \pi_1, \ldots, \pi_{k(\lambda)}) \in \mathrm{Supp}\left(P(1^\lambda, \mathbb{G}, T, x, f_{\mathbb{G},T}(x))\right)$. *If*

$$\Pr_{\rho}\left[\mathsf{ParTime}_V(1^\lambda, \mathbb{G}, T, \mathsf{st}) < \frac{T}{(k(\lambda)+1)} - \log(k(\lambda)+1)\right] \geq \max(2\delta(\lambda), \epsilon(\lambda)),$$

then there exists a standard model probabilistic $\mathsf{poly}(\lambda, k(\lambda), T, 1/\epsilon(\lambda)) \cdot \mathsf{Time}_V(1^\lambda, \mathbb{G}, \mathsf{st})$ *time algorithm* A *such that*

$$\Pr\left[c \leftarrow A\left(1^\lambda, \mathbb{G}, k, T, \mathsf{st}, 1/\epsilon(\lambda)\right) : \mathsf{ord}(\mathbb{G}) \text{ divides } c, c \neq 0\right] \geq 1 - 2^{-\lambda}.$$

Interactive Non-Malleable Codes Against Desynchronizing Attacks in the Multi-Party Setting

Nils Fleischhacker
Ruhr-Universität Bochum, Germany

Suparno Ghoshal
Ruhr-Universität Bochum, Germany

Mark Simkin
Ethereum Foundation, Aarhus, Denmark

Abstract

Interactive Non-Malleable Codes were introduced by Fleischhacker et al. (TCC 2019) in the two party setting with synchronous tampering. The idea of this type of non-malleable code is that it "encodes" an interactive protocol in such a way that, even if the messages are tampered with according to some class $\mathcal{F}$ of tampering functions, the result of the execution will either be correct, or completely unrelated to the inputs of the participating parties. In the synchronous setting the adversary is able to *modify* the messages being exchanged but cannot drop messages nor desynchronize the two parties by first running the protocol with the first party and then with the second party. In this work, we define interactive non-malleable codes in the non-synchronous multi-party setting and construct such interactive non-malleable codes for the class $\mathcal{F}^s_{\mathsf{bounded}}$ of bounded-state tampering functions.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols; Mathematics of computing → Coding theory

Keywords and phrases non-malleability, multi-party protocols

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.5

Funding Nils Fleischhacker and Suparno Ghoshal were supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CASA – 390781972.

1 Introduction

There is a long line of research that aims to make communication resilient to tampering, starting with error correcting codes. Error correcting codes allow a sender to encode a message m into a codeword c, such that a receiver can always recover the message m even from a tampered codeword c' as long as the tampering is done in some restricted way. Specifically, the class of tampering functions tolerated by traditional error correcting codes are those that erase or modify at most a constant fraction of the symbols in codeword c. If the tampering function, however, behaves in any other way, there is no longer any guarantee on the output of the decoding algorithm. Error *detecting* codes are a relaxation that allows the decoder to also output a special symbol $\perp$ when m is not recoverable from c'. But these codes, again, cannot tolerate, i.e. will decode incorrectly when tampered with, many simple tampering functions such as a constant function.

Dziembowski, Pietrzak, and Wichs [30] introduced a further relaxation which they called non-malleable codes (NMC). Very informally, an encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ is an NMC for a class of tampering functions, $\mathcal{F}$, if the following holds: given a tampered codeword $c' = f(\mathsf{Enc}(m))$ for some $f \in \mathcal{F}$, the decoded message $m' = \mathsf{Dec}(c')$ is either the original message m or *completely unrelated* to m. I.e., the tampering function can only "destroy" the

© Nils Fleischhacker, Suparno Ghoshal, and Mark Simkin;
licensed under Creative Commons License CC-BY 4.0

information being transferred, but not modify it in a meaningful way. Obviously, NMCs can still not exist for the set of *all* tampering functions $\mathcal{F}_{\mathsf{all}}$. To see this, consider the tampering function that retrieves $m = \mathsf{Dec}(c)$, chooses a message m' related to m and encodes $c' = \mathsf{Enc}(m')$. This tampering trivially defeats the requirement above. In light of this observation, a rich line of works has dealt with constructing non-malleable codes for different classes of tampering attacks (see Section 1.3 for a discussion).

Non-malleable codes have the obvious advantage that we can obtain meaningful guarantees for larger classes of tampering functions (compared to error correcting codes) and they have also found a number of interesting applications in cryptography such as tamper-resilient cryptography [30, 46, 33, 34]. They have also been useful as a building block in constructing non-malleable encryption [23], round optimal non-malleable commitments [39], and non-malleable secret sharing schemes [37, 38, 7].

Interactive Coding

Traditional codes, whether error correcting, error detecting or non-malleable are only concerned with the scenario where a sender sends a single message to a receiver. Interactive Coding, introduced by Schulman [49, 50, 51], generalizes (error correcting) codes to arbitrary interactive protocols between two or more [48] parties. Consider the following scenario: n parties, each with their own input, are running an interactive protocol to perform some task involving their inputs, such as computing a joint function on them. Now, say an adversary can get access to their communication channels and tamper with the messages being sent in the protocol. An interactive code for a class of tampering functions $\mathcal{F}$ is essentially a wrapper around the protocol that would guarantee that, as long as the tampering is performed using a function $f \in \mathcal{F}$, the protocol will conclude correctly and all participants will be able to recover their correct output.

Interactive Non-Malleable Codes

In interactive coding, just as in the case of error correcting codes, there are strong limits on which classes of tampering can be dealt with. To achieve meaningful guarantees for larger classes of tampering functions Fleischhacker et al. [36] introduced the notion of interactive *non-malleable* codes (INMC). Just as interactive coding generalizes error correcting codes, INMCs generalize NMCs, by encoding "active communication" instead of "passive data". An INMC is supposed to give a similar guarantee as an NMC. Informally, that means that the participants of the protocol should either be able to recover the correct output from the protocol or the correct output would be completely "destroyed" and the participants would recover something completely unrelated to their inputs. Fleischhacker et al. in fact define two seperate notion of non-malleability, weak non-malleability only requires the outputs to be unrelated from *other* parties' inputs, while *strong* non-malleability requires the outputs to be unrelated even to the party's own input. We are only interested in the *strong* non-malleability notion, which we will simply call non-malleability. It turns out that strong non-malleability somewhat counterintuitively actually implies error detection in the interactive case.

Fleischhacker et al. [36] define three classes of tampering functions, bounded-state tampering, a variation of split-state tampering, and sliding-window tampering. For each class they give a construction of a strongly non-malleable INMC.

However, both the definitions and the constructions are limited, because they only apply to the two party setting and they only consider *synchronous* tampering. Consider a protocol between two parties, Alice and Bob. In the synchronous setting, when Alice sends a message

m to Bob, the tampering function can arbitrarily *modify* m, but it *must* then forward it to Bob without further delay. I.e., at all times, even in a tampered execution of the protocol, Alice and Bob remain in sync and the tampering function can *not* choose to, e.g., first finish the protocol with Alice, before later resuming the communication with Bob.

When considering desynchronization between parties in an interactive protocol we need to consider how protocols are modeled. As pointed out by Braverman et al. [13] there are essentially two paradigms for protocols without fully synchronized parties. In a *clock-driven* model, each party has a clock and wakes up with each clock tick, checks for incoming messages, performs some computation, and potentially send messages to the other parties. Here desynchronization can occur because the different parties' clocks may be mismatched or skewed. In a *message-driven* model, the parties sleep until they receive a new message, which triggers them to wake up and perform some action. Here desynchronization can occur because messages are dropped causing one party to be ahead in the protocol. The model for *desynchronization* considered in this paper is of the latter kind. I.e., parties in our model are purely activated by receiving messages and have no sense of time, i.e., they do not notice how long they may have been asleep. This strengthens the possible attacker, since it allows for even the most extreme forms of desynchronization.

1.1 Results and Technical Overview

In this work, we aim to remedy the shortcomings of the previous work by Fleischhacker et al. [36]. In Section 3 we introduce new definitions for arbitrary (potentially desynchronizing) tampering with interactive protocols between $n \geq 2$ parties, define interactive non-malleability and formalize the class of bounded-state tampering functions. In Section 5 we construct an INMC for bounded-state tampering functions.

The "Obvious" Solution

When faced with the task of constructing an interactive non-malleable code it may seem tempting to directly apply the huge body of work around regular non-malleable codes and try to build an INMC by simply applying an NMC on a per-message basis. While we might not get guarantees against the *same* class of tampering functions, we might still hope to get *some* useful guarantees. Sadly, this is not the case for general protocols. Consider a protocol between Alice and Bob, where Alice has input (x_1, x_2) and Bob has no input. In the protocol, Alice first sends x_1 to Bob, Bob replies with some arbitrary message and then Alice sends x_2. At the end Alice outputs nothing and Bob outputs (x_1, x_2).

If we encode the messages in this protocol individually, a tampering function can simply leave all the messages related to x_1 intact and replace the messages related to x_2 with constant messages that will decode to some x_2', causing Bob to output (x_1, x_2'), an output very much *not* unrelated (x_1, x_2). This attack works for any class of tampering functions that allow to tamper with the entire message, no matter how restrictive.

Technical Overview

On a technical level, our construction is heavily inspired by the bounded-state INMC of [36] and follows the same basic idea: At the beginning of the protocol, each pair of parties runs a key-exchange protocol that is secure against a bounded state attacker with full control of the communication channel. Once the key material has been successfully exchanged, the parties will engage in the underlying protocol while encrypting all messages with an information theoretically secure encryption scheme and authenticating each message with an information theoretically secure message authentication code.

However, since [36] was restricted to two parties and worked in a strictly synchronized setting, it is unsurprising that directly lifting their protocol to the multi-party unsynchronized setting causes it to fail in several ways. The key-exchange of [36] works as follows: The two parties P_1, P_2 each choose random strings α_1, β_1 and α_2, β_2 of sufficient length. They then send these in alternating order, use a 2-non-malleable extractor to agree on a key $k := \mathsf{nmExt}(\alpha_1, \beta_1) \oplus \mathsf{nmExt}(\alpha_2, \beta_2)$ and go on to exchange key-confirmation messages to confirm that both parties have received the same key. Once we allow the tampering function to desynchronize the two parties this key exchange becomes trivially broken. Consider the tampering function f that simply ignores P_2 and their messages. Instead, when P_1 sends α_1, the tampering function immediately sends back α_1 and likewise for β_1. Note that this attack works, even though P_1 will now receive α_2 in the "wrong round". This is because, as discussed above, we work in a purely message-driven model where P_1 does not notice how much time has passed between sending α_1 and receiving α_2. This means that P_1 will now derive the constant key $\mathsf{nmExt}(\alpha_1, \beta_1) \oplus \mathsf{nmExt}(\alpha_1, \beta_1) = 0^\kappa$. This key is of course trivially known to the tampering function in future rounds meaning the tampering function can simply engage in the underlying protocol pretending to be P_2 with some arbitrary input y' of its own choice. If the protocol is meant to evaluate a function g on the joint inputs, P_1 will now output $g(x, y')$ which is in general neither the correct result nor independent of (x, y).

To fix this problem, we split each bidirectional communication channel into two unidirectional channels and negotiate separate keys. The two parties still choose random strings α_1, β_1 and α_2, β_2 of sufficient length and send them in separate messages. However the parties agree on two separate keys $k_1 := \mathsf{nmExt}(\alpha_1, \beta_1)$ and $k_2 := \mathsf{nmExt}(\alpha_2, \beta_2)$. Each party P_i then uses k_i to *encrypt* messages *sent to* the other party and to *verify* the authentication tags on messages *received from* the other party. This way, each party always uses a key that is known to be *untampered* to perform the security critical operations.

It is still critical, that keys are confirmed and bound to a specific channel. If keys are not explicitly confirmed, a tampering function could replace one of the keys, say k_2 without any of the parties realizing. If P_2 would now send a message to P_1, this message would be *authenticated* using k_1 which was not tampered with, meaning P_1 would accept it. However they would then go on to *decrypt* the message with an incorrect key k_2'. This would likely result in P_1 passing a random string to the underlying protocol and there is no guarantee how the underlying protocol would behave in that case. If the key was not explicitly bound to a specific channel, a tampering function could potentially "swap" two parties. Say there's a protocol where P_3 and P_2 do not communicate with one another but *do* communicate with P_1. The tampering function could swap all messages from the channel between P_1 and P_2 to the channel between P_1 and P_3 and vice versa. If P_2 and and P_3 behave identically in the protocol and never explicitly identify themselves, this would lead P_1 to output $g(x_1, x_3, x_2)$ which again is obviously neither correct nor independent of the original input (x_1, x_2, x_3) in general.

To prevent all these and other problems introduced by the existence of multiple parties and the ability of the tampering function to desynchronize the parties each message is always authenticated together with the identifier of the channel it is being sent on and the message counter.

Different Message Topologies

The INMC for bounded-state tampering functions presented in Section 5 is still somewhat restricted in the sense that it can only directly be applied to protocols with a fixed message topology. This means the message flow of the protocol is required to be known is a priori and

has to be independent of inputs and randomness. I.e., when party P_i is invoked for the rth time, we a priori know from which parties they *should be receiving* messages and to which parties they *should be sending*.

Restricting ourselves to protocols with a fixed message topology makes our live significantly easier, as it allows us to sidestep many subtle issues. The most obvious problem would be an input dependent message topology. If, whether P_i sends a message to P_j when invoked for the rth time depends on the value x_i, we can easily come up with ways to leak x_i to the tampering function, which would make non-malleability impossible. A more subtle issue are protocols that *misbehave* if messages are reordered or dropped. Consider a protocol between two parties. In an untampered execution P_1 would receive a message from P_2 in its first invocation. At the end both parties output 0. Now we can modify this protocol to *misbehave* if the messages from P_2 never arrives. In this case P_1 could simply output x_1, which is neither correct nor unrelated to (x_1, x_2). If we, however, *know* which messages should be arriving in which order, we can abort any party that did not receive messages as specified in the protocol preventing them from misbehaving.

Obviously, restricting the INMC to protocols with a fixed message topology limits its applicability at least in theory. However, we show in Section 4 that any protocol (with a fixed upper bound on the number of rounds) can be transformed into a protocol with a fixed message topology, thereby extending the applicability of the INMC to (almost) arbitrary protocols. The transformation is fairly straightforward and simply involves sending dummy messages when the original protocol decides *not* to send a message. The transformation naturally comes with a certain blowup in the communication complexity.

1.2 Instantiating the Construction

To instantiate the protocol the main question is how to instantiate the 2-source non-malleable extractor. Before going into details of the instantiation of the non-malleable extractor one needs to understand the amount of key material that will be required by each party in order to carry out the protocol execution efficiently. As per the construction of our protocol every party will use the 2-source non-malleable extractor to extract an authentication key and an encryption key per party it communicates with. The length of those keys will depend on the number of messages the party expects to exchange with each other party. For a rough ballpark estimate, let us assume that the encoded protocol is between n parties, that each party sends the same number r of messages to every other party, and that those messages are all of length $\ell \geq \lambda$.

From the two sources sent from party A to party B, both parties must thus extract a key for a statistically unforgeable $(r+1)$-time MAC and a key for stateful r-time encryption scheme with perfect indistinguishability. Following, Remark 2 and Remark 4 they have to extract at least $(2r+2) \cdot \ell$ bits from each pair of sources. For a 2-source non-malleable extractor with source length κ_{nm} we will later see, that the sources will have min-entropy at least $\kappa_{nm} - (s + 3n\lambda)$. To get the lowest possible overhead, we will need a 2-source non-malleable extractor that can tolerate sources with the lowest possible min-entropy. The best currently known extractor in this regard was described in a recent paper by Xin Li [44]. The description of the construction, as is common for the literature on extractors, unfortunately only makes *asymptotic* statements about the extractor. It is thus hard to find out what the *exact* concrete source length of the extractor needs to be. We can however make an estimate on the *best possible* overhead achievable with the extractor from [44].

The construction described in Theorem 6.3 of [44] requires sources with min-entropy $(2/3+\gamma) \cdot \kappa_{nm}$ for the first source and k with $k \geq C \log \kappa_{nm}$ for the second source, where $0 < \gamma < 1/3$ can be chosen arbitrarily and $C > 1$ is some "large enough" constant. For large

enough κ_{nm} we can choose $k = (2/3 + \gamma) \cdot \kappa_{nm}$ which is convenient as the guaranteed min-entropy of the sources will be balanced in our application. The absolute best-case scenario for Li's extractor, depending on its exact instantiation, is that the output length is $9{\cdot}10^{-6}{\cdot}\kappa_{nm}$ and therefore we must have that $\kappa_{nm} > (10^6/9) \cdot (2r+2)\ell$. However, this is not the only condition. Additionally, we need to consider that we must have $\kappa_{nm} - (s+3n\lambda) \geq (2/3+\gamma) \cdot \kappa_{nm}$ for the non-malleability guarantee to apply. Therefore, we must also have $\kappa_{nm} \geq (s+3n\lambda)/(1/3+\gamma)$. Which of these two bounds is larger depends on the the exact parameters of the protocol and the size of the tampering function's state. However clearly even in the best case scenario the current state of the art makes the encoded protocol incur a multiplicative overhead of roughly $440,000$. It is thus clear that, which the current state of the art, our construction is chiefly of theoretical interest.

1.3 Related Works

To the best of our knowledge, the only previous work on non-malleable codes in the interactive setting has been the already mentioned work of Fleischhacker et al. [36]. In concurrent work Lin [45] used the results of [36] to construct non-malleable *multi-party computation*. However, Lin's results are largely orthogonal to our work. In particular the tampering model is weaker. E.g., while we allow the tampering function to tamper jointly on all parties' concurrent messages, Lin requires a fixed execution order and only allows tampering based on *past* messages. At the same time Lin attempts to achieve not merely a non-malleable encoding but non-malleable MPC, where the same party who controls the tampering function *also* controls a number of corrupted parties. Overall this means that the results of [45] are incomparable even if the used techniques are similar.

In contrast, non-malleable codes in the non-interactive setting have been studied extensively for a large variety of different classes of tampering functions. The most extensively studied class in the non-interactive setting are certainly split-state tampering functions [46, 29, 3, 19, 18, 2, 20, 42, 40, 41, 4]. But other classes of tampering functions have been studied such as tampering circuits of limited size or depth [35, 10, 17, 11, 8], tampering functions computable by decision trees [12], memory-bounded tampering functions [32] where the size of the available memory is a priori bounded, bounded polynomial time tampering functions [9], bounded *parallel*-time tampering functions [26], and non-malleable codes against streaming tampering functions [11]. Non-malleable codes were also generalized in several ways, such as continuously non-malleable codes in [33, 25, 23, 47, 31, 24, 4] and locally decodable and updatable non-malleable codes [28, 15, 27].

As a general rule non-malleable codes are usually considered in the information theoretic setting. However, there has also been some work in the computational setting. [1, 5, 6, 11]

2 Preliminaries

In this section we introduce our notation and recall some definitions needed for our constructions and proofs.

2.1 Notation

We denote the security parameter by $\lambda \in \mathbb{N}$. For an integer $n \in \mathbb{N}$, denote $[n] = \{1, \ldots, n\}$.

Let $\boldsymbol{M}$ be a matrix. We denote by $\mathbf{row}_i(\boldsymbol{M})$ the i-th row vector and by $\mathbf{col}_j(\boldsymbol{M})$ the j-th column vector of $\boldsymbol{M}$. If $\boldsymbol{M}$ is square, we denote by $\mathbf{diag}(\boldsymbol{M})$ the vector representing the main diagonal of $\boldsymbol{M}$.

Let S and S' be sets, let $P : S \to \{\mathsf{true}, \mathsf{false}\}$ be a predicate, let $f : S \to S'$ be a function, and let $L = (x_1, \ldots, x_\ell) \in S^\ell$ be a list. We denote by $\big(f(x) \mid x \in L \wedge P(x)\big)$ the list that contains $f(x_i)$ iff $P(x_i) = \mathtt{true}$ and preserves the relative order of the elements.

For $x' \in S$ we denote by $L \circ x'$ the list $(x_1, \ldots, x_\ell, x')$, i.e. the list resulting from appending x' to L. Further, we write L_i to denote the ith entry of L and $L_{\leq i}$ to denote the length i prefix of L, i.e. $L_{\leq i} = (x_1, \ldots, x_i)$.

Let D be some distribution over S. We denote by $f(D)$ the distribution over S' sampled by first sampling x according to D and then applying f to x. For a pair D_1, D_2 of distributions over a domain S, we denote their statistical distance by

$$\mathsf{SD}(D_1, D_2) = \frac{1}{2} \sum_{v \in S} \Big| \Pr[x \leftarrow D_1 : x = v] - \Pr[x \leftarrow D_2 : x = v] \Big|.$$

If $\mathsf{SD}(D_1, D_2) \leq \epsilon$, we say that D_1, D_2 are ϵ-close. For an arbitrary set S we define the functions $\mathsf{replace} : (S \cup \{\mathsf{same}\}) \times S \to S$ and $\mathsf{indicate} : S \to \{\mathsf{same}, \bot\}$ as

$$\mathsf{replace}(x, y) := \begin{cases} y & \text{if } x = \mathsf{same} \\ x & \text{otherwise} \end{cases} \quad \text{and} \quad \mathsf{indicate}(x) := \begin{cases} \mathsf{same} & \text{if } x \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

We extend $\mathsf{replace}$ and $\mathsf{indicate}$ to n-tuples in the natural way by applying them component-wise, i.e. $\mathsf{replace}(\boldsymbol{x}, \boldsymbol{y}) := (\mathsf{replace}(x_1, y_1), \ldots, \mathsf{replace}(x_n, y_n))$ and $\mathsf{indicate}(\boldsymbol{x}) := (\mathsf{indicate}(x_1), \ldots, \mathsf{indicate}(x_n))$.

2.2 Encryption and Message Authentication Codes

Our constructions relies exclusively on information theoretically secure primitives, specifically perfectly indistinguishable encryption and statistically secure message authentication codes. For notational convenience we formalize encryption as stateful which allows us not burden the description of the protocol with keeping track of key-usage.

▶ **Definition 1** (Stateful q-time Encryption with Perfect Indistinguishability)**.** *A correct stateful q-time encryption scheme $\mathcal{E}$ for message space $\{0,1\}^\ell$ and keyspace $\{0,1\}^\kappa$ consists of a pair of deterministic stateful algorithms $(\mathsf{Enc}, \mathsf{Dec})$, such that for all keys $k \in \{0,1\}^\kappa$ and all messages $(m_1, \ldots, m_q) \in (\{0,1\}^\ell)^r$ we have that for $c_1 := \mathsf{Enc}(k, m_1), \ldots, c_q := \mathsf{Enc}(k, m_q)$ and $m'_1 := \mathsf{Dec}(k, c_1), \ldots, m'_q := \mathsf{Dec}(k, c_q)$ it holds that $m_i = m'_i$ for all $i \in [r]$.*

Let LoR be the stateful "left-or-right" algorithm defined as $\mathsf{LoR}(k, b, m_0, m_1) := \mathsf{Enc}(k, m_b)$ for the first q invocations and as $\bot$ afterwards. A stateful q-time encryption scheme is perfectly indistinguishable if for any unbounded algorithm $\mathcal{A}$ it holds that

$$\Pr\big[k \leftarrow \{0,1\}^\kappa : \mathcal{A}^{\mathsf{LoR}(k,0,\cdot,\cdot)} = 0\big] = \Pr\big[k \leftarrow \{0,1\}^\kappa : \mathcal{A}^{\mathsf{LoR}(k,1,\cdot,\cdot)} = 0\big]$$

For convenience we extend the notation of encryption schemes to vectors in the natural way by applying the algorithm component wise. I.e., for $\boldsymbol{m} \in (\{0,1\}^\ell)^n$ and $k \in (\{0,1\}^\kappa)^n$ we write $\boldsymbol{c} := \mathsf{Enc}(\boldsymbol{k}, \boldsymbol{m})$ to denote the vector consisting of $c_i := \mathsf{Enc}(k_i, m_i)$. Similarly we write $\boldsymbol{m}' := \mathsf{Dec}(\boldsymbol{k}, \boldsymbol{c})$ for the vector consisting of $m'_i := \mathsf{Dec}(k_i, c_i)$.

▶ Remark 2. A stateful q-time encryption with perfect indistinguishability can easily be instantiated using the one-time pad where the key k is split into keys $k_1, \ldots, k_q \in \{0,1\}^\ell$ and c_i is computed as $m_i \oplus k_i$. The perfect indistinguishability follows from the regular perfect secrecy of the one-time pad.[52] In this case $\kappa = q\ell$.

▶ **Definition 3** (Statistically Unforgeable q-time MACs). *A q-time message authentication code $\mathcal{M}$ for message space $\{0,1\}^\ell$ and keyspace $\{0,1\}^\kappa$ consists of a pair of deterministic algorithms $(\mathsf{MAC}, \mathsf{Vf})$, such that for all keys $k \in \{0,1\}^\kappa$ and all messages $m \in \{0,1\}^\ell$ it holds that $\mathsf{Vf}(k, m, \mathsf{MAC}(k,m)) = 1$.*

Let $n \in \mathbb{N}$ and let $\widetilde{\mathsf{MAC}}$ be the algorithm defined as $\widetilde{\mathsf{MAC}}(k_1, \ldots, k_n, i, m) := \mathsf{MAC}(k_i, m)$. A q-time message authentication code is ϵ-unforgeable, if for all unbounded algorithms $\mathcal{A}$ it holds that

$$\Pr\left[\begin{matrix} k_1, \ldots, k_n \leftarrow \{0,1\}^\kappa \\ (i, m, t) \leftarrow \mathcal{A}^{\widetilde{\mathsf{MAC}}(k_1,\ldots,k_n,\cdot,\cdot)}() \end{matrix} : \begin{matrix} \mathsf{Vf}(k_i, m, t) = 1 \\ \wedge\, (m,t) \notin Q_i \wedge |Q_i| \leq q \end{matrix}\right] \leq \epsilon$$

where Q_i denotes the set of message-answer pairs, queried by $\mathcal{A}$ for index i.

Similar to encryption schemes, we extend the notation of message authentication codes to vectors in the natural way by applying the algorithm component wise. I.e., for $\boldsymbol{m} \in (\{0,1\}^\ell)^n$ and $\boldsymbol{k} \in (\{0,1\}^\kappa)^n$ we write $\boldsymbol{t} := \mathsf{MAC}(\boldsymbol{k}, \boldsymbol{m})$ to denote the vector consisting of $t_i := \mathsf{MAC}(k_i, m_i)$.

▶ Remark 4. Statistically unforgeable q-time MACs can be instantiated using any family of $q+1$-wise independent functions such as the family of degree q polynomials over $\mathbb{F}_{2^{\max\{\ell,\lambda\}}}$ [53]. In this case $\kappa = (q+1) \cdot \max\{\ell, \lambda\}$ and $\epsilon = 2^{-\max\{\ell,\lambda\}}$.

2.3 2-Non-Malleable Extractors

Our construction also makes use of 2-non-malleable extractors. These were first defined by Cheraghchi and Guruswami [19, 21] but constructing them was left as an open problem. The definition was finally instantiated by Chattopadhyay, Goyal, and Li [16]. Such an extractor allows to non-malleably extract an almost uniform random string from two sources with a given min-entropy that are being tampered by a split-state tampering function. We closely follow the definition from [16].

▶ **Definition 5** (2-Non-Malleable Extractor). *A function $\mathsf{nmExt} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ is a 2-non-malleable extractor for sources with min-entropy $k < n$ and with error ϵ if it satisfies the following property: If X, Y are independent sources of length n with min-entropy k and $f = (f_0, f_1)$ is an arbitrary 2-split-state tampering function, then there exists a distribution D_f over $\{0,1\}^m \cup \{\mathsf{same}\}$, such that*

$$\mathsf{SD}\big(\big(\mathsf{nmExt}(X,Y), \mathsf{nmExt}(f_0(X), f_1(Y))\big), \big(U_m, \mathsf{replace}(D_f, U_m)\big)\big) \leq \epsilon$$

where both U_m refer to the same uniform m-bit string.

▶ Remark 6. The required 2-non-malleable extractor can be instantiated with the construction of Chattopadhyay Goyal and Li [16] or a number of other construction. [42, 43, 22].

3 Interactive Protocols and Tampering Model

We consider protocols Π between n parties $P_1, \ldots, P_n$ for evaluating functionalities $g = (g_1, \ldots, g_n)$ of the form $g_i : X_1 \times \cdots \times X_n \to Y_i$, where X_i, Y_i are finite domains. Each party P_i holds an input $x_i \in X_i$ and randomness $\omega_i \in \Omega_i$ and the goal of the protocol is to interactively evaluate the functionality, such that at the end of the protocol party P_i outputs $g_i(x_1, \ldots, x_n) \in Y_i$.

Formally, an interactive protocol Π between n parties can be described either using interactive Turing machines, or using next-message functions. The two formalizations are equivalent up to a slight computational overhead. We will switch between the two formalizations whenever this is convenient for exposition.

Interactive Protocols as Interactive Turing Machines

In this formalization an interactive protocol Π between n parties is described by an n-tuple of interactive Turing machines P_i. Each interactive Turing machine P_i has an input tape containing x_i, a random tape containing ω_i, an internal work tape, as well as an incoming communication tape and an outgoing communication tape for each party P_j with $j \neq i$ and an output tape.

Interactive Protocols as Next Message Functions

In this formalization an interactive protocol Π between n parties is described by a an n-tuple of "next message" functions π_i and an n-tuple of output functions out_i. The next message function π_i takes as input the view of P_i, i.e., the input x_i, the randomness ω_i, and the sequence of message vectors received by P_i thus far and outputs the vector $\boldsymbol{s}_i \in \{0,1\}^* \cup \{\bot\}$ of messages to be sent by P_i. The output function out_i takes as input the final view of P_i, i.e., x_i, ω_i, and received message vectors and outputs P_i's protocol output.

Equivalence of Formalizations

The two formalizations are equivalent up to a slight computational overhead. To see this consider the following two simple conversions: Given an interactive Turing machine P_i, the equivalent next message function π_i can be computed on input $x_i, \omega_i, \boldsymbol{m}_i$ by simulating the Turing machine on input x_i and randomness ω_i, writing the received messages for each round on the appropriate incoming communication tapes until the current round is reached. The content of the outgoing communication tapes can then be output as $\boldsymbol{s}_i$. Similarly, given a next message function π_i, the equivalent interactive Turing machine P_i will simply store the contents of its incoming communication tapes on its internal work tape, evaluate π_i on its input x_i, randomness ω_i and all incoming messages, and write the output of π_i to its outgoing communication tapes.

3.1 Correctness and Encodings

We denote by $\Pi(\boldsymbol{x})$ the joint distribution of the outputs of an honest execution of the protocol Π using inputs $\boldsymbol{x}$ and uniformly sampled randomness $\boldsymbol{\omega}$. Further, we denote by $g(\boldsymbol{x})$ the vector $(g_1(x_1, \ldots, x_n), \ldots, g_1(x_1, \ldots, x_n))$.

▶ **Definition 7** (Correctness). *A protocol Π, is said to ϵ-correctly evaluate a functionality $g = (g_1, \ldots, g_n)$ if an untampered execution of the protocol correctly computes g with probability at least $1 - \epsilon$. I.e., for all valid input vectors $\boldsymbol{x}$ it holds that $\Pr[\boldsymbol{y} \leftarrow \Pi(\boldsymbol{x}) : \boldsymbol{y} = g(\boldsymbol{x})] \geq 1 - \epsilon$, where the probability is taken over the uniform choice of the random tape of all parties.*

▶ **Definition 8** (Encoding of an Interactive Protocol). *An encoding $\mathcal{E}$ of n-party interactive protocols is defined by n interactive oracle machines Enc_i.*

Let Π be an arbitrary interactive n-party protocol that ϵ-correctly evaluate a functionality g. The encoded *protocol is then the interactive n-party protocol between interactive Turing machines $(Q_1, \ldots, Q_n)$ defined as follows: On input x_i, Q_i samples uniform randomness ω_i,*

initiates the oracle $\mathsf{O}_{\boldsymbol{x},\boldsymbol{\omega'}} = P_i(x_i;\omega_i)$ *and then executes* $\mathsf{Enc}_i^{\mathsf{O}_{\boldsymbol{x},\boldsymbol{\omega'}}}()$*, giving it direct access to all communication tapes. Once* $\mathsf{Enc}_i^{\mathsf{O}_{\boldsymbol{x},\boldsymbol{\omega'}}}()$ *terminates with some output* y*,* Q_i *also outputs* y*.* $\mathcal{E}$ *is a* δ*-correct protocol encoding for* Π *if for all inputs* $\boldsymbol{x}$*, the protocol* $\mathcal{E}(\Pi) = (Q_1, \ldots, Q_n)$ $\epsilon + \delta$*-correctly evaluates the functionality* g*.*

3.2 Tampering Model

The transcript of a protocol executed under tampering needs to specify for each round of execution both the messages sent by each party and the messages received by each party. Remember that, due to the presence of the tampering function, the messages received are not necessarily related in any way to the messages sent.

We consider a scenario in which each party has a point-to-point channel to each other party, but not to itself. I.e., a protocol among n parties is executed over a complete directed communication graph (excluding loops) with n nodes P_i.

For each round, the transcript needs to label each edge (P_i, P_j) for $i \neq j$ in the graph with the message P_i sent to P_j and the message P_j received from P_i, the two of which need not be related. We will denote this with two $n \times n$ matrices $\boldsymbol{S}$ and $\boldsymbol{R}$ of labels per round of execution, where a label is either an arbitrary bitstring or the special symbol $\perp$ denoting that *no* message was sent or received respectively.

▶ **Definition 9** (Transcripts). *Let* $\mathcal{M} = \{0,1\}^* \cup \{\perp\}$ *be the set of possible labels for the edges of the communication graph. The set of possible transcripts is then the set of lists of pairs of matrices* $\boldsymbol{S}_i, \boldsymbol{R}_i \in \mathcal{M}^{n\times n}$ *such that the diagonal of both matrices only contains* $\perp$*. I.e.,*

$$\mathcal{T} = \left(\left\{\boldsymbol{M} \in \mathcal{M}^{n\times n} \;\middle|\; \mathbf{diag}(\boldsymbol{M}) \in \{\perp\}^n\right\}^2\right)^*.$$

For any transcript $\tau = \big((\boldsymbol{S_1}, \boldsymbol{R_1}), \ldots, (\boldsymbol{S_\ell}, \boldsymbol{R_\ell})\big)$*,* $\mathbf{row}_j(\boldsymbol{S_i})$ *denotes the vector of messages sent by* P_j *in round* i *of the execution, while* $\mathbf{col}_j(\boldsymbol{R_i})^\top$ *denotes the vector of messages received by* P_j *in round* i *of the execution.*
We denote by $\mathsf{Trans}_\Pi(\boldsymbol{x}, \boldsymbol{\omega})$ *the function mapping the input vector* $\boldsymbol{x}$ *along with the randomness* $\boldsymbol{\omega}$ *to the transcript of an honest execution of* Π *with inputs* $\boldsymbol{x}$ *and randomness* $\boldsymbol{\omega}$*.*

A party's view of the transcript consists exactly of the vectors of messages it receives. In particular, if a party does not receive any messages in a particular round of the execution, this round is not included in the party's view. This models that a party is not necessarily capable of detecting that desynchronization happens and allows general tampering functions to arbitrarily desynchronize different parties during protocol execution.

▶ **Definition 10** (Views). *Let* τ *be a transcript. The corresponding view of party* P_i *is then defined as* $V_i(\tau) = \big(\mathbf{col}_i(\boldsymbol{R})^\top \;\big|\; (\boldsymbol{S}, \boldsymbol{R}) \in \tau \wedge \mathbf{col}_i(\boldsymbol{R})^\top \notin \{\perp\}^n\big)$*.*

The interactive non-malleable code presented in Section 5 is restricted to protocols with a fixed message topology. This means that the number of messages exchanged over each channel is fixed, the expected relative ordering of all the messages received by a single party is a priori known, and whether or not a party sends a message along a communication channel does not depend on their input or their received messages. I.e., the "structure" of each vector in a party's view as well as the output vector in any particular round of execution is fixed in an untampered execution. We define this formally as follows.

▶ **Definition 11** (Fixed Message Topology). *An interactive protocol* Π *with* n *parties defined by next message functions* π_i *and output functions* out_i *is said to have a fixed message topology, if there exists a function* $\mu : [n] \times \mathbb{N} \to \{0,1\}^n \times \{0,1\}^n$*, such that for all vectors of inputs* $\boldsymbol{x}$*, all randomness vector* $\boldsymbol{\omega}$ *and the transcript* τ *of an honest untampered execution of* Π *on* $\boldsymbol{x}$ *with* $\boldsymbol{\omega}$*, all* $i \in [n]$*, and all* $r \in [|V_i(\tau)|]$ *it holds that* $\mu(i,r) = (\boldsymbol{v}', \boldsymbol{s}')$*, where*

$$v'_j := \begin{cases} 0 & \text{if } V_i(\tau)_{r,j} = \bot \\ 1 & \text{otherwise} \end{cases} \qquad s'_j := \begin{cases} 0 & \text{if } \pi_i(x_i, \omega_i, V_i(\tau)_{\leq r})_j = \bot \\ 1 & \text{otherwise} \end{cases}$$

for $j \in [n]$ *and for all* $r \geq |V_i(\tau)|$ *it holds that* $\mu(i,r) = (0^n, 0^n)$*. We further define the function* $\nu : [n] \times [n] \to \mathbb{N}$ *as* $\nu(i,j) := \sum_{r \in \mathbb{N}} \mu(i,r)_{1,j} = \sum_{r \in \mathbb{N}} \mu(j,r)_{2,i}$ *as the exact number of messages received by party* i *from* j *during an execution of the protocol.*

Let Π be a protocol with n parties defined by next message functions π_i and output functions out_i. For ease of notation we define the function Next_Π which describes computation of all messages sent during a particular round of execution depending on the protocol specification, the vector of inputs $\boldsymbol{x} = (x_1, \ldots, x_n)$ and the partial transcript $\tau \in \mathcal{T}$. Let $F : \mathcal{T} \times \mathcal{M}^{n \times n} \to \mathcal{M}^{n \times n}$ be an arbitrary tampering function. We describe execution of Π on inputs $\boldsymbol{x} = (x_1, \ldots, x_n)$ under tampering by F using the algorithm $\mathsf{Execute}_{\Pi,F}$.

$\mathsf{Next}_\Pi(\boldsymbol{x}, \boldsymbol{\omega}, \tau)$	$\mathsf{Execute}_{\Pi,F}(\boldsymbol{x}; \boldsymbol{\omega})$
parse $\tau = ((\boldsymbol{S_1}, \boldsymbol{R_1}), \ldots, (\boldsymbol{S_\ell}, \boldsymbol{R_\ell}))$	$\tau := \emptyset$
for $1 \leq i \leq n$ **do**	$\boldsymbol{S} := \mathsf{Next}(\Pi, \boldsymbol{x}, \boldsymbol{\omega}, \tau)$
if $\tau = \emptyset \ \vee \ \mathbf{col}_i(\boldsymbol{R_\ell})^\top \neq \bot^n$	$\boldsymbol{R} := F(\tau, \boldsymbol{S})$
$\boldsymbol{s_i} := \pi_i(x_i, \omega_i, V_i(\tau))$	**while** $\boldsymbol{R} \neq \bot^{n \times n}$
else	$\tau := \tau \circ (\boldsymbol{S}, \boldsymbol{R})$
$\boldsymbol{s_i} := \bot^n$	$\boldsymbol{S} := \mathsf{Next}(\Pi, \boldsymbol{x}, \boldsymbol{\omega}, \tau)$
return $\begin{bmatrix} \boldsymbol{s_1} \\ \vdots \\ \boldsymbol{s_n} \end{bmatrix}$	$\boldsymbol{R} := F(\tau, \boldsymbol{S})$
	return $\big(\mathsf{out}_1(x_1, V_1(\tau)), \ldots, \mathsf{out}_n(x_n, V_n(\tau))\big)$

Let $I : \mathcal{T} \times \mathcal{M}^{n \times n} \to \mathcal{M}^{n \times n}$ be the function defined as $I(\tau, \boldsymbol{S}) := \boldsymbol{S}$. We call I the identity tampering function. Note that the distribution $\Pi(\boldsymbol{x})$ is identical to the distribution $\mathsf{Execute}_{\Pi,I}(\boldsymbol{x})$.

▶ **Definition 12** (Protocol Non-malleability). *An* n*-party protocol* Π *for functionality* g *is* ϵ*-protocol non-malleable for a family* $\mathcal{F}$ *of tampering functions if for every tampering function* $F \in \mathcal{F}$ *there exists a distribution* D_F *over* $\{\bot, \mathsf{same}\}^n$ *such that for all* $\boldsymbol{x}$*, it holds that*

$$\mathsf{SD}(\mathsf{Execute}_{\Pi,F}(\boldsymbol{x}), \mathsf{replace}(D_F, \Pi(\boldsymbol{x}))) \leq \epsilon.$$

▶ **Definition 13** (Interactive Non-Malleable Code). *A protocol encoding* $\mathcal{E}$ *is called a* $(\delta, \epsilon)-$inter-active non-malleable code *for a family* $\mathcal{F}$ *of tampering functions and a class of protocols, if for any protocol* Π *of this class,* $\mathcal{E}(\Pi)$ δ*-correctly encodes* Π *and* $\mathcal{E}(\Pi)$ *is* ϵ*-protocol non-malleable for* $\mathcal{F}$*.*

3.3 Bounded State Tampering

We now define bounded state tampering functions for multi-party protocols. This is a natural model in which the adversary can *arbitrarily* and *jointly* tamper with *all* channels, however there exists an a priori upper bound on the size of the state they can hold. Similar classes of adversaries have already been considered starting with the work of Cachin and Maurer [14] which proposed encryption and key exchange protocols secure against computationally unbounded adversaries. With respect to non-malleable codes Faust et al. [32] introduced the notion of non-malleable codes against space-bounded tampering. Our formalization closely follows the one of Fleischhacker et al. [36] but adapted to the multi-party case. This means that we do not limit the size of the memory available for computing the tampering function in each round of tampering. Instead, we only limit the size of the state that can be carried over to the next round of tampering. I.e., an adversary in this model can jointly tamper with all of the messages exchanged in one round of execution depending on some function of *all* previously exchanged messages. *But* the function can only depend on up to some fixed number of s bits of information about previous messages. This is formalized as follows.

▶ **Definition 14** (Bounded State Tampering Functions). *Functions of the class of s-bounded state tampering functions $F \in \mathcal{F}^{s}_{\mathsf{bounded}}$ for an interactive protocol are defined by a function*

$$f : \{0,1\}^s \cup \{\bot\} \times \mathcal{M}^{n\times n} \to \{0,1\}^s \times \mathcal{M}^{n\times n}$$

The function f takes as input a previous state of the tampering function and a matrix of sent messages and outputs an updated state and a matrix of received messages.

The full tampering function $F : \mathcal{T} \times \mathcal{M}^{n\times n} \to \mathcal{M}^{n\times n}$ is then defined in terms of f as seen below.

$F(\tau, \boldsymbol{S})$
$\sigma \coloneqq \bot$
for $(\boldsymbol{S'}, \boldsymbol{R'})$ **in** τ
  $(\sigma, \boldsymbol{R}) \coloneqq f(\sigma, \boldsymbol{S'})$
$(\sigma, \boldsymbol{R}) \coloneqq f(\sigma, \boldsymbol{S})$
return $\boldsymbol{R}$

4 Arbitrary Message Topologies

The INMC for Bounded-state tampering functions that is introduced in Section 5 requires the underlying protocol to have a fixed message topology. In this section we show that is is not in general a restriction, as any protocol can be transformed protocol with a fixed message topology. Therefore, the INMC can be applied to *any* protocol by first transforming it to a protocol with a fixed message topology and then applying the INMC itself.

For this purpose we first introduce a general definition of a message topology, which for any party and round defines the *probability* that messages are received or sent over each channel, maximized over all possible inputs.

▶ **Definition 15** (Message Topology). *Let Π be interactive protocol with n parties defined by next message functions π_i and output functions out_i. The message topology of Π is defined by a function $\mu : [n] \times \mathbb{N} \to [0,1]^n \times [0,1]^n$, such that for all $i \in [n]$, and all $r \in \mathbb{N}$ it holds that $\mu(i,r) = (\boldsymbol{v}', \boldsymbol{s}')$, where*

$$v'_j := \max_{\boldsymbol{x}} \left\{ \Pr\left[\begin{matrix} \boldsymbol{\omega} \leftarrow \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n \\ \tau \leftarrow \mathsf{Trans}_\Pi(\boldsymbol{x}, \boldsymbol{\omega}) \end{matrix} : |V_i(\tau)| \geq r \wedge V_i(\tau)_{r,j} \neq \bot \right] \right\}$$

$$s'_j := \max_{\boldsymbol{x}} \left\{ \Pr\left[\begin{matrix} \boldsymbol{\omega} \leftarrow \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n \\ \tau \leftarrow \mathsf{Trans}_\Pi(\boldsymbol{x}, \boldsymbol{\omega}) \end{matrix} : |V_i(\tau)| \geq r \wedge \pi_i(x_i, \omega_i, V_i(\tau)_{\leq r})_j \neq \bot \right] \right\}$$

for $j \in [n]$.

A fixed message topology can then be seen as a special case, where μ is defined over $\{0,1\}$ and the probabilities used in the definition are *independent of the input vectors.*

4.1 Transformations from Arbitrary to Fixed Message Topology

We propose three different transformations from an arbitrary message topology (AMT) to a fixed message topology (FMT). The first transformation is very naive, resulting in a very large blowup of the communication complexity but can be applied without any detailed consideration to the original message topology, i.e. it does not even reference the above definition. The second transformation *considers* the original message topology and will result in a lower blowup in the communication complexity for most reasonable protocols. However, in the worst case, for pathological examples, it can still result in the same blowup as the naive transformation. The third transformation finally allows us to limit the blowup, even in the worst case, but at the cost of potentially degrading the correctness of the protocol. Throughout this section, we assume that there exists a fixed upper bound on the number rounds the execution of a protocol may take.

4.1.1 Trivial Transformation

The simplest transformation floods the entire network in every round by sending dummy messages whenever there's no actual message to be sent. To reliably distinguish between real and dummy messages, real messages are marked by a prefix identifying them as real. Specifically, if in any round of the original protocol an actual message is sent by party P_i to party P_j then party P_i just prepends 1 to the message and sends it to the concerned party P_j. If on the other hand *no* message is sent in the original protocol a dummy message consisting of 0 is sent to P_j. To formally describe the next-message functions of the transformed protocol, we first define two functions $\mathsf{addDummies}$ and $\mathsf{remDummies}$ used to add and remove the dummy messages. We define the function

$$\mathsf{addDummies} : [n] \times [r_{\max}] \times (\{0,1\}^* \cup \{\bot\})^n \to (\{0,1\}^* \cup \{\bot\})^n$$

as

$$\mathsf{addDummies}(i, r, \boldsymbol{m}) := \boldsymbol{m}', \quad \text{where} \quad m'_j := \begin{cases} \bot & \text{if } i = j \\ 0 & \text{if } m_j = \bot \text{ and } i \neq j \\ 1 \| m_j & \text{otherwise.}^1 \end{cases}$$

and the function

[1] Note that $\mathsf{addDummies}$ takes an input r which is then ignored. This will make our lives easier when we modify the transformation going forward.

$$\mathsf{remDummies} : (\{0,1\}^* \cup \{\bot\})^n \to (\{0,1\}^* \cup \{\bot\})^n$$

as

$$\mathsf{remDummies}(\boldsymbol{m}) := \boldsymbol{m}', \quad \text{where} \quad m'_j := \begin{cases} \bot & \text{if } m_j \in \{0, \bot\} \\ m'' & \text{if } m_j = 1 \| m'' \end{cases}$$

For ease of notation we will apply remDummies to lists of vectors, which is to be interpreted as component-wise application. Let Π be an arbitrary ϵ-correct protocol described by next message functions π_i with an upper bound of $r_{\max}$ on the number of rounds. The next message function π'_i of the naively transformed protocol can then simply be defined as

$$\pi'_i(x_i, \omega_i, V_i) := \begin{cases} \bot^n & \text{if } |V_i| \geq r_{\max} \\ \mathsf{addDummies}(i, |V_i|, \pi_i(x_i, \omega_i, \mathsf{remDummies}(V_i))) & \text{otherwise.} \end{cases}$$

This trivially transformed protocol works exactly in the same way as the original protocol with the only exception being that in every round, all channels on which the original protocol would not have sent messages, the transformed protocol sends dummy messages and then promptly ignores them. The protocol terminates after exactly $r_{\max}$ rounds. This means that since the transformed protocol doesn't drop any messages and the original views of the parties can easily be reconstructed by ignoring the dummy messages. Hence it will still be ϵ-correct. This transformation clearly serves the purpose of transforming any protocol into a protocol with a fixed message topology. A clear downside, however, is the blowup in communication complexity, especially if the original protocol used a rather sparse communication graph. In every round each party starts sends messages to every other party whether they were expecting messages or not. In the worst case, this means that the expected communication complexity of the protocol blows up infinitely.[2] But even in more reasonable protocols that happen to use a sparse communication graph, the blowup is quite severe. Luckily we can do a bit better at least for reasonable protocols.

4.1.2 Maintaining the Communication Graph

The overhead of the transformation can be reduced if we only flood those channels where messages *could possibly be sent*. In order for that to happen we let party P_i send a dummy message only on those channels where there's a *non-zero* probability of a real message being sent. We can achieve that if we redefine the function addDummies as follows

$$\mathsf{addDummies}(i, r, \boldsymbol{m}) := \boldsymbol{m}', \text{ where } m'_j := \begin{cases} 0 & \text{if } \boldsymbol{m}'_j = \bot \text{ and } \mu(i,r)_{2,j} > 0 \\ 1 \| \boldsymbol{m}'_j & \text{if } \boldsymbol{m}'_j \neq \bot \\ \bot & \text{otherwise.} \end{cases}$$

The next message function is still defined as before. Clearly, this again results in a fixed message topology. This transformed protocol will also be ϵ-correct if the actual protocol is ϵ-correct as no messages are dropped and the original view can be reconstructed. Even though this transformation eliminates quite a lot of redundant messages and will result in a much smaller blowup for many protocols run over a sparse communication graph, the worst case blowup still remains infinite by the same argument as before.

[2] An example of a pathological protocol that exhibits infinite blowup is a protocol with at most one round, where one party sends a message with probability ζ, where ζ tends towards 0.

4.1.3 Dropping Low Probability Messages

To fix this issue of an infinite blowup in the expected communication complexity we can modify the transformation, by introducing a threshold value t and dropping all messages that are sent with probability less than t. In order to achieve this we again redefine the function $\mathsf{addDummies}$ as as

$$\mathsf{addDummies}(i, r, \boldsymbol{m}) := \boldsymbol{m}', \text{ where } m'_j := \begin{cases} 0 & \text{if } \boldsymbol{m}'_j = \bot \text{ and } \mu(i,r)_{2,j} > t \\ 1 \| \boldsymbol{m}'_j & \text{if } \boldsymbol{m}'_j \neq \bot \text{ and } \mu(i,r)_{2,j} > t \\ \bot & \text{otherwise.} \end{cases}$$

This transformation only allows messages to be sent if their probability is above the threshold probability of t. The implementation of the next message function π_i remains the same for this transformation as was for the last transformation. For this transformation potentially degrades the correctness of the protocol. In a protocol with $r_{\max}$ rounds there are $r_{\max}(n^2-n)$ potential messages, each of which may have a probability of being sent infinitesimally less than t. Each message that *is actually sent* by the underlying protocol but them blocked by the transformation can result in the protocol computing an incorrect output. But we can apply a union bound and get that the transformed protocol remains $\epsilon + t r_{\max}(n^2 - n)$-correct. However, this degraded correctness buys us a finite bound on the blowup of the expected communication complexity. In the worst case, each message in the original protocol is sent with probability exactly t, whereas it is sent with probability 1 in the transformed protocol. Therefore, the blowup can be at most $1/t$ therefore allowing us to bound the blowup.

5 An INMC for Bounded-State Tampering Functions

We devise an interactive non-malleable code for bounded state tampering functions that can be applied to any multi-party protocol Π' with *fixed message topology*, i.e., to any protocol where for every party P_i and every invocation r of the next message function π_i, whether or not a message is sent to party P_j is a priori known and does *not* depend on any of $\pi'_i s$ inputs. The basic idea is that each pair of parties will first run a key-exchange in which they will exchange enough key-material to the execute the original protocol encrypted under an information theoretically secure encryption scheme and authenticated with a statistically unforgeable MAC. Besides making sure that the tampering function cannot replay, redirect or omit messages by binding the authentication to a specific channel and including message counters in the authentication, the main challenge is to construct a key exchange that is secure against a computationally unbounded but bounded state adversary. We achieve this using a 2-non-malleable extractor. Essentially each party chooses a key by choosing two random sources α, β which will be much longer than the bounded state of the tampering function and extracting a key $k := \mathsf{nmExt}(\alpha, \beta)$. They will be using this key which they *know* is untampered to *encrypt* messages and to *verify* authentication tags. The two sources α, β are then sent in seperate rounds, ensuring that they cannot be tampered jointly, except for some amount of leakage through the state of the tampering function and potentially conditional aborts. This leakage can be handled by reinterpreting the sources as coming from a different distribution with slightly less min-entropy. Once the keys are exchanged, the parties verify that the keys were not modified in transit by sending a MAC computed over the ID of the channel with the key they *received* from the other party.

$\mathsf{CheckMsgOrder}(V)$

$\boldsymbol{r} = 0^n$
for $j \in [n] \setminus \{i\}$ **do**
  if $V_{|V|,j} \neq \bot$
    $r_j := 1$
if $\mu(i, |V| - 3)_1 \neq \boldsymbol{r}$
  return 0
return 1

Figure 1 The function checking the ordering of messages against the fixed message topology.

$\pi_i^{\mathsf{O}'_{\boldsymbol{x},\boldsymbol{\omega}'}}(\boldsymbol{\omega}, V_i(\tau))$

$\boldsymbol{s} := \bot^n$
if $|V| = 0$
  $\mathsf{rec} := 0^n, \mathsf{sent} := 0^n$
  $(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i) \leftarrow ((\{0,1\}^{\kappa_{nm}})^2)^n$
  $(\boldsymbol{k}_i^{enc}, \boldsymbol{k}_i^{auth}) := \mathsf{nmExt}(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i)$
  $\boldsymbol{s} := \boldsymbol{\alpha}_i$
elseif $|V| = 1$
  if $\bot \in V_1$
    abort
  $\boldsymbol{s} := \boldsymbol{\beta}_i$
elseif $|V| = 2$
  if $\bot \in V_2$
    abort
  $(\tilde{\boldsymbol{k}}_j^{enc}, \tilde{\boldsymbol{k}}_j^{auth}) := \mathsf{nmExt}(V_1, V_2)$
  for $j \in [n] \setminus \{i\}$ **do**
    $s_j := \mathsf{MAC}(\tilde{k}_{i,j}^{auth}, (i, j))$
else
  if $|V| = 3$
    if $\bot \in V_3$
      abort
    for $j \in [n] \setminus \{i\}$ **do**
      if $\mathsf{Vf}(k_i^{auth}, (j, i), V_{3,j}) = 0$
        abort
  else
    $\boldsymbol{m} := \bot^n$
    if $\mathsf{CheckMsgOrder}(V) = 0$
      abort
    else
      for $j \in [n] \setminus \{i\}$ **do**
        if $V_{|V|,j} \neq \bot$
          $(c, t) := V_{|V|,j}, \mathsf{rec}_j := \mathsf{rec}_j + 1$
          if $\mathsf{Vf}(k_i^{auth}, (c, j, i, \mathsf{rec}_j), t) = 0$
            abort
          $m_j := \mathsf{Dec}(\tilde{k}_j^{enc}, c)$
    $\boldsymbol{s}' \leftarrow \mathsf{O}'_{\boldsymbol{x},\boldsymbol{\omega}'}(\boldsymbol{m})$
    for $j \in [n] \setminus \{i\}$ **do**
      if $s'_j \neq \bot$
        $c := \mathsf{Enc}(\tilde{k}_i^{enc}, s'_j), \mathsf{sent}_j := \mathsf{sent}_j + 1$
        $s_j := (c, \mathsf{MAC}(\tilde{k}_j^{auth}, (c, i, j, \mathsf{sent}_j))$
return $\boldsymbol{s}$

Figure 2 The next message function describing the INMC for bounded state tampering functions. For the sake of readability, we write the function as if it were stateful. I.e., in particular the variables rec and sent retain their value accross different invocations of π_i and do not need to be recomputed.

5.1 Defining the Next Message Function

The INMC is restricted to protocols with a fixed message topology as defined in Definition 11. Refer to Section 4 for a discussion on how arbitrary protocols can be transformed into protocols with a fixed message topology. To formally describe the next message function and output function of the INMC, we need an algorithm that checks whether the sequence of messages received from the other parties involved in the protocol confirm to the fixed message topology. The function CheckMsgOrder defined in Figure 1 allows to perform this check. Now that we have defined the CheckMsgOrder function we are ready to define the next message function in Figure 2. Remember, that according to Definition 8 an encoding is specified by an oracle machine or equivalently a next message function that is defined relative to an stateful oracle representing the next message function of the underlying protocol. The next message function $\pi_i^{\mathsf{O}_{i,\boldsymbol{x},\boldsymbol{\omega}'}}$ has three phases. In the initial phase every party shares their keys with the rest of the parties taking part in the protocol. In the next phase all of the parties confirms their respective keys with the other parties by sending a key confirmation value. The last phase of the execution of the next message function just deals with the

actual message exchanges that happens between all the parties taking part in the protocol Π'. The output function out_i of the INMC simply takes the view V' of the underlying protocol that it can extract from it's own view exactly as in the next message function and outputs $\mathsf{out}'_i(\boldsymbol{x}, \boldsymbol{\omega'}, V')$ if the view conforms to the fixed message topology or $\perp$ otherwise.

▶ **Theorem 16.** *Let Π' be a protocol between n parties with fixed message topology, with $r = \max_{i,j\in[n]}\{\nu(i,j)\}$ and message length ℓ. If $(\mathsf{MAC}, \mathsf{Vf})$ is a statistically ϵ_{mac}-unforgeable $r+1$-time message authentication code with with message length $\ell + 2\lceil \log n \rceil + \lceil \log r \rceil$ and key length κ_{mac}, $(\mathsf{Enc}, \mathsf{Dec})$ is a perfectly indistinguishable stateful t-time encryption scheme with message length ℓ and key length κ_{enc}, and $\mathsf{nmExt} : \{0,1\}^{\kappa_{nm}} \times \{0,1\}^{\kappa_{nm}} \to \{0,1\}^{\kappa_{mac}+\kappa_{\mathsf{Enc}}}$ is an ϵ_{nm}-non-malleable 2-source extractor for sources with min-entropy at least $\kappa_{nm} - s - 3n\lambda$, then Π as described by π_i and out_i specified above is a $(0, (2n^2+n)\cdot 2^{-\lambda} + (n^2-n)\cdot\epsilon_{nm} + \epsilon_{\mathsf{MAC}})$-interactive non-malleable code for Π' for the class $\mathcal{F}^s_{\mathsf{bounded}}$ of bounded state tampering functions.*

Due to space constraints the proof is deferred to Appendix A.

References

1. Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 393–417, Tel Aviv, Israel, January 10–13 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49099-0_15.
2. Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 459–468, Portland, OR, USA, June 14–17 2015. ACM Press. doi:10.1145/2746539.2746544.
3. Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 774–783, New York, NY, USA, May 31 – June 3 2014. ACM Press. doi:10.1145/2591796.2591804.
4. Divesh Aggarwal, Nico Döttling, Jesper Buus Nielsen, Maciej Obremski, and Erick Purwanto. Continuous non-malleable codes in the 8-split-state model. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 531–561, Darmstadt, Germany, May 19–23 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17653-2_18.
5. Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 538–557, Santa Barbara, CA, USA, August 16–20 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-47989-6_26.
6. Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 375–397, Warsaw, Poland, March 23–25 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46494-6_16.
7. Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 593–622, Darmstadt, Germany, May 19–23 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17653-2_20.

8 Marshall Ball, Dana Dachman-Soled, Siyao Guo, Tal Malkin, and Li-Yang Tan. Non-malleable codes for small-depth circuits. In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 826–837, Paris, France, October 7–9 2018. IEEE Computer Society Press. doi:10.1109/FOCS.2018.00083.

9 Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, Huijia Lin, and Tal Malkin. Non-malleable codes against bounded polynomial time tampering. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 501–530, Darmstadt, Germany, May 19–23 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17653-2_17.

10 Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 881–908, Vienna, Austria, May 8–12 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49896-5_31.

11 Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes from average-case hardness: AC^0, decision trees, and streaming space-bounded tampering. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 618–650, Tel Aviv, Israel, April 29 – May 3 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78372-7_20.

12 Marshall Ball, Siyao Guo, and Daniel Wichs. Non-malleable codes for decision trees. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 413–434, Santa Barbara, CA, USA, August 18–22 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-26948-7_15.

13 Mark Braverman, Ran Gelles, Jieming Mao, and Rafail Ostrovsky. Coding for interactive communication correcting insertions and deletions. *IEEE Transactions on Information Theory*, 63(10):6256–6270, 2017. doi:10.1109/TIT.2017.2734881.

14 Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 292–306, Santa Barbara, CA, USA, August 17–21 1997. Springer, Heidelberg, Germany. doi:10.1007/BFb0052243.

15 Nishanth Chandran, Bhavana Kanukurthi, and Srinivasan Raghuraman. Information-theoretic local non-malleable codes and their applications. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 367–392, Tel Aviv, Israel, January 10–13 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49099-0_14.

16 Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 285–298, Cambridge, MA, USA, June 18–21 2016. ACM Press. doi:10.1145/2897518.2897547.

17 Eshan Chattopadhyay and Xin Li. Non-malleable codes and extractors for small-depth circuits, and affine functions. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 1171–1184, Montreal, QC, Canada, June 19–23 2017. ACM Press. doi:10.1145/3055399.3055483.

18 Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. In *55th Annual Symposium on Foundations of Computer Science*, pages 306–315, Philadelphia, PA, USA, October 18–21 2014. IEEE Computer Society Press. doi:10.1109/FOCS.2014.40.

19 Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*,

volume 8349 of *Lecture Notes in Computer Science*, pages 440–464, San Diego, CA, USA, February 24–26 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-54242-8_19.

20 Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. *IEEE Transactions on Information Theory*, 62(3):1097–1118, March 2016.

21 Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. *Journal of Cryptology*, 30(1):191–241, January 2017. doi:10.1007/s00145-015-9219-z.

22 Eldon Chung, Maciej Obremski, and Divesh Aggarwal. Extractors: Low entropy requirements colliding with non-malleability. arXiv, 2021. doi:10.48550/arXiv.2111.04157.

23 Sandro Coretti, Yevgeniy Dodis, Björn Tackmann, and Daniele Venturi. Non-malleable encryption: Simpler, shorter, stronger. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 306–335, Tel Aviv, Israel, January 10–13 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49096-9_13.

24 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, *ACNS 19: 17th International Conference on Applied Cryptography and Network Security*, volume 11464 of *Lecture Notes in Computer Science*, pages 3–23, Bogota, Colombia, June 5–7 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-21568-2_1.

25 Sandro Coretti, Ueli Maurer, Björn Tackmann, and Daniele Venturi. From single-bit to multi-bit public-key encryption via non-malleable codes. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 532–560, Warsaw, Poland, March 23–25 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46494-6_22.

26 Dana Dachman-Soled, Ilan Komargodski, and Rafael Pass. Non-malleable codes for bounded parallel-time tampering. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 535–565, Virtual Event, August 16–20 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-84252-9_18.

27 Dana Dachman-Soled, Mukul Kulkarni, and Aria Shahverdi. Tight upper and lower bounds for leakage-resilient, locally decodable and updatable non-malleable codes. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 310–332, Amsterdam, The Netherlands, March 28–31 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-54365-8_13.

28 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, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 427–450, Warsaw, Poland, March 23–25 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46494-6_18.

29 Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 239–257, Santa Barbara, CA, USA, August 18–22 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40084-1_14.

30 Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *ICS 2010: 1st Innovations in Computer Science*, pages 434–452, Tsinghua University, Beijing, China, January 5–7 2010. Tsinghua University Press.

31 Antonio Faonio, Jesper Buus Nielsen, Mark Simkin, and Daniele Venturi. Continuously non-malleable codes with split-state refresh. In Bart Preneel and Frederik Vercauteren, editors, *ACNS 18: 16th International Conference on Applied Cryptography and Network Security*,

volume 10892 of *Lecture Notes in Computer Science*, pages 121–139, Leuven, Belgium, July 2–4 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-93387-0_7.

32 Sebastian Faust, Kristina Hostáková, Pratyay Mukherjee, and Daniele Venturi. Non-malleable codes for space-bounded tampering. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 95–126, Santa Barbara, CA, USA, August 20–24 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-63715-0_4.

33 Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 465–488, San Diego, CA, USA, February 24–26 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-54242-8_20.

34 Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. A tamper and leakage resilient von neumann architecture. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 579–603, Gaithersburg, MD, USA, March 30 – April 1 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46447-2_26.

35 Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 111–128, Copenhagen, Denmark, May 11–15 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_7.

36 Nils Fleischhacker, Vipul Goyal, Abhishek Jain, Anat Paskin-Cherniavsky, and Slava Radune. Interactive non-malleable codes. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 233–263, Nuremberg, Germany, December 1–5 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-36033-7_9.

37 Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th Annual ACM Symposium on Theory of Computing*, pages 685–698, Los Angeles, CA, USA, June 25–29 2018. ACM Press. doi:10.1145/3188745.3188872.

38 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, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 501–530, Santa Barbara, CA, USA, August 19–23 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-96884-1_17.

39 Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 1128–1141, Cambridge, MA, USA, June 18–21 2016. ACM Press. doi:10.1145/2897518.2897657.

40 Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Four-state non-malleable codes with explicit constant rate. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 344–375, Baltimore, MD, USA, November 12–15 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-70503-3_11.

41 Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Non-malleable randomness encoders and their applications. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 589–617, Tel Aviv, Israel, April 29 – May 3 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78372-7_19.

42 Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM*

Symposium on Theory of Computing, pages 1144–1156, Montreal, QC, Canada, June 19–23 2017. ACM Press. doi:10.1145/3055399.3055486.

43 Xin Li. Non-malleable extractors and non-malleable codes: Partially optimal constructions. In *Proceedings of the 34th Computational Complexity Conference*, CCC '19, Dagstuhl, DEU, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2019.28.

44 Xin Li. Two source extractors for asymptotically optimal entropy, and (many) more. arXiv, 2023. doi:10.48550/arXiv.2303.06802.

45 Fuchun Lin. Non-malleable multi-party computation. Cryptology ePrint Archive, Report 2022/978, 2022. URL: https://eprint.iacr.org/2022/978.

46 Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 517–532, Santa Barbara, CA, USA, August 19–23 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-32009-5_30.

47 Rafail Ostrovsky, Giuseppe Persiano, Daniele Venturi, and Ivan Visconti. Continuously non-malleable codes in the split-state model from minimal assumptions. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 608–639, Santa Barbara, CA, USA, August 19–23 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-96878-0_21.

48 Sridhar Rajagopalan and Leonard J. Schulman. A coding theorem for distributed computation. In *26th Annual ACM Symposium on Theory of Computing*, pages 790–799, Montréal, Québec, Canada, May 23–25 1994. ACM Press. doi:10.1145/195058.195462.

49 Leonard J. Schulman. Communication on noisy channels: A coding theorem for computation. In *33rd Annual Symposium on Foundations of Computer Science*, pages 724–733, Pittsburgh, PA, USA, October 24–27 1992. IEEE Computer Society Press. doi:10.1109/SFCS.1992.267778.

50 Leonard J. Schulman. Deterministic coding for interactive communication. In *25th Annual ACM Symposium on Theory of Computing*, pages 747–756, San Diego, CA, USA, May 16–18 1993. ACM Press. doi:10.1145/167088.167279.

51 Leonard J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, November 1996.

52 Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

53 Mark N. Wegman and J. Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981. doi:10.1016/0022-0000(81)90033-7.

A Proof of Main Theorem

Proof. In order to prove that the protocol Π is a $(0, \epsilon)$-interactive non-malleable code we need to prove the *correctness* as well as *non-malleability* of the protocol as stated in Lemma 17 and Lemma 18.

▶ **Lemma 17.** *For any protocol* Π'*,* Π 0*-correctly encodes* Π'*.*

Proof. The extractor is deterministic and hence all the parties involved in the protocol will extract identical keys in an untampered execution. Since the MAC is correct, and tags are computed and verified with the correct keys, all messages will always verify and no party will abort during the protocol. As the correctness of the stateful encryption scheme Enc allows each party decrypt all received messages correctly all the parties will be able to faithfully execute a perfectly honest untampered execution of the underlying protocol Π. Therefore Π evaluates correctly whith the same probability as Π'. ◀

▶ **Lemma 18.** *The interactive protocol* Π *is* ϵ*-protocol non-malleable, where* $\epsilon = (2n^2 + n) \cdot 2^{-\lambda} + (n^2 - n) \cdot \epsilon_{nm} + \epsilon_{\mathsf{MAC}}$.

Proof. In order to show that the coding scheme is non-malleable we need to provide a *distribution* D_F as defined in Definition 12. In order to achieve a sampler for the distribution D_F, we start with the output distribution of an honest execution of the actual protocol and modify it through a serie of hyrids, until we reach a distribution that can be sampled independently of $\boldsymbol{x}$. To define the different hybrid distributions, first define a function $\bar{V}_i$ which essentially gives us the equivalent of a party's current view in the protocol, but replaces all received messages, with the messages that were originally sent.

$\bar{V}(\tau)$

$V := V_i(\tau)$
for $j \in [n]$ **do**
  $s := (S_{i,j} \mid (S, R) \in \tau \wedge S_{i,j} \neq \bot)$
  $c := 1$
  for $q \in [|V|]$ **do**
    if $V_{q,j} \neq \bot$
      $V_{q,j} := t_{j,c}$
      $c = c + 1$
return V

Now, let $F \in \mathcal{F}^s_{\mathsf{bounded}}$ be an arbitrary tampering function. For $i, j \in [n]$ and $o \in \{\alpha, \beta\}$, let $\zeta_{\alpha,i,j}$ be the probability that the tampering function modifies or drops $o_{i,j}$
during an execution of the protocol. We define the modified tampering function F' which behaves exactly like F, but for any (i, j, o) such that $\zeta_{\alpha,i,j} < 2^{-\lambda}$ it always keeps $o_{i,j}$ unmodified. We then further define for $i \in [n]$ and $r \in \{1, 2, 3\}$, $\gamma_{i,r}$ to be the probability that in an execution tampered by F', π_i aborts in execution round r, i.e., after receiving the αs, after receiving the βs, or after receiving the key confirmation values. Finally, let $\boldsymbol{x}' \in X_1 \times \cdots \times X_n$ be arbitrary but fixed. We then define several variants of $\mathsf{Execute}$, Next, and π_i in Figure 3 and Figure 4 respectively and are then finally ready to specify a series of hybrid distribution we construct to reach the distribution that corresponds to $\mathsf{replace}(D_F, \Pi(\boldsymbol{x}))$.

- $\boldsymbol{H_0}$ **:** Hybrid 0 is the original output distribution of a tampered execution. I.e, $H_0 = \mathsf{Execute}_{\Pi,F}(\boldsymbol{x})$.
- $\boldsymbol{H_1}$ **:** Hybrid 1 is still the distribution of a tampered execution, however we replace the tampering function with the modified tampering function F'. I.e., $H_1 = \mathsf{Execute}_{\Pi,F'}(\boldsymbol{x})$.
- $\boldsymbol{H_2}$ **:** In hybrid 2, we switch to using the modified execution algorithm $\mathsf{Execute}^1$ and Π^1. This change gives the next message function access to the message it *should* have received, i.e., those that were originally sent. I.e., $H_2\mathsf{Execute}^1_{\Pi^1,F'}(\boldsymbol{x})$.
- $\boldsymbol{H_3}$ **:** In hybrid 3 we switch to using Π^2, which means that parties that abort with overwhelming probability during the key exchange or key confirmation phase, now abort with probability 1. I.e., $H_3 = \mathsf{Execute}^1_{\Pi^2,F'}(\boldsymbol{x})$.
- $\boldsymbol{H_4}$ **:** In hybrid 4, we switch to using Π^3 which means that the keys are now no longer extracted but instead sampled uniformly at random on the sender's side and according to D_f on the receiver's side, where f is a split state tampering function induced by F'. I.e., $H_4 = \mathsf{Execute}^1_{\Pi^3,F'}(\boldsymbol{x})$.
- $\boldsymbol{H_5}$ **:** Hybrid 5 switches to using Π^4, which means that instead of verifying MACs the next message functions now directly check if messages were modified or not. I.e., $H_5 = \mathsf{Execute}^1_{\Pi^4,F'}(\boldsymbol{x})$.

$\mathsf{Execute}^{\chi}_{\Pi,F}(\boldsymbol{x};\boldsymbol{\omega})$

$\tau := \emptyset$

$\boldsymbol{S} := \mathsf{Next}^{1}(\Pi, \boldsymbol{x}, \boldsymbol{\omega}, \tau)$ // $\chi = 1$

$\boldsymbol{R} := F(\tau, \boldsymbol{S})$

while $\boldsymbol{R} \neq \bot^{n\times n}$

  $\tau := \tau \circ (\boldsymbol{S}, \boldsymbol{R})$

  $\boldsymbol{S} := \mathsf{Next}^{1}(\Pi, \boldsymbol{x}, \boldsymbol{\omega}, \tau)$ // $\chi = 1$

  $\boldsymbol{R} := F(\tau, \boldsymbol{S})$

return $\left(\mathsf{out}_1(x_1, V_1(\tau)), \ldots, \mathsf{out}_n(x_n, V_n(\tau))\right)$ // $\chi = 1$

return $\begin{pmatrix}\mathsf{indicate}(\mathsf{out}_1(x_1, V_1(\tau))),\\ \ldots,\\ \mathsf{indicate}(\mathsf{out}_n(x_n, V_n(\tau)))\end{pmatrix}$ // $\chi = 2$

$\mathsf{Next}^{1}_{\Pi}(\boldsymbol{x}, \boldsymbol{\omega}, \tau)$

parse $\tau = ((\boldsymbol{S_1}, \boldsymbol{R_1}), \ldots, (\boldsymbol{S_\ell}, \boldsymbol{R_\ell}))$

for $1 \leq i \leq n$ **do**

  if $\tau = \emptyset \ \vee \ \mathbf{col}_i(\boldsymbol{R_\ell})^{\top} \neq \bot^n$

    $\boldsymbol{s_i} := \pi^l_i(x_i, \omega_i, V_i(\tau), \bar{V}_i(\tau))$

  else

    $\boldsymbol{s_i} := \bot^n$

return $\begin{bmatrix}\boldsymbol{s_1}\\ \vdots \\ \boldsymbol{s_n}\end{bmatrix}^{\top}$

Figure 3 Variants of Execute and Next as used in the hybrid distributions. Differences from the original are highlighted in gray. The different versions of Execute used in the hybrids are differentiated by the index χ. Each line where differences exist is marked with a comment indicating for which values of χ this line will be executed.

$\boldsymbol{H_6}$ **:** In hybrid 6 we switch to $\mathsf{Execute}^2$. This means that the execution no longer outputs the actual outputs of the parties. Instead it only indicates which parties produced an output and which aborted. The outputs of all non-aborting parties are then replaced by the outputs of an honest untampered execution of $\Pi(\boldsymbol{x})$. I.e., $H_6 = \mathsf{replace}(\mathsf{Execute}^2_{\Pi^4,F'}(\boldsymbol{x}), \Pi(\boldsymbol{x}))$.

$\boldsymbol{H_7}$ **:** Finally in hybrid 7, we replace the input $\boldsymbol{x}$ of the tampered execution with the arbitrary fixed input $\boldsymbol{x'}$. I.e., $H_7 = \mathsf{replace}(\mathsf{Execute}^2_{\Pi^4,F'}(\boldsymbol{x'}), \Pi(\boldsymbol{x}))$.

We note, that in H_7, the distribution of $\mathsf{Execute}^2_{\Pi^4,F'}(\boldsymbol{x'})$ no longer depends on $\boldsymbol{x}$. I.e., we define D_F as $\mathsf{Execute}^2_{\Pi^4,F'}(\boldsymbol{x'})$ and it is then sufficient to bound that $\mathsf{SD}(H_0, H_7)$ to prove the Lemma. We do so by bounding the statistical distance of each pair of neighboring hybrids.

▷ Claim 19. $\mathsf{SD}(H_0, H_1) \leq 2(n^2 - n) \cdot 2^{-\lambda}$.

Proof. In H_1 we replaced F with the modified tampering function F'. This function is modified such that a series of low probability events (that $o_{i,j}$ for $o \in \{\alpha, \beta\}$ and $i, j \in [n]$ is modified by F) does not happen. Each event happens with probability less than $2^{-\lambda}$. The number of events is bounded by two times the number of edges in the communication graph. This is a directed complete graph, i.e., the number of edges is $n^2 - n$. Hence, by a union bound over all events, the statistical distance between hybrids H_0 and H_1 can be bounded by $2(n^2 - n) \cdot 2^{-\lambda}$. ◁

▷ Claim 20. $\mathsf{SD}(H_1, H_2) = 0$

Proof. In hybrids H_1, H_2, it is easy to see that the only differences between the hybrids are syntactic. I.e., the next message function receives the additional input $\bar{V}_i(\tau)$ in H_2, but does not actually use it yet. Therefore the output distributions remain identical. ◁

▷ Claim 21. $\mathsf{SD}(H_2, H_3) \leq 3n \cdot 2^{-\lambda}$

Proof. In hybrid H_3 we eliminate another series of low probability events. If F' causes any of the parties to abort with overwhelming probability $> (1 - 2^{-\lambda})$ in the first three rounds of

$\pi_i^{d,\mathsf{O}'_{\boldsymbol{x},\boldsymbol{\omega}'}}(\boldsymbol{\omega}, V_i(\tau))$

$\boldsymbol{s} := \bot^n$
if $|V| = 0$
  $\mathsf{rec} := 0^n, \mathsf{sent} := 0^n$
  $(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i) \leftarrow ((\{0,1\}^{\kappa_{nm}})^2)^n$
  $(\boldsymbol{k}_i^{enc}, \boldsymbol{k}_i^{auth}) := \mathsf{nmExt}(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i)$ // $d < 3$
  $(\boldsymbol{k}_i^{enc}, \boldsymbol{k}_i^{auth}) \leftarrow \{0,1\}^{\kappa_{enc}+\kappa_{mac}}$ // $d \geq 3$
  $\boldsymbol{s} := \boldsymbol{\alpha}_i$
elseif $|V| = 1$
  if $\bot \in V_1$ // $d < 2$
  if $\bot \in V_1$ **or** $\gamma_{i,1} > 1 - 2^{-\lambda}$ // $d \geq 2$
    abort
  $\boldsymbol{s} := \boldsymbol{\beta}_i$
elseif $|V| = 2$
  if $\bot \in V_2$ // $d < 2$
  if $\bot \in V_2$ **or** $\gamma_{i,2} > 1 - 2^{-\lambda}$ // $d \geq 2$
    abort
  $(\tilde{\boldsymbol{k}}_j^{enc}, \tilde{\boldsymbol{k}}_j^{auth}) := \mathsf{nmExt}(V_1, V_2)$ // $d < 3$
  $(\tilde{\boldsymbol{k}}_j^{enc}, \tilde{\boldsymbol{k}}_j^{auth}) := \mathsf{replace}(D_f, (\boldsymbol{k}_j^{enc}, \boldsymbol{k}_j^{auth}))$ // $d \geq 3$
  for $j \in [n] \setminus \{i\}$ **do**
    $s_j := \mathsf{MAC}(\tilde{k}_{i,j}^{auth}, (i,j))$
else
  if $|V| = 3$
    if $\bot \in V_3$ // $d < 2$
    if $\bot \in V_3$ **or** $\gamma_{i,3} > 1 - 2^{-\lambda}$ // $d \geq 2$
      abort
    for $j \in [n] \setminus \{i\}$ **do**
      if $\mathsf{Vf}(k_i^{auth}, (j,i), V_{3,j}) = 0$ // $d < 4$
      if $(\tilde{k}_i \neq k_i)$ **or** $\bar{V}_{3,j} \neq V_{3,j}$ // $d \geq 4$
        abort
  else
    $\boldsymbol{m} := \bot^n$
    if $(\mathsf{CheckMsgOrder} = 0)$
      abort
    else
      for $j \in [n] \setminus \{i\}$ **do**
        if $V_{|V|,j} \neq \bot$
          $(c,t) := V_{|V|,j}, \mathsf{rec}_j := \mathsf{rec}_j + 1$
          if $\mathsf{Vf}(k_i^{auth}, (c,j,i,\mathsf{rec}_j), t) = 0$ // $d < 4$
          if $\bar{V}_{|V|,j} \neq V_{|V|,j}$ // $d \geq 4$
            abort
          $m_j := \mathsf{Dec}(\tilde{k}_j^{enc}, c)$
    $\boldsymbol{s}' \leftarrow \mathsf{O}'_{\boldsymbol{x},\boldsymbol{\omega}'}(\boldsymbol{m})$
    for $j \in [n] \setminus \{i\}$ **do**
      if $s'_j \neq \bot$
        $c := \mathsf{Enc}(\tilde{k}_i^{enc}, s'_j), \mathsf{sent}_j := \mathsf{sent}_j + 1$
        $s_j := (c, \mathsf{MAC}(\tilde{k}_j^{auth}, (c,i,j,\mathsf{sent}_j))$
return $\boldsymbol{s}$

Figure 4 The modified next message functions used in the hybrid distributions. Differences from the original are highlighted in gray. The different next message functions used in the hybrids are differentiated by the index d. Each line where differences between next message functions exist is marked with a comment indicating for which values of d this line will be executed.

the protocol, i.e., during key-exchange or key-confirmation, the party now aborts at the same point in time with probability 1. I.e., each eliminated event, i.e. the "non-abort", happens with probability less than $2^{-\lambda}$. The number of eliminated events is bounded by three times the number of parties in the protocol. Therefore a union bound over all eliminated events gives us that the statistical distance between H_2 and H_3 can be bounded by $3n \cdot 2^{-\lambda}$. ◁

▷ Claim 22. $\mathsf{SD}(H_3, H_4) \leq (n^2 - n) \cdot \epsilon_{nm}$.

Proof. For any i, j, let $f_{i,j}$ be the tampering function for $\alpha_{i,j}, \beta_{i,j}$ induced by F. We observe that the changes that were made in H_4 are that rather than using the extracted keys the sender uses uniformly chosen keys while the receiver either receives keys that are distributed according to $D_{f_{i,j}}$ that is *independent* of the actual key, or it receives the same uniformly distributed key used by the sender.

Now, *if* $f_{i,j}$ were split state, then the non-malleability of the extractor would imply that the statistical distance caused by each replaced key can be at most ϵ_{nm}. The main issue is

that $f_{i,j}$ is in fact *not* split state. The tampering function can use both, its bounded state as well as conditional aborts (and non-aborts) of the individual parties to leak information from the first part of the tampering function to the second part and from both parts to the rest of the protocol. However, if we can *bound* the amount of information that can be leaked, then we can change our perspective and look at $f_{i,j}$ as a split state tampering functions, that tampers with sources sampled from a distribution defined by sampling almost uniformly, but conditioned on the leakage.

It remains to actually bound the leakage. Clearly a tampering function in $\mathcal{F}^s_{\mathsf{bounded}}$ can leak s bits simply through its persistent state. Additional leakage is obtained by causing any of the parties to abort or not to abort with low probability. However, due to the elination of low probability events in previous hybrids, we know that each of these events happens with probability at *least* $2^{-\lambda}$. Per party there exist three abort/non-abort events, i.e. the tampering function can leak at most $3n \log \frac{1}{2^{-\lambda}} = 3n\lambda$ additional bits of information.

We can thus reinterpret $f_{i,j}$ as a split-state tampering function on sources with min-entropy $\kappa_{nm} - s - 3n\lambda$. Since, nmExt is specified as working with sources of this type, we have that each replaced key increases the statistical distance by at most ϵ_{nm}. As there are, as mentioned before, $(n^2 - n)$ keys to deal with, we can bound the total statistical distance between the hybrids H_1 and H_2 with $(n^2 - n) \cdot \epsilon_{nm}$. ◁

▷ Claim 23. $\mathsf{SD}(H_4, H_5) \leq \epsilon_{\mathsf{MAC}}$

Proof. Here we bound the statistical distance between the hybrids using a reduction from the statistical unforgeability of the MAC. The output distribution of the two hybrids only differs, if at any point one of the parties receives a ciphertext and tag pair (c, t) such that for some (i, j, r), $\mathsf{Vf}(k^{auth}_{i,j}, (c, i, j, r), t) = 1$ but where none of the parties ever computed $\mathsf{MAC}(k^{auth}_{i,j}, (c, i, j, r))$. That means that the statistical distance between the hybrids is equal to the probability that the above event occurs. We can then construct an attacker $\mathcal{A}$ against the MAC scheme as follows: $\mathcal{A}$ executes H_4 as specified, except that it ignores the actual authentication keys and instead uses the MAC oracle to compute all tags. When the event specified above occurs, $\mathcal{A}$ outputs $(c, i, j, r), t, i$. If the event never occurs, $\mathcal{A}$ aborts. Clearly $\mathcal{A}$ forges a MAC with probability $\mathsf{SD}(H_4, H_5)$. Since the MAC is ϵ_{mac}-statistically unforgeable, we therefore have $\mathsf{SD}(H_4, H_5) \leq \epsilon_{\mathsf{MAC}}$ as claimed. ◁

▷ Claim 24. $\mathsf{SD}(H_5, H_6) = 0$.

Proof. Due to the changes in the previous hybrids, we know that all messages received by any party that does not abort are exactly those messages that were originally sent. Further, whenever a party aborts it does not send any more messages, ensuring that all messages that *are* sent are computed solely based on untampered messages. Additionally, since the protocol has a fixed message topology and both the next message function as well as the output function check that the view conforms to this topology, we know that any party that does not abort computed their output based on a complete view consisting of honestly computed messages that were received in the correct order. I.e., in H_5 the outputs of the *non-aborting* parties are distributed according to the same distribution as in a completely untampered execution of Π on $\boldsymbol{x}$. In H_6, $\mathsf{Execute}^2_{\Pi^4,F}(\boldsymbol{x})$ returns $\perp$ for all aborting parties and same for all non-aborting parties. The function $\mathsf{indicate}$ then replaces the same entries with consistent outputs of an honest execution of $\Pi(\boldsymbol{x})$. Therefore the two distributions are identical. ◁

▷ Claim 25. $\mathsf{SD}(H_6, H_7) = 0$.

Proof. Since the message topology is fixed in both the hybrids, the "shape" of the transcripts of the underyling protocol during the execution in both the hybrids are identical, only the *content* of the messages might differ based on the inputs $\boldsymbol{x}$ and $\boldsymbol{x}'$. However, due to the perfect indistingishability of the stateful encryption scheme, the distribution of the *ciphertexts* is identical. Therefore the distributions of the overall transcripts observed by the tampering function are identical and therefore, so are the output distributions. ◁

Using the triangle inequality over the bounds from Claim 19 through Claim 25 we can thus conclude that

$$\begin{aligned}
&\mathsf{SD}(\mathsf{Execute}_{\Pi,F}(\boldsymbol{x}), \mathsf{replace}(D_F, \Pi(\boldsymbol{x})))\\
&= \mathsf{SD}(\mathsf{Execute}_{\Pi,F}(\boldsymbol{x}), \mathsf{replace}(\mathsf{Execute}^2_{\Pi^4,F}(\boldsymbol{x}'), \Pi(\boldsymbol{x})))\\
&= \mathsf{SD}(H_0, H_7) \leq \sum_{i=1}^{7} \mathsf{SD}(H_{i-1}, H_i) = (2n^2+n)\cdot 2^{-\lambda} + (n^2-n)\cdot \epsilon_{nm} + \epsilon_{\mathsf{MAC}}
\end{aligned}$$ ◀

The theorem finally follows immediately from Lemma 17 and Lemma 18. ◀

Asymmetric Multi-Party Computation

Vipul Goyal
NTT Research, Sunnyvale, CA, USA
Carnegie Mellon University, Pittsburgh, PA, USA

Chen-Da Liu-Zhang
NTT Research, Sunnyvale, CA, USA

Rafail Ostrovsky
University of California at Los Angeles, CA, USA

Abstract

Current protocols for Multi-Party Computation (MPC) consider the setting where all parties have access to similar resources. For example, all parties have access to channels bounded by the same worst-case delay upper bound Δ, and all channels have the same cost of communication. As a consequence, the overall protocol performance (resp. the communication cost) may be heavily affected by the slowest (resp. the most expensive) channel, even when most channels are fast (resp. cheap). Given the state of affairs, we initiate a systematic study of *asymmetric* MPC. In asymmetric MPC, the parties are divided into two categories: fast and slow parties, depending on whether they have access to high-end or low-end resources.

We investigate two different models. In the first, we consider asymmetric communication delays: Fast parties are connected via channels with small delay δ among themselves, while channels connected to (at least) one slow party have a large delay $\Delta \gg \delta$. In the second model, we consider asymmetric communication costs: Fast parties benefit from channels with cheap communication, while channels connected to a slow party have an expensive communication. We provide a wide range of positive and negative results exploring the trade-offs between the achievable number of tolerated corruptions t and slow parties s, versus the round complexity and communication cost in each of the models. Among others, we achieve the following results. In the model with asymmetric communication delays, focusing on the information-theoretic (i-t) setting:

- An i-t asymmetric MPC protocol with security with abort as long as $t + s < n$ and $t < n/2$, in a constant number of slow rounds.
- We show that achieving an i-t asymmetric MPC protocol for $t + s = n$ and with number of slow rounds independent of the circuit size implies an i-t synchronous MPC protocol with round complexity independent of the circuit size, which is a major problem in the field of round-complexity of MPC.
- We identify a new primitive, *asymmetric broadcast*, that allows to consistently distribute a value among the fast parties, and at a later time the same value to slow parties. We completely characterize the feasibility of asymmetric broadcast by showing that it is possible if and only if $2t + s < n$.
- An i-t asymmetric MPC protocol with guaranteed output delivery as long as $t + s < n$ and $t < n/2$, in a number of slow rounds independent of the circuit size.

In the model with asymmetric communication cost, we achieve an asymmetric MPC protocol for security with abort for $t + s < n$ and $t < n/2$, based on one-way functions (OWF). The protocol communicates a number of bits over expensive channels that is independent of the circuit size. We conjecture that assuming OWF is needed and further provide a partial result in this direction.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases multiparty computation, asymmetric, delays, communication

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.6

© Vipul Goyal, Chen-Da Liu-Zhang, and Rafail Ostrovsky;
licensed under Creative Commons License CC-BY 4.0
4th Conference on Information-Theoretic Cryptography (ITC 2023).
Editor: Kai-Min Chung; Article No. 6; pp. 6:1–6:25

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

Funding *Rafail Ostrovsky*: The author was supported in part by DARPA under Cooperative Agreement HR0011-20-2-0025, the Algorand Centers of Excellence programme managed by Algorand Foundation, NSF grants CNS-2246355, CCF-2220450 and CNS-2001096, US-Israel BSF grant 2015782, Amazon Faculty Award, Cisco Research Award and Sunday Group. Any views, opinions, findings, conclusions or recommendations contained herein are those of the author(s) and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, the Department of Defense, the Algorand Foundation, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copyright annotation therein.

1 Introduction

Secure Multi-Party Computation [49, 24, 7, 11, 46] allows a set of distrustful parties to compute a function over their private inputs, in such a way that nothing about the inputs is revealed beyond the output of the computation.

Generally speaking, current MPC protocols consider the simplest setting where all parties have network resources with the same guarantees. In particular, the most common *synchronous* network model considers the setting where all channel delays are upper bounded by a single worst-case delay Δ and all channels have the same cost of communication. Even though this model is theoretically interesting, it suffers from important practical limitations. In particular, Δ has to be set large enough to accommodate any possible delay: Even in cases where almost all parties have fast channels with delay $\delta \ll \Delta$, the protocols do not take advantage of this, and the running time of the protocol is affected by the slowest party. This is particularly critical for information-theoretic protocols, where all current solutions have a round complexity that depends on the depth of the circuit to evaluate. Similarly, the protocols designed in this model also fail to take advantage of the cost of communication from channels that are cheap, and the total communication cost is affected by the most expensive channel.

Given the state of affairs, we initiate the study of *asymmetric* MPC. In asymmetric MPC, the parties are divided into two categories. We consider *fast* parties, which are parties that have access to high-end network resources (e.g. channels with small delay, cheap channels, etc), and *slow* parties, which are parties that have access only to low-end network resources. One can think about the fast parties as parties that are in some sense privileged and have access to fast and cheap internet connection, e.g. with fiber, while slow parties are not so privileged, and only have access to slow and expensive connectivity, e.g. one may think of mobile devices or IoT devices such as sensors collecting data. Considering parties with asymmetric resources allows us to not only model more realistic scenarios, where parties have access to different levels of resources, but also to design more refined protocols that exploit such asymmetries, thereby improving the performance and communication cost of protocols, while at the same time achieving more refined levels of security and assumptions.

2 Our Contributions

We initiate a systematic study of asymmetric MPC with respect to two different models for network resources. In the first model, we consider a network with asymmetric communication delay. Fast parties are connected via channels with small delay δ among themselves, while all other channels, which are connected to a slow party, have a large delay Δ. In the second

model, we consider a network with asymmetric communication cost. This means that fast parties benefit from channels with cheap communication, while other channels incur a high cost of communication.

Our focus is on minimizing the complexity from the slow parties (minimizing the number of slow rounds, respectively the usage of expensive channels) while at the same time tolerating as many corruptions and slow parties as possible. This allows us to give a first overall study of asymmetric MPC protocols in a clean manner.

What are the achievable trade-offs between the number of tolerated corruptions and slow parties compared to the number of slow rounds in the model with asymmetric delays? And similarly, trade-offs with respect to the number of bits transmitted over expensive channels in the model with asymmetric communication cost?

To the best of our knowledge, no previous work has addressed the setting of asymmetric MPC, for any of the resource models. For example, synchronous protocols assume that all channels have the same worst-case delay and also that the cost of communication is the same for all channels. Similarly, asynchronous protocols assume that all channels have *eventual* delivery, and that all channels have the same cost of communication.

Note, however, that existing protocols using standard cryptographic assumptions do achieve a constant number of slow rounds [49, 5], and FHE-based protocols [40, 1, 26] achieve low communication over expensive channels, so our focus is on information-theoretic protocols when minimizing the number of slow rounds, and non-FHE protocols when minimizing the communication over expensive channels. See a more detailed discussion in Appendix A.

2.1 MPC with Asymmetric Delays

The Model. As mentioned above, in this model we divide the parties into two categories: fast and slow parties. Parties have access to a complete network of point-to-point (P2P) channels. The channels between fast parties have a small delay δ and are denoted *fast channels*, and all the other channels, i.e. the channels that contain at least one slow party, have a large delay Δ and are denoted *slow channels*.

We are interested in counting the number of slow P2P-rounds, which are the number of P2P communication steps via all channels, and the number of fast rounds, which are the number of P2P communication steps via the fast channels. Optimally, we would like to find protocols that have a constant number of slow P2P-rounds, or at least a number of slow P2P-rounds that is independent of the circuit to evaluate. Throughout the paper, we will omit the P2P term, and simply denote such rounds as fast and slow rounds.

Existing Solutions. Current constant-round solutions based on cryptographic assumptions [49, 5, 37, 35, 42, 34, 20, 8], are already asymmetric MPC protocols with a constant number of slow rounds. This is because any constant-round synchronous MPC protocol trivially implies an asymmetric MPC protocol in our setting with a constant number of slow rounds.

Information-Theoretic Protocols. Information-theoretic protocols are much more interesting, since all current synchronous solutions require a number of rounds proportional to the depth of the circuit to evaluate. This is in stark contrast with protocols in our model, where we will be able to achieve a number of slow rounds that is independent of the circuit (sometimes even constant). We propose several information-theoretic protocols for the setting of malicious security, with abort and with guaranteed output delivery.

In the following, we let C be the circuit to evaluate, and let n, s, t be the number of parties, bound on the number of slow parties, and bound on the number of corruptions.

Security with Abort. We first present a protocol that achieves security with abort and is secure as long as $t + s < n$ and $t < n/2$. The round complexity is $O(1)$ slow rounds and $O(\mathsf{depth}(C))$ fast rounds.

▶ **Theorem 1.** *Let n, s, t be natural numbers such that $t + s < n$ and $t < n/2$. There is an information-theoretic asymmetric MPC protocol among n parties that securely evaluates circuit C with security with abort, in the presence of up to t malicious corruptions and s slow parties. The round complexity is $O(1)$ slow rounds and $O(\mathsf{depth}(C))$ fast rounds.*

We then show that improving the resiliency of our asymmetric MPC protocol requires a breakthrough. Assume that $t + s = n$. Then, the following result implies that information-theoretic asymmetric MPC in a constant number of slow rounds implies constant-round information-theoretic MPC in the synchronous model resilient up to 1 corruption, which is known to be a major barrier in information-theoretic MPC.

▶ **Theorem 2.** *Let $n, s, t > 0$ be natural numbers such that $t + s = n$. Then n-party information-theoretic asymmetric MPC with security with abort (resp. guaranteed output delivery), resilient up to t corruptions and s slow parties in R slow rounds, implies $(s+1)$-party information-theoretic synchronous MPC with security with abort (resp. guaranteed output delivery), resilient up to 1 corruption in R rounds.*

Guaranteed Output Delivery. In the setting of malicious security with guaranteed output delivery, we present two results.

A Protocol for $2t+s < n$ and $t < n/3$. We start by presenting a solution for an information-theoretic protocol with guaranteed output delivery in the regime where $2t + s < n$ and $t < n/3$ (with no setup nor broadcast).

▶ **Theorem 3.** *Let κ be a security parameter. Let n, s, t be natural numbers such that $2t+s < n$ and $t < n/3$. There is an asymmetric MPC protocol among n parties that securely evaluates circuit C with guaranteed output delivery in the presence of up to t malicious corruptions and s slow parties. The round complexity is $O(\kappa)$ slow rounds and $O(\mathsf{depth}(C) \cdot \kappa)$ fast rounds.*

In the above theorem statement, if synchronous broadcast channels are assumed as setup (or alternatively, a setup for i-t signatures [44]), the condition $t < n/3$ is not necessary. The above protocol inherently requires the condition $2t + s < n$. However, optimally one would wish to require an honest majority *overall*, rather than among the fast parties. We therefore want to find protocols that deal with a dishonest majority among the fast parties.

Broadcast with Asymmetric Delays. To overcome this bound, we identify a natural primitive in our setting with asymmetric delays, called *asymmetric broadcast*. This primitive ensures that all the fast parties obtain the output within d_{bc} fast rounds, while slow parties obtain the same output much later, within D_{bc} slow rounds. We call the quantity d_{bc} the fast asymmetric broadcast delay, and D_{bc} the slow asymmetric broadcast delay.

Our first step is to investigate the possible trade-offs for asymmetric broadcast. Our results completely characterize the feasibility of asymmetric broadcast from point-to-point channels, by showing matching positive and negative results.

First, in Section 5.2, we show a simple construction of n-party asymmetric broadcast with a fast sender, where d_{bc} and D_{bc} are $O(\kappa)$, and security holds up to t corruptions and s slow parties, as long as $2t + s < n$, assuming a PKI setup for signatures. The theorem holds also unconditionally, if the setup consists of information-theoretic signatures [44].

▶ **Theorem 4.** *Let n, s, t be natural numbers such that $2t + s < n$. Assuming a PKI setup for signatures, there is an n-party asymmetric broadcast protocol with $d_{bc} = D_{bc} = O(\kappa)$, tolerating t malicious corruptions and s slow parties.*

Perhaps surprisingly, this is the best trade-off one can achieve, and tolerating $2t + s = n$ for any non-trivial parameters $t > 0$ and $s > 0$ is impossible, even with setup, and for any number of fast and slow rounds.

▶ **Theorem 5.** *Let $n, s, t > 0$ be natural positive numbers such that $2t + s = n$. Then, asymmetric broadcast is impossible against t malicious corruptions and s slow parties, even with setup.*

A Protocol for $t + s < n$ and $t < n/2$. We now present an asymmetric MPC protocol where parties have access to asymmetric broadcast channels with fast and slow delays d_{bc} and D_{bc}, and achieve a protocol that is secure as long as $t + s < n$ and $t < n/2$. The round complexity is $O(n \cdot D_{bc})$ slow rounds and $O(\mathsf{depth}(C) \cdot n \cdot d_{bc})$ fast rounds. In the optimistic case where no party is corrupted, we save a factor n in the round complexity. That is, in this case the protocol incurs $O(D_{bc})$ slow rounds and $O(\mathsf{depth}(C) \cdot d_{bc})$ fast rounds.

▶ **Theorem 6.** *Let n, s, t be natural numbers such that $t + s < n$ and $t < n/2$. Assuming asymmetric broadcast with slow and fast delays D_{bc} and d_{bc}, there is an asymmetric MPC protocol among n parties that securely evaluates circuit C with guaranteed output delivery, in the presence of up t corruptions and s slow parties. The round complexity is $O(n \cdot D_{bc})$ slow rounds and $O(\mathsf{depth}(C) \cdot n \cdot d_{bc})$ fast rounds. In the optimistic case where no party is corrupted, the round complexity is $O(D_{bc})$ slow rounds and $O(\mathsf{depth}(C) \cdot d_{bc})$ fast rounds.*

The number of rounds in the protocol above has a linear dependency in the number of parties in the worst case. This linear dependency can be removed at the cost of requiring any constant fraction of honest parties among the fast parties.

▶ **Corollary 7.** *Let $\epsilon > 0$ and n, s, t be natural numbers such that $t < \min\{(1-\epsilon)(n-s), n/2\}$. Assuming asymmetric broadcast with slow and fast delays D_{bc} and d_{bc}, there is an asymmetric MPC protocol among n parties that securely evaluates circuit C with guaranteed output delivery, in the presence of up to t corruptions and s slow parties. The round complexity is $O(D_{bc})$ slow rounds and $O(\mathsf{depth}(C) \cdot d_{bc})$ fast rounds.*

2.2 MPC with Asymmetric Communication Cost

The Model. Similar to the previous model, we divide the parties into two categories: fast and slow parties. Parties have access to a complete network of standard synchronous point-to-point channels with the same delay upper bound. However, now the channels are differentiated with respect to the cost of communication. The channels between fast parties have a small cost and are denoted by *cheap channels*, and all other channels, i.e. the channels that contain at least one slow party as sender or receiver, have a high cost and are denoted by *expensive channels*.

We are interested in minimizing the number of bits transmitted over the expensive channels. Note that slow parties need to distribute their inputs, so the number of transmitted bits over expensive channels will definitely depend (at least) on the total number of slow parties and their input size. Our main focus is therefore that the number of bits over expensive channels does not depend on the circuit to evaluate.

Existing Solutions. Current solutions that make use of (multi-key) fully-homomorphic encryption (FHE) [40, 1, 26] already transmit a number of bits over expensive channels that is independent of the circuit to evaluate, given that the computation is performed under the homomorphic evaluation with no interaction.

A Protocol From One-Way Functions. Our focus is then on protocols that do not make use of FHE. Even more, we will focus on protocols that make use of as weaker cryptography assumptions as possible (preferably one-way functions, or no assumptions).

We provide a protocol that is resilient as long as $t+s < n$ and $t < n/2$ and achieves security with abort. The protocol communicates $O(\mathsf{poly}(n, \kappa))$ bits over the expensive channels and assumes the existence of one-way functions.

▶ **Theorem 8.** *Let n, s, t be natural numbers such that $t + s < n$ and $t < n/2$. Assuming the existence of one-way functions, there is an asymmetric MPC protocol among n parties that securely evaluates circuit C with security with abort, in the presence of up to t malicious corruptions and s slow parties. The communication complexity is $O(\mathsf{poly}(n, \kappa))$ bits of expensive communication and $O(\mathsf{poly}(n, \kappa)|C|)$ bits of cheap communication.*

Interestingly, our communication-efficient protocol requires the existence of one-way functions, in contrast to our round-efficient protocols. We conjecture that this is necessary, and provide a partial result in this direction.

▶ **Lemma 9.** *Assume an n-party asymmetric MPC, secure up to t semi-honest corruptions and s slow parties such that the slow parties perform constant computation, for any $n, s, t > 0$ natural numbers such that $n - s > 1$, $t + s < n$ and $t < n/2$. Then, this implies the existence of one-way functions.*

2.3 Open Questions

Our work initiates the area of asymmetric MPC and leaves several exciting new directions for future research. We highlight some of them:

1. General network topologies: We divide the parties into slow and fast parties, where fast parties have high-end channels among themselves, while all other channels are low-end. One can generalize this setting and consider more general network topologies, where a party may at the same time have some high-end and some low-end channels.
2. Other resource asymmetries: Our work considers models with respect to network asymmetries. One can explore other types of resources. For example, one can explore trade-offs with respect to asymmetric computation or memory resources.
3. Communication-efficient protocol with guaranteed output delivery: Our communication efficient protocols achieve security with abort. It would be interesting to see if our techniques extend to the case of guaranteed output delivery.
4. Concrete efficiency: Our protocols serve as a basis for feasibility of asymmetric MPC, and our focus is on minimizing the complexity coming from the slow parties. In practice, one may consider more refined settings where the cost difference is explicit (for example, expensive channels cost as twice as cheap channels). Understanding which concrete cost differences are relevant in practice and designing protocols tailored to such settings is an exciting open research direction.

3 Technical Overview

In this section we describe the techniques used to achieve our theorem statements.

3.1 Information-Theoretic MPC with Asymmetric Delays

We present several information-theoretic solutions for the settings of security with abort and guaranteed output delivery.

Security with Abort. We provide a simple solution that achieves security with abort in a constant number of slow rounds, when $t + s < n$ and $t < n/2$. The protocol lets all parties generate during a pre-processing phase OT correlations among the fast parties, using an information-theoretic MPC protocol for honest majority and security with abort, with round complexity linear in the depth of the circuit (e.g. [46, 21]). Since the OTs can be generated in parallel and can be computed with a constant-depth circuit, this takes a constant number of slow rounds. Then, the parties execute an information-theoretic MPC protocol in the client-server model, that achieves security with abort against a dishonest majority in the OT-hybrid model: The parties (acting as clients) distribute their inputs to the fast parties (acting as servers), who will compute the corresponding outputs and send them back to the respective parties. Existing protocols [36, 15] run in a constant number of rounds during the input and output phases, and the number of rounds during the computation is linear in the depth of the circuit to evaluate. This leads to Theorem 1.

We then show that our protocol achieves the optimal resilience: any asymmetric MPC protocol secure when $t + s = n$ among n parties and with constant number of slow rounds, implies a synchronous MPC protocol with security with abort, resilient up to 1 corruption in constant number of rounds, which is a major open problem in round complexity of MPC. The proof implication follows from a simple emulation argument: In order to design a synchronous MPC protocol among $s + 1$ parties, we simply let s parties emulate each of the slow parties in the asymmetric MPC protocol, and the last party to emulate all fast parties altogether (in total t parties). The resulting synchronous protocol has a round-complexity that is the same as the number of slow rounds in the asymmetric MPC protocol, and therefore the implication follows. This corresponds to Theorem 2.

Asymmetric Broadcast. We briefly sketch the arguments that exactly characterize the feasibility of asymmetric broadcast.

First observe that it is clear that for a sender that is a slow party, it is impossible to expect the fast parties to obtain output fast, even when all parties are honest. Therefore, we focus on the case where the sender is a fast party.

The protocol to achieve asymmetric broadcast (for a fast sender), for any $2t + s < n$, assuming a PKI for signatures, is quite simple: Fast parties run a synchronous broadcast protocol [33] among themselves, and reach agreement on a value v. All fast parties send the value v to the slow parties, who take a majority decision. Since there is honest majority among the fast parties, all slow parties output the same value. This is presented in Theorem 4.

In order to show that asymmetric broadcast (for a fast sender) is impossible when $2t + s = n$, even with setup, we make use of two ideas. First, observe that fast parties must output before the slow parties are even able to communicate with any fast party. This is because fast rounds may be much faster than slow rounds, i.e., the delay $\delta \ll \Delta$ of fast channels could be much smaller than the delay Δ of slow channels, and asymmetric broadcast requires fast parties to output fast. Let v denote the value that the fast parties output.

Second, since there is a dishonest majority among the fast parties, the corrupted parties (including the sender) can simulate towards the slow parties an execution with input value $v' \neq v$. As a consequence, the honest slow parties cannot decide on a consistent output value. A precise scenario-based proof is presented in Theorem 5.

Guaranteed Output Delivery. We present two results. We first present a somewhat simple solution for an information-theoretic protocol in the regime where $2t + s < n$ and $t < n/3$. The protocol works as follows: First, since $t < n/3$, the protocol generates a setup for information-theoretic signatures to emulate synchronous broadcast channels with guaranteed termination [44, 33] from slow parties to all parties in $O(\kappa)$ slow rounds, and from fast parties to themselves in $O(\kappa)$ fast rounds, where κ is the security parameter. Using these broadcast channels, parties can execute an existing synchronous protocol in the client-server model as follows: All parties initially play the role of a client, while each fast party in addition plays the role of a server. The clients distribute their inputs towards the $n - s$ servers, where each synchronous round corresponds to a slow round. The servers then perform the protocol computation, where each synchronous round corresponds to a fast round. Finally, the fast parties robustly reconstruct each output to the respective clients, where each synchronous round corresponds to a slow round. Standard information-theoretic protocols [46, 14] tolerate up to half of the corrupted servers (we assume $2t < n - s$) and any number of clients, and have a constant number of rounds and broadcast invocations, during the input and the output phase, and a number of rounds proportional to the circuit depth during the computation phase. This results in an asymmetric MPC protocol with $O(\kappa)$ slow rounds and a number of fast rounds proportional to the circuit depth times κ, corresponding to Theorem 3.

We then present our information-theoretic asymmetric MPC protocol with guaranteed output delivery, and resilience $t + s < n$ and $t < n/2$, which assumes asymmetric broadcast.

The protocol follows the sharing-based paradigm, and has a preprocessing phase and an online phase. During the preprocessing phase, the parties generate raw data that is independent of the inputs. During the online phase, the parties receive their inputs and perform the protocol evaluation.

In the preprocessing phase, we generate *certified* Beaver multiplication triples, using the MPC protocol by Cramer et. al. [14].

Background. Let us first recap their VSS protocol Π_{vss} [14]. The protocol follows traditional verifiable secret sharing schemes [19, 7] with bivariate polynomials, but uses so-called *information-checking* (IC) signatures, instead of error correction. One can think about such signatures as information-theoretic signatures that can only be forwarded once. These can be generated unconditionally without setup, and also have a *linearity* property, where given signatures for values x and y, one can compute a signature on $x + y$.

In order to share a value v, the dealer D creates a random bivariate polynomial $f(x, y)$ of degree at most t, with $f(0, 0) = v$. The univariate polynomial projections $f(x, i)$ and $f(i, y)$ are sent to party P_i in a signed manner (by sending all the points $(a_{i1}, \ldots, a_{in}) = (f(i, 1), \ldots, f(i, n))$ and $(b_{1i}, \ldots, b_{ni}) = (f(1, i), \ldots, f(n, i))$, where each point is signed using IC-signatures). After this, the parties can bilaterally compare the cross-point values between them, and expose inconsistent behavior by the dealer by broadcasting the signatures. If an inconsistency is detected, the dealer is disqualified.

After the checking process, the values held by honest parties are consistent, and since there are at least $n - t \geq t + 1$ honest parties, these values uniquely define a bivariate polynomial $f'(x, y)$ of degree at most t, which in turn defines a fixed secret v' (which is $v' = v$ if the dealer is honest). Therefore, this already ensures that the dealer is committed to a value after the sharing phase.

Still, the reconstruction might fail if the adversary sends corrupted shares (the adversary can send arbitrary shares). To avoid that, each share of P_i is also signed by the other parties. This will in turn prevent the adversary from corrupting the secret at reconstruction time.

At the end of the VSS, each party P_i holds sub-shares $(a_{i1}, \ldots, a_{in})$, where a_{ij} is signed by P_j. This implicitly defines a share a_i, which in the case of an honest dealer is $f(i, 0)$.

With the above VSS, one can process addition gates locally (using the fact that the IC-signatures are linear). The multiplication gates are processed using the well-known method by Gennaro, Rabin and Rabin [22]: Each party P_i locally multiplies his shares a_i and b_i of the input wires a and b, and shares the result $d_i = a_i b_i$ using VSS. This results in n VSSs and a proper sharing of the output wire c can be computed as a fixed linear combination of these. The authors show a way for P_i to share a secret d_i, such that $d_i = a_i b_i$ and to prove that he has done so properly. The details can be found in [14].

The Online Protocol. At the start of the online phase, enough triples (x, y, z) have been shared using the protocol described above, where each of the shares x_i, y_i, z_i are held (implicitly) by party P_i via the corresponding sub-shares, which are IC-signed by the other parties. Note that generating such certified Beaver triples takes $O(1)$ invocations of broadcast, since they can be generated in parallel.

The online phase proceeds as follows. Parties distribute their inputs using Π_{vss}. The addition gates are locally computed (simply adding the shares and the IC-signatures, since they are linear). In the multiplication gates, fast parties publicly open two random values, $(a - x)$ and $(b - y)$, where a and b are the values of the input wires to the multiplication gate, by running the same reconstruction procedure of Π_{vss}, except that they distribute their shares using asymmetric broadcast. Since the values are IC-signed, corrupted fast parties can only withhold their shares. Since fast parties distribute their shares via the asymmetric broadcast channel, this implies that all parties, fast and slow, reach agreement on the set of parties that did not contribute their share and are corrupted.

Note that since the threshold is $t < n - s$, if all fast parties contribute their shares, the slow parties do not need to participate (and the protocol can proceed between between the fast parties, without incurring additional slow rounds). However, if not all shares are received, a process to identify and kick out corrupted parties is performed: fast parties wait for the slow parties to help opening the shares (note that $n - t > t$, and therefore all honest parties can jointly open the shares). The corrupted identified parties are then kicked out of the computation, and the protocol is restarted without the kicked parties. This process incurs and overhead of a constant number of slow asymmetric broadcast delays. And since every time at least one corrupted party is kicked out, the incurred overhead on the total number of slow rounds is linear in the number of parties. This corresponds to Theorem 6.

3.2 MPC with Asymmetric Communication Cost

We describe the protocol for MPC that communicates $O(\mathsf{poly}(n, \kappa))$ bits over slow connections, and achieves resilience $t + s < n$ and $t < n/2$. The protocol is based on one-way functions, and is similar to the simple protocol mentioned in Theorem 1 in the asymmetric delay model.

In that protocol, the step that is communication expensive, is the generation of the OT correlations, which depends on the circuit size. In order to solve that, we will make use of OT-extension protocols [41, 2], which can be based on one-way functions. More concretely, since $t < n/2$, parties can jointly create κ OT correlations among each pair of fast parties using an honest-majority MPC protocol [46, 14]. This step communicates $O(\mathsf{poly}(n, \kappa))$ bits over slow connections. The fast parties then perform an OT-extension protocol to set up

an OT channel between each pair of fast parties [41, 2]. With this setup, parties can then perform an unconditional protocol achieving dishonest majority in the OT-hybrid model [36, 15] among the fast parties. This is stated in Theorem 8.

The protocol described above makes use of one-way functions. We conjecture that this is necessary, and provide a partial result: we show that any asymmetric MPC protocol containing at least two fast parties, where the slow parties perform little computation, and with resilience $t + s < n$, implies the existence of one-way functions.

The high-level idea is to build an OT extension protocol from an asymmetric MPC protocol. Since OT extension implies the existence of one-way functions [38], the claim follows. Assume that there is an asymmetric MPC protocol that outputs a large number of OTs. We can emulate the computation of each slow party using a protocol for dishonest majority (e.g. [24]). Note that since each slow party performs a small amount of computation, the circuit that is used to emulate the computation uses a small number of (seed) OTs as well. This is stated in Lemma 9.

4 Models and Definitions

We consider a set of n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$. We partition the set of parties into two known categories, slow parties and fast parties, $\mathcal{P} = \mathcal{S} \sqcup \mathcal{F}$. Let κ be the security parameter.

4.1 Communication Network and Adversary

We consider a complete network of point-to-point secure channels. Parties have access to synchronized clocks, and messages sent by honest parties are guaranteed to be delivered within some known upper bound delay. We consider two asymmetric network models.

Network with Asymmetric Delays. In the first model, we consider a network with asymmetric delays. The channels between fast parties deliver messages within a small delay δ, and are denoted fast channels. And all channels containing at least one slow party have a large delay Δ, and are denoted slow channels. We measure the round complexity as the number of slow P2P-rounds (communication steps via all channels), and the number of fast P2P-rounds (communication steps via fast channels). We will omit mentioning the P2P term, and simply denote such rounds as fast and slow rounds.

Network with Asymmetric Communication Cost. In the second model, we consider a network with asymmetric communication cost. Here, all the channels have the same delay upper bound, similar to the standard synchronous network model, but the cost of communication is asymmetric. We will consider expensive communication, the number of bits transmitted via channels that contain at least one slow party, and cheap communication, the number of bits transmitted via channels that contain only fast parties.

Adversary. We consider a static adversary who corrupts parties in an arbitrary manner at the beginning of the protocol.

4.2 Broadcast

Broadcast allows a designated party called the *sender* to consistently distribute a message among a set of receivers.

Synchronous Broadcast. The synchronous broadcast channel with guaranteed termination delivers the output to the set of receivers after a fixed number of rounds.[1] Synchronous broadcast protocols with guaranteed termination can be achieved within $O(\kappa)$ rounds, when there are up to a third fraction of corrupted parties [19]. This is also the case for honest majority, if a setup is available [33]. In the dishonest majority setting, synchronous broadcast is achievable in $O(n)$ rounds with a PKI setup [18], and even unconditionally with a setup for information-theoretic signatures [44].

These protocols, when run in the asymmetric network delay model, achieve an actual number of rounds that is proportional to the slowest channel. This means, that if all the parties involved (sender and receivers) are connected via fast channels, the output is received after a fixed number of fast rounds. However, when some of the channels between the considered parties are slow, the protocols guarantee that the receivers obtain the output in a fixed number of slow rounds.

4.3 Secret Sharing

In some of our protocols, we make use of Shamir secret sharing scheme [48]. This is a t-out-of-n linear secret-sharing scheme over a finite field $\mathcal{F}$, consisting of two protocols, $(\texttt{Sh}, \texttt{Rec})$, called share and reconstruct.

Protocol `Sh` allows a designated party, called the dealer, to distribute a value $s \in \mathcal{F}$ among n parties, $P_1, \ldots, P_n$. For that, the dealer samples a uniform random polynomial $f \in \mathcal{F}[x]$ with degree at most t, and subject to the fact that $f(0) = s$. Then, the dealer sends the value $f(i) = s_i$ to P_i. We denote s_i the share of P_i, and the vector $[s]_t = (s_1, \ldots, s_n)$ is called a degree-t sharing of s. We may omit the degree if it is clear from the context. Note that any set of t shares does not reveal anything about the secret. Protocol `Rec` allows parties to jointly reconstruct a secret s', which corresponds to the original secret s if the dealer is honest. Shamir secret sharing scheme satisfies in addition the following properties:

- Additive Homomorphism: $\forall [x]_t, [y]_t,\ [x+y]_t = [x]_t + [y]_t$.
- Local Multiplication of Degree-t Sharings: $\forall [x]_t, [y]_t,\ [x \cdot y]_{2t} = [x]_t \cdot [y]_t$.

4.4 Oblivious Transfer

Oblivious transfer [45] is a two-party primitive between a sender S, and a receiver R. The sender has two inputs $x_0, x_1 \in \{0,1\}$, called the messages, and the receiver R has an input $c \in \{0,1\}$, called the selection bit. The oblivious transfer guarantees that R outputs $x_c = c(x_0 \oplus x_1) \oplus x_0$, and that no party learns any other information.

5 MPC with Asymmetric Delays

In this section we introduce protocols in the model with asymmetric delays. We are interested in protocols that incur as few slow rounds as possible, preferably a constant, and tolerating a high number of corruptions and slow parties.

[1] There are also protocols with probabilistic termination [19, 33], where the parties obtain output after an *expected-constant* number of rounds. However, composing such protocols involves many subtleties. See for example [13] for a nice discussion.

5.1 Security with Abort

Protocol Description. We present a naive protocol that achieves security with abort in a constant number of slow rounds, when $t + s < n$ and $t < n/2$.

The protocol lets all parties generate during a pre-processing phase OT correlations among the fast parties, using a (synchronous) information-theoretic MPC protocol for honest majority and security with abort (e.g. [46, 21]), with round complexity linear in the circuit depth to evaluate. Note that since the OTs can be generated in parallel and can be computed with a constant-depth circuit, this is possible in a constant number of slow rounds. Then, the parties execute an information-theoretic MPC protocol in the client-server model, that achieves security with abort against a dishonest majority in the OT-hybrid model: All parties act as clients and distribute their inputs to the fast parties, who also act as servers. The fast parties will then compute the corresponding outputs and send them back to the respective parties. Existing protocols [36, 15] run in a constant number of rounds during the input and output phases, and the number of rounds during the computation is linear in the depth of the circuit to evaluate. As a consequence, the overall protocol incurs a constant number of slow rounds, and a number of fast rounds proportional to the depth of the circuit. This leads to the following theorem, and the proof follows from the security of [46, 21] and [15].

▶ **Theorem 1.** *Let n, s, t be natural numbers such that $t + s < n$ and $t < n/2$. There is an information-theoretic asymmetric MPC protocol among n parties that securely evaluates circuit C with security with abort, in the presence of up to t malicious corruptions and s slow parties. The round complexity is $O(1)$ slow rounds and $O(\mathsf{depth}(C))$ fast rounds.*

Barrier Result. Our result shows that improving the resiliency achieved by the protocols in the above sections would be a breakthrough in the area of information-theoretic synchronous MPC. In particular, if there is an information-theoretic protocol that is constant in the number of slow rounds for $t + s = n$, this implies a constant-round information-theoretic synchronous MPC protocol secure up to 1 corruption.

▶ **Theorem 2.** *Let $n, s, t > 0$ be natural numbers such that $t + s = n$. Then n-party information-theoretic asymmetric MPC with security with abort (resp. guaranteed output delivery), resilient up to t corruptions and s slow parties in R slow rounds, implies $(s+1)$-party information-theoretic synchronous MPC with security with abort (resp. guaranteed output delivery), resilient up to 1 corruptions in R rounds.*

Proof. Let Π be the protocol for fast and slow parties with R slow rounds. We want to construct a synchronous protocol Π' among $s + 1$ parties with R rounds.

Let us call the $s + 1$ *virtual* parties $P_1, \ldots, P_{s+1}$. Each party P_i, $i \in [s]$ emulates a slow party, and the last party P_{s+1} emulates all the fast parties. The parties then execute the protocol Π, where the messages between fast parties are emulated internally by P_{s+1}.

The resulting protocol Π' is a secure synchronous protocol up to 1 corruption and with R rounds. This follows from the fact that Π is a secure protocol tolerating t corruptions, and each virtual party contains at most t parties from Π. ◀

5.2 Broadcast with Asymmetric Delays

The *asymmetric* broadcast channel guarantees the delivery of a consistent message fast to the fast parties and slow to the slow parties. More precisely, an asymmetric broadcast channel achieves guaranteed output after D_{bc} slow rounds for slow parties, and d_{bc} fast rounds for

the fast parties. We will make use of this channel in the protocol for guaranteed output delivery with resilience $t + s < n$ and $t < n/2$, but in this section we study the feasibility of this primitive from a complete network of point-to-point channels as a stand-alone question.

Functionality $\mathcal{F}_{\mathsf{sBC}}$

1: On input x from the sender P^*, output x to the adversary. Then, output x to all parties in $\mathcal{S}$ after D_{bc} slow rounds and all parties in $\mathcal{F}$ after d_{bc} fast rounds.

First, note that when the sender is slow, it is impossible to achieve asymmetric broadcast, since one needs at least a slow round to distribute the value towards the fast parties. Therefore, in the following we focus on the more interesting case where the sender is a fast party.

Feasibility. Assuming setup, if $2t + s < n$, it is easy to see that asymmetric broadcast with a fast sender is achievable. The protocol proceeds as follows: The sender with input s uses a synchronous broadcast protocol to distribute his value among all the fast parties. Since there is an honest majority of fast parties, one can for example use the protocol by Katz and Koo [33]. All the fast parties reach agreement on a value s' (which is s if the sender is honest) within $O(\kappa)$ fast rounds, and they send their value to the slow parties, who will take a majority decision and output the result. Therefore, the slow parties output after $O(\kappa)$ fast rounds, and 1 slow round.

▶ **Theorem 4.** *Let κ be a security parameter. Further let n, s, t be natural numbers such that $2t + s < n$. Assuming a PKI setup for signatures, there is an n-party asymmetric broadcast protocol with $d_{bc} = D_{bc} = O(\kappa)$, tolerating t malicious corruptions and s slow parties.*

The protocol can be achieved with unconditional security, if a setup for information-theoretic signatures is assumed [44].

Impossibility. We show that asymmetric broadcast is impossible when $2t + s = n$ for a fast sender (even with setup), for any non-trivial parameters $t > 0$ and $s > 0$. Note that this is in contrast to synchronous broadcast, which is achievable for any number of corruptions assuming a PKI setup [18]. (Or even unconditionally, assuming information-theoretic signatures [44].) The main challenge is to achieve agreement between fast and slow parties. Intuitively, since asymmetric broadcast requires fast parties to obtain the output fast, they need to decide their output value (let us denote it v) without having received any value from the slow parties. Moreover, since there is dishonest majority among the fast parties, they can act towards the slow parties as if their output value was $v' \neq v$. As a consequence, the honest slow parties will not output v and consistency is broken. The proof of the following theorem can be found in Appendix B.

▶ **Theorem 5.** *Let $n, s, t > 0$ be natural positive numbers such that $2t + s = n$. Then, asymmetric broadcast is impossible against t malicious corruptions and s slow parties, even with setup.*

5.3 Guaranteed Output Delivery

In this section, we present two protocols. The first protocol achieves a lower resilience, but operates only assuming point-to-point channels. The second protocol has a higher resilience, but makes use of asymmetric broadcast.

Protocol for $2t + s < n$ and $t < n/3$. In the regime where $2t + s < n$ and $t < n/3$, it is easy to design an asymmetric MPC protocol, by simply delegating the computation to the fast parties. Note that since $t < n/3$, the parties can create a setup of information-theoretic signatures, which can be used to construct a synchronous broadcast channel [44, 33] from slow parties to all parties in $O(\kappa)$ slow rounds, and from fast parties among themselves in $O(\kappa)$ fast rounds. With the emulated synchronous broadcast channels, parties can then execute an honest majority protocol such as [46] as follows: the slow parties, using the emulated broadcast channel, use a verifiable secret sharing scheme to share their input towards the fast parties (with threshold t), who will robustly evaluate the circuit among themselves. The fast parties can then robustly reconstruct the output towards the respective recipients. The total round complexity is $O(\kappa)$ slow rounds and $O(\mathsf{depth}(C) \cdot \kappa)$ fast rounds. The proof of the following theorem follows from the security of [46] and [33].

▶ **Theorem 3.** *Let κ be a security parameter. Let n, s, t be natural numbers such that $2t+s < n$ and $t < n/3$. There is an asymmetric MPC protocol among n parties that securely evaluates circuit C with guaranteed output delivery in the presence of up to t malicious corruptions and s slow parties. The round complexity is $O(\kappa)$ slow rounds and $O(\mathsf{depth}(C) \cdot \kappa)$ fast rounds.*

We note that if a synchronous broadcast channel is given (or alternatively a setup for information-theoretic signatures), the condition $t < n/3$ is not necessary.

Protocol for $t + s < n$ and $t < n/2$. In this section, we present a protocol that achieves guaranteed output delivery with the higher trade-off $t + s < n$ and $t < n/2$. The resulting protocol has round complexity $O(n \cdot D_{bc})$ slow rounds and $O(\mathsf{depth}(C) \cdot n \cdot d_{bc})$ fast rounds, assuming asymmetric broadcast.

Generating Certified Beaver Triples. We generate Beaver multiplication triples (a, b, c), that are shared among all parties. The triples are certified, in the sense that all shares are signed via *information-checking* signatures. There can be thought of as signatures that are information-theoretic, and can only be forwarded once. The signatures are also homomorphic, in the sense that for two values that can be verified by the scheme, any linear combination of them can also be verified with no additional information. See [14] for a concrete construction.

We generate the triples using the protocol by Cramer et al. [14], which makes use of such a IC-signature scheme. The protocol takes $O(1)$ invocations to asymmetric broadcast (incurring a total of $O(D_{bc})$ slow rounds), since the multiplication triples can be generated in parallel. In this protocol, a value s is shared using a bivariate polynomial $f(x, y)$ with degree at most t. At the end of the sharing protocol, each honest party P_i holds the values $s_{i1}, \ldots, s_{in}$ that lie on a degree-t polynomial, which in the case the dealer is honest, corresponds to the values $f(i, 1), \ldots, f(i, n)$. This implicitly defines the share of P_i, which is $s_i = f(i, 0)$. Moreover, each value s_{ij} is signed by party P_j, and we denote such a signature (from P_j to P_i) by $\sigma(s_{ij}, P_j, P_i)$. This signature allows P_i to forward the value s_{ij} in an authentic way. In Appendix C, we recap the protocol in detail.

The Online Phase. At the start of the online phase, enough triples (a, b, c) have been shared using the protocol in [14]. This means, that each party P_i implicitly holds each of the shares a_i, b_i, c_i via the corresponding sub-shares, which are signed by the other parties.

The online phase proceeds as follows. Parties distribute their inputs using Π_{vss}, the VSS scheme in [14]. The addition gates can locally be computed (simply by locally adding the shares and locally adding the signatures, since they are linear). In the multiplication gates,

fast parties robustly open two random values, $(x-a)$ and $(y-b)$, where x and y are the values of the input wires to the multiplication gate, by running the same reconstruction procedure of Π_{vss}, except that they distribute their shares using asymmetric broadcast. Since the values are signed, corrupted fast parties can only withhold their shares. Since fast parties distribute their shares via the asymmetric broadcast channel, this implies that all parties, fast and slow, reach agreement on the set of parties that did not contribute their share and are corrupted. Note that since the threshold is $t < n - s$, if all fast parties contribute their shares, the slow parties do not need to participate (and therefore we do not need to incur additional slow rounds). However, if not all shares are received, a process to identify and kick out corrupted parties is performed: fast parties wait for the slow parties to help opening the shares (note that $n - t > t$, and therefore all honest parties can jointly open the shares). The corrupted identified parties are then kicked out of the computation, and the protocol is restarted without the kicked parties. This process involves $O(D_{bc})$ slow rounds. And since every time at least one corrupted party is kicked out, the incurred total number of slow rounds is linear in the number of parties times the broadcast slow delay. We formally describe the protocol in Appendix D, and a proof of the following theorem appears in Appendix E.

▶ **Theorem 6.** *Let n, s, t be natural numbers such that $t + s < n$ and $t < n/2$. Assuming asymmetric broadcast with slow and fast delays D_{bc} and d_{bc}, Π_{rgod} is an asymmetric MPC protocol among n parties that securely evaluates circuit C with guaranteed output delivery, in the presence of up t corruptions and s slow parties. The round complexity is $O(n \cdot D_{bc})$ slow rounds and $O(\mathsf{depth}(C) \cdot n \cdot d_{bc})$ fast rounds. In the optimistic case where no party is corrupted, the round complexity is $O(D_{bc})$ slow rounds and $O(\mathsf{depth}(C) \cdot d_{bc})$ fast rounds.*

Although the number of slow rounds is independent of the depth of the circuit, it has the drawback that it depends on the number of parties in the worst case. However, if we assume that among the fast parties there is a constant fraction of parties that are honest, then we can modify the above protocol to achieve round complexity $O(D_{bc})$ slow rounds and $O(\mathsf{depth}(C) \cdot d_{bc})$ fast rounds. Then, the adversary needs $\epsilon(n-s)$ corrupted parties to not send messages in order to execute Steps 3 and 4 in a multiplication step, which will be all identified and kicked out of the computation. This can only happen at most $1/\epsilon$ times.

▶ **Corollary 7.** *Let $\epsilon > 0$ and n, s, t be natural numbers such that $t < \min\{(1-\epsilon)(n-s), n/2\}$. Assuming asymmetric broadcast with slow and fast delays D_{bc} and d_{bc}, there is an asymmetric MPC protocol among n parties that securely evaluates circuit C with guaranteed output delivery, in the presence of up to t corruptions and s slow parties. The round complexity is $O(D_{bc})$ slow rounds and $O(\mathsf{depth}(C) \cdot d_{bc})$ fast rounds.*

6 MPC with Asymmetric Communication Cost

In this section we introduce protocols in the model with asymmetric communication cost. We are interested in protocols that transmit as few bits as possible over expensive channels (channels containing at least one slow party), and that tolerate a high number of corrupted parties and slow parties. Looking closer, our protocol from Theorem 3 already achieves small expensive communication, independent of the circuit since expensive communication only occurs in the input and output stages. However, the resiliency is only $2t + s < n$. Therefore, we turn our attention to protocols achieving resiliency $t + s < n$ and $t < n/2$.

6.1 Security with Abort

We provide a protocol that achieves security with abort and communicates $O(\mathsf{poly}(n,\kappa))$ bits over the expensive channels. The protocol assumes the existence of one-way functions, and follows from existing results. The idea is to generate pair-wise OT correlations among the fast parties during a pre-processing phase. This is possible [46, 21] since there is an honest majority $t < n/2$ of parties, and the total number of communicated bits over expensive channels is independent of the circuit size. More concretely, all parties will prepare κ OT correlations per fast connection (in total $O(n^2\kappa)$ OTs[2]. Each pair of fast parties can then use OT extension protocols [4, 32, 41] to set up OT channels between them. Once the OT channels are prepared, we can execute a standard unconditional OT-based protocols for dishonest majority [36, 15] to perform the computation among the fast parties, and deliver the outputs to all the parties, leading to the following theorem statement.

▶ **Theorem 8.** *Let n, s, t be natural numbers such that $t + s < n$ and $t < n/2$. Assuming the existence of one-way functions, there is an asymmetric MPC protocol among n parties that securely evaluates circuit C with security with abort, in the presence of up to t malicious corruptions and s slow parties. The communication complexity is $O(\mathsf{poly}(n,\kappa))$ bits of expensive communication and $O(\mathsf{poly}(n,\kappa)|C|)$ bits of cheap communication.*

6.2 Barriers on Communication Complexity

In this section, we show that if one assumes an asymmetric MPC, where the slow parties perform a small amount of computation, then this implies OT extension.

▶ **Lemma 9.** *Assume an n-party asymmetric MPC, secure up to t semi-honest corruptions and s slow parties such that the slow parties perform constant amount of computation, for any $n, s, t > 0$ natural numbers such that $n - s > 1$, $t + s < n$ and $t < n/2$. Then, this implies the existence of one-way functions.*

Proof. The proof strategy is to build a semi-honest OT extension protocol from a semi-honest asymmetric MPC protocol. Since OT extension for semi-honest corruption implies the existence of one-way functions [38], the claim follows. Let c the total circuit size to represent the computation of the slow parties (this has polynomial size). Assume that there is an n-party asymmetric MPC protocol Π with at least 2 fast parties and secure up to $t = n - s - 1$ and $t < n/2$. Further assume that the protocol outputs $c + 1$ OT correlations. We first describe an $(n - s)$-party reactive functionality $\mathcal{F}_{\mathsf{slow}}$ that emulates the computation of the slow parties. Concretely, this is a reactive functionality which keeps the internal joint state of the slow parties, and which, upon giving inputs from the fast parties, it updates its internal joint state and computes the outputs to the fast parties according to the protocol description Π for the slow parties. Now we create a two-party protocol Π' for OT extension, where both parties have access to the reactive functionality described above, and where: P_1 emulates $n - s - 1$ fast parties, and P_2 emulates the 1 remaining fast party. The two parties execute the asymmetric MPC protocol, where any interaction with the slow parties is performed via the reactive functionality $\mathcal{F}_{\mathsf{slow}}$ instead. First, note that the reactive functionality can be realized using a dishonest majority synchronous MPC protocol such as GMW among

[2] This can be reduced to $O(n\kappa)$ OTs if there is a constant fraction of honest fast parties, i.e. $t < (1-\epsilon)(n-s)$, for $\epsilon > 0$ constant, using the results of Harnik, Ishai and Kushilevitz [30], which combines from distributing computations among several committees from Bracha [10], techniques for combining oblivious transfers from Harnik et al. [31], and constructions of dispersers [25, 47])

the fast parties. Further note that since we are assuming that the slow parties performs computation represented by a total circuit size c. This means that the number of OTs needed to realize $\mathcal{F}_{\mathsf{slow}}$ is c. Finally, in terms of security, note that since the asymmetric MPC protocol tolerates $t = n - s - 1$ corruptions, the resulting two-party protocol tolerates 1 corruption. In conclusion, the resulting two-party protocol where the reactive functionality is emulated among the two fast parties is a secure OT extension protocol against a semi-honest static adversary, which implies one-way functions. This concludes the proof. ◀

References

1 Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_29.

2 Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer extensions with security for malicious adversaries. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 673–701. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46800-5_26.

3 Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Secure MPC: Laziness leads to GOD. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 120–150. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64840-4_5.

4 Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488. ACM Press, May 1996. doi:10.1145/237814.237996.

5 Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990. doi:10.1145/100216.100287.

6 Zuzana Beerliová-Trubíniová and Martin Hirt. Perfectly-secure MPC with linear communication complexity. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 213–230. Springer, Heidelberg, March 2008. doi:10.1007/978-3-540-78524-8_13.

7 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988. doi:10.1145/62212.62213.

8 Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78375-8_17.

9 Erica Blum, Jonathan Katz, Chen-Da Liu-Zhang, and Julian Loss. Asynchronous byzantine agreement with subquadratic communication. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 353–380. Springer, Heidelberg, November 2020. doi:10.1007/978-3-030-64375-1_13.

10 Gabriel Bracha. An o (log n) expected rounds randomized byzantine generals protocol. *Journal of the ACM (JACM)*, 34(4):910–920, 1987.

11 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988. doi:10.1145/62212.62214.

12 Arka Rai Choudhuri, Aarushi Goel, Matthew Green, Abhishek Jain, and Gabriel Kaptchuk. Fluid MPC: Secure multiparty computation with dynamic participants. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 94–123, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84245-1_4.

13 Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 240–269. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53015-3_9.

14 Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 311–326. Springer, Heidelberg, May 1999. doi:10.1007/3-540-48910-X_22.

15 Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 110–123. Springer, Heidelberg, August 1995. doi:10.1007/3-540-44750-4_9.

16 Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 572–590. Springer, Heidelberg, August 2007. doi:10.1007/978-3-540-74143-5_32.

17 Ivan Damgård, Daniel Escudero, and Antigoni Polychroniadou. Phoenix: Secure computation in an unstable network with dropouts and comebacks. Cryptology ePrint Archive, Report 2021/1376, 2021. URL: https://ia.cr/2021/1376.

18 Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

19 Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *20th ACM STOC*, pages 148–161. ACM Press, May 1988. doi:10.1145/62212.62225.

20 Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78375-8_16.

21 Daniel Genkin, Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In David B. Shmoys, editor, *46th ACM STOC*, pages 495–504. ACM Press, May / June 2014. doi:10.1145/2591796.2591861.

22 Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In Brian A. Coan and Yehuda Afek, editors, *17th ACM PODC*, pages 101–111. ACM, June / July 1998. doi:10.1145/277697.277716.

23 Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. YOSO: You only speak once - secure MPC with stateless ephemeral roles. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 64–93, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84245-1_3.

24 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.

25 Oded Goldreich and Avi Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Structures & Algorithms*, 11(4):315–343, 1997.

26 S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 63–82. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7_4.

27 Vipul Goyal, Yanyi Liu, and Yifan Song. Communication-efficient unconditional MPC with guaranteed output delivery. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 85–114. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7_4.

28 Vipul Goyal, Yifan Song, and Chenzhi Zhu. Guaranteed output delivery comes free in honest majority MPC. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 618–646. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56880-1_22.

29 Yue Guo, Rafael Pass, and Elaine Shi. Synchronous, with a chance of partition tolerance. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 499–529. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26948-7_18.

30 Danny Harnik, Yuval Ishai, and Eyal Kushilevitz. How many oblivious transfers are needed for secure multiparty computation? In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 284–302. Springer, Heidelberg, August 2007. doi:10.1007/978-3-540-74143-5_16.

31 Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 96–113. Springer, Heidelberg, May 2005. doi:10.1007/11426639_6.

32 Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_9.

33 Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 445–462. Springer, Heidelberg, August 2006. doi:10.1007/11818175_27.

34 Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, Heidelberg, August 2004. doi:10.1007/978-3-540-28628-8_21.

35 Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round efficiency of multi-party computation with a dishonest majority. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 578–595. Springer, Heidelberg, May 2003. doi:10.1007/3-540-39200-9_36.

36 Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988. doi:10.1145/62212.62215.

37 Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 171–189. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_10.

38 Yehuda Lindell and Hila Zarosim. On the feasibility of extending oblivious transfer. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 519–538. Springer, Heidelberg, March 2013. doi:10.1007/978-3-642-36594-2_29.

39 Chen-Da Liu-Zhang, Julian Loss, Ueli Maurer, Tal Moran, and Daniel Tschudi. MPC with synchronous security and asynchronous responsiveness. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 92–119. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64840-4_4.

40 Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_26.

41 Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 681–700. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_40.

42 Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *44th FOCS*, pages 404–415. IEEE Computer Society Press, October 2003. doi:10.1109/SFCS.2003.1238214.

43 Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 3–33. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78375-8_1.

44 Birgit Pfitzmann and Michael Waidner. Unconditional byzantine agreement for any number of faulty processors. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 337–350. Springer, 1992.

45 Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. URL: https://eprint.iacr.org/2005/187.

46 Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, May 1989. doi:10.1145/73007.73014.

47 Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 3–13. IEEE, 2000.

48 Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

49 Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. doi:10.1109/SFCS.1986.25.

Appendix

A Related Work

To the best of our knowledge, all current protocols consider a *symmetric* network resource model, where all parties have access to the same types of channels: all channels have the same delay and the same communication cost. In contrast, we consider a more refined *asymmetric* resource model, by making a distinction between two party categories. In the following, we show how our results compare to previous existing results.

Round Complexity of Information-Theoretic Synchronous MPC. The round complexity of MPC has a significant line of works. We focus on the information-theoretic setting. Here, all known protocols incur a number of rounds, that is linear in the depth of the computed circuit [7, 11, 46], and each round corresponds to the worst-case delay upper bound Δ. This is the case even when only one channel has delay Δ, and all other channels have delay $\delta \ll \Delta$. In contrast, we provide protocols where the number of slow Δ-rounds is independent of the circuit size (even constant in some cases). Note that a constant number of slow rounds is necessary in order to distribute the inputs of all slow parties.

Responsive MPC. A line of works focused on the problem of obtaining MPC protocols that are *responsive*. These are protocols where the running time is as fast as the actual network delay. Note that any asynchronous protocol is responsive, but there are also synchronous protocols that remain responsive when the number of corruptions is small [43, 39]. These protocols inherently cannot achieve input completeness, i.e., the inputs of up to t honest parties may be ignored from the computation, and are responsive up to $t < n/3$ corruptions (above $n/3$ corruptions, the running time is similar to that of synchronous protocols). In contrast, our protocols achieve input completeness, and benefit from the fast channels even up to $t < n/2$ corruptions.

Communication Complexity of MPC. The communication complexity of MPC has a significant line of works as well. Traditional solutions incur $\mathsf{poly}(n)|C|$ communication, and since then, there has been a significant progress in the communication efficiency (see e.g. [16, 6, 27, 28]). All these works consider the communication cost to be the same for all channels, and, as a consequence, their communication depends on the circuit size. Our work makes a distinction between cheap and expensive channels, and we provide protocols where the amount of bits transmitted over expensive channels is independent of the circuit size.

MPC with Sporadic Participation. A line of works [29, 3, 9, 12, 23, 17] considered MPC protocols where not all parties need to be online and/or participate at all steps of the protocol. In such protocols, a main challenge is to keep the state of the honest parties consistent throughout the protocol execution, and the protocol efficiency is counted uniformly (there is no asymmetry between network resources). In our case all parties are always online, and the main challenge is on minimizing the use of resources from a fixed known set of (slow) parties.

B Proof of Theorem 5

We prove by contradiction. Assume that there is such a secure asymmetric broadcast protocol resilient up to t corruptions and s slow parties such that $2t+s=n$, for $s,t>0$ and $n-s>1$. We partition the set of parties into three sets: the slow parties $\mathcal{S}$, and two sets of size t $\mathcal{F}_0$ and $\mathcal{F}_1$ for the fast parties. We make the argument for an external fast sender role P^*.

The adversary corrupts the sender P^* and chooses a random bit b. Then, it corrupts $\mathcal{F}_b$. In addition, the adversary runs the following two executions Exec_0 and Exec_1 at the same time.

- In Exec_b, the corrupted parties $\mathcal{F}_b$ and the corrupted sender P^* run an execution with the other parties where the sender has input b, but where $\mathcal{F}_b$ ignores every message from $\mathcal{F}_{1-b}$ and does not communicate with them either.
- In Exec_{1-b}, the corrupted sender P^* runs an execution with the slow parties and parties in $\mathcal{F}_{1-b}$, where the sender has input $1-b$, and the parties in $\mathcal{F}_b$ do not send any message to $\mathcal{F}_{1-b}$.

Consider an execution with the above adversary strategy. First, note that no matter what value b is chosen, the view of the slow parties $\mathcal{S}$ is exactly the same. This is because the view of the slow parties consist of two independent executions, where $\mathcal{F}_b$ and $\mathcal{F}_{1-b}$ do not communicate, but otherwise follow an execution where the sender has input b and $1-b$, respectively.

Second, note that when the bit b is chosen, the parties in $\mathcal{F}_{1-b}$ output $1-b$. This is because the view of $\mathcal{F}_{1-b}$ is identically distributed as in an execution where the sender is honest with input $1-b$, and parties in $\mathcal{F}_b$ crashed. Note that $\mathcal{F}_{1-b}$ must output fast, before receiving any message from $\mathcal{S}$.

The above observations imply that the slow parties output a value that is inconsistent with $\mathcal{F}_{1-b}$, with probability $1/2$. Note that even though $\mathcal{S}$ sees the sender is corrupted, he does not see whether $\mathcal{F}_0$ or $\mathcal{F}_1$ is honest (these sets output different bits).

C Recap of Protocol [14]

In this section, we describe the details of the protocol by Cramer et al. [14], which is used in the preprocessing phase of the protocol Π_{rgod} in Section 5.3.

IC-Signatures. Information checking (IC) [14] is a tool for authenticating data that is information-theoretic. We make use of information checking among n parties $P_1, \ldots, P_n$. Protocol Dist will be carried out by the dealer D with intermediary INT and the receivers $P_1, \ldots, P_n$, each with the same input value s. The information sent by D to INT will be called an IC-signature and we denote such a signature as $\sigma(s, D, INT)$. In order to verify a signature among n parties, the AuthVal protocol is executed bilaterally by INT and each party P_i. Then, in protocol Reveal, INT broadcasts s and the authentication information, and if $t+1$ parties accept s then we say that the signature has been confirmed. These signatures enable D to give INT a signature which only INT can use to convince the other parties about the authenticity of a value received from the dealer. Therefore, we can use these IC-signatures as signatures given specifically from D to INT, so that INT can prove authenticity of a received value to any party.

Verifiable Secret Sharing. We recall the definition of verifiable secret sharing (VSS).

▶ **Definition 10.** *A t-secure VSS scheme for sharing a secret $s \in \mathbb{F}$ is a pair* $(\texttt{Sh}, \texttt{Rec})$ *n-party protocols that satisfy the following properties, even in the presence of an adversary corrupting up to t parties:*

- *Correctness: Once all honest parties terminate protocol* Sh*, there exists a fixed value, $s' \in \mathbb{F} \cup \perp$, such that the following requirements hold:*
  - *If the dealer D is honest, then $s' = s$, and each honest party outputs s' in protocol* Rec*.*
  - *If the dealer is corrupted then each honest party outputs s' in protocol* Rec*.*
- *Privacy: If the dealer is honest and no honest party has yet started* Rec*, then the adversary has no information about the shared secret s.*
- *Termination: If the dealer D is honest then all honest parties terminate* Sh*, and if the honest parties invoke* Rec*, then each honest party eventually terminates* Rec*.*

Protocol Description. The protocol is based on classical protocols [19, 7], but using IC-signatures, instead of error correction.

In order to share a value s, the dealer D will make use of a bivariate polynomial $f(x, y)$ of degree at most t. The projections $f(x, i)$ and $f(i, y)$ will be sent to party P_i (where all the points are signed using IC-signatures). The parties can now bilaterally compare the cross-point values between them, and expose inconsistent behavior by the dealer using the signatures. This implies that the values held by honest parties are consistent, and since there are at least $n - t > t + 1$ honest parties, these values uniquely define a bivariate polynomial $f'(x, y)$ of degree at most t, which in turn defines the secret. Therefore, this already ensures that the dealer is committed to a value after the sharing phase.

However, the reconstruction might still fail if the adversary sends corrupted shares. In order to avoid that, each share of P_i is also signed by the other parties. This will in turn prevent the adversary from corrupting the secret at reconstruction time.

Protocol Π_{vss}

Share Let s be the input for the dealer D.

1: D chooses a random bivariate polynomial $f(x, y)$ of degree at most t in each variable, such that $f(0, 0) = s$. Let $s_{ij} = f(i, j)$. The dealer sends to party P_i the values $a_{1i} = s_{1i}, \ldots, a_{ni} = s_{ni}$ and $b_{i1} = s_{i1}, \ldots, b_{in} = s_{in}$. For each value a_{ji}, b_{ij}, D attaches IC-signatures $\sigma(a_{ji}, D, P_i)$, and $\sigma(b_{ij}, D, P_i)$.
2: Party P_i checks that the two sets $a_{1i}, \ldots, a_{ni}$ and $b_{i1}, \ldots, b_{in}$ are t-consistent. If the values are not t-consistent, Pi broadcasts these values with D's signature on them. If a party hears a broadcast of inconsistent values with the dealer's signature then D is disqualified and execution is halted.
3: P_i sends a_{ji} and a signature which he generates on a_{ij}, $\sigma(a_{ji}, P_i, P_j)$ privately to P_j.
4: Party P_i compares the value a_{ij} which he received from P_j in the previous step to the values b_{ij} received from D. If there is an inconsistency, P_i broadcasts b_{ij} and $\sigma(b_{ij}, D, P_i)$.
5: Party P_i checks if P_j broadcasted a value b_{ji}, $\sigma(b_{ji}, D, P_j)$ which is different than the value a_{ji} which he holds. If such a broadcast exists then P_i broadcasts a_{ji} and $\sigma(a_{ji}, D, Pi)$.
6: If for an index pair (i, j) a party hears two broadcasts with signatures from the dealer on different values, then D is disqualified and execution is halted.

Reconstruct

1: Party P_i broadcasts the values $b_{i1}, \ldots, b_{in}$ with the signature for value b_{ij} which he received from party P_j.
2: Party P_i checks whether P_j's shares broadcasted in the previous step are t-consistent and all the signatures are valid. If not then P_j is disqualified.
3: The values of all non-disqualified parties are taken and interpolated to compute the secret.

Multi-Party Computation. Using the VSS scheme Π_{vss} from above, it is easy to come up with an MPC protocol. Addition gates are straightforward and parties can process them locally (using the fact that the IC-signatures are linear). Multiplication gates are processed using the well-known method by Gennaro, Rabin and Rabin [22]: Each party P_i locally multiplies his shares a_i and b_i of the input wires a and b, and shares the result $d_i = a_i b_i$ using VSS. This results in n VSSs and a proper sharing of the output wire c can be computed as a fixed linear combination of these. The authors show a way for P_i to share a secret d_i, such that $d_i = a_i b_i$ and to prove that he has done so properly. The details can be found in [14].

D Description of Protocol Π_{rgod}

Protocol Π_{rgod}

Initialize $t' = t$, $\mathcal{F}' = \mathcal{F}$.

Preprocessing Phase

1: Parties use the protocol [14] to create:
- n_m random sharings of certified Beaver triples $([a_k], [b_k], [c_k])$, where n_m is the number of multiplication gates, as explained before. This means, that for each triple (a, b, c), each honest party P_i implicitly holds his share a_i by holding sub-shares $a_{i1}, \ldots, a_{in}$, where each sub-share is IC-signed by P_j. These sub-shares lie on a degree-t polynomial; and the shares a_i also lie on a degree-t polynomial f with $f(0) = a$. And similarly, for the values b and $c = ab$.

Input Phase Let x_j be the input from party P_j.

1: P_j uses the protocol Π_{vss} (described in Section C) to share his input x_j.

Addition Gates

1: Fast parties locally add the shares using linearity of the sharing scheme, and compute the corresponding IC-signatures using its linearity property.

Multiplication Gates Let $[x]$ and $[y]$ be sharings of the inputs to the gate.

1: Fast parties in $\mathcal{F}'$ use a multiplication triple $([a],[b],[c])$ to publicly reconstruct the values $x-a$ and $y-b$ among the fast parties, using the reconstruction procedure of Π_{vss} (where only the fast parties start). That is, the fast parties use asymmetric broadcast to distribute their share towards all parties, and the corresponding IC-signatures.
2: After time d_{bc}, all the fast parties reach agreement on whether all the shares received have correct IC-signatures are t-consistent. If not, they keep waiting for a total of $2D_{bc}$ time. Otherwise, execute Step 5.
3: After time D_{bc}, all slow parties reach agreement on whether the shares distributed by the fast parties were correct (t-consistent and with correct signatures). If not, slow parties participate in the reconstruction by broadcasting their shares and IC-signatures.
4: After time $2D_{bc}$, either there were enough shares to reconstruct at Step 2, or all honest parties (fast and slow) identify at least one corrupted fast party $P_k \in \mathcal{F}'$ that did not contribute its share. In this case, parties kick out the identified corrupted party, and restart the protocol, now with threshold $t' = t' - 1$, and set $\mathcal{F}' = \mathcal{F} \setminus \{P_k\}$.
5: Fast parties in $\mathcal{F}'$ locally compute a share of the output to the gate as $[z] = (x-a)[b] + (y-b)[a] + [c] + (x-a)(y-b)$ (via locally adding each of the sub-shares), and update the IC-signatures accordingly using the linearity property.

Output Phase

1: To reconstruct a sharing $[x]$ towards P_j, parties robustly reconstruct the secret to the P_j, using the VSS reconstruction protocol in Π_{vss}.

E Proof of Theorem 6

We describe the simulator for the online phase of the protocol, as the preprocessing phase is executed using the protocol [14] and can also be simulated.

Simulation of the Input Phase. For each honest party P_i, the simulator emulates an execution of Π_{vss} with input 0 towards the adversary. This includes sending random projected univariate polynomials to the adversary, along with IC-signatures. Each time a complain is received by the adversary, the simulator responds by broadcasting the corresponding stored share and signature.

For each corrupted party P_i, the simulator emulates an honest execution of Π_{vss} on behalf of the honest parties: it receives univariate polynomials with signatures on behalf of the honest parties. It then checks that they are all of degree-t, and if not, it broadcasts the values with P_i's signatures. Finally, for any share and signature that are broadcasted, if the corresponding party holding the cross-point share is honest, it checks whether that received share is consistent, and if not, it broadcasts these values with P_i's signature on them.

Simulation of Addition Gates. The addition gates require no simulation, since they constitute local computation only.

Simulation of Multiplication Gates. During each multiplication gate, the simulator emulates the reconstruction procedure of Π_{vss} from the fast parties. That is, it emulates an execution of asymmetric broadcast, and gives to the adversary the shares and signatures corresponding

to $x - a$ and $y - b$, where x and y are the input wire values in the simulated execution, and (a, b, c) is a multiplication triple. (Note that since a and b are random, the reconstructed values are also random.)

Simulation of the Output Stage. During the output phase, the simulator receives the output y_i for P_i, and the shares of the corrupted parties. With these, the simulator can consistently reconstruct the shares of the honest parties and send them to the adversary.

Analysis of the Simulation. We first argue about correctness. First, the pre-processing phase is successful, and all parties hold correct shares of the multiplication triples with correct signatures (with high probability). Similarly, after the input phase, parties hold correct shares of their inputs and signatures. This correctness trivially propagates to the rest of the wires in the circuit, since all operations are linear.

Now let us argue that the protocol securely computes the function. During the input phase for an honest party P_i, the adversary receives the projections of univariate polynomials $f(x, j)$ and $f(j, y)$, for up to t indices j. By properties of bivariate polynomials, this reveals no information about the secret, and is therefore identically distributed as the univariate polynomials that the simulator reveals. In the multiplication phase, as hinted above, since the multiplication triple (a, b, c) contains uniform random values a and b, the reconstructed values are also distributed uniformly at random, identically as the simulated execution. This is also the case for the output gates, where the simulator outputs honest shares and signatures that are consistent with the corresponding reconstructed output.

Phoenix: Secure Computation in an Unstable Network with Dropouts and Comebacks

Ivan Damgård
Aarhus University, Denmark

Daniel Escudero
J.P. Morgan AI Research & J.P. Morgan AlgoCRYPT CoE, New York, NY, USA

Antigoni Polychroniadou
J.P. Morgan AI Research & J.P. Morgan AlgoCRYPT CoE, New York, NY, USA

Abstract

We consider the task of designing secure computation protocols in an unstable network where honest parties can drop out at any time, according to a schedule provided by the adversary. This type of setting, where even honest parties are prone to failures, is more realistic than traditional models, and has therefore gained a lot of attention recently. Our model, Phoenix, enables a new approach to secure multiparty computation with dropouts, allowing parties to drop out and re-enter the computation on an adversarially-chosen schedule and without assuming that these parties receive the messages that were sent to them while being offline - features that are not available in the existing models of Sleepy MPC (Guo et al., CRYPTO '19), Fluid MPC (Choudhuri et al., CRYPTO '21) and YOSO (Gentry et al. CRYPTO '21). Phoenix does assume an upper bound on the number of rounds that an honest party can be off-line – otherwise protocols in this setting cannot guarantee termination within a bounded number of rounds; however, if one settles for a weaker notion, namely guaranteed output delivery only for honest parties who stay on-line long enough, this requirement is not necessary.

In this work, we study the settings of perfect, statistical and computational security and design MPC protocols in each of these scenarios. We assume that the intersection of online-and-honest parties from one round to the next is at least $2t + 1$, $t + 1$ and 1 respectively, where t is the number of (actively) corrupt parties. We show the intersection requirements to be optimal. Our (positive) results are obtained in a way that may be of independent interest: we implement a traditional stable network on top of the unstable one, which allows us to plug in *any* MPC protocol on top. This approach adds a necessary overhead to the round count of the protocols, which is related to the maximal number of rounds an honest party can be offline. We also present a novel, perfectly secure MPC protocol in the preprocessing model that avoids this overhead by following a more "direct" approach rather than first building a stable network and then using existing protocols. We introduce our network model in the UC-framework, show that the composition theorem still holds, and prove the security of our protocols within this setting.

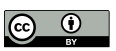
2012 ACM Subject Classification Theory of computation → Cryptographic protocols

Keywords and phrases Secure Multiparty Computation, Unstable Networks

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.7

Related Version *Full Version*: `https://eprint.iacr.org/2021/1376`

Acknowledgements This paper was prepared in part for information purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co and its affiliates ("JP Morgan"), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of



© Ivan Damgård, Daniel Escudero, and Antigoni Polychroniadou;
licensed under Creative Commons License CC-BY 4.0

participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful. 2023 JP Morgan Chase & Co. All rights reserved.

1 Introduction

Secure Multiparty Computation (MPC) is a technique that allows multiple mutually distrustful parties to compute a function of their inputs without leaking anything else beyond the output of the computation. Most protocols in the MPC literature assume that the parties communicate over a *synchronous network*, that is, all the parties have access to a global clock. This allows the parties to follow the protocol specification based on time. A protocol under such network model proceeds in *communication rounds*, each of which has a fixed duration and where each party can send a message to each other party.

Synchronous networks are natural for describing protocols and may make sense in many contexts, but the model is not resilient to sudden slowdowns: if a party fails to send a message within the allocated time for a specific round, this message will not be taken into account, and what is worse, in the context of an active adversary this will be considered a deviation from the protocol specification. Hence an honest party who accidentally misses a deadline will be classified as corrupt. The first problem with this is that an MPC protocol can only tolerate a certain maximal number of corruptions. Tagging parties as corrupt because of natural network issues that may appear in practice leaves little room for real corruptions. For instance, MPC over unstable mobile network connections or denial of service attacks might consume all the corruptions we can handle. The second problem is that once a party is tagged as corrupt, the protocol may now reveal her secret inputs, which seems unfair if the party was actually honest but suffered a random network delay. An alternative model is an *asynchronous network*, where the parties are not assumed to have a clock anymore. This modeling is more resilient to the type of attacks described above since the communication network allows for parties to be slow and no deadlines are set. However, this model comes with its own set of issues since, when dealing with an active adversary, the parties cannot distinguish a delayed message sent by a slow party, from a message that an actively corrupt party decided not to send in the first place. As a result asynchronous protocols tend to tolerate a smaller number of corruptions [3], and, what is worse, an asynchronous protocol cannot guarantee that all honest parties get to contribute inputs to the computation.

Therefore, it seems to be a better approach of considering an imperfect synchronous network where the adversary is allowed to cause some parties to go offline temporarily, and require protocols to not classify such parties as corrupt. In such a setting we may still hope to get (1) optimal corruption thresholds, (2) allow all parties to contribute input, and (3) guarantee termination at a certain time. A series of works has studied MPC in different variant of this model, see Section 1.3 and also the Full Version of this work for a detailed comparison of prior works. However, it is still an open question whether we can have MPC protocols with optimal security and corruption thresholds in the most adversarial, but also most realistic setting, that we call an *unstable network* in this paper. In such a network parties go offline and come back according to an adversarially chosen schedule (not a schedule prescribed by the protocol specifications), and parties are not assumed to receive messages sent while they were offline. Not receiving messages while being offline introduces more challenges since one can only rely on the parties that are online in the current round and were also online in the previous round.

1.1 Unstable Networks

As we have mentioned, there are multiple attempts in the literature to model what a realistic network where parties can dropout and return should represent concretely. In this work we are interested in studying the setting of MPC over an *unstable network*, which is a type of synchronous network we introduce where, in contrast to a *stable network* (*i.e.* a standard synchronous network), the adversary can choose in each round a subset of parties that will be offline in that specific round, and hence may not be able to send or receive messages. This models honest parties dropping out in that specific round, possibly due to network errors or malicious attacks, which serves to represent certain failures like weak mobile connections or DDoS attacks. We remark that our "timing model" is still synchronous in that the parties have a synchronized clock and know which current protocol step is being run, but crucially, they may drop and re-join in every round.

Given that over an unstable network the set of offline parties can be different in every round, an MPC protocol in such setting must allow parties to rejoin the computation after being offline. Furthermore, these parties may not know they are under network attack, so a missing message can mean that either (1) they are under attack, (2) the sender is under attack, or (3) the sender is malicious. This ambiguity is crucial to maintain a strong and realistic model, but it turns out to heavily complicate protocol design. This is further accentuated by the fact that, in an unstable network – and in stark contrast with previous networking models for tolerating dropouts – parties who rejoin the computation do not necessarily receive the messages sent to them while being offline, which is an important property to model settings like peer-to-peer networks where the parties do not count on "always-running" servers that can queue messages for them. This is an important scenario to consider in practice, since one might argue that counting on communication servers that never fail can be equivalent to assuming parties who never drop.

1.2 Our Contribution

In this work we formally introduce the notion of an *unstable network*, which we believe to be an appropriate communication model to capture realistic settings where parties join and leave an ongoing computation according to a potentially adversarial schedule. Our first contribution lies in the formal definition of this novel networking model, and we present a rigorous treatment of this notion within the confines of the UC framework, which in particular involves re-proving the UC theorem to ensure that composability still holds in this new setting.

Our second contribution – and where most of our work is devoted – consists of a full characterization of what types of security properties (*i.e.* perfect, statistical or computational) can be achieved by MPC protocols over unstable networks in terms of the underlying adversarial schedule. More precisely, we show that the minimum amount of honest parties that remain online from one round to the next is the crucial metric that determines whether a given level of security is attainable or not, and we show both impossibility and correspondingly matching feasibility results for each one of the three security notions: computational, statistical and perfect security. We believe our novel model and initial set of results open an exciting and interesting research direction on the design of MPC protocols over realistic networks.

In order to discuss what the characterizations above are in detail, let us introduce some notation. Let n be the number of parties and let t be the number of corrupt parties. Let $\mathcal{O}_r$ denote the set of online parties in round r, and let $\mathcal{H}$ denote the set of honest parties. Our goal is to determine if we can construct MPC protocols for an unstable network which enjoy

	Perfect security	Statistical security	Computational security
Passive adversary $\lvert\mathcal{O}_r \cap \mathcal{O}_{r+1}\rvert \geq$	$t+1$	$t+1$	1
Active adversary $\lvert\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}\rvert \geq$	$2t+1$	$t+1$	1

Figure 1 Overview of the required intersection sizes for each setting considered in this paper. The result for statistical and passive security follows from the one for perfect and passive security.

the same security guarantees as protocols over a stable network and if so, what constraints we must assume on the unstable network to make this happen. To be able to talk more concretely about this, we will say that two protocols π, π' are *equivalent* if they tolerate the same number of corruptions, achieve the same type of security (computational/statistical/perfect) and the same security guarantee (security with abort/fairness/guaranteed output delivery). Our first set of results is as follows:

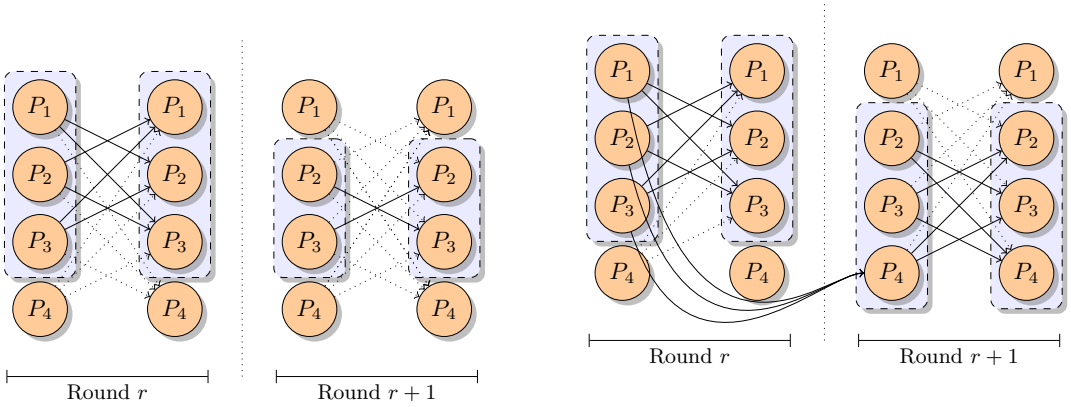
Perfect security. (Section 2) Given any perfectly secure synchronous MPC protocol against t corruptions, we construct an equivalent protocol over an unstable network, assuming that $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq 2t+1$ for all $r > 0$. Furthermore, this condition is required for any MPC protocol with perfect security to exist over an unstable network.

Statistical security. (Section 3) Given any statistically secure synchronous MPC protocol against t corruptions, we construct an equivalent protocol over an unstable network, assuming that $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq t+1$ for all $r > 0$. This condition is required for any MPC protocol with statistical security to exist over an unstable network.

Computational security. (Full Version) Given any computationally secure synchronous MPC protocol secure against t corruptions, we construct an equivalent protocol over an unstable network, assuming that $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq 1$ for all $r > 0$ (and, for malicious security, assuming a PKI and public key encryption). The intersection condition is required for any computationally secure MPC protocol to exist over an unstable network.

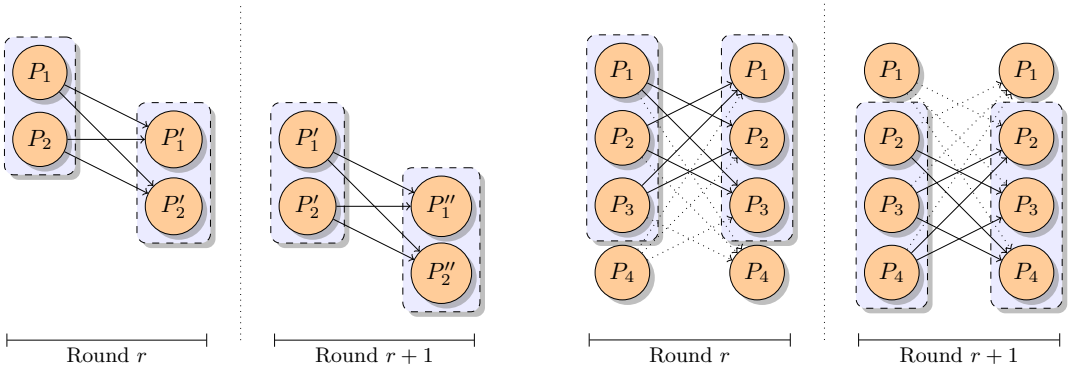
An overview of the intersection sizes required in each of the settings considered in our work is presented in Fig 1. Notice that our results imply a necessary tradeoff between instability and corruptions: taking perfect security as an example, it is well known that we must have $n \geq 3t+1$ to have perfect security at all. So for a maximal value of t, we have only $2t+1$ honest parties, and the result above then says that all honest parties must stay online all the time. On the other hand, as we increase n above $3t+1$, an increasing number of honest players can be sent offline. Also, note that even if the (minimal) assumptions in our results say that a minimum amount of parties must stay online from one round to the next, this does not imply that any *particular party* stays online for more than one round. This makes protocol design considerably difficult, as in particular, the following scenario may occur: a given party can be offline for a while, not receiving any messages, then it is set to be online in a given round, but the scheduling[1] is such that this party only receives messages in this round after he or she has sent their own message, so this message can only depend on outdated information this party learned before going offline. Furthermore, this party may be

[1] As in the standard synchronous network, the adversary is allowed to choose the ordering of the messages received by honest parties.



(a) Lazy-MPC model.

(b) Guo et al. (Sleepy) model.



(c) Fluid MPC model.

(d) Our model.

Figure 2 Our model compared to other models in the literature. Parties inside the marked region are online, and messages represented by dashed arrows are dropped. In Lazy-MPC, Fig. 2a, the parties cannot return. In the model of Guo et al., Fig. 2b, the parties can return but it is assumed they receive the messages sent to them while they were offline. In the Fluid-MPC model, Fig. 2c, in each round the set of parties who send messages may differ from the set of parties who receive these messages, but the identities of these parties must be known by the protocol. In our model, Fig. 2d, the parties can return to the computation and it is not assumed that they receive the messages sent to them while they were offline.

set to be offline for the next round immediately after sending their message, which makes the contribution of this party to the protocol meaningless. The honest parties in $\mathcal{O}_r \cap \mathcal{O}_{r+1}$ are these who are able to receive the messages in round r, and simultaneously are able to send a derived message in round $r+1$, so having enough honest parties in this intersection is what enables us to design MPC protocols in this difficult networking setting.

1.3 Related Work

In what follows we discuss some of the works that study a similar problem to the one we address in this work. The description in this section is relatively lightweight, and we defer a more detailed analysis to the Full Version.

Fail-stop adversaries that may cause some parties to stop during a computation were considered for the first time in [5], but this and subsequent works assume parties know when a given party fail-stopped, plus these parties are not able to return the computation. A recent

model in [1] considers an adversary that can set parties to be offline at any round, but as before these parties cannot return the computation, plus that work focuses on computational assumptions, making use of strong homomorphic encryption tools. In the "sleepy model" of [8] parties who drop can return. However, a crucial difference with our model is that, in our case, parties who return after being offline may not receive the messages sent to them before becoming onine, while in [8] these parties (who are not "offline" but "slow") do receive these messages. This makes the problem considerably easier, plus the authors consider only computational assumptions. Finally, in [2, 12] a new model is considered where the set of parties can change dynamically from one round to the next. In that work, the set of "online" parties in a given round is not adversarially chosen, but rather set in advance and used in the design of the protocol. As a result, this work may not model adversarial attacks to the underlying network, and may be less realistic in these settings. Furthermore, the protocol in [2], although statistically secure, only achieves security with abort. Our compilation-based techniques allows us to transfer any result in the standard synchronous setting (*e.g.* protocols with guaranteed output delivery) to the unstable networking setting.

The "You Only Speak Once" (YOSO) model for MPC is introduced in [7]. Our model assumes a somewhat less powerful adversary who must allow a physical party to come back after being offline, while in [7] this adversary can take a party down as soon as they speak, and progress is guaranteed by means of assigning roles "on-the-fly" in certain randomized fashion. Their model does not allow for perfect security, while in our case, on top of achieving much easier protocol design, we can obtain information theoretic security based only on point-to-point secure channels, and we allow for termination such that all parties can provide input and get output. Finally, the "constrained parties" and "full-omission parties" from [10] and [13] are such that whose messages are selectively blocked by the adversary, as in our setting. However, in these works the adversary choses the subset of offline parties at the beginning of the protocol execution, while in our case this subset can change adaptively as the protocol is run. This is in fact one of the main sources of difficulties when designing protocols in our setting, since a party who is "full-omission-corrupt" can stop being so, and non-corrupted parties can later on become full-omission-corrupt. We remind the reader to visit the Full Version for a more detailed discussion on related work.

We present in Figure 2 a more graphical comparison of our model with respect to the works of [1, 8, 2].

1.4 Preliminaries and Organization

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of all parties, and $\mathcal{H}$ be the set of honest parties. We assume that the adversary corrupts t out of the n parties. Let $\mathbb{F}$ be a finite field with $|\mathbb{F}| > n$. Due to space limitations we assume background on Shamir secret-sharing, with details given in Section A and in the Full Version. For our results in the computational setting, we assume the existence of a CPA-secure public key encryption scheme $(\mathsf{enc}, \mathsf{dec})$, and a EUF-CMA signature scheme $(\mathsf{sign}, \mathsf{verify})$. The formal definitions of these primitives and their security is standard and can be found in any modern book in Cryptography (e.g. [9]).

Unstable Networks

Now we provide the different functionalities we will make use of in our work. More thorough definitions and considerations, including the proof of the composition theorem, are given in the Full Version. Our timing model is synchronous, meaning there parties have a global clock and there is a known upper bound Δ on the time it takes for a message to be transmitted

between any pair of parties. The communication pattern proceeds in rounds, identified with integers $1, 2, 3, \ldots$, each taking Δ time and consisting of all parties sending messages to each other at the beginning of each round, and receiving some of these messages in a way we will specify later before the end of that round. We use $\mathcal{F}_{\mathsf{StableNet}}$ to denote the functionality that models a stable network in which all of the messages between honest parties are always delivered. We also consider a family of functionalities $\{\mathcal{F}_{\mathsf{StableNet}}^{P_i \to P_j}\}_{i,j=1}^{n}$ that models a synchronous channel from P_i to P_j only. In this work we take the following approach in order to obtain MPC over unstable networks: first, we instantiate the $\mathcal{F}_{\mathsf{StableNet}}$ functionality on top of an unstable network, that is, we design a way for each pair of parties to communicate reliably over an unstable network. Then, we take off-the-shelf MPC protocols set in the stable/synchronous model and compose them with our protocol for emulating the stable network, to get MPC protocols that are set in the unstable networking model. In the Full Version we elaborate on which protocols we use, and on why the modular approach sketched above works via the composition theorem. In this version we focus on instantiating the stable networking model only.

An unstable network is formalized as a functionality, that we denote by $\mathcal{F}_{\mathsf{UnstableNet}}$. In each round, the functionality proceeds as follows: (1) At the beginning of the round the environment, denoted by $\mathcal{Z}$, specifies a subset of parties $\mathcal{O}_r \subseteq \mathcal{P}$; (2) For every $P_i, P_j \in \mathcal{O}_r \cap \mathcal{H}$, the functionality delivers messages sent from P_i to P_j in the given round; (3) For every P_i and P_j with either one of the two parties in $(\mathcal{O}_r)^c \cap \mathcal{H}$, the environment can choose whether to drop the message sent from P_i to P_j in the given round.

If the adversary is allowed to set a given party P_i as offline forever, it is obvious that no stable channel to or from P_i could be instantiated. To address this we introduce the *B-assumption*, which states that the maximum amount of consecutive rounds that a party can be offline is B. The protocols we present here require this assumption in order to produce output, but in the Full Version we discuss alternative protocols that do not require this during the entire computation.

2 Instantiating $\mathcal{F}_{\mathsf{StableNet}}^{P_S \to P_R}$ with Perfect Security

In this section we take care of instantiating the functionality for a stable network with perfect security. First, in Section 2.1 we discuss the simplest setting of passive security. Then, in Section 2.2 we extend this to active security, while retaining perfect simulation.

2.1 Passive Security

Assuming a passive adversary, and assuming that $|\mathcal{O}_r \cap \mathcal{O}_{r+1}| \geq t + 1$ for all $r > 0$, our protocol to instantiate $\mathcal{F}_{\mathsf{StableNet}}^{P_S \to P_R}$ with perfect security is obtained as follows. At every round, P_S tries to secret-share its message m towards all the parties, which succeeds in the round in which P_S comes online. In the following rounds, the parties try to send their shares of m to P_R, who is able to get them when it comes online, and hence is able to reconstruct m. The only missing step is that, when P_S secret-shares m, only the parties online in the current round are able to receive the shares. To alleviate this issue, the parties in each round "transfer" the shared secret to the parties that are online in the next round. This is done via a simple resharing protocol. Details are in Protocol $\Pi_{\mathsf{StableNet}}^{\mathsf{perf},\mathsf{passive}}(P_S, P_R, m)$.

We remark that, although it is not explicitly written in the protocol description, whenever it is written that P_i sends a message to P_j, this is done by invoking the $\mathcal{F}_{\mathsf{UnstableNet}}$ functionality.

Protocol $\Pi^{\mathsf{perf,passive}}_{\mathsf{StableNet}}(P_S, P_R, m)$

- On input (m), P_S samples random elements $c_{ij} \in \mathbb{F}$ for $i, j = 0, \ldots, t$, subject to $c_{0,0} = m$ and $c_{ij} = c_{ji}$, and lets $f(x, y) = \sum_{i,j=0}^{t} c_{ij} x^i y^j$. Then, in rounds $1, \ldots, B$, P_S sends $f(x, i)$ to each party P_i.
- Every party P_i initializes a variable $\mathtt{f}_i = \perp$. In rounds $1, \ldots, 2B$, P_i does the following:
  - If $\mathtt{f}_i$ is not set already:
    - If P_i receives a polynomial $f_i(x) = f(x, i)$ from P_S, then P_i sets $\mathtt{f}_i = f_i$.
    - Else, if P_i receives messages $m_j \in \mathbb{F}$ from at least $t+1$ parties P_j, then P_i sets $\mathtt{f}_i$ to be the polynomial $f_i(x)$ such that $f_i(j) = m_j$ for the first $t+1$ messages m_j.
  - If $\mathtt{f}_i \neq \perp$, then P_i sends $\mathtt{f}_i(j)$ to each party P_j and $\mathtt{f}_i(0)$ to P_R.
- In rounds $B+1, \ldots, 2B$, P_R does the following: If P_R receives messages $m_j \in \mathbb{F}$ from at least $t+1$ parties P_j, then P_R computes the polynomial $f_0(x)$ such that $f_0(j) = m_j$ for the first $t+1$ messages m_j, and outputs $m = f_0(0)$.

▶ **Theorem 1.** *Assume that $|\mathcal{O}_r \cap \mathcal{O}_{r+1}| \geq t+1$ for every $r > 0$. Then, protocol $\Pi^{\mathsf{perf,passive}}_{\mathsf{StableNet}}(P_R, P_S)$ instantiates the functionality $\mathcal{F}^{P_R \to P_S}_{\mathsf{StableNet}}$ in the $\mathcal{F}_{\mathsf{UnstableNet}}$-hybrid model with perfect security against an adversary passively corrupting $t < n$ parties.*

Proof. We claim that, in an execution of protocol $\Pi^{\mathsf{perf,passive}}_{\mathsf{StableNet}}(P_R, P_S)$, P_R learns the value of m at the end of the interaction, and the adversary does not learn the value of m, unless P_S or P_R are passively corrupt.

To see this, let $r_S \in \{1, \ldots, B\}$ be the smallest value such that $P_S \in \mathcal{O}_{r_S}$, which exists due to the B-assumption. We claim the following invariant: at the end of every round r with $r_S \leq r \leq 2B$, each $P_i \in \mathcal{O}_r$ has $\mathtt{f}_i \neq \perp$, and these polynomials satisfy that $\mathtt{f}_i(x) = f(x, i)$, where $f(x, y)$ is the polynomial sampled by P_S at the beginning of the protocol. To see this we argue inductively. First, notice that the invariant holds for $r = r_S$ given that parties $P_i \in \mathcal{O}_{r_S}$ receive this directly from P_S. For the inductive step assume that the invariant holds for some round r, that is, each party $P_i \in \mathcal{O}_r$ has set its variable $\mathtt{f}_i$, and $\mathtt{f}_i(x) = f(x, i)$. In particular, this is held by the parties in $\mathcal{O}_r \cap \mathcal{O}_{r+1}$, so each party P_i in this set sends $\mathtt{f}_i(j)$ to every other party P_j in round $r+1$, which is received by the parties in $\mathcal{O}_{r+1}$. Since $|\mathcal{O}_r \cap \mathcal{O}_{r+1}| \geq t+1$, we see that each party $P_j \in \mathcal{O}_{r+1}$ receives at least $t+1$ values $\mathtt{f}_i(j) = f(j, i) = f(i, j)$, which enables P_j to interpolate $f(x, j)$, which is set to $\mathtt{f}_j$. We see then that the invariant is preserved.

Finally, let $r_R \in \{B+1, \ldots, 2B\}$ be a round in which $P_R \in \mathcal{O}_{r_R}$, which is guaranteed from the B-assumption. By the invariant, the parties in $\mathcal{O}_{r_R - 1}$ have set their variables $\mathtt{f}_i$ at the end of round $r_R - 1$ correctly, so in particular the parties in $\mathcal{O}_{r_R-1} \cap \mathcal{O}_{r_R}$ will send $\mathtt{f}_i(0) = f(0, i)$ to P_R in round $\mathcal{O}_{r_R}$. Since there are at least $t+1$ such parties, this means that P_R gets at least $t+1$ values $f(0, i)$, which allows P_R to interpolate $m = f(0, 0)$.

The fact that the adversary does not learn anything if both P_S and P_R are honest follows from the fact that its view is limited to t polynomials of the form $f(x, i)$, which look uniformly random. We remark that with the analysis above, it is straightforward to set up a simulator $\mathcal{S}$ for the proof. ◀

Optimality of $|\mathcal{O}_r \cap \mathcal{O}_{r+1}| \geq t+1$

Now we show that, in order to instantiate $\mathcal{F}^{P_S \to P_R}_{\mathsf{StableNet}}$ with perfect security against a passive adversary, the assumption that the adversary's schedule satisfies $|\mathcal{O}_r \cap \mathcal{O}_{r+1}| \geq t+1$ in every round r is necessary. However, we have to be careful about what this should actually mean:

consider an adversary who respects the B-assumption and breaks the intersection condition in one, or some finite number of rounds. Now, if the sender happens to start our protocol for sending a message after the last bad round, it will clearly succeed. So we cannot hope to show that communication between sender and receiver is impossible, unless we consider an adversary who keeps breaking the intersection condition "for ever". So we construct below an adversary that breaks this condition once every B rounds, and by doing so it is able to learn the message sent by an honest sender using *any* instantiation of $\mathcal{F}_{\mathsf{StableNet}}^{P_S \to P_R}$.

Assume the existence of an implementation of $\mathcal{F}_{\mathsf{StableNet}}^{P_S \to P_R}$ with perfect security that tolerates an adversary that schedules the parties as follows: (1) The adversary chooses a set $A_1 \subset \mathcal{P}$ such that $|A_1| = t+1$, $P_S \in A_1$ and $\mathcal{O}_{k \cdot B} = A_1$ for $k > 0$, and (2) the adversary chooses a set A_2 such that $A_1 \cup A_2 = \mathcal{P}$ and $|A_1 \cap A_2| \leq t$ such that $P_R \in A_2$, $P_S \notin A_2$ and $\mathcal{O}_r = A_2$ for every r that is not of the form $k \cdot B$. Notice that this scheduling respects the B-assumption. Now, suppose that P_R learns the output in round $r_R = k \cdot B + \ell$ for some k and ℓ with $1 \leq \ell \leq B$. Since during the whole protocol P_R only hears from the parties in A_2, this means that these parties together had enough information to reconstruct the secret in round r_R. However, these parties only hear from P_S through $A_1 \cap A_2$, which means that at a given point in the protocol this set had enough information to reconstruct the secret. This is a contradiction since $|A_1 \cap A_2| \leq t$ and $P_S, P_R \notin A_1 \cap A_2$, and due to privacy no set of at most t parties that does not contain the sender nor the receiver can reconstruct the message.

We remark that this lower bound rules out general MPC over unstable networks when $|\mathcal{O}_r \cap \mathcal{O}_{r+1}| \leq t$, since $\mathcal{F}_{\mathsf{StableNet}}^{P_S \to P_R}$ is a particular case of general MPC. This can be seen even more clearly since what the lower bound actually shows is that, if the minimum intersection size is not met, then the "state" of the computation is either leaked, or lost, which rules out general MPC. Indeed, our perfectly secure protocol from Section B, which does not use $\mathcal{F}_{\mathsf{StableNet}}$ directly, still requires $|\mathcal{O}_r \cap \mathcal{O}_{r+1}| \geq t+1$ to hold for every round.

2.2 Active Security

The construction we presented in the previous section does not carry over to the actively secure setting, given that a corrupted party P_i is not forced to send correct evaluations $\mathtt{f}_i(j)$. In this section we show an extension of this protocol that rules out this case. We assume that, for every r, $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq 2t+1$, which should be contrasted with the weaker condition in the passively secure setting of $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq t+1$. The use of a larger threshold allows us to make use of *error correction*, which allows the parties to reconstruct the right polynomials at each step of the protocol regardless of any incorrect value sent by corrupt parties.

The protocol for active security, Protocol $\Pi_{\mathsf{StableNet}}^{\mathsf{perf,active}}(P_S, P_R, m)$, is similar to Protocol $\Pi_{\mathsf{StableNet}}^{\mathsf{perf,passive}}(P_S, P_R, m)$, except for the following crucial change: when each P_i collects the messages $m_j \in \mathbb{F}$ for P_j received in a given round, only if there are at least $2t+1$ such messages, P_i performs error correction on these to reconstruct a polynomial $f_i(x)$ such that $f_i(j) = m_j$ for every received message m_j, and if this succeeds, then P_i sets $\mathtt{f}_i = f_i$. Similarly, only if P_R receives at least $2t+1$ messages $\{m_j\}_j$, then P_R performs error correction to recover a polynomial $f_0(x)$ such that $f_0(j) = m_j$ for every received message m_j, and if this succeeds then P_R outputs $m = f_0(0)$.

▶ **Theorem 2.** *Assume that $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq 2t+1$ for every $r > 0$. Then, protocol $\Pi_{\mathsf{StableNet}}^{\mathsf{perf,active}}(P_R, P_S)$ instantiates the functionality $\mathcal{F}_{\mathsf{StableNet}}^{P_R \to P_S}$ in the $\mathcal{F}_{\mathsf{UnstableNet}}$-hybrid model with perfect security against an adversary actively corrupting $t < n/3$ parties.*[2]

[2] In principle the restriction is simply $t < n$, but we have that $n - t = |\mathcal{H}| \geq |\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq 2t+1$, so $n \geq 3t+1$.

Proof. We claim that, in an execution of protocol $\Pi^{\mathsf{perf,active}}_{\mathsf{StableNet}}(P_R, P_S)$, P_R learns the value of m at the end of the interaction, and, if P_R and P_S are honest, the adversary does not learn the value of m.

To see this, let $r_S \in \{1, \ldots, B\}$ be the smallest value such that $P_S \in \mathcal{O}_{r_S}$. We claim the following invariant: at the end of every round r with $r_S \leq r \leq 2B$, each $P_i \in \mathcal{O}_r \cap \mathcal{H}$ has $\mathtt{f}_i \neq \perp$, and these polynomials satisfy that $\mathtt{f}_i(x) = f(x, i)$, where $f(x, y)$ is the polynomial sampled by P_S at the beginning of the protocol. We use induction in order to show that the invariant holds. First, notice that the invariant is true for $r = r_S$ given that parties $P_i \in \mathcal{O}_{r_S} \cap \mathcal{H}$ receive the polynomial directly from P_S. For the inductive step assume that the invariant holds for some round r, and we show that it holds for round $r + 1$. By the hypothesis assumption each party $P_i \in \mathcal{O}_r \cap \mathcal{H}$ has set its variable $\mathtt{f}_i$, and $\mathtt{f}_i(x) = f(x, i)$. In particular, this holds for the parties in $\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}$, which means that each party P_i in this set sends $\mathtt{f}_i(j)$ to every other party P_j in round $r + 1$, which is received by the parties in $\mathcal{O}_{r+1}$. Since $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq 2t + 1$, each party $P_j \in \mathcal{O}_{r+1} \cap \mathcal{H}$ receives at least $2t + 1$ correct values $\mathtt{f}_i(j) = f(j, i) = f(i, j)$. Even if P_j receives more shares, some of them potentially incorrect, P_j can still recover $f(x, j)$ via error correction, as instructed by the protocol. We see then that for P_j $\mathtt{f}_j = f(x, j)$, so the invariant is preserved.

Now, let $r_R \in \{B + 1, \ldots, 2B\}$ be a round in which $P_R \in \mathcal{O}_{r_R}$. By the invariant, the parties in $\mathcal{O}_{r_R - 1}$ have set their variables $\mathtt{f}_i$ at the end of round $r_R - 1$ correctly, so in particular the parties in $\mathcal{O}_{r_R - 1} \cap \mathcal{O}_{r_R} \cap \mathcal{H}$ will send $\mathtt{f}_i(0) = f(0, i)$ to P_R in round $\mathcal{O}_{r_R}$. Since there are at least $2t + 1$ such parties, this means that P_R gets at least $2t + 1$ correct values $f(0, i)$, which allows P_R to error-correct $m = f(0, 0)$. The fact that the adversary does not learn anything if both P_S and P_R are honest follows as in the proof of Theorem 1.

As with the case with passive security, the analysis above enables the construction of a simulator $\mathcal{S}$ for the proof in a straightforward manner. The main complication with the actively secure setting in contrast to the scenario with passive security is that a corrupt P_S may send inconsistent shares in the first round in which it becomes online. However, in this case, $\mathcal{S}$ can simply emulate the protocol exactly as the honest parties would do, and check if the receiver would be able to error-correct or not at the end of the execution. Only if this is the case, $\mathcal{S}$ would make use of the change command in the $\mathcal{F}^{P_S \to P_R}_{\mathsf{StableNet}}$ functionality to set P_S's message to be the one that is recovered by P_R, and then it would clock-out P_R if P_R is honest. ◀

Optimality of $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq 2t + 1$

As in Section 2.1, we show that the bound $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq 2t+1$ is necessary for essentially all rounds by presenting an adversary that breaks the correctness of any perfectly secure implementation of $\mathcal{F}^{P_S \to P_R}_{\mathsf{StableNet}}$ against active adversaries, by using a scheduling that breaks the condition above while still respecting the B-assumption.

The adversary's scheduling is as follows. For simplicity let us assume that $n = 5$ and $t = 1$, although the argument can be extended easily to any number of parties. Assume that P_1 is the sender, P_5 is the receiver.

- Let $\mathcal{O}_{k \cdot B} = \{P_1, P_2, P_3, P_4\}$ for $k = 0, 1, \ldots$.
- Let $\mathcal{O}_r = \{P_2, P_3, P_4, P_5\}$ for every r that is not of the form $r_0 + k \cdot B$. Notice that $|\mathcal{O}_{k \cdot B} \cap \mathcal{O}_{k \cdot B + 1} \cap \mathcal{H}| = |\{P_3, P_4\}| = 2 = 2t$ where $\mathcal{O}_{k \cdot B} \cap \mathcal{O}_{k \cdot B + 1} = \{P_2, P_3, P_4\}$.

Notice that this scheduling respects the B-assumption. Suppose that there is a protocol that instantiates $\mathcal{F}^{P_S \to P_R}_{\mathsf{StableNet}}$ with perfect security against an active adversary, supporting the scheduling above. We will show a contradiction arising from the fact that the adversary can actively cheat.

Suppose that P_R learns the output in round $r_R = k_0 \cdot B + \ell$ for some k_0 and ℓ with $1 \leq \ell \leq B$. Consider two different messages $m \neq m'$, and let M_j and M'_j for $j = 2, 3, 4$ be the concatenation of the messages sent by P_j in round $k \cdot B$ to the parties in $\mathcal{O}_{k \cdot B} \cap \mathcal{O}_{k \cdot B+1} = \{P_2, P_3, P_4\}$ for $k = 0, \ldots, k_0$, when the inputs of P_S to the protocol are m and m' respectively.

First, we claim that the messages (M_2, M_3, M_4) (resp. (M'_2, M'_3, M'_4)) must uniquely determine the secret m (resp. m'). To see why this is the case, observe that the receiver, P_5, only ever hears from the parties P_2, P_3, P_4, but these in turn only hear from the sender, P_1, through the messages (M_2, M_3, M_4) (resp. (M'_2, M'_3, M'_4)), so these messages have to carry enough information to determine the secret.

Now, due to privacy, no single party must be able to determine whether the message sent is m or m'. If P_3 was corrupt and if $M_3 \neq M'_3$ for all possible initialization of all random tapes, then the adversary would be able to distinguish the message by simply looking at whether M_3 or M'_3 is being sent by P_3. Hence, we see that there must exist an initial random tape for which $M_3 = M'_3$. For the rest of the attack we assume this is the case.

With the observations we have seen so far, a corrupt party P_2 can mount the following attack: If P_2 sees it needs to send M_2, it will send M'_2 instead. Since the protocol withstands an active attack, the transcript (M_2, M_3, M_4), which would be transformed to (M'_2, M_3, M_4) after the attack, would uniquely determine m. On the other hand, the very same transcript can arise from an actively corrupt P_4 that modifies the message M'_4 when the message is m' to M_4 (recall that $M'_3 = M_3$). In this case, due to the resilience of the protocol against one active attack, (M'_2, M_3, M_4) should reconstruct to the same message as (M'_2, M'_3, M'_4), which is m'. This is, however, a contradiction, since the same transcript cannot lead to two different messages.

3 Instantiating $\mathcal{F}_{\mathsf{StableNet}}^{P_S \to P_R}$ with Statistical Security

The goal of this section is to develop an information-theoretic protocol that instantiates $\mathcal{F}_{\mathsf{StableNet}}^{P_S \to P_R}$ against *active* adversaries, but replacing the condition $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq 2t + 1$ from Section 2.2 with $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq t + 1$. As shown in Section 2.2, perfect security cannot be achieved in this setting, so we settle with statistical security.

Our construction at a high level works as follows. First, we design a pair of functions $f(m) = (m_1, \ldots, m_n)$ and $g(m'_1, \ldots, m'_n) = m'$ such that, if $m'_i = m_i$ for at least $t + 1$ (unknown) indices, then $m' = m$. Also, it should hold that no set of at most t values m_i leaks anything about m. Assuming the existence of such pair of functions, we can envision a simple construction of a protocol $\Pi_1(P_S, P_R, m)$ that guarantees that a receiver P_R gets the message m sent by a sender P_S, as long as P_R comes online either in the same round where P_S is, or in the next one. This operates as follows: P_S computes $(m_1, \ldots, m_n) = f(m)$, and, in every round, P_S sends m_i to party P_i, as well as m to P_R. Once a party P_i receives m_i, it sends this value to P_R in the next round. Let $m'_1, \ldots, m'_n$ be the values received by P_R when it comes online, where $m'_i = \perp$ if P_R does not receive a message from P_i (notice that m'_i could differ from m_i if P_i is actively corrupt). Since $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq t + 1$, we see that at least $t + 1$ of the m'_i are equal to m_i, so P_R can output $m = g(m'_1, \ldots, m'_n)$.

Now, we would like to "bootstrap" the protocol Π_1 into a protocol $\Pi_2(P_S, P_R, m)$ that guarantees that a receiver P_R gets the message m sent by a sender P_S, as long as P_R comes online either in the same round where P_S is, in the next one, *or in the one after that.* To this end, the parties run $\Pi_1(P_S, P_R, m)$, which guarantees that P_R gets m if it comes online in the same round as P_S, or at most in the round after. However, to deal with the case in which P_R comes online two rounds after P_S, the parties also execute the following *in parallel*: P_S

computes $(m_1, \ldots, m_n) = f(m)$ and executes $\Pi_1(P_S, P_R, m_i)$ for $i = 1, \ldots, n$. This ensures that every $P_i \in \mathcal{O}_2$ will get m_i, and at this point, the parties in $\mathcal{O}_3 \cap \mathcal{O}_2$ can send these to P_R in the third round. Upon receiving m'_i, P_R outputs $m = g(m'_1, \ldots, m'_n)$.

To analyze the protocol Π_2, assume for simplicity that $P_S \in \mathcal{O}_1$. We first observe that if $P_R \in \mathcal{O}_1 \cup \mathcal{O}_2$, then P_R gets m as $\Pi_1(P_S, P_R, m)$ is being executed. If, on the other hand, $P_R \in \mathcal{O}_3$, P_R gets m as $g(m_1, \ldots, m_n)$ since the parties $P_i \in \mathcal{O}_2$ get m_i from $\Pi_1(P_S, P_R, m_i)$. This idea can be iterated to obtain protocols that deliver messages as long as P_R comes online at most k rounds after P_S comes online.

In what follows we present the tools necessary to formalize this idea, and later discuss the actual protocols for instantiating $\mathcal{F}_{\mathsf{StableNet}}^{P_S \to P_R}$.

3.1 Robust Secret Sharing

The functions f and g discussed above are instantiated using robust secret-sharing, which are techniques that enables a dealer to distribute a secret among multiple nodes in such a way that (1) no subset of at most t nodes learn the secret and (2) if each node sends its share to a receiver, no subset of at most t corrupt nodes can stop the receiver from learning the correct secret.

The definition we consider here is more general than standard definitions from the literature since, at reconstruction time, we allow for missing shares, and if there are many of these we allow the reconstruction algorithm to output an error signal $\perp$. However, if there are enough honest non-missing shares, then reconstruction of the correct message must be guaranteed. This is needed since, in our protocols, there are some rounds in which parties may not receive enough shares to reconstruct the right secret, and they must be able to detect this is the case to wait for subsequent rounds where more shares are available.

▶ **Definition 3.** *Let $A \subseteq \{1, \ldots, n\}$ with $|A| \leq t$. A robust secret-sharing (RSS) scheme with deletions having message space M and share space S is made up of two randomized polytime functions, $\mathsf{share} : \mathsf{M} \to \mathsf{S}^n$ and $\mathsf{rec} : \mathsf{S}^n \to \mathsf{M}$, satisfying the properties below for any not-necessarily-polytime algorithm $\mathcal{A}$. Let $(s_1, \ldots, s_n) = \mathsf{share}(m)$. Let $B^c = \mathcal{A}(\mathsf{missing}, \{s_j\}_{j \in A}) \subseteq \mathcal{P}$ denote a set chosen by $\mathcal{A}$ of shares to be deleted. Let $(s'_1, \ldots, s'_n)$ be defined as follows: $s'_i = \perp$ for $i \in B^c$, $s'_i = \mathcal{A}(i, \{s_j\}_{j \in A}) \in \mathsf{S}$ for $i \in A \cap B$ and $s'_i = s_i$ for $i \in A^c \cap B$.*

- ***Privacy.*** *The distribution of $\{s_i\}_{i \in A}$ is independent of m.*
- ***Error detection.*** *With probability $1 - \mathsf{negl}(\kappa)$, $\mathsf{rec}(s'_1, \ldots, s'_n)$ outputs either m or $\perp$.*
- ***Guaranteed reconstruction.*** *If $|A^c \cap B| > t$ then, with probability $1 - \mathsf{negl}(\kappa)$, it holds that $m = \mathsf{rec}(s'_1, \ldots, s'_n)$.*

Several robust secret-sharing constructions can be found in the literature. However, since we consider a non-standard version of robust secret-sharing, we present below a concrete construction that fits Definition 3, which is motivated on the so-called information-checking signatures from [11]. We remark that *any* instantiation of Definition 3 will suffice for our stable network construction, with better parameters such as share length of computational complexity directly leading to direct improvements on our protocols.

The following proposition shows that the scheme $(\mathsf{share}, \mathsf{rec})$ is an RSS scheme with error detection.

▶ **Proposition 4.** *The construction $(\mathsf{share}, \mathsf{rec})$ from above is an RSS scheme with deletions.*

RSS scheme with deletions: $(\mathsf{share}, \mathsf{rec})$

$\mathsf{share}(m)$: Compute Shamir shares $m_1, \ldots, m_n$ of m. For each $i \in \{1, \ldots, n\}$, sample $(\alpha_i, \{\beta_{ij}\}_{j=1}^n)$, and let, for every $i, j \in \{1, \ldots, n\}$, $\tau_{ij} = \alpha_j m_i + \beta_{ji}$. Return $(s_1, \ldots, s_n)$, with $s_i = (m_i, (\alpha_i, \{\beta_{ij}\}_{j=1}^n, \{\tau_{ij}\}_{j=1}^n)$.

$\mathsf{rec}(s'_1, \ldots, s'_n)$. Let $B = \{i : s'_i \neq \perp\}$. Parse each s'_i for $i \in B$ as $(m'_i, (\alpha'_i, \{\beta'_{ij}\}_{j=1}^n, \{\tau'_{ij}\}_{j=1}^n)$. Then proceed as follows:

1. If $|B| \geq t+1$: for every $i \in B$ do the following. If $\alpha'_j m'_i + \beta'_{ji} \stackrel{?}{=} \tau'_{ij}$ does not hold for at least $t+1$ values of $j \in B$, then set $m'_i = \perp$.[a]
2. After this process, if $|\{m'_i : m'_i \neq \perp\}| > t$, then using any subset of this set of size $t+1$ to interpolate a polynomial $f(x)$ of degree at most t, and output $m = f(0)$. Else, output $\perp$.

[a] In particular, if $0 \leq |B| \leq t$ then all m'_i would be set to $\perp$ as the check would always fail.

Proof. Let $\mathsf{share}(m) = (s_1, \ldots, s_n)$ with $s_i = (m_i, (\alpha_i, \{\beta_{ij}\}_{j=1}^n), \{\tau_{ij} = \alpha_j m_i + \beta_{ji}\}_{j=1}^n)$. First we argue privacy. It is clear that the n Shamir shares $m_1, \ldots, m_n$ do not leak anything about the secret m towards the adversary. Additionally, the keys $(\alpha_i, \{\beta_{ij}\}_{j=1}^n)$ are simply random values, which do not leak anything either. Finally, each P_i receives $\{\tau_{ij} = \alpha_j m_i + \beta_{ji}\}_{j=1}^n$, but these only involve m_i, which is already known by P_i. Notice that, since β_{ji} is uniformly random and unknown to P_i (if $j \neq i$), P_i learns no information about α_j. This will be crucial since, as we show below, α_j is used to prevent P_i from changing their share.

Now, to see the guaranteed reconstruction property, let $(s'_1, \ldots, s'_n)$ be as in Definition 3. Assume that $|A^c \cap B| > t$, we want to show that $\mathsf{rec}(s'_1, \ldots, s'_n)$ outputs m in this case. Let us write each s'_i for $i \in A \cap B$ as $s'_i = (m'_i, (\alpha'_i, \{\beta'_{ij}\}_{j=1}^n), \{\tau'_{ij}\}_{j=1}^n)$. We claim that if $m'_i = m_i + \delta_i$ with $\delta_i \neq 0$, then $\tau'_{ij} = \alpha_j m'_i + \beta_{ji}$ for at least $j \in A^c \cap B$ can only happen with negligible probability. To see why this holds, let us write $\tau'_{ij} = \tau_{ij} + \epsilon_{ij}$, so $\tau'_{ij} = (\alpha_j m_i + \beta_{ji}) + \epsilon_{ij} = (\alpha_j m'_i + \beta_{ji}) - \alpha_j \delta_i + \epsilon_{ij}$. For this to be equal to $\alpha_j m'_i + \beta_{ji}$, it has to hold that $\alpha_j = \delta_i^{-1} \epsilon_{ij}$. However, δ_i and ϵ_{ij} are functions of $\{s_\ell\}_{\ell \in A}$, so they are computed independently of the uniformly random value α_j since $j \notin A$. This shows that the equation $\alpha_j = \delta_i^{-1} \epsilon_{ij}$ for at least $j \in A^c \cap B$ can only hold with probability at most $1/|\mathbb{F}| = \mathsf{negl}(\kappa)$, so in particular the claim above holds (recall that $n = \mathsf{poly}(\kappa)$).

From the above we see that if $m'_i \neq m_i$ then, with overwhelming probability, $\tau'_{ij} \neq \alpha_j m'_i + \beta_{ji}$ for every $j \in A^c \cap B$, so in particular $\tau'_{ij} = \alpha_j m'_i + \beta_{ji}$ can only be satisfied for $j \in A \cap B$, but since $|A \cap B| \leq t$, we see that m'_i would be set to $\perp$ from the definition of $\mathsf{rec}(\cdot)$. As a result, only values with $m'_i = m_i$ remain, and since there are at least $|A^c \cap B| > t$ of these, we see that $\mathsf{rec}(\cdot)$ outputs m correctly in this case.

The argument above also shows the error detection property: the extra assumption $|A^c \cap B| > t$ was only used at the end to show that the set $\{m'_i : m'_i \neq \perp\}$ will have at least $t+1$ elements, in which case the correct m could be reconstructed. If this does not hold, then $\mathsf{rec}(\cdot)$ outputs $\perp$. ◀

3.2 Delivering within 2 rounds

Let $(\mathsf{share}, \mathsf{rec})$ be a robust secret-sharing scheme with deletions. We begin by presenting a protocol $\Pi_1(P_S, P_R, m)$ that guarantees that P_R gets the message m sent by P_S as long as P_R comes online either in the same round as P_S, or at most one round later. First, we define the concept of k-delivery, which formalizes and generalizes this notion.

Protocol $\Pi_1(P_S, P_R, m)$

P_S does the following:
- Let $(s_1, \ldots, s_n) = \mathsf{share}(m)$. Send s_i to P_i in every round.
- Send m to P_R.

Every party P_i does the following:
- P_i sets an internal variable $\mathtt{s}_i = \bot$. In every round, if P_i receives s_i from P_i, then it sets $\mathtt{s}_i = s_i$.
- In every round, if $\mathtt{s}_i \neq \bot$, then P_i sends $\mathtt{s}_i$ to P_R.

P_R does the following in every round:
- If P_R receives m from P_S, then P_R outputs m.
- Let s'_i be the message P_R receives from P_i, setting $s'_i = \bot$ if no such message arrives. If $\mathsf{rec}(s'_1, \ldots, s'_n) \neq \bot$, then P_R outputs this value.

▶ **Definition 5** (k-delivery)**.** *A protocol Π is said to satisfy k-delivery if it instantiates the functionality $\mathcal{F}_{\mathsf{StableNet}}^{P_S,P_R}$ (with statistical security), modified so that P_R is only guaranteed to receive the message sent by P_S if $P_R \in \bigcup_{r=0}^{k} \mathcal{O}_{r_S+r}$, where r_S is the first round in which $P_S \in \mathcal{O}_{r_S}$. If $P_R \notin \bigcup_{r=0}^{k} \mathcal{O}_{r_S+r}$, then P_R cannot output an incorrect message.*

▶ **Proposition 6.** *$\Pi_1(P_R, P_S, m)$ satisfies 1-delivery.*

Proof. Privacy holds from the privacy of the robust secret-sharing scheme.

Now, assume that $P_R \in \mathcal{O}_{r_S} \cup \mathcal{O}_{r_S+1}$. If $P_R \in \mathcal{O}_{r_S}$, then P_R gets m as it is being sent by P_S directly. On the other hand, if $P_R \in \mathcal{O}_{r_S+1}$, the argument is the following. First, each $P_i \in \mathcal{O}_{r_S}$ receves s_i from P_S, which in particular means that the parties in $\mathcal{O}_{r_S} \cap \mathcal{O}_{r_S+1} \cap \mathcal{H}$ send the correct s_i to P_R. P_R receives at least $t+1$ correct shares s_i and at most t incorrect ones, hence, by the guaranteed reconstruction property of the RSS, P_R obtains s from these shares.

Finally, the fact that if $P_S \notin \mathcal{O}_{r_S} \cup \mathcal{O}_{r_S+1}$ then P_S does not output an incorrect message follows from the error detection property of $(\mathsf{share}, \mathsf{rec})$. ◀

3.3 From $(k-1)$-delivery to k-delivery

Now we show that, given a protocol $\Pi_{\mathsf{k}-1}(P_R, P_S, \cdot)$ that achieves $(k-1)$-delivery, one can obtain a protocol that achieves k-delivery. This is achieved by Protocol $\Pi_{\mathsf{k}}(P_R, P_S, m)$.

Protocol $\Pi_{\mathsf{k}}(P_R, P_S, m)$

In the following, multiple protocols will be executed in parallel. We assume that messages are tagged with special identifiers so that they can be effectively distinguished.

The parties execute $\Pi_{\mathsf{k}-1}(P_S, P_R, m)$. In parallel, they execute the following.
- Let $(s_1, \ldots, s_n) = \mathsf{share}(m)$. The parties run n protocol instances $\Pi_{\mathsf{k}-1}(P_S, P_i, s_i)$ for $i = 1, \ldots, n$.
- Each P_i, upon outputting s_i from $\Pi_{\mathsf{k}-1}(P_S, P_i, s_i)$, send (s_i) to P_R in all subsequent rounds.
- P_R initializes variables $\mathtt{s}_1, \ldots, \mathtt{s}_n = \bot$. Then P_R does the following in every round:
  - Upon outputting s_i from some execution $\Pi_{\mathsf{k}-1}(P_S, P_i, s_i)$, P_R sets $\mathtt{s}_i = s_i$.
  - Upon receiving s'_i from some party, sets $\mathtt{s}_i = s'_i$.
  - P_R outputs $\mathsf{rec}(\mathtt{s}_1, \ldots, \mathtt{s}_n)$ if this value is not $\bot$.

▶ **Proposition 7.** *Protocol* $\Pi_{\mathsf{k}}(P_S, P_R, m)$ *achieves* k*-delivery.*

Proof. Let r_S be the first round in which $P_S \in \mathcal{O}_{r_S}$, and assume that $P_R \in \bigcup_{r=0}^{k} \mathcal{O}_{r_S+r}$. If $P_R \in \bigcup_{r=0}^{k-1} \mathcal{O}_{r_S+r}$, then P_R would receive m correctly from the properties of $\Pi_{\mathsf{k}-1}$.

Given the above, it remains to analyze the case in which $P_R \in \mathcal{O}_{r_S+k}$. From the properties of $\Pi_{\mathsf{k}-1}$, every party $P_i \in \mathcal{O}_{r_S+(k-1)}$ receives s_i from P_S in round $r_S + (k-1)$. In particular, each party $P_i \in \mathcal{O}_{r_S+(k-1)} \cap \mathcal{O}_{r_S+k}$ sends s_i to P_R in round $r_S + k$. An analysis similar to the one in the proof of Proposition 6 shows that P_R is able to recover m from this information, and it also shows that if $P_R \notin \bigcup_{r=0}^{k} \mathcal{O}_{r_S+r}$, then P_R cannot be fooled into reconstructing an incorrect message. ◀

Combining Propositions 6 and 7, we obtain the following corollary:

▶ **Corollary 8.** *For every* k*, there exists a protocol* Π_{k} *satisfying* k*-delivery.*

Now, recalling that the B-assumption implies that there is one round among $1, \ldots, B$ in which P_S will come online, and a round among $B+1, \ldots, 2B$ in which P_R is online as well, we obtain the following theorem as a corollary.

▶ **Theorem 9.** *Assume that* $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq t+1$ *for every* $r > 0$*. Then, protocol* $\Pi_{\mathsf{2B}}(P_R, P_S, \cdot)$ *instantiates the functionality* $\mathcal{F}_{\mathsf{StableNet}}^{P_R \to P_S}$ *in the* $\mathcal{F}_{\mathsf{UnstableNet}}$*-hybrid model with statistical security against an adversary actively corrupting* $t < n/2$ *parties.*[3]

▶ Remark 10. The communication complexity of Π_{k} is $\Theta(n^k)$. This is because, in the execution of Π_{k}, P_S must use $\Pi_{\mathsf{k}-1}$ to communicate a share to each single party, adding a factor of n with respect to the communication complexity of this protocol. This is too inefficient for large values of k. We leave is an open problem the challenging task of obtaining instantiations of $\mathcal{F}_{\mathsf{StableNet}}^{P_S,P_R}$ with statistical security in the setting in which $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq t+1$ having communication complexity that is polynomial in the bound B.

References

1. Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Secure MPC: Laziness leads to GOD. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 120–150. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64840-4_5.
2. Arka Rai Choudhuri, Aarushi Goel, Matthew Green, Abhishek Jain, and Gabriel Kaptchuk. Fluid mpc: Secure multiparty computation with dynamic participants. In *Annual International Cryptology Conference*, pages 94–123. Springer, 2021.
3. Ivan Damgård, Martin Geisler, Mikkel Krøigaard, and Jesper Buus Nielsen. Asynchronous multiparty computation: Theory and implementation. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 160–179. Springer, Heidelberg, March 2009. doi:10.1007/978-3-642-00468-1_10.
4. Ivan Damgård, Daniel Escudero, and Divya Ravi. Information-theoretically secure mpc against mixed dynamic adversaries. Thheory of Cryptography Conference, 2021.
5. Matthias Fitzi, Martin Hirt, and Ueli M. Maurer. Trading correctness for privacy in unconditional multi-party computation (extended abstract). In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 121–136. Springer, Heidelberg, August 1998. doi:10.1007/BFb0055724.

[3] As with Theorem 2, in principle the restriction is simply $t < n$, but we have that $n - t = |\mathcal{H}| \geq |\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq t+1$, so $n \geq 2t+1$.

6 Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Information processing letters*, 43(4):169–174, 1992.

7 Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. YOSO: you only speak once - secure MPC with stateless ephemeral roles. In *CRYPTO 2021*, 2021.

8 Yue Guo, Rafael Pass, and Elaine Shi. Synchronous, with a chance of partition tolerance. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 499–529. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26948-7_18.

9 Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

10 Chiu-Yuen Koo. Secure computation with partial message loss. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 502–521. Springer, Heidelberg, March 2006. doi:10.1007/11681878_26.

11 Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, May 1989. doi:10.1145/73007.73014.

12 Rahul Rachuri and Peter Scholl. Le mans: Dynamic and fluid mpc for dishonest majority. Cryptology ePrint Archive, Paper 2021/1579, 2021. URL: https://eprint.iacr.org/2021/1579.

13 Vassilis Zikas, Sarah Hauser, and Ueli M. Maurer. Realistic failures in secure multi-party computation. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 274–293. Springer, Heidelberg, March 2009. doi:10.1007/978-3-642-00457-5_17.

A Shamir Secret Sharing

Throughout this work we will make use of Shamir secret sharing in order to distribute data among different parties. To secret-share a value $s \in \mathbb{F}$ among the n parties $P_1, \ldots, P_n$ using threshold t, a dealer proceeds as follows: (1) sample a uniformly random polynomial $f(x) \in \mathbb{F}[x]$ of degree at most t, subject to $f(0) = s$, and (2) send to P_i its share $s_i := f(i)$. It is well known that for every set of $t+1$ points (i, s_i) there exists a unique polynomial $f(x)$ of degree at most t such that $f(i) = s_i$ for all i, which implies that any set of at least $t+1$ shares can recover the secret, and any set of t shares does not reveal anything about the secret.

Bivariate sharings

Sometimes we will make use of *bivariate sharings*, in which the dealer, to distribute a secret $s \in \mathbb{F}$, samples a random symmetric bivariate polynomial $f(x, y)$ of degree at most t in each variable subject to $f(0, 0) = s$, and sends the polynomial $f(x, i)$ to P_i. As before, given at most t of these polynomials nothing is leaked about the secret s since any secret could be chosen so that it looks consistent with the given polynomials.

Error-detection and error-correction

Given m shares among which at most t can be incorrect, then the parties output $f(0)$ as the secret, where $f(x)$ is the reconstructed polynomial. Given m shares $\{s_i\}$ among which at most t are incorrect we have the following two possibilities:

- If at least $t+1$ are guaranteed to be correct, *error-detection* can be performed by checking if these shares all lies in a polynomial of degree at most t, and if this is the case, the reconstructed polynomial is guaranteed to be correct since it is determined by the $t+1$ correct shares.

- If at least $2t+1$ are guaranteed to be correct, *error-correction* is possible by looping through all possible subsets of these shares of size $2t+1$ and checking if all shares in the given subset are consistent with a polynomial of degree at most t. The subset used for reconstructing this polynomial has $2t+1$ points among which at least $t+1$ are correct (since at most t shares are assumed to be incorrect), which guarantees that the reconstructed polynomial is the correct one. Although the process of looping through all subsets of size $2t+1$ can be too inefficient if m is much larger than $2t+1$, this can be made polynomial in m by using error-detection algorithms like Berlekamp-Welch [6].

In some of our protocols we will need a version of error-correction, which we call *enhanced error-correction*, in which the correct polynomial is recovered if there are enough correct shares, and else an error is output. To this end, given $m \geq 2t+1$ shares as above among which at most t are incorrect, all possible subsets of $2t+1$ shares are inspected, checking if all these shares are consistent with a polynomial of degree at most t. If one such subset is found, then its corresponding polynomial is output, and else, an error $\perp$ is produced as the result. By the same analysis as above, this either results in the correct polynomial or an error. The main complication is that error-correcting algorithms like Berlekamp-Welch are not designed to handle this setting in which not enough correct shares may be available, but one can easily modify this algorithm to handle this case (see for example [4]).

B A More Efficient Protocol with Perfect Security

Recall that in Section 2.2 we presented a protocol to instantiate the functionality $\mathcal{F}_{\mathsf{StableNet}}$, which is intended to represent a traditional stable and secure network among the n parties. This is the typical communication model used in several MPC protocols, and, assuming $t < n/3$, we can find perfectly secure protocols in this model which can be used together with our protocol $\Pi^{\mathsf{perf,active}}_{\mathsf{StableNet}}(P_S, P_R)$ from Section 2.2 to obtain a perfectly secure protocol over an unstable network.

In order to instantiate the functionality $\mathcal{F}_{\mathsf{StableNet}}$, we required that the scheduling the adversary provides allows each party to come online at least *once* within certain amount of rounds, say B. This is necessary since $\mathcal{F}_{\mathsf{StableNet}}$ requires each message between honest parties to be delivered, and if the receiver never comes online such guarantee cannot hold. Unfortunately, our protocol $\Pi^{\mathsf{perf,active}}_{\mathsf{StableNet}}(P_S, P_R)$ requires $2B$ rounds to deliver a message between a sender and a receiver, which ultimately means that the final protocol after composing $\Pi^{\mathsf{perf,active}}_{\mathsf{StableNet}}(P_S, P_R)$ with an existing perfectly secure protocol would lead to a multiplicative overhead of $2B$ in the number of rounds.

Round-count is a very sensitive metric in distributed protocols, especially in high-latency scenarios where every communication trip incurs in a noticeable waiting time. Furthermore, the $\theta(B)$ overhead may not be so noticeable if the higher level protocol has a low round count, but unfortunately, it is a well-known open problem to achieve constant round protocols with *perfect security* for functionalities outside NC^1 while achieving polynomial computation and communication complexity. Motivated by this, we develop in this section a perfectly secure protocol over an unstable network whose number of rounds corresponds to the depth of the circuit being computed plus a term that depends on B, but is independent of the size of the circuit, matching the round complexity of existing protocols over stable networks. Furthermore, after the inputs have been provided, our protocol does not require anymore the

assumption that each party has to be online at least once every B rounds.[4] This is because, as we will see, our protocol only relies on the assumption that $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq 2t+1$ for every round r in order to transmit and advance the *secret-shared state* of the computation from one round to the next. Intuitively, it is irrelevant if certain specific parties become online at certain points of the protocol, and the only thing that matters is that *enough* parties remain online from one round to the next one, irrespectively of their identities.

B.1 Bivariate Sharings and Transition of Shares

We describe the input and preprocessing phases of our protocol in Section B.2, and in Section B.3 we describe its computation phase. However, before we dive into the protocols themselves, we need to present certain primitives that will be useful for these constructions. These are bivariate sharings, together with methods for transmitting bivariate shared values from one round to the next. This will allow the parties to "transmit" the state of the computation from the parties that are online in a given round, to these online in the next one, making progress in one layer of the circuit at the same time.

We say that the parties have bivariate shares of a value s if there exists a symmetric bivariate polynomial $f(x,y)$ of degree at most t in both variables such that (1) each party $P_i \in \mathcal{P}$ has $f(x,i)$ and (2) it holds that $f(0,0) = s$. We denote this by $\langle s \rangle$. Observe that this scheme is linear, i.e. parties can locally compute additions of secret shared values, which is denoted by $\langle x+y \rangle \leftarrow \langle x \rangle + \langle y \rangle$.

Bivariate sharings were used indirectly in Section 2.2 to instantiate $\mathcal{F}_{\mathsf{StableNet}}^{P_S \to P_R}$ with perfect security against an active adversary. This type of sharings proved useful in Protocol $\Pi_{\mathsf{StableNet}}^{\mathsf{perf,active}}(P_S, P_R)$ to "transfer" a state between a set of parties to another one, and this is the purpose of this primitive in this section as well. In a bit more detail, during the execution of our protocol it will not hold that all parties have shares of certain given values, but rather only specific subsets corresponding to online parties will do. Since the set of online parties potentially changes from round to round, a crucial primitive our protocol relies on is what we call *transition of shares*, which takes care of transmitting the shared state from one set of parties to another.

We first formalize the notion that only (part of) the online parties hold shares of a given value. We say that the parties have a bivariate-shared value s *in round* r if there exists a symmetric bivariate polynomial $f(x,y)$ of degree at most t in both variables such that (1) there exists a subset $\mathcal{S}_r \subseteq \mathcal{O}_r \cap \mathcal{H}$ with $|\mathcal{S}_r| \geq 2t+1$ such that each $P_i \in \mathcal{S}_r$ has $f(x,i)$, (2) each $P_i \in (\mathcal{O}_r \cap \mathcal{H}) \setminus \mathcal{S}_r$ has set their share to either $f(x,i)$, or a predefined value $\perp$, and (3) it holds that $f(0,0) = s$. This is denoted by $\langle s \rangle^{\mathcal{O}_r}$. Observe that nothing is required from parties outside $\mathcal{O}_r \cap \mathcal{H}$. Also, notice that if all the parties have bivariate shares of a value s, which we denote by $\langle s \rangle$, then it holds that $\langle s \rangle^{\mathcal{O}_r}$ for every r.

A protocol for transition of shares is a one-round protocol in which the parties start with $\langle s \rangle^{\mathcal{O}_r}$ in round r, and they obtain $\langle s \rangle^{\mathcal{O}_{r+1}}$ in the next round $r+1$. In what follows we present a protocol for transition of shares, which is motivated in the perfectly secure protocol for instantiating $\mathcal{F}_{\mathsf{StableNet}}^{P_S \to P_R}$ from Section 2.

[4] However, the output will be received only by the parties who happen to be online at the output phase.

Protocol Π_{transfer}

Input: $\langle s \rangle^{\mathcal{O}_r}$ in round r
Output: $\langle s \rangle^{\mathcal{O}_{r+1}}$ in round $r+1$.

Parties do the following:

1. For each $i = 1, \ldots, n$, if P_i has a share $f(x, i)$ of $\langle s \rangle^{\mathcal{O}_{r+1}}$ (different to $\perp$), then P_i sends $f(j, i)$ to P_j for $j = 1, \ldots, n$.
2. For each $j = 1, \ldots, n$, if P_j receives at least $2t + 1$ messages $\{f(j, i)\}_i$, then P_j performs *enhanced* error correction (see Section A) to either recover $f(j, x)$ or output an error $\perp$.

▶ **Theorem 11.** *If executed in round r, protocol Π_{transfer} guarantees that the parties get sharings $\langle s \rangle^{\mathcal{O}_{r+1}}$.*

Proof. Let $\mathcal{S}_r \subseteq \mathcal{O}_r \cap \mathcal{H}$ with $|\mathcal{S}_r| \geq 2t + 1$ be the set of honest parties P_i having $f(x, i)$, guaranteed from the definition of bivariate sharings. Since the protocol above is executed in round r, each party $P_i \in \mathcal{S}_r$ will send $f(j, i)$ to each other party P_j, which in particular is received by the parties $P_j \in \mathcal{O}_{r+1} \cap \mathcal{O}_r \cap \mathcal{H}$, and given that $|\mathcal{S}_r| \geq 2t + 1$, the enhanced error-correction algorithm executed by P_j will result in P_j recovering $f(j, x)$, which is equal to $f(x, j)$. Let $\mathcal{S}_{r+1} := \mathcal{O}_{r+1} \cap \mathcal{O}_r \cap \mathcal{H}$ and note that (1) $|\mathcal{S}_{r+1}| \geq 2t + 1$ and also each $P_j \in \mathcal{S}_{r+1}$ has $f(x, j)$, (2) each $P_j \in (\mathcal{O}_{r+1} \cap \mathcal{H}) \setminus \mathcal{S}_{r+1}$ set their share to either $f(x, j)$ or $\perp$ due to the properties of the enhance error-correction mechanism, and (3) it (still) holds that $f(0, 0) = s$. From the definition of bivariate sharings, it holds that $\langle s \rangle^{\mathcal{O}_{r+1}}$. ◀

Transitioned Reconstruction

Another primitive that we will need in our protocol, besides transferring shares from one set of parties to another, consists of reconstructing a bivariate-shared value. Assume that the parties in round r have $\langle s \rangle^{\mathcal{O}_r}$. If all parties in round r send their shares $\{f(0, j)\}_j$ to all other parties, they can perform (enhanced) error correction to reconstruct $s = f(0, 0)$. In this way, the parties in $\mathcal{O}_r \cap \mathcal{H}$ are guaranteed to learn s. In particular, s is known by the parties in $\mathcal{O}_{r+1} \cap \mathcal{O}_r \cap \mathcal{H}$, which contains at least $2t + 1$ parties. This protocol is denoted by $s \leftarrow \Pi_{\mathsf{rec}}(\langle s \rangle^{\mathcal{O}_r})$.

▶ Remark 12. An important fact about the proof of Theorem 11 is that, it holds that $\mathcal{S}_{r+1} \subseteq \mathcal{O}_{r+1} \cap \mathcal{O}_r \cap \mathcal{H}$. In addition, the reconstruction protocol from above ensures that the parties in $\mathcal{O}_{r+1} \cap \mathcal{O}_r \cap \mathcal{H}$, so in particular the parties in $\mathcal{S}_{r+1}$, learn the secret. This will be important in our main protocol in Section B.3.

B.2 Preprocessing and Input Phases

We assume that the functionality to be computed is given by a layered circuit $(x_1^{(L)}, \ldots, x_{\ell_L}^{(L)}) = F(x_1^{(0)}, \ldots, x_{\ell_0}^{(0)})$. Considering layered circuits, in contrast to more general circuits, is useful for our construction since in this case the values in a given layer completely determine the current *state* of the computation, that is, the next layer, and in particular the remainder of the computation, is fully determined by these values. This is important since, as we will see, at the heart of our construction lies the possibility of a given set of online parties to transmit their shared state to the online parties in the next round, and, from the structure of the protocol, this state is comprised by the shared values in a given layer.

For our main protocol, we assume that *all* the parties have certain bivariate-shared multiplication triples (as specified below), plus bivariate shares of the inputs of the computation. By making use of the B-assumption, these shares can be computed by using *any* generic

MPC protocol for these tasks, together with our compiler from Section 2.2. This would incur a multiplicative overhead of B in the number of rounds, however, the circuit representing this computation is constant-depth, so this does not affect the overall result of this section. Notice that this does not require all the parties to be online during the computation of these sharings, but instead, the B-assumption, that requires every honest party to come online once every B rounds, suffices.

The correlation required for the computation consists of secret-shared values $(\langle a\rangle, \langle b\rangle, \langle c\rangle)$, one tuple for every multiplication gate in the circuit, where $a, b \in_R \mathbb{F}$ and $c = a \cdot b$.

B.3 Computation Phase

With the primitives described above, the protocol for computing the given functionality F is relatively straightforward: by making use of the Π_{transfer} and Π_{rec} protocols, the parties can use the standard approach to secure computation based on multiplication triples, making progress from round to round depending on the set of parties that is online. This is possible since, at the end of the execution of the method described in Section B.2, *all* the parties hold the preprocessing material and shares of the inputs (even if some parties were offline during certain parts of the execution), together with the fact that $|\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}| \geq 2t+1$ for every round r, which enables share transfer and reconstruction. The protocol is described in detail below. The security proof follows straightforwardly from existing techniques, together with the properties proven in Section B.1, and a sketch of this proof can be found towards the end of this section. Observe that the protocol requires only L rounds, which, added to the $O(1)$ rounds from the preprocessing and input phases, leads to a protocol with comparable round efficiency to protocols in the stable (i.e. traditional) model.

Protocol Π_{MPC}

Input: Secret-shared inputs $\langle x_1^{(0)}\rangle, \ldots, \langle x_{\ell_0}^{(0)}\rangle$, where ℓ_0 is the number of input wires.
Preprocessing: A multiplication triple $(\langle a\rangle, \langle b\rangle, \langle c = a \cdot b\rangle)$ for every multiplication gate in the circuit.
Output: Let L be the final round of the protocol. The parties have $\langle x_1^{(L)}\rangle^{\mathcal{O}_L}, \ldots, \langle x_{\ell_L}^{(L)}\rangle^{\mathcal{O}_L}$ in round L, where $(x_1^{(L)}, \ldots, x_{\ell_L}^{(L)}) = F(x_1^{(0)}, \ldots, x_{\ell_0}^{(0)})$.

For rounds $r = 1, \ldots, L$:

- The parties in round $r-1$ already have shares $\langle x_1^{(r-1)}\rangle^{\mathcal{O}_{r-1}}, \ldots, \langle x_{\ell_{r-1}}^{(r-1)}\rangle^{\mathcal{O}_{r-1}}$.
- The parties in round r obtain shares $\langle x_1^{(r)}\rangle^{\mathcal{O}_r}, \ldots, \langle x_{\ell_r}^{(r)}\rangle^{\mathcal{O}_r}$ as follows:
  1. For every addition gate with inputs $\langle x\rangle^{\mathcal{O}_{r-1}}$ and $\langle y\rangle^{\mathcal{O}_{r-1}}$, the parties locally obtain $\langle x+y\rangle^{\mathcal{O}_{r-1}}$ and call $\langle x+y\rangle^{\mathcal{O}_r} \leftarrow \Pi_{\mathsf{transfer}}(\langle x+y\rangle^{\mathcal{O}_{r-1}})$.
  2. For every multiplication gate with inputs $\langle x\rangle^{\mathcal{O}_{r-1}}$ and $\langle y\rangle^{\mathcal{O}_{r-1}}$, the parties proceed as follows:
     a. Let $(\langle a\rangle, \langle b\rangle, \langle c\rangle)$ be the next available multiplication triple. The parties in round $r-1$ locally compute $\langle d\rangle^{\mathcal{O}_{r-1}} = \langle x\rangle^{\mathcal{O}_{r-1}} - \langle a\rangle^{\mathcal{O}_{r-1}}$ and $\langle e\rangle^{\mathcal{O}_{r-1}} = \langle y\rangle^{\mathcal{O}_{r-1}} - \langle b\rangle^{\mathcal{O}_{r-1}}$.
     b. The parties in round r learn d and e by calling $d \leftarrow \Pi_{\mathsf{rec}}(\langle d\rangle^{\mathcal{O}_{r-1}})$ and $e \leftarrow \Pi_{\mathsf{rec}}(\langle e\rangle^{\mathcal{O}_{r-1}})$.
     c. The parties in round r compute $\langle x \cdot y\rangle^{\mathcal{O}_r}$ as $d \cdot \langle b\rangle^{\mathcal{O}_r} + e \cdot \langle a\rangle^{\mathcal{O}_r} + \langle c\rangle^{\mathcal{O}_r} + d \cdot e$.[a]
  3. For every identity gate with input $\langle x\rangle^{\mathcal{O}_{r-1}}$ the parties call $\langle x\rangle^{\mathcal{O}_r} \leftarrow \Pi_{\mathsf{transfer}}(\langle x\rangle^{\mathcal{O}_{r-1}})$.

[a] Here is where Remark 12 becomes relevant: parties in $\mathcal{O}_r$ (or rather $\mathcal{S}_r$) can compute the linear combination defining $\langle x \cdot y\rangle^{\mathcal{O}_r}$ since both the constants and the sharings are known to the parties in $\mathcal{S}_r$.

▶ Remark 13 (About the output). In our protocol above, the parties in $\mathcal{O}_L$ obtain shares $\langle x_1^{(L)}\rangle^{\mathcal{O}_L}, \ldots, \langle x_{\ell_L}^{(L)}\rangle^{\mathcal{O}_L}$ in round L, where $(x_1^{(L)}, \ldots, x_{\ell_L}^{(L)}) = F(x_1^{(0)}, \ldots, x_{\ell_0}^{(0)})$ is the result of the computation. This output can be dealt with in multiple different ways:

- The parties in $\mathcal{O}_L$ can reconstruct the output to each other. This way, the parties in $\mathcal{O}_L$ are guaranteed to learn the output, but parties outside this set may not satisfy this.
- If the B-assumption holds for some B, the parties can reconstruct and transfer this sharing for B more rounds so that all parties learn the output.

B.4 Security Analysis

Now we provide a *sketch* of the security properties of protocol Π_{MPC} from Section B.3. Recall that the function to be computed is assumed to be given by a layered circuit $(x_1^{(L)}, \ldots, x_{\ell_L}^{(L)}) = F(x_1^{(0)}, \ldots, x_{\ell_0}^{(0)})$. Furthermore, it is assumed that the parties have bivariate shares of the inputs $\langle x_1^{(0)}\rangle, \ldots, \langle x_{\ell_0}^{(0)}\rangle$, and also, for every multiplication gate, a triple $(\langle a\rangle, \langle b\rangle, \langle c = a \cdot b\rangle)$ with a, b uniformly random in $\mathbb{F}$.[5] Recall that $\langle s\rangle^{\mathcal{O}_r}$ means that there is a large enough subset $\mathcal{S}_r \subseteq \mathcal{O}_r \cap \mathcal{H}$ such that every party $P_i \in \mathcal{S}_r$ has $f(x, i)$ such that $f(0, 0) = s$, and parties in $(\mathcal{O}_r \cap \mathcal{H}) \setminus \mathcal{S}_r$ either have $f(x, i)$ or a special symbol $\perp$.

Assume the protocol starts in round 0. We claim that the following invariant holds: In round r, the parties in $\mathcal{O}_r$ have shares of the intermediate results in layer r, namely $\langle x_1^{(r)}\rangle^{\mathcal{O}_r}, \ldots, \langle x_{\ell_r}^{(r)}\rangle^{\mathcal{O}_r}$. To see this we argue inductively. For $r = 0$ this follows trivially as we assumed that the parties start with shares $\langle x_1^{(0)}\rangle, \ldots, \langle x_{\ell_0}^{(0)}\rangle$, which in particular means they have shares $\langle x_1^{(0)}\rangle^{\mathcal{O}_0}, \ldots, \langle x_{\ell_0}^{(0)}\rangle^{\mathcal{O}_0}$.

Assume the invariant holds for r, and let us show it also holds for $r + 1$. Let $k \in \{1, \ldots, \ell_{r+1}\}$. From the definition of a layered circuit, the value $x_k^{(r+1)}$ can be computed in either one of three ways:

- *Identity gate* $x_k^{(r+1)} = x_i^{(r)}$. In this case the protocol instructs that the parties must call $\langle x_k^{(r+1)}\rangle^{\mathcal{O}_{r+1}} \leftarrow \Pi_{\mathsf{transfer}}(\langle x_i^{(r)}\rangle^{\mathcal{O}_r})$.
- *Addition gate* $x_k^{(r+1)} = x_i^{(r)} + x_j^{(r)}$. In this case the protocol dictates the parties to compute $\langle x_k^{(r)}\rangle^{\mathcal{O}_r} = \langle x_i^{(r)}\rangle^{\mathcal{O}_r} + \langle x_j^{(r)}\rangle^{\mathcal{O}_r}$, followed by $\langle x_k^{(r+1)}\rangle^{\mathcal{O}_{r+1}} \leftarrow \Pi_{\mathsf{transfer}}(\langle x_k^{(r+1)}\rangle^{\mathcal{O}_r})$.
- *Multiplication gate* $x_k^{(r+1)} = x_i^{(r)} \cdot x_j^{(r)}$. Here, the parties in $\mathcal{O}_r$ first compute locally $\langle d\rangle^{\mathcal{O}_r} = \langle x_i^{(r)}\rangle^{\mathcal{O}_r} - \langle a\rangle^{\mathcal{O}_r}$ and $\langle e\rangle^{\mathcal{O}_r} = \langle x_j^{(r)}\rangle^{\mathcal{O}_r} - \langle b\rangle^{\mathcal{O}_r}$, and call $d \leftarrow \Pi_{\mathsf{rec}}(\langle d\rangle^{\mathcal{O}_r})$ and $e \leftarrow \Pi_{\mathsf{rec}}(\langle e\rangle^{\mathcal{O}_{r-1}})$, which enables the parties in $\mathcal{O}_r \cap \mathcal{H}$, which include $\mathcal{O}_r \cap \mathcal{O}_{r+1} \cap \mathcal{H}$, to learn d and e. Observe that this does not reveal anthing about $x_i^{(r)}$ and $x_j^{(r)}$ to the adversary since a and b are assumed to be uniformly random and unknown to the adversary. Finally, these parties, which define the set $\mathcal{S}_{r+1}$, compute $d \cdot \langle b\rangle^{\mathcal{O}_{r+1}} + e \cdot \langle a\rangle^{\mathcal{O}_{r+1}} + \langle c\rangle^{\mathcal{O}_{r+1}} + d \cdot e$, which can be easily checked to be equal to $\langle x_i^{(r)} \cdot x_j^{(r)}\rangle^{\mathcal{O}_{r+1}}$, which is the same as $\langle x_k^{(r+1)}\rangle^{\mathcal{O}_{r+1}}$.

Since the invariant holds for every layer, in particular it holds for $r = L$, which shows that, after L rounds, the parties obtain $\langle x_1^{(L)}\rangle^{\mathcal{O}_L}, \ldots, \langle x_{\ell_L}^{(L)}\rangle^{\mathcal{O}_L}$. As mentioned in Remark 13 in Section B.3, these shared outputs can be handled in different ways, depending on the application under consideration.

[5] A simple "optimization" is that these shares do not need to be held by *all* the parties, but rather by these that will make use of these sharings in each corresponding round.

Weighted Secret Sharing from Wiretap Channels

Fabrice Benhamouda
Algorand Foundation, New York, NY, USA

Shai Halevi
Algorand Foundation, New York, NY, USA

Lev Stambler
Independent Researcher, NJ, USA

Abstract

Secret-sharing allows splitting a piece of secret information among a group of shareholders, so that it takes a large enough subset of them to recover it. In *weighted* secret-sharing, each shareholder has an integer weight, and it takes a subset of large-enough weight to recover the secret. Schemes in the literature for weighted threshold secret sharing either have share sizes that grow linearly with the total weight, or ones that depend on huge public information (essentially a garbled circuit) of size (quasi)polynomial in the number of parties.

To do better, we investigate a relaxation, (α, β)-ramp weighted secret sharing, where subsets of weight βW can recover the secret (with W the total weight), but subsets of weight αW or less cannot learn anything about it. These can be constructed from standard secret-sharing schemes, but known constructions require long shares even for short secrets, achieving share sizes of $\max\left(W, \frac{|\text{secret}|}{\epsilon}\right)$, where $\epsilon = \beta - \alpha$. In this note we first observe that simple rounding let us replace the total weight W by N/ϵ, where N is the number of parties. Combined with known constructions, this yields share sizes of $O\big(\max(N, |\text{secret}|)/\epsilon\big)$.

Our main contribution is a novel connection between weighted secret sharing and wiretap channels, that improves or even eliminates the dependence on N, at a price of increased dependence on $1/\epsilon$. We observe that for certain additive-noise $(\mathcal{R}, \mathcal{A})$ wiretap channels, any semantically secure scheme can be naturally transformed into an (α, β)-ramp weighted secret-sharing, where α, β are essentially the respective capacities of the channels $\mathcal{A}, \mathcal{R}$. We present two instantiations of this type of construction, one using Binary Symmetric wiretap Channels, and the other using additive Gaussian Wiretap Channels. Depending on the parameters of the underlying wiretap channels, this gives rise to (α, β)-ramp schemes with share sizes $|\text{secret}| \cdot \log N/\text{poly}(\epsilon)$ or even just $|\text{secret}|/\text{poly}(\epsilon)$.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives; Security and privacy → Information-theoretic techniques

Keywords and phrases Secret sharing, ramp weighted secret sharing, wiretap channel

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.8

Related Version *Full Version*: `IACR ePrint 2022/1578` [8]

Funding *Lev Stambler*: Work done while in Algorand Foundation, USA.

Acknowledgements We thank the anonymous reviewers for their comments, which vastly improved this work.

1 Introduction

Secret sharing [24, 10] allows a dealer to split some secret information among multiple parties, giving each party an individual share, so that large enough subsets of shareholder can recover the secret, but small subsets cannot learn any partial information about it. Such schemes are typically parametrized by the number of parties N and a threshold $T \leq N$, such that it takes at least T parties to recover the secret.

© Fabrice Benhamouda, Shai Halevi, and Lev Stambler;
licensed under Creative Commons License CC-BY 4.0

Weighted secret sharing (WSS) is similar, except that each shareholder j has an integer weight w_j, it takes a "heavy enough" subsets to recover the secret, while "light" subsets cannot learn any partial information about it. The threshold $T \in [N]$ is replaced by $\tau \in (0,1)$, such that it takes shareholders of aggregate weight τW to recover the secret (where W is the total weight, $W = \sum_{j\in[N]} w_j$).

One method of implementing WSS is to rely on standard secret-sharing with $N' = W$ and $T' = \tau W$, giving w_j shares to a shareholder j with weight w_j. While this solution can achieve good rate for long secrets (see Section 3.1), it is very wasteful for short ones, as its share sizes grow linearly with the weight. Prior work on weighted secret sharing explored other solutions (e.g., using Chinese remaindering) or limited models (e.g., specific weight hierarchies). But they all still feature either linear dependency of the share-size on W, severe restrictions to the access structures that can be realized, or huge public information that must be broadcasted to everyone alongside the individual shares. (See more discussion in Section 1.2 below.)

In an attempt to do better, in this work we consider the relaxed model of ramp secret-sharing [9], that has a fuzzy threshold. Specifically, an (α,β)-ramp weighted secret sharing scheme allows any subset of aggregate weight at least βW to recover the secret, but subsets of weight αW or less cannot learn any information about it. Such gaps were considered often in the literature for standard secret-sharing schemes, but to our knowledge were not studied in the context of weighted secret sharing.

It is not hard to see (and we describe it explicitly in Section 3) that this relaxation enables shorter secrets, just by keeping only a $1/\epsilon$ precision for the weights, where $\epsilon = \beta - \alpha$. Rather than linear dependence on the weights, we now get linear dependence on N/ϵ (where the dependence on the number of parties N is due to the accumulation of rounding errors in this limited-precision approximation).

Beyond this simple observation, the main technical meat in this work is a novel blueprint for (α,β)-ramp WSS schemes, by exploring a surprising connection to secure transmission schemes for wiretap channels. These constructions reduce or even eliminate the dependence on N, at the price of potentially worse (but still polynomial) dependence on $1/\epsilon$. We note that the field of wiretap coding is an ongoing line of research with an aim of decreasing dependence on $1/\epsilon$. Any advances in wiretap coding can easily be applied to WSS with our construction.

1.1 Overview of Our Techniques

The starting point for our new blueprint is the following approach: On input $\boldsymbol{s}$, the dealer gives each shareholder j an independent noise vector $\boldsymbol{e}_j$, whose magnitude depends on their weight, and publishes the value $\boldsymbol{g} = \mathsf{Enc}(\boldsymbol{s}) + \sum_j \boldsymbol{e}_j$, where $\mathsf{Enc}(\cdot)$ is some encoding function.[1] Given the public $\boldsymbol{g}$ and their individual $\boldsymbol{e}_j$'s, the only information that a set T of shareholder has on the secret $\boldsymbol{s}$ is the value

$$\boldsymbol{g}_T = \boldsymbol{g} - \sum_{j\in T} \boldsymbol{e}_j = \mathsf{Enc}(\boldsymbol{s}) + \sum_{j\notin T} \boldsymbol{e}_j.$$

We can therefore associate with each subset T an additive-noise channel $C_T : x \mapsto x + \sum_{j\notin T} \boldsymbol{e}_j$, such that the information that T learns about $\boldsymbol{s}$ is exactly the received value $C_T(\mathsf{Enc}(\boldsymbol{s}))$. We are seeking an encoding function $\mathsf{Enc}(\cdot)$ so that:

[1] Publishing $\boldsymbol{g}$ can be done by sending it to all the shareholders, which will only double the size of the shares that each one holds. It may be possible to use information dispersal to do even better.

- Any qualified set S can recover $\boldsymbol{s}$ from $C_S(\mathsf{Enc}(\boldsymbol{s}))$;
- For any unqualified set T, seeing $C_T(\mathsf{Enc}(\boldsymbol{s}))$ yields no information on $\boldsymbol{s}$.

Intuitively, the smaller (or "lighter") the set is, the more error components it is missing, so the more noisy its channel will be. Consider now $\mathcal{R}$ which is "the most noisy channel" for any qualified set, and $\mathcal{A}$ which is "the least noisy channel" for any unqualified set.

We can hope that $\mathcal{R}$ is less noisy than $\mathcal{A}$, and use a good transmission scheme for the wiretap channel $(\mathcal{R}, \mathcal{A})$, with receiver channel $\mathcal{R}$ and adversary channel $\mathcal{A}$. (Recall that a wiretap scheme for a pair of channels $(\mathcal{R}, \mathcal{A})$ consists of an encoding function $\mathsf{Enc}(\cdot)$ such that a secret $\boldsymbol{s}$ can be recovered from $\mathcal{R}(\mathsf{Enc}(\boldsymbol{s}))$ whp, but where $\mathcal{A}(\mathsf{Enc}(\boldsymbol{s}))$ yields almost no information on $\boldsymbol{s}$.)

Trying to flesh out this approach, we need to associate an error distribution $\mathcal{D}_{w_j}$ to every weight $w_j \in \mathbb{N}$, so that whenever $\sum_{j\in A} w_j > \sum_{j\in B} w_j$ it holds that $\sum_{j\in A} \mathcal{D}_{w_j}$ has "more error" than $\sum_{j\in B} \mathcal{D}_{w_j}$. Then we need to find two concrete channels $\mathcal{R}, \mathcal{A}$ such that

- $\mathcal{R}$ is at least as noisy as C_Q for any qualified set Q with weight $\geq \beta W$.
- $\mathcal{A}$ is at most as noisy as C_U for any unqualified set U with weight $\leq \alpha W$.

If $\mathcal{R}$ is less noisy than $\mathcal{A}$, then we can use a good transmission scheme for the wiretap channel $(\mathcal{R}, \mathcal{A})$ to implement our (α, β)-ramp WSS scheme. The parameters of this WSS scheme can be derived from those of the underlying wiretap scheme.

1.1.1 Binary Symmetric Channels

Trying to instantiate this approach with binary symmetric channels, we associate with each weight w_j an error probability p_j and the corresponding Bernoulli random variable

$$\mathcal{D}_j = \begin{cases} 1 & \text{with probability } p_j \\ 0 & \text{with probability } 1 - p_j. \end{cases}$$

One problem to overcome is that for "the most natural mapping" of weights to probabilities, the error probability does not add up linearly: If we set (say) $p_j = w_j/W$, it is not hard to find instances where $\sum_{j\in A} w_j > \sum_{j\in B} w_j$ and yet $\sum_{i\in A} \mathcal{D}_j \bmod 2$ has smaller error probability than $\sum_{j\in B} \mathcal{D}_j \bmod 2$, as the following example shows.

A problematic example

Consider three parties with $w_1 = w_2 = 13$ and $w_3 = 24$, so $W = 50$ and we have $\Pr[\mathcal{D}_1 = 1] = \Pr[\mathcal{D}_2 = 1] = 13/50 = 0.26$ and $\Pr[\mathcal{D}_3 = 1] = 24/50 = 0.48$. Let $A = \{1, 2\}$ and $B = \{3\}$, so the aggregate weight of A is 26, larger that the weight of B which is 24. On the other hand, we have

$$\Pr[\mathcal{D}_1 \oplus \mathcal{D}_2 = 1] = 0.26 + 0.26 - 0.26^2 = 0.4525 < 0.48 = \Pr[\mathcal{D}_3 = 1],$$

so the error rate for A is *lower* that for B.

Clearly, the reason for this example is the cancellation due to the term 0.26^2, namely the fact that the error probabilities do not simply add up. This cancellation effect can be reduced it by scaling down the probabilities, setting $p_j = \gamma w_j/W$ for some $\gamma < 1$ (that may depend on α, β). For example, if we set $p_j = w_j/2W$ rather than $p_j = w_j/W$, then we get $\Pr[\mathcal{D}_1 = 1] = \Pr[\mathcal{D}_2 = 1] = 0.13$ and $\Pr[\mathcal{D}_3 = 1] = 0.24$, and therefore

$$Pr[\mathcal{D}_1 \oplus \mathcal{D}_2 = 1] = 0.13 + 0.13 - 0.13^2 = 0.2431 > 0.24 = \Pr[\mathcal{D}_3 = 1].$$

While this can be made to work, it has a drawback that the error rates of the scaled $\mathcal{R}$ and $\mathcal{A}$ become quite close, of distance only $O(\epsilon^2)$ (where $\epsilon = \beta - \alpha$). This would require the codes of fairly large block-length, making the share-size a large polynomial in $1/\epsilon$. Instead, we describe here a different variant that was pointed out to us by the anonymous reviewers, that improves the dependence on $1/\epsilon$ by using a better mapping from weights to probabilities.

1.1.1.1 The BSC construction

The BSC-based construction that we present in Section 5 gives a weight-w_j shareholder an error variable $\mathcal{D}(w_j)$ which is the sum modulo 2 of w_j IID Bernoulli random variables, all with the same head probability of $\tau < 1/2$ (where τ can depend on α, and β and the total weight $W = \sum_j w_j$). With this definition, it is clear that we get additivity, namely $\mathcal{D}(w_1 + w_2) = \mathcal{D}(w_1) + \mathcal{D}(w_2) \pmod 2$. Therefore, the error sum of a shareholder set with cumulative weight w is exactly $\mathcal{D}(w)$, Also, it is not hard to show that for this construction, the head probability of $\mathcal{D}(w_j)$ is

$$\Pr[\mathcal{D}(w_j) = 1] = \frac{1}{2} \cdot (1 - \exp(-\gamma \cdot w_j/W)),$$

where γ is some constant that depends on τ. As we show in Section 5, optimizing the constant γ in this construction yields a wiretap channel $(\mathcal{R}, \mathcal{A})$ where the capacity gap between $\mathcal{R}$ and $\mathcal{A}$ is $\Theta(\beta - \alpha)$.

1.1.2 Additive Gaussian Channels

Another natural attempt to instantiate our blueprint is using additive white Gaussian noise (AWGN) channels. For these channels, the noise is natively additive: adding Gaussian variables with variance σ_1^2 and σ_2^2 yields another Gaussian with variance $\sigma_1^2 + \sigma_2^2$. The AWGN-noise construction therefore associates each weight, $w \in \mathbb{N}$ with the Normal random variable $\mathcal{N}(0, w/W)$, i.e. zero-mean with variance w/W (stdev $= \sqrt{w/W}$). Due to additivity, the aggregate random variable for a set A is itself a Normal variable,

$$\sum_{j \in A} \mathcal{N}(0, w_j/W) = \mathcal{N}(0, \sum_{j \in A} w_j/W).$$

This implies that whenever S has higher weight than T, the channel C_T has more error than the channel C_S.

For any $\beta > \alpha$, we can therefore construct an (α, β)-ramp WSS scheme from a good transmission scheme for the AWGN wiretap channel $(\mathcal{R}, \mathcal{A})$, where

$$\mathcal{R} : x \mapsto x + \mathcal{N}(0, 1 - \beta) \quad \text{and} \quad \mathcal{A} : x \mapsto x + \mathcal{N}(0, 1 - \alpha).$$

Indeed, since $\beta > \alpha$ then $\mathcal{A}$ is more noisy than $\mathcal{R}$.

One problem to solve when using AWGN channels is that they natively deal with real numbers with infinite precision, whereas we can only use finite precision for our construction. In Appendix A we therefore sketch an approach that uses discrete Gaussians instead. That construction achieves somewhat worse rate than the BSC construction for long secrets, but it can plausibly offer concrete parameter benefits for short secrets.

1.2 Prior Work

Ramp secret-sharing (without weights) was introduced by Blakley and Meadows [9]. A textbook construction for a ramp-scheme with good rate based on standard "packed secret sharing" can be found, e.g., in [12, 11.4.2] (and is described in Section 3.1 below).

Some early work on weighted secret sharing was cast against the backdrop of general access structures. Beimel et al. [3] characterized the weighted (strict) thresholds access structures that admit *ideal* schemes, where the share size is equal to the secret size, proving that only few specific threshold structures can be realized this way.

Beimel and Weinreb [4] showed that any threshold access structure can be realized using shares of size quasiPoly($N \log W$) times the secret size, or even just $\mathsf{poly}(N \log W) \cdot \lambda$ if computational security is enough (λ is the security parameter). They did that by describing monotone circuits that compute every threshold function, and using known monotone-circuits-to-secret-sharing compilers [7, 27].[2] Works such as [16] and [25] propose an explicit scheme for hierarchical threshold structures, those are solving a different (albeit somewhat related) problem than ours.

Another notable prior work is due to Zou et al. [29], they use the Chinese Remainder Theorem to improve some efficiency parameters of weighted multi-secret sharing, but secret sizes are still the same as in the simple scheme based on Shamir sharing.

Also, noisy channels were used in many prior works as a tool for achieving secure computation, starting with [14, 13]. The goals in that line of works are quite different from ours, however, and the connection that we draw between ramp secret-sharing and wiretap channels is new.

Organization

We present some background in Section 2, then define (α, β)-ramp WSS and describe a simple rounding-based protocol for realizing it in Section 3. We formulate our blueprint for WSS schemes from wiretap schemes in Section 4, then describe instantiations of this blueprint from binary symmetric channels in Section 5 and from additive white Gaussian noise channels in Appendix A.

2 Background

Notations. For an integer n, we denote $[n] = \{1, 2, \ldots, n\}$. The ℓ'th entry in a vector $\boldsymbol{e}$ is denoted $\boldsymbol{e}[\ell]$. For two distributions $\mathcal{D}, \mathcal{E}$, we denote by $SD(\mathcal{D}, \mathcal{E})$ their statistical distance. Namely $SD(\mathcal{D}, \mathcal{E}) = \frac{1}{2} \sum_{x \in X} |\mathcal{D}(x) - \mathcal{E}(x)|$, where X is the union of their support.

For a real number x and an integer η, we denote by $\lfloor x \rfloor_{2^{-\eta}}$, $\lceil x \rceil_{2^{-\eta}}$, $\lceil x \rfloor_{2^{-\eta}}$ the rounding of x down, up, or to the nearest number with precision $2^{-\eta}$, respectively. Namely, $\lfloor x \rfloor_{2^{-\eta}}$ is the largest number of the form $i/2^\eta$ (with i an integer) which is not larger than x, and similarly $\lceil x \rceil_{2^{-\eta}}$ is the smallest number of this form which is not smaller than x, and $\lceil x \rfloor_{2^{-\eta}}$ is one of the above which is closer to x (breaking ties arbitrarily). Omitting the $2^{-\eta}$ parameter means rounding to an integer (same as using 2^0).

2.1 Channels and Error Correcting Codes

A communication channel with input set $\mathcal{X}$ and output set $\mathcal{Y}$ is a transform that maps each input symbol $x \in \mathcal{X}$ to a distribution over the output symbols $\mathcal{Y}$. In this work we deal with additive-noise channels where $\mathcal{X} = \mathcal{Y}$ is an additive group, and the channel just adds to its input some random noise, chosen from a known distribution $\mathcal{D}$. Namely, $\mathsf{Ch} : x \mapsto x + \mathcal{D}$.

[2] Those compilers essentially construct a garbled circuit for the threshold function, with the secret being the output label. Hence, they require a very large public information, namely the garbled circuit itself.

We assume a memoryless channel: when sending a sequence of symbols, each symbol is transformed according to the channel Ch independently of the others (and their order is maintained).

An error-correction scheme is meant to facilitate reliable transmission of a sequence of symbols $m \in \mathcal{X}^k$ (for some k) over the channel Ch. For any input length k it consists of a code, defined by an encoding $\mathsf{Enc} : \mathcal{X}^k \to \mathcal{X}^n$ that adds redundancy, mapping the information sequence m to a longer code-word $w \in \mathcal{X}^n$ that will be sent over the channel, and by a matching decoding routine $\mathsf{Dec} : \mathcal{X}^n \to \mathcal{X}^k$ that attempts to recover the original information from the received sequence $\mathsf{Ch}(w)$. An error-correction scheme is a sequence of codes for increasing k.

The rate of a code is k/n, and the channel capacity is the highest possible rate (asymptotically as $k \to \infty$) of any scheme that achieves vanishing decoding error probability. For additive noise channels with noise distribution $\mathcal{D}$, the channel capacity is $1 - h(\mathcal{D})$ where h is the Shannon entropy function. In particular, for any channel Ch and any $\nu > 0$, there exist schemes with rate ν away from capacity (perhaps with inefficient encoding/decoding), in which the decoding error probability is bounded below $2^{-\Theta(n \cdot \nu^2)}$.

In this work we will be concerned with *Binary Symmetric Channels* (BSC, see Section 5) and *Additive White Gaussian Noise* channels (AWGN, see Appendix A). For those channels, there exist schemes with efficient encoding/decoding procedures that approach capacity and achieve vanishing error probability. (The dependence on the slackness parameter ν = capacity-minus-rate, affects the parameters that our blueprint can achieve, and will be discussed in the sequel.)

The "more noisy" relation

We say that a channel Ch' is more noisy than another channel Ch (or Ch is less noisy than Ch'), and denote $\mathsf{Ch} \preceq \mathsf{Ch}'$ or $\mathsf{Ch}' \succeq \mathsf{Ch}$, if there is some transform T such that $\mathsf{Ch}' = T(\mathsf{Ch})$. An example is when Ch' is obtained from Ch by adding more noise, $\mathsf{Ch}'(x) = \mathsf{Ch}(x) + \mathcal{D}$ for some noise distribution $\mathcal{D}$. It is easy to see that the capacity of Ch is at least as high as that of Ch'. Moreover, any error-correction scheme for Ch' also works for Ch.[3]

2.2 Wiretap Channel Transmission Schemes

A wiretap channel is a pair of communication channels $(\mathcal{R}, \mathcal{A})$ with the same input and output sets $\mathcal{X}, \mathcal{Y}$, where $\mathcal{R}$ is a channel from the sender to an intended receiver and $\mathcal{A}$ is the wiretap that goes to the adversary. Given a message m that the sender wants to send to the receiver, the goal is to encode it as $w = \mathsf{Enc}(m)$, so that m can be recovered (whp) from $\mathcal{R}(w)$, but not from $\mathcal{A}(w)$.

Bellare et al. defined in [6] the notion of semantically secure encryption scheme for a wiretap channel (that we prefer to call a *transmission scheme*[4]). The following is essentially their definition of distinguishing security. In our setting, it is sufficient to work with what they call a "seeded" scheme, where encoding and decoding depend on a public random seed.

[3] In theory, to use a decoder for Ch' we may need to apply T to the output of $\mathsf{Ch}(w)$ before we can decode it. In practice, decoders for the high-noise Ch' always work as-is also for the low-noise Ch.

[4] This is a keyless scheme, so it differs from cryptographic encryption.

▶ **Definition 1** (Secure Wiretap Transmission Schemes). *Let $(\mathcal{R}, \mathcal{A})$ be a wiretap channel (for message space $\mathcal{M}$), a secure transmission scheme for it consists of (seed-dependent[5]) encoding and decoding procedures $\mathsf{Enc}_{\mathsf{sd}}, \mathsf{Dec}_{\mathsf{sd}}$ such that*

Correctness. *For all $m \in \mathcal{M}$, $\Pr[\mathsf{Dec}_{\mathsf{sd}}(\mathcal{R}(\mathsf{Enc}_{\mathsf{sd}}(m))) = m] \geq 1 - \mathsf{negl}(|\mathsf{sd}|)$,*

Secrecy. *For all $m, m' \in \mathcal{M}$, $SD\left((\mathsf{sd}, \mathcal{A}(\mathsf{Enc}_{\mathsf{sd}}(m))), (\mathsf{sd}, \mathcal{A}(\mathsf{Enc}_{\mathsf{sd}}(m')))\right) \leq \mathsf{negl}(|\mathsf{sd}|)$,*

where the probability is over the channel randomness as well as the random selection of the seed sd, and negl is some negligible function.

The literature contains many constructions of wiretap channel schemes from error-correcting schemes, some of which we will be using in Section 5 and Appendix A. For the abstract blueprint that we present in Section 4, we need the "obvious" property of all the schemes in the literature, where if they work for one wiretap channel then they also work for all "easier channels." Namely, they are monotone in terms of the more-noisy relation:

▶ **Definition 2** (Monotone Schemes). *A secure transmission scheme $(\mathsf{Enc}, \mathsf{Dec})$ for a channel $(\mathcal{R}, \mathcal{A})$ is* noise-monotone *if it is also a secure transmission scheme for any channel $(\mathcal{R}', \mathcal{A}')$ such that $\mathcal{R}' \preceq \mathcal{R}$ and $\mathcal{A} \preceq \mathcal{A}'$.*

Clearly, the secrecy condition of a transmission scheme is always monotone. The correctness condition is monotone as long as the decoding error of the underlying code is not increased by *reducing* the noise level of the channel (which is true for all coding schemes that we know of).

3 Weighted Secret Sharing

A secret-sharing scheme is a two-phase multi-party protocol for $N + 1$ parties, a dealer and N shareholders. In the dealing phase, the dealer has a secret input $\boldsymbol{s}$, and it outputs a share for each shareholder, and optionally also a public share. In the reconstruction phase, a subset of the shareholders collect all their shares (and the public share if any) and attempt to use them in order to reconstruct the secret.

Each secret-sharing schemes comes with an *access structure*, consisting of a collection of qualified subsets $\Gamma \subset 2^{[N]}$ that should be able to reconstruct the secret, and a collection of *unqualified* subsets $\Psi \subset 2^{[N]}$ that should not be able to learn anything about the secret.[6] Non-perfect realizations of secret sharing come with a security parameter λ that is given as input to all the parties, and we require that the imperfections are negligible in λ.

Below we denote by $\mathsf{View}_S(\boldsymbol{s})$ the view of a subset of the shareholders $S \subset [N]$ when the secret $\boldsymbol{s}$ is shared, consisting of their own shares and the public share (if any). For a qualified set S we also denote by $\mathsf{Recover}(\mathsf{View}_S(\boldsymbol{s}))$ the value that these shareholders compute when trying to recover the secret.

▶ **Definition 3** (Secret Sharing). *A secret-sharing scheme for the access structure (Γ, Ψ) and the space of secrets $\mathcal{S}$, satisfies the following (for some negligible function $\mathsf{negl}(\cdot)$):*

Correctness. *For any qualified subset $S \in \Gamma$ and any secret $\boldsymbol{s} \in \mathcal{S}$,*

$$\Pr[\mathsf{Recover}(\mathsf{View}_S(\boldsymbol{s})) = \boldsymbol{s}] \geq 1 - \mathsf{negl}(\lambda).$$

[5] We use the seed length as the security parameter for this definition.
[6] Sometimes we have $\Psi = \overline{\Gamma}$, but rump schemes have $\Psi \subsetneq \overline{\Gamma}$.

Parameters: Weights $w_1, w_2, \ldots, w_N \in \mathbb{N}$, thresholds $0 < \alpha < \beta < 1$. Let $W := \sum_{i\in[N]} w_i$.

Sharing a secret $\boldsymbol{s} \in \{0,1\}^k$: Let $r = \lceil (\beta - \alpha)W \rceil$ and $k' = \lceil k/r \rceil$.

1. Break the secret into r chunks of length $\leq k'$, let $\vec{s} \in (\{0,1\}^{k'})^r$ be the resulting vector;
2. Share $\vec{s}$ using $(\alpha W, \beta W; r, W)$ multi-secret sharing, party, $j \in [N]$, gets w_j shares.

Reconstructing the secret by a qualified set S:

3. Use multi-secret reconstruction with all revealed shares to recover $\vec{s}$;
4. Concatenate the entries of $\vec{s}$ to get $\boldsymbol{s}$.

Figure 1 A rate-efficient (α, β)-ramp WSS from multi-secret sharing.

Secrecy. *For any unqualified subset $T \in \Psi$ and any two secrets $\boldsymbol{s}, \boldsymbol{s}' \in \mathcal{S}$, the view of T when sharing $\boldsymbol{s}$ is statistically close to the view when sharing $\boldsymbol{s}'$,*

$$SD\left(\mathsf{View}_T(\boldsymbol{s}), \mathsf{View}_T(\boldsymbol{s}')\right) \leq \mathsf{negl}(\lambda).$$

In this work we study a relaxation of threshold weighted secret sharing, (α, β)-ramp weighted secret sharing.

▶ **Definition 4** ((α, β)-ramp weighted secret sharing)**.** *A (α, β)-ramp weighted secret sharing for $0 < \alpha < \beta < 1$, N shareholders, and weights $w_1, \ldots, w_N \in \mathbb{N}$, is a secret-sharing scheme for the access structure*

$$\Gamma = \{S \subseteq [N] : \sum_{i\in S} w_i \geq \beta W\} \text{ and } \Psi = \{T \subseteq [N] : \sum_{i\in T} w_i < \alpha W\},$$

where $W = \sum_{i\in[N]} w_i$.

Below we often use the notation $\epsilon = \beta - \alpha$ when discussing the parameters of ramp WSS schemes.

3.1 Ramp WSS from Multi-Secret Sharing

A $(T_1, T_2; r, N)$ multi-secret sharing scheme shares r secrets (from some domain) among N shareholders, with secrecy when T_1 or less of the shares are revealed and recovery when T_2 or more shares are revealed. A packed Shamir sharing, where multiple secrets are encoded in different evaluation points of a degree-$(T-1)$ polynomial, yields a $(T-r, T; r, N)$ multi-secret sharing scheme over any field of size $\geq N + r$, where each share is only a single field element. Hence it achieves a "rate" of $|\text{secret}|/|\text{share}| = r$.

This can be converted to a ramp WSS scheme using the obvious approach of giving w shares to a weight-w shareholder. This construction is described in Figure 1. To get an (α, β)-ramp WSS we need a multi-secret scheme with $N := W$, $T_1 := \alpha W$, and $T_2 := \beta W$. Using the above construction, we can pack $r = T_2 - T_1 = \epsilon W$ field elements while each underlying share is a single element.

Each shareholder in the resulting WSS scheme holds at most W shares of the underlying scheme, so we get a WSS scheme with share size $\leq W$ element that can handle secrets of size upto ϵW elements. This yields encoding rate of

$$|\text{secret}|/|\text{share}| \geq \frac{\epsilon W}{W} = \epsilon,$$

Parameters: Weights $w_1, w_2, \ldots, w_N \in \mathbb{N}$, thresholds $0 < \alpha < \beta < 1$. Let $W := \sum_{i\in[N]} w_i$.

1. Let $\eta := \left\lceil \log \frac{5N}{\beta-\alpha} \right\rceil$. For all $j \in [N]$, set $w'_j := 2^\eta \cdot \left\lceil \frac{w_j}{W} \right\rceil_{2^{-\eta}}$.
2. Use the Ramp WSS from Figure 1 with the w'_j's and thresholds $\alpha' = \alpha + \frac{\beta-\alpha}{5}$ and $\beta' = \beta - \frac{\beta-\alpha}{4}$.

Figure 2 A rounded (α, β)-ramp weighted secret sharing.

as long as the secret is long enough (i.e., at least ϵW field elements). This scheme is not very useful for short secrets, however, as its efficiency depends on breaking the secret into many chunks. In particular, the size of shares is still W (or more) in the worst case, regardless of how small is the secret. [7]

3.2 A Rounding-Based (α, β)-ramp WSS Protocol

We note that simple rounding can be used to roughly replace the dependence on W in the above scheme by dependent on N/ϵ. Specifically, we use the construction from Figure 1 to implement a modified version of the system, with weights that are rounded to precision of only about $(\beta - \alpha)/N$. Due to rounding errors, the modified version has a smaller gap $\epsilon' < \beta - \alpha$, but the increase can be controlled by setting the precision appropriately. Specifically, with precision of $(\beta - \alpha)/5N$ we can get $\epsilon' \geq \epsilon/2$. This simple protocol is described in Figure 2.

▶ **Lemma 5.** *The protocol outline in Figure 2 is an (α, β)-ramp weighted secret sharing scheme.*

Proof. By our choice of η we get $N/2^\eta \leq (\beta - \alpha)/5$, and for every set $J \subseteq [N]$ we have

$$2^\eta \sum_{j\in J} w_j/W \;\leq\; \sum_{j\in J} w'_j \;<\; |J| + 2^\eta \sum_{j\in J} w_j/W.$$

In particular for $J = [N]$ we have $W' = \sum_{j\in[N]} w'_j \in [2^\eta, 2^\eta + N]$. For any non-qualified set $J \subseteq [N]$ with $\sum_{j\in J} w_j \leq \alpha W$ we therefore have

$$\sum_{j\in J} w'_j/W' \leq \frac{N + 2^\eta \sum_{j\in J} w_j/W}{2^\eta} \leq \frac{N + 2^\eta \cdot \alpha}{2^\eta} \leq (\beta - \alpha)/5 + \alpha = \alpha'.$$

Similarly, for any qualified set $J \subseteq [N]$ with $\sum_{j\in J} w_j \geq \beta W$ we have

$$\sum_{j\in J} w'_j/W' \geq \frac{2^\eta \sum_{j\in J} w_j/W}{N + 2^\eta} \geq \frac{\beta}{1 + (N/2^\eta)} \geq \frac{\beta}{1 + (\beta-\alpha)/5} \overset{(*)}{\geq} \beta - (\beta - \alpha)/4 = \beta'.$$

To see why inequality $(*)$ holds, note that

$$\frac{\beta}{1 + (\beta-\alpha)/5} = \frac{\beta(1 + (\beta-\alpha)/5)}{1 + (\beta-\alpha)/5} - \frac{\beta(\beta-\alpha)/5}{1 + (\beta-\alpha)/5} = \beta - \frac{\beta(\beta-\alpha)}{5 - (\beta-\alpha)} \geq \beta - \frac{\beta-\alpha}{4}. \quad ◀$$

In terms of performance for the protocol of Figure 2, the number of shares a party can receive is upper-bounded by $W' < N + 2^\eta \leq N\left(1 + \frac{10}{\beta-\alpha}\right)$. Hence, the size of shares in this scheme grows with $O(N/\epsilon)$ instead of the total weight W.

[7] In other contexts it is sometimes helpful to use algebraic-geometric codes instead of the Reed-Solomon codes of Shamir sharing, as it enables the use of smaller fields. In our case this does not seem to help, since the inefficiency comes from the number of field elements and not their size.

Sharing a secret $\boldsymbol{s} \in \{0,1\}^k$, with security parameter λ:

1. If the wiretap scheme is seeded, choose a random seed sd of length λ;
2. $\forall j \in [N]$, draw $\boldsymbol{e}_j \leftarrow \mathcal{D}_{w_j}$ and send to party j;
3. Publish sd and $\boldsymbol{g} = Enc_{\mathsf{sd}}(\boldsymbol{s}) + \sum_{j\in[N]} \boldsymbol{e}_j$.

Reconstructing the secret by a qualified set S:

Set $\boldsymbol{g}' = \boldsymbol{g} - \sum_{j\in S} \boldsymbol{e}_j$ and output $\mathsf{Dec}_{\mathsf{sd}}(\boldsymbol{g}')$.

Figure 3 The generic framework for ramp weighted secret sharing from wiretap channels.

4 A Blueprint for WSS from Wiretap Channels

Let $w_1, \ldots, w_N$ be the concrete weights that we want to implement and $0 < \alpha < \beta < 1$ be the parameters that we want to achieve. Denote $W = \sum_{i\in[N]} w_i$. An instance of our blueprint operates in some additive group $\mathcal{X}$, and consists of two components:

- A mapping from weights $w \in \mathbb{N}$ to noise distributions $\mathcal{D}_w$ over $\mathcal{X}$.
- A (seeded) noise-monotone secure transmission scheme $(\mathsf{Enc}, \mathsf{Dec})$ for a wiretap channel $(\mathcal{R}, \mathcal{A})$ (cf. Definition 1), such that:
  - For any qualified subset $S \subseteq [N]$ with $\sum_{i\in S} w_i \geq \beta W$, the channel $\mathcal{R}$ is more noisy than adding all the noise distributions *outside* S. Namely, $C_S \preceq \mathcal{R}$ where $C_S : x \mapsto x + \sum_{i\notin S} \mathcal{D}_{w_i}$.
  - For any unqualified subset $T \subseteq [N]$ with $\sum_{i\in S} w_i \leq \alpha W$, the channel $\mathcal{A}$ is less noisy than adding all the noise distributions *outside* T. Namely, $C_T \succeq \mathcal{A}$ where $C_T : x \mapsto x + \sum_{i\notin T} \mathcal{D}_{w_i}$.

Given these components, our WSS scheme is described in Figure 3.

▶ **Lemma 6.** *If* $(\mathsf{Enc}, \mathsf{Dec})$ *is a noise-monotone secure transmission scheme for a wiretap channel* $(\mathcal{R}, \mathcal{A})$*, as per Definitions 1 and 2, that satisfy the conditions above. Then the scheme from Figure 3 is a secure* (α, β)*-ramp weighted secret-sharing scheme.*

Proof. This holds more or less by definition. Consider an arbitrary qualified set S and an arbitrary unqualified set T. Then by construction we have $C_S \preceq \mathcal{R}$ and $\mathcal{A} \preceq C_T$, and since $(\mathsf{Enc}, \mathsf{Dec})$ is noise-monotone then it is also a secure transmission scheme for the wiretap channel (C_S, C_T). This means on one hand that for the qualified set S, seeing $y = C_S(\mathsf{Enc}(\boldsymbol{s}))$, we have $\mathsf{Dec}(y) = \boldsymbol{s}$ with all but negligible probability. On the other hand, the unqualified set T, seeing only $C_T(\mathsf{Enc}(\boldsymbol{s}))$, cannot distinguish it from $C_T(\mathsf{Enc}(\boldsymbol{s}'))$ except with a negligible advantage. ◀

The public share

Our solutions, as well as some solutions from the literature (such as [4]), use a public share, which is known to everyone, in addition to the individual shares of the shareholders. Clearly, it is possible to eliminate the public share by adding it to each individual share, and in our case this will at most double the share size of parties. In some solutions in the literature, however, the public share is much larger than the individual shares. Here we chose to account for the public share separately and only count it once (rather than once per shareholder).

5 Constructions from Binary Symmetric Wiretap Channels

5.1 Background

5.1.1 Binary Symmetric Channels

A binary symmetric channel (BSC) is used for sending bits. It is associated with a "crossover probability" $p \leq 1/2$, which is the probability that the received bit differs from the one that was sent. Namely, we have a Bernoulli error variable B_p with $\Pr[B_p = 1] = p$ and $\Pr[B_p = 0] = 1 - p$, and the channel is defined on message space $\{0,1\}$ as $BSC_p : x \mapsto x + B_p \bmod 2$.

The capacity of BSC_p is $1 - h(p)$, where h is the binary entropy function. If we are not concerned with efficient decoding, then random linear codes (with ML/MAP decoding) have rates that approach the channel capacity with exponentially small error probability. Capacity-approaching constructions with efficient decoding are known using concatenated codes [17] or polar codes [1, 18], with somewhat weaker bounds on the decoding error. For example, [2, 19] show that the error probability for block-length n and rate $1 - h(p) - \nu$ is at most $\exp(-\Theta(\sqrt{n}))$, where the constant in the exponent depends on p and ν. Later results feature stronger bounds in terms of the block-length n with polynomial dependence on the slackness ν. In particular, we have

▶ **Lemma 7** (Corollary of [11], Thm 17). *For any $p < 1/2$, $\nu < 1 - h(p)$, and $\mu < 1$, there exists a code for BSC_p with rate $1 - h(p) - \nu$, block length $n = \mathsf{poly}_\mu(1/\nu)$ (for some polynomial that depends on μ), error probability $\exp(-n^\mu)$, and decoding complexity $O(n \log n)$.*

It is known that the polynomial dependence on $1/\nu$ is quadratic for any discrete memoryless channel, while for some efficient constructions there is evidence that $\mathrm{poly}_{1/2}(x) \leq x^{4.6}$ [23, 28].

5.1.2 Wiretap Schemes for Binary Symmetric Channels

Bellare et al. also described in [6] a construction called ItE (Invert-then-Encode) for discrete wiretap channels, building on error-correction. The construction realizes Definition 1 for the channels $(\mathcal{R}, \mathcal{A})$, using a code with low decoding error probability for $\mathcal{R}$, at a rate noticeably larger than the capacity of $\mathcal{A}$. (In particular, if the code rate approaches the capacity of $\mathcal{R}$ then this construction approaches the secrecy capacity of the wiretap channel.)

The ItE construction has integer parameters $b < k < n$ (with values as set later in this section). Identifying $\{0,1\}^k$ with the finite field $\mathbb{F}_{2^k}$, this is a seeded construction with seed space the multiplicative group $\mathbb{F}_{2^k} \setminus \{0^k\}$ and message space $\{0,1\}^b$. In addition, it uses error-correction encoding $\mathsf{Enc} : \{0,1\}^k \to \{0,1\}^n$ and the corresponding decoding $\mathsf{Dec} : \{0,1\}^n \to \{0,1\}^k$. The encoding and decoding routines of the ItE construction (denoted $\mathsf{Enc}'_{\mathsf{sd}}, \mathsf{Dec}'_{\mathsf{sd}}$) are described in Figure 4. The following is a re-phrasing of Lemma 5.3 and Lemmas 5.5-5.6 from [5]:

▶ **Lemma 8** ([5], Lemma 5.3). *If $(\mathsf{Enc}, \mathsf{Dec})$ is an error-correction scheme with decoding-error probability at most ϵ for the channel $\mathcal{R}$, then the ItE scheme $(\mathsf{Enc}_{\mathsf{sd}}, \mathsf{Dec}_{\mathsf{sd}})$ from Figure 4 is correct for $(\mathcal{R}, \mathcal{A})$ with correctness holding with probability $\geq 1 - \epsilon$.* ◀

▶ **Lemma 9** (Corollary of [5], Lemmas 5.5-5.6). *Let $\mathcal{A}$ be a symmetric memoryless channel with capacity $c(\mathcal{A})$. Assume that $\frac{k}{n}$ (the rate of Enc) is larger than $c(\mathcal{A})$, denote the slackness by $\rho = \frac{k}{n} - c(\mathcal{A})$, and let λ be the security parameter. Then for any $0 < \delta < \rho - \frac{2\lambda}{n}$, setting $b := \lfloor n(\rho - \delta) - 2\lambda - 2 \rfloor$ in the ItE construction yields a wiretap transmission scheme with secrecy upto statistical distance $4 \cdot 2^{-\delta^2 n/11} + 2 \cdot 2^{-\lambda}$.*

Encoding: $\mathsf{Enc}'_{\mathsf{sd}}(M \in \{0,1\}^b)$ with seed $\mathsf{sd} \in \mathbb{F}_{2^k} \setminus \{0^k\}$:
1. Choose a random $R \leftarrow \{0,1\}^{k-b}$, let $Y = (M|R) \in \mathbb{F}_{2^k}$ be the concatenation;
2. Set $X := Y/\mathsf{sd} \in \mathbb{F}_{2^k}$;
3. Send the message $W := \mathsf{Enc}(X) \in \{0,1\}^n$.

Decoding: $\mathsf{Dec}'_{\mathsf{sd}}(W' \in \{0,1\}^n)$:
1. Use error-correction to get $X' = \mathsf{Dec}(W')$;
2. Compute $Y' := X' \cdot \mathsf{sd}$;
3. Output M', the first b bits of Y'.

Figure 4 The ItE construction from [6].

Plugging the coding parameter from above, we get the following instantiation:

▶ **Corollary 10.** *For a binary symmetric wiretap channel (BSC_{p_R}, BSC_{p_A}) with $0 \le p_R < p_A < 1/2$, denote $\xi := h(p_A) - h(p_R)$. There exists an instance of the ItE scheme $(\mathsf{Enc}_{\mathsf{sd}}, \mathsf{Dec}_{\mathsf{sd}})$ with security parameter λ and*

- *Encoding size $n = \max\left(\mathsf{poly}_{\frac{1}{2}}(\frac{4}{\xi}), \lambda^2, \frac{44\lambda}{\xi^2}\right)$;* [8]
- *Seed space $\mathbb{F}_{2^k} \setminus \{0^k\}$ with $k = (1 - h(p_A) + \frac{3\xi}{4})n = (1 - h(p_R) - \frac{\xi}{4})n$; and*
- *Message space $\{0,1\}^b$, $b \ge (\frac{\xi}{4} - \frac{2}{\lambda})n - 2$;*

such that

- *For all $m \in \{0,1\}^b$, $\Pr[\mathsf{Dec}_{\mathsf{sd}}(BSC_{p_R}(\mathsf{Enc}_{\mathsf{sd}}(m))) = m] \ge 1 - 2^{-\lambda}$;*
- *For all $m, m' \in \{0,1\}^b$,*

$$SD\big((\mathsf{sd}, BSC_{p_A}(\mathsf{Enc}_{\mathsf{sd}}(m))),\ (\mathsf{sd}, BSC_{p_A}(\mathsf{Enc}_{\mathsf{sd}}(m')))\big) \le 6 \cdot 2^{-\lambda}.$$

Proof. Recall that k determines both the seed space of the ItE construction and the input space for the underlying error-correcting code. The rate of the underlying code is therefore $k/n = 1 - h(p_R) - \frac{\xi}{4}$, and by Lemma 7 we can find such codes as soon as the encoding-length exceeds $\mathsf{poly}_{1/2}(4/\xi)$, with decoding error probability at most $\exp(-n^{1/2}) < 2^{-\sqrt{n}}$. If $n \ge \lambda^2$ then this is bounded below $2^{-\lambda}$, and due to Lemma 8 the same holds for correctness of the ItE construction.

For the secrecy part, we have rate $k/n = 1 - h(p_A) + \frac{3\xi}{4}$, and we use $\delta = \frac{\xi}{2}$ in Lemma 9. This yields $b = \frac{\xi}{4}n - 2\lambda - 2$, and since $n \ge \lambda^2$ then $b \ge (\frac{\xi}{4} - \frac{2}{\lambda})n - 2$. If we also have $n \ge \frac{44\lambda}{\xi^2}$, then $\delta^2 n/11 \ge (\xi/2)^2 \cdot (44\lambda/\xi^2)/11 = \lambda$, and therefore the statistical distance is bounded by

$$4 \cdot 2^{-\delta^2 n/11} + 2 \cdot 2^{-\lambda} \le 4 \cdot 2^{-\lambda} + 2 \cdot 2^{-\lambda} = 6 \cdot 2^{-\lambda}.$$ ◀

5.1.2.1 Remark

Different from most works in the literature, in the setting above we do not aim at achieving the secrecy capacity in the limit. Rather, we try to maintain a small encoding size n relative not just to the message size b, but also to the security parameter λ and the parameters p_R, p_A. [9]

[8] $\mathsf{poly}_{\frac{1}{2}}$ is the polynomial from Lemma 7 for $\mu = \frac{1}{2}$.

[9] In particular, we opted for losing a constant factor in the ratio b/n in return for better dependency on λ and ξ.

Parameters:
Weights $w_1, w_2, \ldots, w_N \in \mathbb{N}$, thresholds $0 < \alpha < \beta < 1$, security parameter λ.
- Let $W := \sum_{i\in[N]} w_i$ and $\gamma := \frac{0.4}{1-\alpha}$.
- Denote $g(x) := \frac{1-\exp(-x)}{2}$, and let $p_R := g(\gamma(1-\beta))$ and $p_A := g(\gamma(1-\alpha))$.
- Let $(\mathsf{Enc}, \mathsf{Dec})$ (with parameters n, k, b) be as in the ItE construction from Corollary 10 for the wiretap channel (BSC_{p_R}, BSC_{p_A}).

Sharing a secret $\boldsymbol{s} \in \{0,1\}^k$:
1. $\forall j \in [N]$, set $p_j := g(\frac{\gamma \cdot w_j}{W})$, draw $\boldsymbol{e}_j \leftarrow (\mathsf{Bernoulli}_{p_j})^n$ and send to party j;
2. Draw a random $\mathsf{sd} \in \mathbb{F}_{2^k} \setminus \{0^k\}$, publish sd and $\boldsymbol{g} := \mathsf{Enc}_{\mathsf{sd}}(\boldsymbol{s}) + \sum_{j\in[N]} \boldsymbol{e}_j \bmod 2$.

Reconstructing the secret by a qualified set S:
Set $\boldsymbol{g}' = \boldsymbol{g} + \sum_{j\in S} \boldsymbol{e}_j \bmod 2$ and output $\mathsf{Dec}_{\mathsf{sd}}(\boldsymbol{g}')$.

Figure 5 Weighted secret sharing from symmetric binary wiretap channels.

5.2 Our Construction

In Figure 5 we show how to use the ItE instance from Corollary 10 to get an (α, β)-ramp WSS for given weights $w_1, w_2, \ldots, w_N$ and thresholds $0 < \alpha < \beta < 1$.

Clearly, this construction is an instance of the blueprint from Figure 3, instantiated over the additive group $\mathbb{F}_{2^k}$, using the noise distributions $D_w = \mathsf{Bernoulli}_{g(\gamma\cdot w/W)}$ and the ItE construction from Corollary 10 for the wiretap channel (CSB_{p_R}, CSB_{p_A}). It is also clear that the ItE construction is noise-monotone (since the underlying error-correction codes are).

The only thing left to prove in order to use Lemma 6, is that for any qualified S and unqualified T, the corresponding channels satisfy $C_S \preceq BSC_{p_R}$ and $C_T \succeq BSC_{p_A}$. To that end, we use the following technical lemma:

▶ **Lemma 11.** *Let $\mathcal{B}_1, \ldots, \mathcal{B}_t$ be independent Bernoulli random variables with $\Pr[\mathcal{B}_j = 1] = \frac{1-\exp(-u_j)}{2}$, and denote $\mathcal{S} := \sum_{j\in[t]} \mathcal{B}_j \bmod 2$ then $\mathcal{S}$ is a Bernoulli random variable with:*

$$\Pr[\mathcal{S} = 1] = \frac{1 - \exp(-\sum_{j\in[t]} u_j)}{2}.$$

Proof. We prove the lemma by induction on t. The base case, where $t = 1$, is trivial. For $t > 1$, we have that

$$\begin{aligned}
\Pr[\mathcal{S} = 1] &= \Pr\left[\sum_{j\in[t-1]} \mathcal{B}_j \bmod 2 = 1 \;\&\; \mathcal{B}_t = 0\right] + \Pr\left[\sum_{j\in[t-1]} \mathcal{B}_j \bmod 2 = 0 \;\&\; \mathcal{B}_t = 1\right] \\
&= \frac{1 - \exp\left(\sum_{j\in[t-1]} u_j\right)}{2} \cdot \frac{1 + \exp(x_t)}{2} + \frac{1 + \exp\left(\sum_{j\in[t-1]} u_j\right)}{2} \cdot \frac{1 - \exp(x_t)}{2} \\
&\qquad \text{(by the inductive hypothesis)} \\
&= \frac{1 - \exp\left(-\sum_{j\in[t]} u_j\right)}{2}.
\end{aligned}$$

Thus, the inductive step holds. ◀

We can now complete the proof that the ItE-based construction above satisfies all the conditions of Lemma 6.

▶ **Corollary 12.** *With the parameters as set in Figure 5 and a straightforward application of Lemma 11:*

(A) *For every subset $S \subseteq [N]$ with $\sum_{j\in S} w_j \geq \beta W$, we have $C_S \preceq BSC_{p_R}$ where $C_S : x \mapsto x + \sum_{j\notin S} \mathsf{Bernoulli}_{p_j}$.*

(B) *For every subset $T \subseteq [N]$ with $\sum_{j\in S} w_j \leq \alpha W$, we have $C_T \succeq BSC_{p_A}$ where $C_T : x \mapsto x + \sum_{j\notin T} \mathsf{Bernoulli}_{p_j}$.*

An Alternative Presentation

An alternative way of describing the scheme from Figure 5, is that the noise component for party j with weight w_j is set as the sum (modulo 2) of w_j IID random vectors, all of the form $\left(\mathsf{Bernoulli}_{g(\gamma/W)}\right)^n$. By Lemma 11, this noise vector indeed has the form $\left(\mathsf{Bernoulli}_{p_j}\right)^n$, where $p_j = g(\gamma \cdot w_j/W)$.

5.3 Performance Characteristics of This Construction

Let $\epsilon = \beta - \alpha > 0$ and $h : p \mapsto -p\log p - (1-p)\log(1-p)$ the binary entropy function (recall that log is in base 2). To get the best parameters from Corollary 10, we want to set the parameter γ so as to maximize $\xi := h(p_A) - h(p_R)$, where:

$$p_A = g(\gamma(1-\alpha)), \qquad p_R = g(\gamma(1-\beta)), \qquad \text{recalling that } g : x \mapsto \frac{1-\exp(-x)}{2}.$$

Denote the function $f : x \mapsto h(g(x))$. The mean value theorem implies that:

$$\begin{aligned} \xi &= f(\gamma(1-\alpha)) - f(\gamma(1-\beta)) && (1)\\ &\geq (\gamma(1-\alpha) - \gamma(1-\beta)) \cdot \inf_{\gamma(1-\beta)<x<\gamma(1-\alpha)} f'(x) \\ &= \epsilon\cdot\gamma\cdot \inf_{\gamma(1-\beta)<x<\gamma(1-\alpha)} f'(x). && (2)\end{aligned}$$

where f' is the derivative of f.

Let us now compute f'. We have $h'(p) = \log(1/p-1)$ and $g'(x) = \exp(-x)/2$. Thus:

$$f'(x) = \frac{1}{2}\exp(-x)\cdot\log\left(\frac{2}{1-\exp(-x)} - 1\right).$$

We remark that f' is decreasing, because $\exp(-x)$ is decreasing and $\log\left(\frac{2}{1-\exp(-x)} - 1\right)$ is decreasing. Therefore Equation (2) implies:

$$\xi \geq \epsilon\cdot\gamma\cdot f'(\gamma(1-\alpha)) = \frac{\epsilon}{1-\alpha}\cdot\gamma'\cdot f'(\gamma') \geq \epsilon\cdot\gamma'\cdot f'(\gamma')$$

where $\gamma' := \gamma(1-\alpha)$.

To maximize our lower bound of ξ, we just need to maximize $\gamma'\cdot f'(\gamma')$. The optimal γ' is about 0.4. In particular, setting $\gamma' = 0.4$ gives $\gamma'\cdot f'(\gamma') \geq 0.31$. Thus we can set $\gamma = \frac{0.4}{1-\alpha}$, which implies

$$\xi = h(p_A) - h(p_R) \geq 0.31\cdot\epsilon. \tag{3}$$

By Corollary 10, there is a transmission scheme $(\mathsf{Enc}_{\mathsf{sd}}, \mathsf{Dec}_{\mathsf{sd}})$ for the wiretap channel (BSC_{p_R}, BSC_{p_A}) with correctness/secrecy upto $O(2^{-\lambda})$ and parameters

- **Encoding length:** $n \leq \max\left(\mathsf{poly}_{\frac{1}{2}}(\frac{13}{\epsilon}), \lambda^2, \frac{458\lambda}{\epsilon^2}\right)$;
- **Message length:** $b \geq (\frac{\epsilon}{13} - \frac{2}{\lambda})n - 2$;
- **Seed length:** $k = \left\lceil(1 - h(p_A) + \frac{\epsilon}{4})n\right\rceil$.

Recall that for this scheme, we have secrets of length b, each shareholder gets a share of length n, and the public share is of size $n + k$. Note also that n, k, b depend only the thresholds α, β and not on the weights themselves. Thus, we get a scheme where the share sizes are independent of the weights, and the rate is $b/(2n + k) = \Theta(\epsilon)$.

When the gap $\epsilon = \beta - \alpha$ is a constant, we can obtain this constant rate already for constant-size secret. As the gap gets smaller, the share sizes grow as a polynomial in $1/\epsilon$, so we can only get $\Theta(\epsilon)$ rate for longer secrets. For example, assuming that constructions such as [28] yield good binary codes with efficient decoding and $\mathsf{poly}_{\frac{1}{2}}(x) = x^{4.6}$, we get $n \approx (13/\epsilon)^{4.6}$.

6 Conclusions

In this work, we study a ramp weighted secret sharing, with a gap between the qualified and unqualified sets, and described two different types of constructions, one based on rounding and the other using a new connection to wiretap schemes. Both types have share size independent of total weight, and dependent only the gap between qualified and unqualified sets. We described in detail a construction based on binary symmetric wiretap channels, and sketched one based on AWGN. It may be interesting to explore other channels as well, to see if any of then can offer concrete parameter improvements.

References

1 Erdal Arikan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory*, 55(7):3051–3073, 2009. doi:10.1109/TIT.2009.2021379.

2 Erdal Arikan and Emre Telatar. On the rate of channel polarization. In *2009 IEEE International Symposium on Information Theory*, pages 1493–1495, 2009. doi:10.1109/ISIT.2009.5205856.

3 Amos Beimel, Tamir Tassa, and Enav Weinreb. Characterizing ideal weighted threshold secret sharing. In *Theory of Cryptography Conference*, pages 600–619. Springer, 2005.

4 Amos Beimel and Enav Weinreb. Monotone circuits for monotone weighted threshold functions. *Information Processing Letters*, 97(1):12–18, 2006. doi:10.1016/j.ipl.2005.09.008.

5 Mihir Bellare and Stefano Tessaro. Polynomial-time, semantically-secure encryption achieving the secrecy capacity. *IACR Cryptology ePrint Archive*, page 22, 2012. URL: http://eprint.iacr.org/2012/022.

6 Mihir Bellare, Stefano Tessaro, and Alexander Vardy. Semantic security for the wiretap channel. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2012. Also available from https://arxiv.org/abs/1201.2205. doi:10.1007/978-3-642-32009-5_18.

7 Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer, 1988. doi:10.1007/0-387-34799-2_3.

8 Fabrice Benhamouda, Shai Halevi, and Lev Stambler. Weighted secret sharing from wiretap channels. *IACR Cryptol. ePrint Arch.*, page 1578, 2022. URL: https://eprint.iacr.org/2022/1578.

9 G. R. Blakley and Catherine A. Meadows. Security of ramp schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 242–268. Springer, 1984. doi:10.1007/3-540-39568-7_20.

10 George R. Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318, 1979. doi:10.1109/MARK.1979.8817296.

11 Jaroslaw Blasiok, Venkatesan Guruswami, and Madhu Sudan. Polar codes with exponentially small error at finite block length. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018*, volume 116 of *LIPIcs*, pages 34:1–34:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.34.

12 Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015. URL: http://www.cambridge.org/de/academic/subjects/computer-science/cryptography-cryptology-and-coding/secure-multiparty-computation-and-secret-sharing?format=HB&isbn=9781107043053.

13 Claude Crépeau. Efficient cryptographic protocols based on noisy channels. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 306–317. Springer, 1997. doi:10.1007/3-540-69053-0_21.

14 Claude Crépeau and Joe Kilian. Achieving oblivious transfer using weakened security assumptions (extended abstract). In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 42–52. IEEE Computer Society, 1988. doi:10.1109/SFCS.1988.21920.

15 U. Erez and R. Zamir. Achieving 1/2 log(1+SNR) on the AWGN channel with lattice encoding and decoding. *IEEE Transactions on Information Theory*, 50(10):2293–2314, 2004. doi:10.1109/TIT.2004.834787.

16 Oriol Farras and Carles Padró. Ideal hierarchical secret sharing schemes. *IEEE transactions on information theory*, 58(5):3273–3286, 2012.

17 Venkatesan Guruswami and Atri Rudra. Concatenated codes can achieve list-decoding capacity. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 258–267, USA, 2008. Society for Industrial and Applied Mathematics.

18 Venkatesan Guruswami and Patrick Xia. Polar codes: Speed of polarization and polynomial gap to capacity. *IEEE Trans. Inf. Theory*, 61(1):3–16, 2015. doi:10.1109/TIT.2014.2371819.

19 S. Hamed Hassani, Ryuhei Mori, Toshiyuki Tanaka, and Rüdiger L. Urbanke. Rate-dependent analysis of the asymptotic behavior of channel polarization. *IEEE Transactions on Information Theory*, 59(4):2267–2276, 2013. doi:10.1109/TIT.2012.2228295.

20 Ling Liu, Yanfei Yan, and Cong Ling. Achieving secrecy capacity of the gaussian wiretap channel with polar lattices. *IEEE Transactions on Information Theory*, 64(3):1647–1665, 2018. Also available at https://arxiv.org/abs/1503.02313. doi:10.1109/TIT.2018.2794327.

21 Ling Liu, Yanfei Yan, Cong Ling, and Xiaofu Wu. Construction of capacity-achieving lattice codes: Polar lattices. *IEEE Transactions on Communications*, 67(2):915–928, 2019. Also available at https://arxiv.org/abs/1411.0187. doi:10.1109/TCOMM.2018.2876113.

22 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.

23 Marco Mondelli, S. Hamed Hassani, and Rüdiger L. Urbanke. Unified scaling of polar codes: Error exponent, scaling exponent, moderate deviations, and error floors. *IEEE Transactions on Information Theory*, 62(12):6698–6712, 2016. doi:10.1109/TIT.2016.2616117.

24 Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

25 Tamir Tassa. Hierarchical threshold secret sharing. *Journal of cryptology*, 20(2):237–264, 2007.

26 Himanshu Tyagi and Alexander Vardy. Explicit capacity-achieving coding scheme for the gaussian wiretap channel. In *2014 IEEE International Symposium on Information Theory*, pages 956–960, 2014. See also https://arxiv.org/abs/1412.4958. doi:10.1109/ISIT.2014.6874974.

27 Vinod Vaikuntanathan, Arvind Narayanan, K. Srinathan, C. Pandu Rangan, and Kwangjo Kim. On the power of computational secret sharing. In Thomas Johansson and Subhamoy Maitra, editors, *Progress in Cryptology - INDOCRYPT 2003*, volume 2904 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 2003. doi:10.1007/978-3-540-24582-7_12.

28 Hsin-Po Wang, Ting-Chun Lin, Alexander Vardy, and Ryan Gabrys. Sub-4.7 scaling exponent of polar codes. arXiv 2204.11683, 2022. URL: https://arxiv.org/abs/2204.11683.

29 Xukai Zou, Fabio Maino, Elisa Bertino, Yan Sui, Kai Wang, and Feng Li. A new approach to weighted multi-secret sharing. In *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–6. IEEE, 2011.

A Constructions from AWGN Wiretap Channels

Below we sketch an AWGN-based construction. It is plausible that such construction can provide somewhat better performance than BSC-based construction, since AWGN-code use "soft decoding" vs. the "hard decoding" that's inherent in BSC code. We do not know if the existing codes actually realize such improvement, however. Moreover, the construction below features logarithmic dependence on the number of parties.

A.1 Background

A.1.1 Additive White Gaussian Noise Channels

Additive white Gaussian noise channels (AWGN) communicate real numbers rather than bits. For each symbol $x \in \mathbb{R}$ transmitted over the channel, the received symbol is $y = x + e$ (addition over the reals), where e is a zero-mean Normal random variable. The variance σ^2 of e is the noise level of the channel.

Symbols transmitted over the channel are chosen subject to some power constraint, specifically their (expected) square is bounded by the *power parameter* P of the sender. The quality of the channel is determined by the ratio between the power and the noise, called the signal-to-noise ratio: $SNR = P/\sigma^2$. [10] Below it will be convenient to fix the power to $P = 1$ and set the variance accordingly. We denote the AWGN channel with variance σ^2 (and power $P = 1$) by $AWGN_{\sigma^2} : x \mapsto x + \mathcal{N}(0, \sigma^2)$. The capacity of this channel (denoted $c(\sigma)$ below) is

$$c(\sigma) := \mathsf{capacity}(AWGN_{\sigma^2}) = \ln\left(1 + \frac{1}{\sigma^2}\right).$$

(The general formula is $\ln\left(1 + \frac{P}{\sigma^2}\right)$ but we are fixing $P = 1$.) There are known constructions of error-correcting codes with efficient decoding for the AWGN that approach capacity, see for example [15, 21]. While AWGN codes can perhaps achieve somewhat better performance than BSC codes (since they use "soft decoding") this improvement has little effect on their asymptotic behavior. In particular, for slackness parameter $\nu < c(\sigma)$, there exist codes for $AWGN_{\sigma^2}$ with rate $c(\sigma) - \nu$, block length $n = \mathsf{poly}(1/\nu)$ (for some polynomial), error probability $\exp(-\sqrt{n})$, and decoding complexity polynomial in n.

[10] Clearly, scaling P and σ^2 by the same factor has no effect on the channel quality.

A.1.2 AWGN Wiretap Channels

Tyagi and Vardy described in [26] a modular construction (in the same spirit as [6]) that combines AWGN codes with randomness extractors. If the underlying code approaches the receiver channel capacity, then the Tyagi-Vardy scheme can be made to approach the secrecy capacity of the wiretap channel. A different approach for a secrecy-capacity-approaching schemes was provided by Liu et al. [20].

These AWGN constructions may be practically more efficient than their BSC counterparts, but as far as we know the improvement has little effect on their asymptotic behavior. Namely, for an AWGN wiretap channel $(AWGN_{\sigma_r^2}, AWGN_{\sigma_a^2})$ with $0 \leq \sigma_r < \sigma_a$, denote $\xi := c(\sigma_r) - c(\sigma_a)$. Then the constructions in [26, 20] provide seeded wiretap transmission schemes $(\mathsf{Enc}_{\mathsf{sd}}, \mathsf{Dec}_{\mathsf{sd}})$ with security parameter λ and

- Encoding size $n = \max\left(\mathsf{poly}(\frac{1}{\xi}), \lambda^2, O(\frac{\lambda}{\xi^2})\right)$;
- Seed size $k = (c(\sigma_a) + \Theta(\xi))n = (c(\sigma_r) - \Theta(\xi))n$; and
- Message space $\{0,1\}^b$, $b \geq (\Theta(\xi) - \frac{2}{\lambda})n$;

such that

- For all $m \in \{0,1\}^b$, $\Pr[\mathsf{Dec}_{\mathsf{sd}}(AWGN_{\sigma_r^2}(\mathsf{Enc}_{\mathsf{sd}}(m))) = m] \geq 1 - 2^{-\lambda}$;
- For all $m, m' \in \{0,1\}^b$,

$$SD\big((\mathsf{sd}, AWGN_{\sigma_a^2}(\mathsf{Enc}_{\mathsf{sd}}(m))),\ (\mathsf{sd}, AWGN_{\sigma_a^2}(\mathsf{Enc}_{\mathsf{sd}}(m')))\big) \leq 2^{-\lambda}.$$

A.1.3 Using Discrete Gaussian Distributions

Continuous Gaussian distributions cannot be used directly in our blueprint, since they require working with real numbers with infinite precision. We therefore need to "quantize" these numbers in some form. The two natural approaches for doing that are either to round them to some finite precision, or to switch working with discrete Gaussian distributions [22]. Either way, the share sizes will grow linearly with the precision that we use, so it is crucial to analyze the precision needed for error-correction. The effect of rounding on error correction is harder to gauge, especially since the magnitude of the rounding errors grows with $\sqrt{n}$ (where n is the code dimension). Below we therefore sketch an approach that uses discrete Gaussians.

Recall that a discrete Gaussian distribution over a point lattice $\Lambda \subset \mathbb{R}^n$ is a probability distribution over Λ where each point $\vec{x} \in \Lambda$ is assigned probability mass proportional to the Gaussian probability density function. Namely, for a parameter $s \in \mathbb{R}$, denote $\rho_s(\vec{x}) := \exp(-\pi\|\vec{x}/s\|^2)$ and $\rho_s(\Lambda) = \sum_{\vec{x}\in\Lambda} \rho_s(\vec{x})$. Then the discrete Gaussian distribution over Λ with parameter $s \in \mathbb{R}$ (centered at the origin), denoted $D_{\Lambda,s}$, assigns to each $\vec{x} \in \Lambda$ the probability mass $D_{\Lambda,s}(\vec{x}) := \rho_s(\vec{x})/\rho_s(\Lambda)$.

An extensive line of work, starting with Micciancio and Regev [22], established that Discrete Gaussians inherit most of the statistical properties of their continuous counterparts, as long as the parameter s is "sufficiently larger that the precision of Λ". Specifically, [22] defined the *smoothing parameter* of Λ (relative to some target deviation ϵ), that captures how large the parameter s needs to be for $D_{\Lambda,s}$ to resemble the continuous distribution upto $O(\epsilon)$. Here we only use the fact that for the integer lattice $\mathbb{Z}^n$ and any $\epsilon, \gamma \in \mathbb{R}$, we have $\eta_\epsilon(\gamma \cdot \mathbb{Z}^n) \leq \gamma \cdot \sqrt{\ln(2n(1+1/\epsilon))/\pi}$ (cf. [22, Lemma 3.3]). Specifically, setting $\epsilon = 2^{-\lambda}$ we get

$$\eta_{2^{-\lambda}}(\gamma \cdot \mathbb{Z}^n) \leq \gamma \cdot \sqrt{\frac{\ln(2n(1+2^\lambda))}{\pi}} < \gamma \cdot \sqrt{\ln n + \lambda}.$$

While we could not find in the literature any treatment of error correction as applied to discrete Gaussians, we take the extensive literature on the statistical properties as evidence that the error-correction techniques for continuous Gaussians should still work. Specifically, for discrete Gaussians over $\gamma \cdot \Lambda^n$, the secrecy/correctness error should not increase by more than $O(\epsilon)$, provided that we always use parameter $s \geq \eta_\epsilon(\gamma \cdot \mathbb{Z}^n)$.

A.2 A Discrete AWGN Construction

Instantiating the approach above with precision γ seem to require that *all* the distributions that we use will have parameter of at least the smoothness factor. Since in our construction we give a party with weight w an error component with parameter $s_w \sim \sqrt{w}$, then we need to use a small enough γ so that even for the smallest non-zero weight (which could be $w = 1$) already has a large enough parameter $s_w \geq \eta_{2^{-\lambda-\log N}}(\gamma \cdot \mathbb{Z}^n)$. (The $\log N$ factor comes due to the fact that we have N such distributions, one per party.) That is, we roughly need $\gamma \approx 1/\sqrt{\ln n + \lambda + \log N}$.

On the other hand, for the largest weights and (which could be as large as $\Omega(W)$), and certainly for the public share, we need to use numbers of size at least $\sqrt{W}$. Hence, each entry in our code would require $O(\log(\sqrt{W}/\gamma)) = O(\log(W) + \log\lambda + \log\log n + \log\log N)$ bits to specify. We could use the rounding technique from Figure 2 to remove the dependence on W, replacing the $\log W$ term by $\log(N/\epsilon)$. This means that the number of bits to specify each entry is $O\left(\log N + \log\lambda + \log(1/\epsilon) + \log\log n\right)$. Since we always have $n = \mathsf{poly}(\lambda)$, we can ignore the $\log\log n$ term above.

We now can set $\sigma_r = \sqrt{1-\beta}$ and $\sigma_a = \sqrt{1-\alpha}$, and consider the wiretap channel with receiver channel $D_{\gamma\mathbb{Z}^n,\sigma_r}$ and adversary channel $D_{\gamma\mathbb{Z}^n,\sigma_a}$. Since these distributions are above the smoothing parameter (wrt $\epsilon = 2^{-\lambda/N}$), we can expect the gap between their capacities to be similar to their continuous counterparts, namely we expect $\xi := c(D_{\gamma\mathbb{Z}^n,\sigma_r}) - c(D_{\gamma\mathbb{Z}^n,\sigma_a}) = \Theta(\epsilon)$.

Plugging the parameters from Appendix A.1.2 we would get $n = \max\left(\mathsf{poly}(\frac{1}{\epsilon}), \lambda^2, O(\frac{\lambda}{\epsilon^2})\right)$, so the share size is

$$\max\left(\mathsf{poly}(\frac{1}{\epsilon}), \lambda^2, O(\frac{\lambda}{\epsilon^2})\right) \cdot O\left(\log N + \log\lambda + \log(1/\epsilon)\right).$$

With seed size $k = \Theta(\epsilon)n$ and message size $b \approx (\Theta(\epsilon) - \frac{2}{\lambda})n$. This implies a rate $|\text{secret}|/|\text{share}| = O(\epsilon/(\log N + \log\lambda + \log(1/\epsilon)))$, which is not as good as for the BSC. However, it is plausible that the $\mathsf{poly}(1/\epsilon)$ term for decoding AWGN channels is better than for BSC, in which case the concrete share sizes or short secrets could still be smaller.

Quantum Security of Subset Cover Problems

Samuel Bouaziz-Ermann
LIP6, Paris, France
Sorbonne Université, Paris, France
CNRS, Paris, France

Alex B. Grilo
LIP6, Paris, France
Sorbonne Université, Paris, France
CNRS, paris, France

Damien Vergnaud
LIP6, Paris, France
Sorbonne Université, Paris, France
CNRS, Paris, France

Abstract

The subset cover problem for $k \geq 1$ hash functions, which can be seen as an extension of the collision problem, was introduced in 2002 by Reyzin and Reyzin to analyse the security of their hash-function based signature scheme HORS. The security of many hash-based signature schemes relies on this problem or a variant of this problem (e.g. HORS, SPHINCS, SPHINCS+, ...).

Recently, Yuan, Tibouchi and Abe (2022) introduced a variant to the subset cover problem, called restricted subset cover, and proposed a quantum algorithm for this problem. In this work, we prove that any quantum algorithm needs to make $\Omega\left((k+1)^{-\frac{2^k}{2^{k+1}-1}} \cdot N^{\frac{2^k-1}{2^{k+1}-1}}\right)$ queries to the underlying hash functions with codomain size N to solve the restricted subset cover problem, which essentially matches the query complexity of the algorithm proposed by Yuan, Tibouchi and Abe.

We also analyze the security of the general (r,k)–subset cover problem, which is the underlying problem that implies the unforgeability of HORS under a r-chosen message attack (for $r \geq 1$). We prove that a generic quantum algorithm needs to make $\Omega\left(N^{k/5}\right)$ queries to the underlying hash functions to find a $(1,k)$-subset cover. We also propose a quantum algorithm that finds a (r,k)-subset cover making $O\left(N^{k/(2+2r)}\right)$ queries to the k hash functions.

2012 ACM Subject Classification Security and privacy → Cryptography

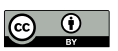
Keywords and phrases Cryptography, Random oracle model, Quantum information

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.9

Related Version *Full Version*: `https://eprint.iacr.org/2022/1474`

Funding This work was partially funded by PEPR integrated project EPiQ ANR-22-PETQ-0007 part of Plan France 2030. This work is part of HQI initiative (www.hqi.fr) and is supported by France 2030 under the French National Research Agency award number "ANR-22-PNCQ-0002".
Alex B. Grilo: ABG is supported by ANR JCJC TCS-NISQ ANR-22-CE47-0004, and by the PEPR integrated project EPiQ ANR-22-PETQ-0007 part of Plan France 2030.

Acknowledgements We thanks the anonymous reviewers for their valuable comments that helped improving the quality of this paper.



© Samuel Bouaziz-Ermann, Alex B. Grilo, and Damien Vergnaud;
licensed under Creative Commons License CC-BY 4.0

1 Introduction

Cryptographic hash functions are functions mapping arbitrary-length inputs to fixed-length outputs and are one of the central primitives in cryptography. They serve as building blocks for numerous cryptographic primitives such as key-establishment, authentication, encryption, or digital signatures. In particular, *one-time signatures* – i.e. in which the signing key can be used only once – based only on hash functions were proposed by Lamport as soon as in 1979 [9]. The basic idea is to evaluate a cryptographic hash function on secret values to generate the public verification key and to authenticate a single message by revealing a subset of those secret pre-images.

With the development of quantum technologies, which may bring drastic attacks against widely deployed cryptographic schemes based on the hardness of integer factorization or the discrete logarithm [12], hash-based signatures have regained interest within the realm of "post-quantum" cryptography and the recent NIST standardization process. In particular, the SPHINCS+ candidate [3] has been selected in 2022 for standardization by NIST and other constructions are standardized by IETF/IRTF. The SPHINCS+ signature scheme and its predecessor SPHINCS [2] make use of a Merkle-hash tree and of HORST, a variant of a hash-based scheme called HORS [11]. HORS (for "Hash to Obtain Random Subset") uses a hash function to select the subset of secret pre-images to reveal in a signature and the knowledge of these secrets for several subsets may not be enough to produce a forgery, a property that makes HORS a *few-time signature* scheme.

More concretely, the security of HORS (and HORST) relies on the hardness of finding a subset cover (SC) for the underlying hash function. More formally, to define the (r, k)*–SC problem*, we consider the hash function as the concatenation of $k \geq 1$ hash functions $h_1, \ldots, h_k$ (with smaller outputs) and the problem is to find, for some integer $r \geq 1$, $r + 1$ elements $x_0, x_1 \ldots, x_r$ in the hash function domain such that $x_0 \notin \{x_1, \ldots, x_r\}$, and

$$\{h_i(x_0) | 1 \leq i \leq k\} \subseteq \bigcup_{j=1}^{r} \{h_i(x_j) | 1 \leq i \leq k\}\,.$$

The hardness of this problem for concrete popular hash functions has not been studied in depth but Aumasson and Endignoux [1] proved in 2017 a lower bound on the number of queries to hash functions for the SC problem in the Random Oracle Model (ROM). However, the exact security of HORS (and more generally HORST, SPHINCS and SPHINCS+) with respect to quantum attacks is still not clear. Since quantum computing provides speedups for many problems (e.g. Grover's search algorithm [8] and Brassard, Høyer, and Tapp [6] collision search algorithm), it is important to provide lower bounds in a quantum world.

1.1 Our results

In this paper, we explore the difficulty of finding subset cover for idealized hash functions for quantum algorithms. We also consider a variant called the k-restricted subset cover (k–RSC) problem where, given k functions $h_1, \ldots, h_k : \mathcal{X} \to \mathcal{Y}$ such that $N = |\mathcal{Y}|$, one has to find $k + 1$ elements $x_0, x_1 \ldots, x_k$ such that:

$$\forall 1 \leq i \leq k, h_i(x_0) = h_i(x_i)$$

and $x_0 \notin \{x_1, \ldots, x_k\}$. This variant was defined recently by Yuan, Tibouchi and Abe [14], who showed a quantum algorithm to solve it. The main contributions of this work are:

1. **Lower bound on k–RSC**: we prove that $\Omega\left((k+1)^{-\frac{2^k}{2^{k+1}-1}} \cdot N^{\frac{2^k-1}{2^{k+1}-1}}\right)$ quantum queries to the idealized hash functions are needed to find a k–RSC with constant probability. (Theorem 14)
2. **Lower bound on $(1,k)$–SC**: we prove that $\Omega\left((k!)^{-1/5} \cdot N^{k/5}\right)$ quantum queries to the idealized hash functions are needed to find a $(1,k)$–SC with constant probability. (Theorem 21)
3. **Upper bound on (r,k)–SC:** we present a quantum algorithm that finds a (r,k)–SC with constant probability with $O\left(N^{k/(2+2r)}\right)$ queries to the hash functions when k is divisible by $r+1$, and $O\left(N^{k/(2+2r)+1/2}\right)$ otherwise. (Theorem 29)

1.2 Technical Overview

To prove our lower bounds on the query complexity, we use the technique called *compressed random oracle model* introduced by Zhandry in [15]. Its goal is to record information about the queries of an adversary A in the quantum random oracle model and it permits "on-the-fly" simulation of random oracles (or *lazy sampling*) by considering the uniform superposition of all possible random oracles instead of picking a single random oracle at the beginning of the computation. The technique uses a register to keep a record of a so-called *database* of the random oracle and this register is updated whenever A makes a query to the random oracle. At the end of A's computation, the reduction can measure the register of the database, and the distribution of the outputs is uniformly random, as if we had chosen a random oracle at the beginning of its computation. This new register that contains the database is at the gist of our lower bounds.

In Section 3, we prove the lower bound on the query complexity to solve the RSC problem. We consider an algorithm A after i quantum queries to the random oracle and call its state at this moment $|\psi_i\rangle$. Our goal is to compute an upper bound for the value $|P_k^{RSC}|\psi_i\rangle|^2$, where P_k^{RSC} is the projection onto the databases that contain a k–RSC. Computing such a bound leads to a lower bound on the number of queries needed for solving k–RSC with constant probability. To prove our bound, we proceed by induction: assuming we proved a bound for the k'–RSC problem for all $k' < k$, we prove a bound for the k–RSC problem. The analysis is naturally divided into two parts: whenever A finds a k–RSC after i quantum queries, it means that either:

1. A finds it after $i-1$ quantum queries;
2. or A finds it with the i^{th} quantum query.

The first case is recursive and it remains to bound $|P_k^{RSC}|\psi_i\rangle|$ in the second case. Here, the database (after $i-1$ quantum queries) must contain a certain number of k'–RSC (for some $k' < k$), in order for A to find k–RSC with the i^{th} query. Using this strategy, we obtain a recursive formula from which we can deduce the bound on $|P_k^{RSC}|\psi_i\rangle|$.

In Section 4.1, we prove a lower bound for the $(1,k)$–SC problem. The idea of the proof is similar to the proof for the lower bound of the k–RSC problem but we have to compute a bound for another problem that we define: the *j–repetition* problem.

Finally in Sections 4.2 and 4.3, we design a family of quantum algorithms for finding a (r,k)–SC. These algorithms are inspired by the algorithm from [14] to solve the k–RSC problem and [10]'s algorithm for finding multi-collisions. These algorithms are recursive and take as input two parameters $t, k' \in \mathbb{N}$ and perform the following:

1. Find t distinct $(r-1,k')$–SC;
2. Find the (r,k)–SC.

The parameters t and k' are chosen in order to optimize the complexity of the algorithm. The first step is done by applying $r-1$ times the algorithm for the value k', and the second step uses Grover's algorithm [8][5].

Full version of the paper

In this paper, most of the proofs are omitted in the interest of space. The proofs of all the lemmas and theorems stated in this paper can be found in the full version of the paper, available on eprint.

1.3 Related works, discussion and open problems

Collision-finding

The link between finding a multi-collision and finding a subset cover was first discussed in [14], since their algorithm is inspired from the one for finding multi-collisions in [10]. In the latter, they also show a lower bound for finding multi-collisions, and our proof of lower bounds uses the same technique they used. We make use of the compressed oracle technique, first introduced by Zhandry in [15], and generalize the proof of the lower bound on multi-collisions to the RSC and SC problems.

Restricted Subset Cover

There is currently only one quantum algorithm for finding RSC [14]. Our lower bound for finding a RSC matches their upper bound when k, the number of functions, is constant. However when k is not a constant, their algorithm makes $O\left(k \cdot N^{\frac{2^k-1}{2^{k+1}-1}}\right)$ queries to $h_1, \ldots, h_k$, which roughly leaves a $k^{3/2}$ gap between the best known attack and our lower bound. To the best of our knowledge, this is the first lower bound on the RSC problem for a quantum algorithm, and there are no such result for classical algorithms. It would be interesting to see if we can close this gap further.

Tighter bounds for $(1,k)$–SC

When k is constant, the lower bound for $(1,k)$–SC is $\Omega\left(N^{k/5}\right)$, while our algorithm for this problem makes $O\left(N^{k/4}\right)$ queries to the oracle (when k is even). It would be interesting to tighten this gap, especially since the results for $(1,k)$–SC are probably necessary to prove the lower bounds (r,k)–SC for $r \geq 2$.

For non-constant k, our lower bound for $(1,k)$–SC is $\Omega\left(C_k^{-1/5} \cdot N^{k/5}\right)$, where $C_k = \sum_{j=2}^{k} \frac{k!}{(j-1)!} \leq k! \cdot e$. Notice that this term cannot be neglected for large values of k. For example with $k = \log(N)$, we have $C_k \geq N$. In comparison, our best algorithm for $(1,k)$–RSC, the factor in k is $\binom{k}{(k+1)/2}^{-1/2} \leq \frac{2^{(k+1)/2}}{\left(\frac{k+1}{2} \cdot \pi\right)^{1/4}}$, which is very far from our bound on C_k. It would also be interesting to see if we can tighten this gap.

Bounds for (r,k)–SC

Unfortunately, expanding our result for the (r,k)–SC problem is much more complicated than the case $r = 1$ and actually even proving the case $r = 2$ is not simple. To prove such a result, one would need a bound for the problem of finding j distinct $(1,k)$–SC problem. While proving such a bound is challenging, it is also unclear what the problem of finding

j distinct $(1,k)$–SC is. Indeed, an important property for our technique in the first lower bound proofs is that by making one query to the oracle, the adversary cannot find two or more k–RSC. The same property must hold for the problem of finding j distinct $(1,k)$–SC, and this definition and subsequent analysis remain open.

Security of SPHINCS and SPHINCS+

The signature scheme SPHINCS relies on the HORST scheme (for "HORS with trees") which adds a Merkle tree to the HORS scheme to compress the public key. The security of HORST also relies on the (r,k)–SC problem but the security of SPHINCS rely on different security notions of the underlying hash functions. In particular, it depends on a variation of the SC problem classed the *target subset cover* (TSC) problem [11]. The main difference comes from the fact that the message signed using HORST is an unpredictable function of the actual message and this prevents an attacker to construct a subset cover beforehand.

Nevertheless, the authors of [2] stated an existential unforgeability result for SPHINCS [2, Theorem 1] under q_s-adaptive chosen message attacks. The success probability in such attacks is roughly upper-bounded by:

$$\sum_{r=1}^{\infty} \min\left(2^{r(\log q_s - h)+h}, 1\right) \cdot Succ_A((r,k) - SC),$$

where h is the height of the tree used in SPHINCS, and $Succ_A((r,k) - SC)$ denotes the success probability of an adversary A to find a (r,k)–SC. The authors made the assumption that this term is negligible for any probabilistic adversary A and our quantum lower bound on the query number to find a $(1,k)$–SC can be seen as a first step towards proving this assumption (for idealized hash functions). To assess the security of SPHINCS from [2, Theorem 1] for concrete parameters such as those proposed in [2] (namely $h = 60, q_s = 2^{30}$), it would also be necessary to upper-bound the success probabilities $Succ_A((2,k) - SC)$ and $Succ_A((3,k) - SC)$, which we leave for future work. For example, one could try to apply [13, Theorem 4.12] to get a lower bound for (r,k)–SC more easily, but the obtained bound will most likely not be tight.

SPHINCS+ is an enhancement of SPHINCS, which makes the scheme more efficient and its security relies on another variant of the SC problem, namely the interleaved target subset cover (ITSC) problem. It would also be interesting to see if our methods can be used to prove similar bounds for the TSC and ITSC problems. At last, one could also try to design algorithms for these two problems, as no quantum algorithms for them exist yet to the best of our knowledge.

2 Preliminaries

We assume the reader is familiar with the theory of quantum information. We denote the concatenation by $||$.

2.1 Compressed oracle technique

We now present the key ingredients of Zhandry's compressed oracle technique, first defined in [15] and refined in [7]. As mentioned in the introduction, the technique uses a register to keep a record of a so-called *database* of the random oracle and this register is updated whenever an adversary A makes a query to the random oracle. This new register that contains the database is at the gist of our lower bounds.

We consider the *Quantum Random Oracle Model*, first defined in [4]. In this model, we are given black-box access to a *random* function $H : \mathcal{X} \to \mathcal{Y}$. For our model, the adversary will work on three different registers $|x, y, z\rangle$. The first register is the query register, the second register is the answer register and the third register is the work register. The first two registers are used for queries and answers to the oracle, while the last register is for the adversary's other computations. We first define the unitary StO that represents the *Standard Oracle* and that computes as follows:

$$\mathsf{StO} \sum_{x,y,z} \alpha_{x,y,z} |x, y, z\rangle \to \sum_{x,y,z} \alpha_{x,y,z} |x, y + H(x), z\rangle$$

This unitary corresponds to a query to H.

Now, we define Zhandry's compressed oracle. In this model, instead of starting with a random function H, we start with the uniform superposition of all random functions $|H\rangle$, where $|H\rangle$ encodes the truth table of the function H. In this model, there is a register for each $x \in \mathcal{X}$, and the value of this register in the state $|H\rangle$ corresponds to $H(x)$. That is, we have that $|H\rangle = \bigotimes_{x \in \mathcal{X}} |H(x)\rangle_x$ Let $\mathcal{H} = \{H : \mathcal{X} \to \mathcal{Y}\}$ be the set of all possible functions H. We define a new register, the database register $|H\rangle$, that starts in the uniform superposition $\frac{1}{|\mathcal{H}|} \sum_{H \in \mathcal{H}} |H\rangle$. This register starts in product state with the other registers, and Zhandry's idea is that instead of modifying the adversary's register when querying the oracle, we will modify the database register instead. To do so, we simply consider the *Fourier basis* for the y and the H register before querying the Standard Oracle.

We write this unitary O and it works as follows:

$$\mathsf{O} \sum_{x,\hat{y},z} \alpha_{x,\hat{y},z} |x, \hat{y}, z\rangle \otimes \sum_{\hat{H} \in \mathcal{H}} \alpha_{\hat{H}} \left|\hat{H}\right\rangle \to \sum_{x,\hat{y},z} \alpha_{x,\hat{y},z} |x, \hat{y}, z\rangle \otimes \sum_{\hat{H} \in \mathcal{H}} \alpha_{\hat{H}} \left|\hat{H} \ominus (x, \hat{y})\right\rangle,$$

where, for any fixed $x \in \mathcal{X}$ and $z \in \mathcal{Y}$, $H \ominus (x, z) : \mathcal{X} \to \mathcal{Y}$ is defined as:

$$H \ominus (x, z)(x') = \begin{cases} H(x') & \text{if } x' \neq x \\ H(x) - z & \text{if } x' = x. \end{cases}$$

In other words, $H \ominus (x, z)$ is obtained by replacing the value of $H(x)$ by $H(x) - z$ in H.

This unitary can be implemented by applying the QFT to the registers $|y\rangle$ and $|H\rangle$, applying the Standard Oracle, then applying the $QFT^\dagger$ again on the $|y\rangle$ and $|H\rangle$ registers.

Finally, we define the compression part. The idea behind the compression is that for every x in the database mapped to $\left|\hat{0}\right\rangle$, we remap it to $|\perp\rangle$, where $\perp$ is a new value outside of $\mathcal{Y}$. More formally, the compression part is done by applying:

$$\mathsf{Comp} = \bigotimes_x \left(|\perp\rangle \left\langle\hat{0}\right| + \sum_{\hat{y}:\hat{y} \neq \hat{0}} |\hat{y}\rangle \langle\hat{y}| \right)$$

in the Fourier basis.

Since at the start of the computation, the database will be initiated with the uniform superposition over all $\mathcal{H}$ possible, then after q queries the state of the database can be described with q vectors. In order to apply the compression as a unitary, we declare that $\mathsf{Comp} |\perp\rangle = \left|\hat{0}\right\rangle$.

Now, we can define the *Compressed Oracle*:

$$\mathsf{cO} = \mathsf{Comp} \circ \mathsf{O} \circ \mathsf{Comp}^\dagger.$$

Of course the compression part inevitably creates some losses, compared to only using the Standard Oracle. The precise characterization of these losses is given in one of Zhandry's lemma, and can be stated as follows:

▶ **Lemma 1** (Lemma 5 from [15]). *Let A be an algorithm that makes queries to a random oracle $H : \mathcal{X} \to \mathcal{Y}$, and output $(x_1, \ldots, x_k, y_1, \ldots, y_k) \in \mathcal{X}^k \times \mathcal{Y}^k$. Let p be the probability that $\forall 1 \leq i \leq k$, $H(x_i) = y_i$. Similarly, consider the algorithm A running with the Compressed Oracle cO, and output $(x'_1, \ldots, x'_k, y'_1, \ldots, y'_k) \in \mathcal{X}^k \times \mathcal{Y}^k$. Let p' be the probability that $\forall 1 \leq i \leq k$, $H'(x'_i) = y'_i$, where H' is obtained by measuring the H register at the end of the execution of the algorithm A. Then:*

$$\sqrt{p} \leq \sqrt{p'} + \sqrt{\frac{k}{|\mathcal{Y}|}}.$$

In the rest of the paper, we will have that $\sqrt{\frac{k}{|\mathcal{Y}|}}$ is negligible, and thus we will neglect this term.

We also have the following lemma from [7] that describes the operator $\mathsf{cO}_{(x,\hat{y})} : \mathcal{H} \to \mathcal{H}$, which is defined as the operator applied on $|H\rangle$ when applying cO to $|x\rangle |\hat{y}\rangle \otimes |H\rangle$. More formally, we have that:

$$\mathsf{cO} |x\rangle |\hat{y}\rangle \otimes |H\rangle = |x\rangle |\hat{y}\rangle \otimes \mathsf{cO}_{(x,\hat{y})} |H\rangle .$$

▶ **Lemma 2** (Lemma 4.3 from [7]). *For any $\hat{y} \neq \hat{0}$, the operator $\mathsf{cO}_{(x,\hat{y})}$ is represented by the following matrix:*

	$\perp$	r
$\perp$	0	$\frac{\omega_N^{-ry}}{\sqrt{\lvert\mathcal{Y}\rvert}}$
y'	$\frac{\omega_N^{yy'}}{\sqrt{\lvert\mathcal{Y}\rvert}}$	$\begin{cases} \left(1 - \frac{2}{\lvert\mathcal{Y}\rvert}\right)\omega_N^{yy'} + \frac{1}{\lvert\mathcal{Y}\rvert} & \text{if } y' = r \\ \frac{1-\omega_N^{yy'}-\omega_N^{ry}}{\lvert\mathcal{Y}\rvert} & \text{if } y' \neq r \end{cases}$

For $\hat{y} = \hat{0}$, we have that $\mathsf{cO}_{(x,\hat{0})}$ is the identity.

We also define, for any compressed $H : \mathcal{X} \to \mathcal{Y} \cup \{\perp\}$, for any fixed $x \in \mathcal{X}$ and $z \in \mathcal{Y}$, $H \cup (x, z) : \mathcal{X} \to \mathcal{Y}$ as:

$$H \cup (x, z)(x') = \begin{cases} H(x') & \text{if } x' \neq x \\ z & \text{if } x' = x. \end{cases}$$

In other words, $H \cup (x, z)$ is obtained by replacing the value of $H(x)$ by z in H.

In the following, we will model the adversary (A) as a series of computation alternating between unitaries and oracle calls. The adversary's quantum state will first be initialized to $|0\rangle^{\otimes N}$. Then, his computation will be decomposed as:

$$A = U_k \mathsf{cO} U_{k-1} \mathsf{cO} \ldots \mathsf{cO} U_2 \mathsf{cO} U_1 \tag{1}$$

So that, if $|\psi_i\rangle = \sum_{x,y,z,D} \alpha_{x,y,z,D} |x, y, z, D\rangle$ is the state of the adversary after i quantum queries to cO, then U_{i+1} operates on the registers x, y and z only. We also define *database properties*:

▶ **Definition 3** (Database property). *A database property is a subset of $\mathcal{H}$. Any database property D can be seen as a projector on $\mathcal{H}$, as follows:*

$$\sum_{d\in D} |d\rangle\langle d|$$

We write $\mathcal{D} = \{I | I \subseteq \mathcal{H}\}$ the set of all subspaces of $\mathcal{H}$, that also corresponds to the set of all database properties.

We now state and prove two lemmas adapted from [10] that we will use thoroughly in this paper. The first lemma will allow us to ignore the unitaries that the adversary A applies on the first registers of the state.

▶ **Lemma 4** (adapted from Lemma 8 from [10]). *For any unitary U, any projector P, and any state $|\phi\rangle$,*

$$|(I\otimes P)\cdot(U\otimes I)|\phi\rangle| = |(I\otimes P)|\phi\rangle|$$

The second lemma bounds the amplitude of measuring a database that satisfies a property P at the i^{th} step of the algorithm, i.e. just after the i^{th} query to the oracle. In this bound, the first term captures the case where we succeed to find a database that satisfies P before the i^{th} query. The second term captures the case where we did not have it before the i^{th} query, but found it with the i^{th} one.

▶ **Lemma 5** (adapted from Lemma 9 from [10]). *Let $|\phi_i\rangle$ be the state of an algorithm A just before the i^{th} quantum query to cO, and $|\psi_i\rangle$ the state of the same algorithm right after the i^{th} quantum query to cO. Let P be any projector on D. We have that:*

$$|P|\psi_i\rangle| \leq |P|\phi_i\rangle| + |P\mathsf{cO}(I-P)|\phi_i\rangle|$$

Proof.

$$\begin{aligned}|P|\psi_i\rangle| = |P\mathsf{cO}|\phi_i\rangle| &= |P\mathsf{cO}(P|\phi_i\rangle + (I-P)|\phi_i\rangle)| \\ &\leq |P|\phi_i\rangle| + |P\mathsf{cO}(I-P)|\phi_i\rangle)|\,,\end{aligned}$$

where the inequality comes from the triangle inequality and the fact that $P\mathsf{cO}P \leq P$. ◀

▶ Remark. In the next section and in the rest of the paper, we will consider multiple functions $h_1,\ldots,h_k : \mathcal{X}\to\mathcal{Y}$ for some fixed k. Note that this is equivalent to considering one function $H : \mathcal{X}\to\mathcal{Y}^k$, such that we interpret, for any $x\in\mathcal{X}$, the output $H(x)$ as the concatenation of values of the functions applied to x, i.e. $H(x) = h_1(x)||h_2(x)||\cdots||h_k(x)$. Hence, in this setting, the compressed oracle is used on the function H, and a query to any of the h_i is a query to all of the h_i's. Thus, in our results, we count the number of queries to the function H and thus the number of queries to all of the h_i's. It may seem that we lose some accuracy in this setting, however this is with the same method that multiple random functions are implemented in the literature.

2.2 The problem of subset cover and its variants

We define the problem of subset cover.

▶ **Definition 6** ((r,k)–SC). *Let $k,r\in\mathbb{N}^*$. Let $h_1,\cdots,h_k : \mathcal{X}\to\mathcal{Y}$. A (r,k)–SC for $(h_1,\cdots,h_k)$ is a set of $r+1$ elements $x_0,x_1,x_2,\cdots,x_r$ in $\mathcal{X}$ such that:*

$$\{h_i(x_0)|1\leq i\leq k\} \subseteq \bigcup_{j=1}^{r}\{h_i(x_j)|1\leq i\leq k\}$$

In other words, for each $1 \leq i \leq k$, there exists a $1 \leq j \leq r$ and a $1 \leq \ell \leq k$ such that $h_i(x_0) = h_\ell(x_j)$.

We notice two facts regarding the parameters of (r,k)–SC. First, we have that the problem becomes easier when r increases. Secondly, we have that when $r > k$, a (r,k)–SC contains a (k,k)–SC. Thus finding a (r,k)–SC when $r > k$ is the same as when $r = k$. For simplicity, we use k–SC as a shorthand of (k,k)–SC.

We also define the database properties $P^{SC}_{(r,k)}$ of containing a (r,k)–SC, that is the set of databases that contains a (r,k)–SC. More formally, we have that:

$$P^{SC}_{(r,k)} = \left\{ D \in \mathcal{D} \middle| \exists x_0, x_1, \ldots, x_r, \forall i \neq 0, x_0 \neq x_i, H(x_0) \subseteq \bigcup_{i=1}^{r} H(x_i) \right\},$$

where for $x \in \mathcal{X}, H(x) = \{h_1(x), \ldots, h_k(x)\}$.

We follow now with the definition of a harder variation of the k-subset cover called the *k-restricted subset cover* (k–RSC).

▶ **Definition 7** (k–RSC). *Let $k \in \mathbb{N}^*$. Let $h_1, \ldots, h_k : \mathcal{X} \to \mathcal{Y}$. A k-*restricted subset cover *(k–RSC) for $(h_1, \ldots, h_k)$ is a set of $k+1$ elements $x_0, x_1, x_2, \ldots, x_k$ in $\mathcal{X}$ such that:*

$$\forall i \in \{1, \ldots, k\}, h_i(x_0) = h_i(x_i) \text{ and } x_0 \neq x_i.$$

We also define the database properties $P^{RSC}_{k,\ell}$ of k distinct ℓ–RSC, that is the set of databases that contains k distinct ℓ–RSC. More formally, we have that:

$$P^{RSC}_{k,\ell} = \left\{ D \in \mathcal{D} \middle| \begin{array}{l} \exists x_{0,1}, \ldots, x_{\ell,1}, \forall i \neq 0, x_{0,1} \neq x_{i,1}, \forall i, h_i(x_{0,1}) = h_i(x_{i,1}) \\ \exists x_{0,2}, \ldots, x_{\ell,2}, \forall i \neq 0, x_{0,2} \neq x_{i,2}, \forall i, h_i(x_{0,2}) = h_i(x_{i,2}) \\ \vdots \\ \exists x_{0,\ell}, \ldots, x_{\ell,k}, \forall i \neq 0, x_{0,k} \neq x_{i,k}, \forall i, h_i(x_{0,k}) = h_i(x_{i,k}) \\ \forall i \neq j, (h_1(x_{0,i}), \ldots, h_\ell(x_{0,i})) \neq (h_1(x_{0,j}), \ldots, h_\ell(x_{0,j})) \end{array} \right\} \tag{2}$$

The problem of finding a k–RSC was introduced in [14], in which the authors describe an algorithm that finds a k–RSC in $O\left(kN^{\frac{1}{2}\left(1-\frac{1}{2^{k+1}-1}\right)}\right)$ quantum queries to $h_1, \ldots, h_k$ when the h_i's are such that $|\mathcal{X}| \geq (k+1)|\mathcal{Y}|$.

We discuss now the last condition in Equation (2). We remark that while such condition was not explicitly imposed in [10] for their lower bound for finding multi-collisions, this property is implicitly and extensively used in their proof. Such a property is needed because when they count k-collisions (that is, k distinct $x_1, \ldots, x_k$ such that $H(x_1) = \cdots = H(x_k)$), they are actually interested in the number of possible *images* that would be helpful to reach a $(k+1)$-collision. In particular, this is helpful since one query can only transform *one* k-collision (with such a property) into a $(k+1)$-collision.

In our case, the last line of (2) ensures that the "supporting set" of the k–RSC (i.e. the set of images of the $x_{0,i}$ by the different random functions $h_1, \ldots, h_k$) is unique. As in the multi-collision case, this condition will be crucial to extend a k–RSC to a $(k+1)$–RSC, and for this reason we define it explicitly in $P^{RSC}_{k,\ell}$.

Finally, we state a result from [10], regarding the amplitude of finding j distinct 2-collisions:

▶ **Lemma 8** (adapted from [10], Corollary 11). *Given a random function $h : \mathcal{X} \to \mathcal{Y}$ where $|N| = \mathcal{Y}$, let $f^{col}_{i,j}$ be the amplitude of the D containing at least j distinct 2-collisions after i quantum queries. Then:*

$$f^{col}_{i,j} \leq \left(\frac{4e \cdot i^{3/2}}{j\sqrt{N}}\right)^j.$$

3 Lower bound on the k-restricted subset cover problem

In this section, we prove a lower bound for the k–RSC problem defined in Definition 7. This section follows closely [10]'s proof of their lower bound on finding multi-collisions. We first prove a lower bound for finding k distinct 2–RSC, which will be necessary in our induction step. Finally, we will prove the induction step in the last subsection and obtain a lower bound on finding s distinct k–RSC.

3.1 Finding k distinct 2-restricted subset cover

We want to bound the number of queries needed to find k distinct triplets that satisfy a 2–RSC. We have the following result:

▶ **Theorem 9.** *Given two random functions $h_1, h_2 : \mathcal{X} \to \mathcal{Y}$ where $N = |\mathcal{Y}|$, a quantum algorithm needs to make $\Omega(k^{4/7} \cdot N^{3/7})$ queries to h_1 and h_2 to find k distinct 2–RSC with constant probability, for any $k \leq N^{1/8}$.*

To prove this theorem, we first introduce some notation. We denote $P_{2,k,\ell}$ the set of databases that satisfies k distinct 2–RSC, and that contain exactly ℓ collisions on h_1. We denote $g_{i,k} = \left| P_{k,2}^{RSC} |\psi_i\rangle \right|$ and $\widehat{g}_{i,k,\ell} = |P_{2,k,\ell} |\psi_i\rangle|$, where $|\psi_i\rangle$ is the state just after the i^{th} query to $H = (h_1, h_2)$.

Our goal is to bound $g_{i,k}$, and we will first prove a recursive formula stated in the next lemma.

▶ **Lemma 10.** *For every $i \in \mathbb{N}$, and every $k \in \mathbb{N}$, we have that:*

$$g_{i,k} \leq g_{i-1,k} + \sqrt{2 \sum_{\ell \geq 0} \frac{\ell}{N} \widehat{g}_{i-1,k-1,\ell}^2} + \frac{(i-1)}{N} g_{i-1,k-1}.$$

We will split the sum in two using $\mu_3(j)$ as a threshold. We also define a new notation that will simplify expressions:

▶ **Definition 11.**

$$A_i = \sum_{\ell=0}^{i-1} \sqrt{2} \left(\sqrt{\frac{\mu_3(\ell-1)}{N}} + \sqrt{8} \frac{\ell - 1}{N} \right),$$

where

$$\mu_3(\ell) = \max \left\{ 8e \frac{\ell^{3/2}}{\sqrt{N}}, 10N^{1/8} \right\}.$$

Before bounding $g_{i,k}$, we first prove a bound on A_i.

▶ **Lemma 12.** *For every $i \in \mathbb{N}$, we have that:*

$$A_i \leq 8\sqrt{e} \frac{i^{7/4}}{N^{3/4}} + 4\frac{i^2}{N} + O\left(N^{-1/48}\right).$$

It follows that $A_i < 2eN^{1/8}$ for $i \leq N^{1/2}$.

We can now state the lemma that bounds $g_{i,k}$.

▶ **Lemma 13.** *For every $i \in \mathbb{N}$ and $k \in \mathbb{N}$, we have that:*

$$g_{i,k} < \frac{A_i^k}{k!} + \sqrt{2} \cdot 2^{-N^{1/8}}.$$

We can now prove the main theorem of this subsection.

Proof of Theorem 9. Following from Lemma 13, we have that:

$$g_{i,k} \leq \frac{A_i^k}{k!} + \sqrt{2} \cdot 2^{-N^{1/8}} \leq \left(\frac{A_i \cdot e}{k}\right)^k + \sqrt{2} \cdot 2^{-N^{1/8}}.$$

We now use the bound on A_i of Lemma 12:

$$g_{i,k} \leq \left(\frac{8e^{3/2}}{k} \cdot \frac{i^{7/4}}{N^{3/4}} + \frac{4e}{k} \cdot \frac{i^2}{N} + \frac{e}{k} \cdot O\left(N^{-1/48}\right)\right)^k + \sqrt{2} \cdot 2^{-N^{1/8}}.$$

So if $i = o(k^{4/7} \cdot N^{3/7})$, then $g_{i,k} = o(1)$. Hence if we want $g_{i,k}$ to be a constant, i.e. not $o(1)$, we must have $i = \Omega\left(k^{4/7} \cdot N^{3/7}\right)$. ◀

3.2 Finding k distinct s-restricted subset cover

In this section, we generalize the result to the problem of finding k distinct s–RSC, for any $s \geq 3$ and any $k \geq 1$. We are given s random functions $h_1, \ldots, h_s$ such that for any $i \in [1, s]$, $h_i : \mathcal{X} \to \mathcal{Y}$. We will prove the following theorem.

▶ **Theorem 14.** *Given s random functions $h_1, \ldots, h_s : \mathcal{X} \to \mathcal{Y}$ where $N = |\mathcal{Y}|$, a quantum algorithm needs to make $\Omega\left((s+1)^{-\frac{2^s}{2^{s+1}-1}} \cdot k^{\frac{2^s}{2^{s+1}-1}} \cdot N^{\frac{2^s-1}{2^{s+1}-1}}\right)$ queries to $h_1, \ldots, h_s$ to find k distinct s–RSC with constant probability, for any $s \leq \log(\log(N))$ and any $k \geq N^{1/2^{s+1}}$.*

And naturally we have the following corollary for $k = 1$:

▶ **Corollary 15.** *Given s random functions $h_1, \ldots, h_s : \mathcal{X} \to \mathcal{Y}$ where $N = |\mathcal{Y}|$, a quantum algorithm needs to make $\Omega\left((s+1)^{-\frac{2^s}{2^{s+1}-1}} \cdot N^{\frac{2^s-1}{2^{s+1}-1}}\right)$ queries to $h_1, \ldots, h_s$ to find one s–RSC with constant probability, for any $s \leq \log(\log(N))$.*

In order to prove Theorem 14, we first define some notations, starting with the notations for the amplitudes. We define:

1. $f_{i,j}$ as the amplitude of the databases D containing at least j distinct $(s-1)$–RSC after i quantum queries.
2. $\widehat{g}_{i,j,k}$ as the amplitude of the databases D containing at least j distinct $(s-1)$–RSC and exactly k distinct s–RSC after i quantum queries.
3. $g_{i,k}$ as the amplitude of the databases D containing exactly k distinct s–RSC after i quantum queries.

More formally, let $|\phi_i\rangle$ (resp. $|\psi_i\rangle$) be the state of the algorithm just before (resp. after) the i^{th} query to the oracle. We have:

$$f_{i,j} = \left\| P^{RSC}_{j,(s-1)} |\psi_i\rangle \right\|,$$
$$\widehat{g}_{i,j,k} = \left\| P^{RSC}_{j,(s-1)} P^{RSC}_{k,s} \neg P^{RSC}_{k+1,s} |\psi_i\rangle \right\|,$$
$$g_{i,k} = \left\| P^{RSC}_{k,s} \neg P^{RSC}_{k+1,s} |\psi_i\rangle \right\|.$$

We want to bound $g_{i,k}$, and to do so, we define some convenient notation. We start by defining Π_s, a term that appears in the bound of $g_{i,k}$.

▶ **Definition 16.** *Let Π_s be defined as follows:*

$$\begin{cases} \Pi_1 = 1 \\ \Pi_2 = 1 \\ \forall s \geq 2, \quad \Pi_{s+1} = 2 \cdot \sqrt{s} \cdot \sqrt{\Pi_s} \end{cases}$$

We define $A_{i,s}$ and $\mu_s(\ell)$ as follows:

▶ **Definition 17.**

$$A_{i,s} = \sum_{\ell=0}^{i-1} B_{\ell,s-1},$$

where

$$B_{\ell,s} = \sqrt{s \cdot \frac{\mu_{s+1}(\ell)}{N}} + 4\left(\frac{\ell}{N}\right)^{s/2} + \left(\sum_{r=2}^{s} \frac{\ell}{N^r}\right)^{1/2},$$

and

$$\mu_s(\ell) = \max\left\{\Pi_{s-1} \cdot (8e)^{\frac{2^{s-2}-1}{2^{s-3}}} \frac{\ell^{(2^{s-1}-1)/2^{s-2}}}{N^{(2^{s-2}-1)/2^{s-2}}}, 40 \cdot s^2 \cdot \Pi_{s-1} \cdot N^{1/2^s}\right\}.$$

We can now state the bound on $g_{i,k}$ that we will need to prove Theorem 14:

▶ **Lemma 18.** *For every $i \in \mathbb{N}$ and every $k \in \mathbb{N}$, we have that:*

$$g_{i,k} \leq \frac{A_{i,s+1}^k}{k!} + O\left(2^{-(s+1)^2 \cdot \Pi_s \cdot N^{1/2^{s+1}}}\right).$$

In order to prove Lemma 18, we first prove a bound on $A_{i,s}$.

▶ **Lemma 19.** $A_{i,s} \leq (8e)^{\frac{2^{s-2}-1}{2^{s-2}}} \frac{i^{(2^s-1)/2^{s-1}}}{N^{(2^{s-1}-1)/2^{s-1}}} \cdot \Pi_s + O\left(s^4 \cdot \Pi_s \cdot N^{-1/(2^s(2^s-2))}\right)$

At last we bound Π_s to conclude the analysis.

▶ **Proposition 20.** *We have for any $s \in \mathbb{N}$ that:*

$$\Pi_s \leq 4s$$

Proof. The statement is true for $s = 1, 2$. Assume it is true for $s \geq 2$. Then,

$$\Pi_{s+1} = 2\sqrt{s} \cdot \sqrt{\Pi_s} \leq 2\sqrt{s} \cdot \sqrt{4s} \leq 4(s+1). \qquad \blacktriangleleft$$

Finally, we can prove Theorem 14:

Proof of Theorem 14. From Lemma 19, we have:

$$A_{i,s} \leq (8e)^{\frac{2^{s-2}-1}{2^{s-2}}} \frac{i^{(2^s-1)/2^{s-1}}}{N^{(2^{s-1}-1)/2^{s-1}}} \cdot \Pi_s + O\left(s^4 \cdot \Pi_s \cdot N^{-1/(2^s(2^s-2))}\right).$$

Hence we can bound $g_{i,k}$ for any i, k, by:

$$\begin{aligned}
g_{i,k} &\leq \frac{A_{i,s+1}^k}{k!} + O\left(2^{-(s+1)^2\cdot\Pi_s\cdot N^{1/2^{s+1}}}\right) \\
&\leq \left(\frac{e\cdot A_{i,s+1}}{k}\right)^k + O\left(2^{-(s+1)^2\cdot\Pi_s\cdot N^{1/2^{s+1}}}\right) \\
&\leq \left(\frac{e}{k}(8e)^{\frac{2^{s-1}-1}{2^{s-1}}}\frac{i^{(2^{s+1}-1)/2^s}}{N^{(2^s-1)/2^s}}\cdot\Pi_{s+1} + \frac{e}{k}\cdot O\left((s+1)^4\Pi_{s+1}\cdot N^{-1/(2^{s+1}(2^{s+1}-2))}\right)\right)^k \\
&\quad + O\left(2^{-(s+1)^2\cdot\Pi_s\cdot N^{1/2^{s+1}}}\right) \\
&\leq \left(\frac{e}{k}\cdot(8e)^{\frac{2^{s-1}-1}{2^{s-1}}}\frac{i^{(2^{s+1}-1)/2^s}}{N^{(2^s-1)/2^s}}\cdot 4(s+1) + \frac{e}{k}\cdot O\left(4(s+1)^5\cdot N^{-1/(2^{s+1}(2^{s+1}-2))}\right)\right)^k \\
&\quad + O\left(2^{-4s(s+1)^2\cdot N^{1/2^{s+1}}}\right),
\end{aligned}$$

where the first inequality comes from Lemma 18, the third inequality comes from Lemma 19 and the last inequality comes from Proposition 20.

If $i = o\left((s+1)^{-\frac{2^s}{2^{s+1}-1}}\cdot k^{\frac{2^s}{2^{s+1}-1}}\cdot N^{\frac{2^s-1}{2^{s+1}-1}}\right)$, then $g_{i,k} = o(1)$. Hence if we want $g_{i,k}$ to be constant, i.e. not $o(1)$, we must have $i = \Omega\left(s^{-\frac{2^s}{2^{s+1}-1}}\cdot k^{\frac{2^s}{2^{s+1}-1}}\cdot N^{\frac{2^s-1}{2^{s+1}-1}}\right)$. ◀

4 The (r,k)-subset cover problem

In this section, we prove some upper and lower bounds on the (r,k)–SC problem. As far as we know, there is no quantum algorithm to find a (r,k)–SC problem, except for [14]'s algorithm when $k = r$, and for the harder problem of finding a k–RSC. We first prove a lower bound on the $(1,k)$–SC problem, then design new algorithms for finding a (r,k)–SC.

4.1 Lower bound on finding a $(1,k)$-subset cover

In this subsection, we will prove a lower bound on the $(1,k)$–SC problem. We are given k random functions $h_1,\ldots,h_k$ such that for $i\in[1,k]$, $h_i : \mathcal{X}\to\mathcal{Y}$. We write $N = |\mathcal{Y}|$ and for $x\in\mathcal{X}$, we write $H(x) = \{h_i(x) | i\in[1,k]\}$. The goal of this subsection is to prove the following theorem.

▶ **Theorem 21.** *Given k random functions $h_1,\ldots,h_k : \mathcal{X}\to\mathcal{Y}$ where $N = |\mathcal{Y}|$, a quantum algorithm needs to make $\Omega\left(C_k^{-1/5}\cdot N^{k/5}\right)$ queries to $h_1,\ldots,h_k$ to find one (1,k)–SC with constant probability, where $C_k = \sum_{j=2}^k \frac{k!}{(j-1)!}$.*

To prove Theorem 21, we introduce the problem of finding a *j-repetition* on $h_{i_1},\ldots,h_{i_j}$, that consists in finding an $x\in\mathcal{X}$ such that $h_{i_1}(x) = \cdots = h_{i_j}(x)$. More formally, we define the following database property:

▶ **Definition 22.**

$$\forall \ell, j, P_{\ell,j}^{rep} = \left\{ D\in\mathcal{D} \;\middle|\; \begin{array}{l} \exists x_1, x_2, \ldots, x_\ell, \forall i, \forall 1\leq \ell\leq j, h_1(x_i) = h_\ell(x_i) \\ \forall i\neq p, x_i\neq x_p \end{array} \right\}.$$

Note that we define the property only for ℓ distinct j-repetition on $h_1, \ldots, h_j$, because by symmetry, the probability of finding a j-repetition on $h_1, \ldots, h_j$ is the same as finding a j-repetition on $h_{i_1}, \ldots, h_{i_\ell}$.

We also define:

1. $\widetilde{f}^{rep}_{i,\ell,j}$ as the amplitude of the databases D containing *at least* ℓ distinct j-repetitions on $h_1, \ldots, h_j$ after i quantum queries.
2. $f^{rep}_{i,\ell,j}$ as the amplitude of the databases D containing *exactly* ℓ distinct j-repetitions on $h_1, \ldots, h_j$ after i quantum queries.
3. $g_{i,k}$ as the amplitude of the databases D containing at least one $(1,k)$-SC after i quantum queries.

More formally, let $|\psi_i\rangle$ be the state just after the i^{th} query to the oracle, then $\widetilde{f}^{rep}_{i,\ell,j} = \left|P^{rep}_{\ell,j}|\psi_i\rangle\right|$, $f^{rep}_{i,\ell,j} = \left|P^{rep}_{\ell,j}\neg P^{rep}_{\ell+1,j}|\psi_i\rangle\right|$, and $g_{i,k} = \left|P^{SC}_{(1,k)}|\psi_i\rangle\right|$.

Our goal is to bound $g_{i,k}$ and for that we will bound $\widetilde{f}^{rep}_{i,\ell,j}$.

▶ **Lemma 23.** *For all $i, \ell, j \in \mathbb{N}$, we have that:*

$$\widetilde{f}^{rep}_{i,\ell,j} \leq \left(\frac{4e \cdot i}{\ell \cdot N^{\frac{j-1}{2}}}\right)^{\ell}.$$

We now bound the amplitude $g_{i,k}$ with an inductive formula, as for the RSC problem.

▶ **Lemma 24.** *For all $i \in \mathbb{N}$ and $k \in \mathbb{N}$, we have that:*

$$g_{i,k} \leq g_{i-1,k} + 4\left(k^k \frac{i-1}{N^k}\right)^{1/2} + \left(\sum_{j=2}^{k}\sum_{\ell\geq 0} \frac{\ell}{N^{k+1-j}} \cdot \frac{k!}{(j-1)!} {f^{rep}_{i-1,\ell,j}}^2\right)^{1/2}.$$

We now bound $g_{i,k}$ in the following lemma.

▶ **Lemma 25.** *For every $i \in \mathbb{N}$ and $k \in \mathbb{N}$, we have that:*

$$g_{i,k} \leq 4k^{k/2} \cdot \frac{i^{3/2}}{N^{k/2}} + \sqrt{\sum_{j=2}^{k} \frac{k!}{(j-1)!}} \cdot \frac{4e \cdot i^{5/2}}{N^{k/2}}.$$

We can now prove Theorem 21.

Proof of Theorem 21. From Lemma 25, we have that:

$$g_{i,k} \leq 4k^{k/2} \cdot \frac{i^{3/2}}{N^{k/2}} + \sqrt{\sum_{j=2}^{k} \frac{k!}{(j-1)!}} \cdot \frac{4e \cdot i^{5/2}}{N^{k/2}}.$$

Writing $C_k = \sum_{j=2}^{k} \frac{k!}{(j-1)!}$, this rewrites as:

$$g_{i,k} \leq 4k^{k/2} \cdot \frac{i^{3/2}}{N^{k/2}} + \sqrt{C_k} \cdot \frac{4e \cdot i^{5/2}}{N^{k/2}}.$$

If $i = o\left(C_k^{-1/5} \cdot N^{k/5}\right)$, then $g_{i,k} = o(1)$. Hence if we want $g_{i,k}$ to be constant, i.e. not $o(1)$, we must have $i = \Omega\left(C_k^{-1/5} \cdot N^{k/5}\right)$. ◀

4.2 Algorithm for finding a $(1, k)$-subset cover

We now describe an algorithm that finds a $(1, k)$–SC, assuming $|\mathcal{X}| = |\mathcal{Y}|^k = N^k$. We first notice that an algorithm that finds a collision on H also finds a $(1, k)$–SC in an expected $O(N^{k/3})$ number of queries. We show now that there is a more efficient algorithm, as stated in the following theorem:

▶ **Theorem 26.** *There exists a quantum algorithm that finds a $(1, k)$–SC in expected $O\left(N^{k/4}\right)$ quantum queries if k is even, and $O(N^{k/4+1/12})$ if k is odd.*

To prove this theorem, we describe the following algorithm (which takes as parameters j and t, whose values will be chosen later):

▶ **Algorithm 27.** *Input: $j \in \{2, \ldots, k\}$ and $t \in \mathbb{N}$.*

1. *Define $F_1 : \mathcal{X} \to \{0, 1\}$ as follows:*

$$F_1(x) = \begin{cases} 1, & \text{if } h_1(x) = h_2(x) = \cdots = h_j(x) \\ 0, & \text{otherwise.} \end{cases}$$

   *(Note that an element $x \in \mathcal{X}$ such that $F_1(x) = 1$ is a j-*repetition*.)*
2. *Execute Grover's algorithm t times on F_1 to find t distinct j-repetitions in H. Let $T = \{x_1, \ldots, x_t\}$ be the set of these j-repetitions.*
3. *Define $F_2 : \mathcal{X} \to \{0, 1\}$ as follows:*

$$F_2(x) = \begin{cases} 1, & \text{if there exists } x_0 \in T \text{ such that } h_1(x) = h_1(x_0) \\ & \text{and for } 1 \leq m \leq k - j, h_{m+1}(x) = h_{j+m}(x_0) \\ 0, & \text{otherwise.} \end{cases}$$

4. *Execute Grover's algorithm to find an x such that $F_2(x) = 1$*
5. *Find x_0 in T corresponding to x, and output (x, x_0).*

▶ **Lemma 28.** *Algorithm 27 makes an expected number of $O\left(N^{(2k-j+1)/6}\right)$ queries to the oracle when $j \leq \frac{k+2}{2}$ for $t = N^{(k-2j+2)/3}$.*

We now prove Theorem 26

Proof of Theorem 26. From Lemma 28, the complexity of Algorithm 27 is $O(N^{(2k-j+1)/6})$ when $j \leq \frac{k+2}{2}$.

- If k is even, then we pick $j = \frac{k+2}{2}$ to reach a complexity of $O(N^{k/4})$.
- If k is odd, then we pick $j = \frac{k+1}{2}$ to reach a complexity of $O(N^{k/4+1/12})$.

Note that if $j > \frac{k+1}{2}$, then the second step of the algorithm is expected to make at least $O\left(N^{\frac{k+1}{4}}\right)$ quantum queries, which is worse than $O(N^{k/4+1/12})$. ◀

▶ Remark. Note that we do not reach the lower bound of Theorem 21, and it would be interesting to see if the gap can be further reduced by either improving our lower bounds or designing a more efficient algorithm.

4.3 Algorithm for finding a (r, k)-subset cover

In this section, we describe an algorithm for solving the (r, k)–SC problem. We consider the case where $|\mathcal{X}| = |r \cdot \mathcal{Y}|^k = r^k \cdot N^k$. The result is stated as follows:

▶ **Theorem 29.** *There exists a quantum algorithm that finds a (r, k)–SC in $O\left(N^{k/(2+2r)}\right)$ quantum queries to H, if k is divisible by $r+1$, and $O\left(N^{k/(2+2r)+1/2}\right)$ otherwise.*

The idea of the algorithm is essentially the same as Algorithm 27 of Section 4.2:

1. we first find t distinct $(r-1, k')$–SC for some integers t and k';
2. we then find the (r, k)–SC.

The first step is done recursively, using the algorithm defined for lower values of k' and $r-1$. The second step uses Grover's algorithm. The algorithm can be defined for any value of k' and t, and we pick them to optimize the complexity.

More formally, we define the algorithm recursively. Assume that we have an algorithm that can output a $(r-1, k')$–SC in $O\left(N^{k'/2r}\right)$ queries, for any $k' < k$ such that k' is divisible by r. Then, we can find a (r, k)–SC as follows:

▶ **Algorithm 30.** *Input: $t \in \mathbb{N}$, $k' \in \mathbb{N}$.*

1. *Execute the $(r-1, k')$–SC algorithm t times to find t distinct $(r-1, k')$–SC in H. Let $T = \{(x_{1,0}, x_{1,1}, \ldots, x_{1,r-1}), \ldots, (x_{t,0}, x_{t,1}, \ldots, x_{t,r-1})\}$ be the set of these $(r-1, k')$–SC.*
2. *Define $F : \mathcal{X} \to \{0, 1\}$ as follows:*

$$F(x) = \begin{cases} 1, & \text{if there exists } (x_{i,0}, x_{i,1}, \ldots, x_{i,r-1}) \in T \text{ such that} \\ & \forall 1 \leq m \leq k - k', h_m(x) = h_{k'+m}(x_{i,0}), \\ 0, & \text{otherwise.} \end{cases}$$

3. *Execute Grover's algorithm to find an x such that $F(x) = 1$*
4. *Find $(x_{i,0}, x_{i,1}, \ldots, x_{i,r-1})$ in T and output $(x_{i,0}, x_{i,1}, \ldots, x_{i,r-1}, x)$.*

▶ **Lemma 31.** *Algorithm 30 makes an expected number of $O\left(N^{k/(2+2r)}\right)$ queries to the oracle, when k is divisible by r, and $O\left(N^{k/(2+2r)+1/2}\right)$ otherwise.*

The proof of Theorem 29 follow directly from Lemma 31.

References

1 Jean-Philippe Aumasson and Guillaume Endignoux. Clarifying the subset-resilience problem. Cryptology ePrint Archive, Report 2017/909, 2017. URL: `https://eprint.iacr.org/2017/909`.

2 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 Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 368–397. Springer, Heidelberg, April 2015. `doi:10.1007/978-3-662-46800-5_15`.

3 Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS$^+$ signature framework. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2129–2146. ACM Press, November 2019. `doi:10.1145/3319535.3363229`.

4 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. `doi:10.1007/978-3-642-25385-0_3`.

5 Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505, June 1998. doi:10.1002/(sici)1521-3978(199806)46:4/5<493::aid-prop493>3.0.co;2-p.

6 Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN '98: Theoretical Informatics, Third Latin American Symposium, Campinas, Brazil, April, 20-24, 1998, Proceedings*, volume 1380 of *Lecture Notes in Computer Science*, pages 163–169. Springer, 1998. doi:10.1007/BFb0054319.

7 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. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 598–629. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77886-6_21.

8 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th ACM STOC*, pages 212–219. ACM Press, May 1996. doi:10.1145/237814.237866.

9 L. Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

10 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. doi:10.1007/978-3-030-17659-4_7.

11 Leonid Reyzin and Natan Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In Lynn Margaret Batten and Jennifer Seberry, editors, *ACISP 02*, volume 2384 of *LNCS*, pages 144–153. Springer, Heidelberg, July 2002. doi:10.1007/3-540-45450-0_11.

12 Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997. doi:10.1137/s0097539795293172.

13 Takashi Yamakawa and Mark Zhandry. Classical vs quantum random oracles. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 568–597. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77886-6_20.

14 Quan Yuan, Mehdi Tibouchi, and Masayuki Abe. On subset-resilient hash function families. *Designs, Codes and Cryptography*, 90, March 2022. doi:10.1007/s10623-022-01008-4.

15 Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 239–268. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7_9.

Distributed Shuffling in Adversarial Environments

Kasper Green Larsen
Aarhus University, Denmark

Maciej Obremski
National University of Singapore, Singapore

Mark Simkin
Ethereum Foundation, Aarhus, Denmark

Abstract

We study mix-nets in the context of cryptocurrencies. Here we have many computationally weak shufflers that speak one after another and want to joinlty shuffle a list of ciphertexts $(c_1, \ldots, c_n)$. Each shuffler can only permute $k << n$ ciphertexts at a time. An adversary $\mathcal{A}$ can track some of the ciphertexts and adaptively corrupt some of the shufflers.

We present a simple protocol for shuffling the list of ciphertexts efficiently. The main technical contribution of this work is to prove that our simple shuffling strategy does indeed provide good anonymity guarantees and at the same time terminates quickly.

Our shuffling algorithm provides a strict improvement over the current shuffling strategy in Ethereum's block proposer elections. Our algorithm is secure against a stronger adversary, provides provable security guarantees, and is comparably in efficiency to the current approach.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Distributed Computing, Shuffling

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.10

Related Version *Full Version*: https://eprint.iacr.org/2022/560

Funding *Kasper Green Larsen*: Supported by Independent Research Fund Denmark (DFF) Sapere Aude Research Leader grant No 9064-00068B.
Maciej Obremski: Funded by MOE2019-T2-1-145 Foundations of quantum-safe cryptography.

1 Introduction

Shuffling the elements of a long vector efficiently is a problem that appears in various shapes and forms throughout many different domains of cryptography. In most applications, the vector entries are either commitments or ciphertexts and each position in the vector is associated with a corresponding identity. The process of shuffling the vector produces a new vector that contains the same multi-set of committed or encrypted values, but hides which value is associated to which identity. In anonymous communication systems [11, 26, 20], for instance, a set of senders would each like to communicate one message to a set of receivers without revealing who is talking to who. In electronic voting [26, 20, 23], we have a long list of votes and we would like to determine the election outcome without revealing who voted for who. In the domain of cryptocurrencies [21, 6], we have multiple payers, who would like to transfer money to multiple payees without revealing who is paying who.

A popular approach for achieving anonymity in the above applications are mix-nets [11]. Here, we assume the existence of one or more shufflers that shuffle the input vector one after another. If only one shuffler was honest, then even an adversary that corrupts all other shufflers cannot tell which entry in the input vector belongs to which entry in the output vector. From a security perspective this approach is great, but unfortunately such strong


© Kasper Green Larsen, Maciej Obremski, and Mark Simkin;
licensed under Creative Commons License CC-BY 4.0

anonymity guarantees do not come for free. The required memory and the computational overhead of each shuffler grows linearly in the length of the vector that should be shuffled. In applications like electronic voting, the length of a vector of votes could easily be in the millions, which places a significant memory burden on each shuffler. In addition, shufflers often need to provide computationally expensive zero-knowledge proofs attesting the correctness of their performed shuffle [26, 15, 23, 2] to show that no values in the vector have been changed by them. These high costs make mix-nets unsuitable for applications, where shuffling needs to terminate in a timely fashion and where the shufflers are restricted in terms of memory or computational power.

1.1 Our Contribution

In this work, we study mix-nets in the context of cryptocurrencies. Here we have many shufflers, but all of them are computationally weak, in the sense that they can only read and shuffle k entries in a vector of length n, where k is potentially much smaller than n. Initially, a vector of ciphertexts $(c_1, \ldots, c_n)$ is written on a public bulletin board, accessible to all. The shufflers speak one after another and each shuffler chooses k entries, re-randomizes, and permutes them. We assume that shuffling takes place in the presence of an adversary $\mathcal{A}$. At the start of the protocol, $\mathcal{A}$ is allowed to corrupt a subset of indices $I \subset \{1, \ldots, n\}$ with $|I| \leq \alpha$ and can track all ciphertexts c_i for $i \in I$ throughout the shuffling process. Additionally, the adversary can adaptively corrupt up to β shufflers throughout the execution. The goal of the shuffling protocol is to hide the output location of the uncorrupted entries in the input vector from the adversary. In terms of efficiency, we would like to minimize the number of shuffles of size k that need to be performed.

We present a very simple shuffling mechanism, where each shuffler picks k uniformly random entries and permutes them. The main technical contribution of this work is a upper bound that shows that this shuffling process terminates quickly and provides good anonymity guarantees. The following informal theorem is a corollary of our main theorem.

▶ **Theorem 1** (Informal). *Let $(c_1, \ldots, c_n)$ be a vector of ciphertexts. Let $\mathcal{A}$ be a PPT adversary that tracks $\alpha = C \cdot n$ ciphertexts, where C is a constant, and adaptively corrupts β shufflers adaptively. If each shuffler randomly permutes $k \in \Omega(\ln^2(n))$ random ciphertexts, then shuffling terminates in $\mathcal{O}(n/k \cdot \ln(n) + \beta)$ steps with a constant success probability.*

To underline the practicality of our distributed shuffling protocol, we implemented our solution and we provide benchmarks, which show that shuffling is not only asymptotically, but also practically efficient.

1.2 Applications

1.2.1 Single Secret Leader Elections

In the single secret leader election (SSLE) problem, introduced by Boneh et al. [4], we have a public bulletin board and n parties that would like to elect exactly one leader among them. The leader should be fairly chosen, in the sense that each party should have a roughly equal probability of becoming the leader. Additionally, the leader should remain hidden until they decide to reveal themselves.

Boneh et al. present three solutions to this problem. The first two solutions are based on indistinguishability obfuscation [16] and threshold fully homomorphic encryption [5] respectively. Both of these solutions are theoretically interesting, but concretely too inefficient to be useful in a practical setting.

The third presented solution is based on a distributed shuffling protocol. In their protocol, each of the n participants publishes a commitment c_i only they can open on the bulletin board. Then, each participant's commitment is assigned a bucket of size $\sqrt{n}$, which is shuffled once. The protocol guarantees that each entry in the output vector could come from $\sqrt{n}$ possible locations in the input vector. The authors mention that stronger anonymity guarantees may be achieved using Håstad's square shuffle [17, 18], but leave the analysis of such an approach as an explicit open question. Håstad's square shuffle is an algorithm that shuffles a vector of length n using shuffles of size $k = \sqrt{n}$ in a benign setting. The algorithm itself does not provide any security guarantees in a setting, where an adversary may track some of the commitments or where the adversary can adaptively prevent some of the shuffles from happening. Even worse, it is straightforward to design an adaptive adversary with a relatively small budget of allowed corruptions that can prevent certain commitments from being shuffled at all.

Using our distributed shuffling protocol, which works for various choices of k beyond just $k = \sqrt{n}$, we obtain a new SSLE protocol that is secure against adaptive adversaries, where the elected leader is hidden not only among $\sqrt{n}$ other participants, but instead among close to all n of them.

Ethereum Block Proposer Elections

A variant of the SSLE problem has recently been considered in the context of the Ethereum blockchain, where we are not only interested in electing one, but rather a ordered list of γ leaders. Two real-world efficiency constraints are important to point out here. Every shufflers needs to speak in a timely manner, yet at the same time they need to provide zero-knowledge proofs attesting the correctness of their performed shuffle. These two constraints mean that no shuffler has enough time to permute the full vector at once.

The currently proposed protocol [14] for potential deployment in Ethereum is effectively a direct implementation of Håstad's square shuffle along with some other minor steps that are not relevant for the discussion here. The protocol is heuristically claimed to be secure against non-adaptive corruptions. However, the protocol is only discussed in an informal model and no security proofs are provided. Similarly to the plain square shuffle of Håstad, the proposed construction is not secure against an adaptive adversary that may adaptively target specific shufflers during the protocol execution. Especially in the context of a blockchain, where shufflers are known entities, and an adversary that may have the ability to target some of them adaptively, we believe that a stronger adaptive security notion and provable security guarantees are of crucial importance.

Our distributed shuffling protocol, which provides *provable* security guarantees against a stronger adversary, can be used as a direct replacement of the current proposal. Our experimental results show that the efficiency of our protocol is comparable to the current proposal of Ethereum. We discuss this application in more depth in Section 5.

1.3 Related Works

Multiple research domains are related to our work here.

1.3.1 Benign Shuffling

A series of existing works [27, 12, 1, 17, 18, 25, 22] has studied the question of how long it takes to shuffle the elements of a vector via either smaller or restricted shuffling operations. There, the problem is studied in a benign setting and it is unclear what security can be achieved in the presence of an adversarial entity.

Conceptually, the work of Diaconis and Shahshahani [12], which considers shuffling a deck of cards by repeatedly picking two random cards and switching them, is closest to our algorithm. In their work, the authors are interested in determining the required number of rounds until the resulting permutation looks close to uniformly random to a distinguisher that does not see which cards were swapped. In contrast to their work, we want to determine the required number of shuffles of size k until any uncorrupted card is at an unpredictable location, even if the adversary gets to see all subsets of k elements that were shuffled in the protocol.

1.3.2 Single Secret Leader Elections

After the first three initial approaches for solving the SSLE problem by Boneh et al. [4], an alternative solution based on functional encryption was proposed by Catalano, Fiore, and Giunta [9]. Their solution has many attractive properties, but requires an expensive initial setup to be performed between the parties participating in the elections. In situations where elections are performed periodically and many participants may join or leave between elections, the setup needs to be repeatedly renewed. Shuffling based solutions on the other hand, gracefully deal with joining and leaving participants, since no setup is required.

In a recent independent work by Catalano, Fiore, and Giunta [10], the authors also study SSLE in the presence of an adaptive adversary. Their work focuses on providing a full formalization of the SSLE problem in the universal composability framework [8]. The authors provide a solution based on shuffling that requires each shuffler to speak multiple times and to permute the full vector. In contrast to their work, we focus on distributed shuffling protocols, where shufflers have bounded memory and only speak once. We believe that our model is closer to how shufflers would actually operate in a real blockchain like Ethereum.

2 Preliminaries

2.1 Notation

We write $[n]$ to denote the set $\{1, \ldots, n\}$. We denote the computational security parameter by λ. For a set X, we write $x \leftarrow X$ to denote the process of sampling a uniformly random element x from X. For a randomized algorithm A we write $A(x; r)$ to explicitly specify the random tape r when A is executed on some input x. Otherwise, we write $A(x)$ and simply assume that r is implicitly chosen uniformly at random. We write $\perp \leftarrow A(x)$ to denote that an algorithm A failed to produce an output. We write $A^{\mathcal{O}(\cdot)}$ to denote algorithm A with oracle access to algorithm $\mathcal{O}$.

2.2 Encryption Schemes

We define the minimal security properties that are sufficient for proving our shuffling algorithm secure in our model. For the remainder of this work, we focus on shuffling a vector of ciphertexts, but all of our results easily carry over to commitments.

▶ **Definition 2.** *A public-key encryption scheme* $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a comprised of the following algorithms:*

$(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)$*: The key generation algorithm takes the security parameter* 1^λ *as input and outputs a public encryption key* ek *and a secret decryption key* dk*.*

$c \leftarrow \mathbf{Enc}(\mathsf{ek}, m)$*: The encryption algorithm takes the key* ek *and a message* m *as input and outputs a ciphertext* c*.*

$m \leftarrow \mathbf{Dec}(\mathsf{dk}, c)$*: The decryption algorithm takes key* dk *and ciphertext* c *as input and outputs message* m*.*

▶ **Definition 3** (Semantic Security)**.** *We say* $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is semantically secure, if for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{r}(\mathsf{ek},\mathsf{dk}) \leftarrow \mathsf{Gen}(1^\lambda)\\ (m_0,m_1) \leftarrow \mathcal{A}(\mathsf{ek})\\ b \leftarrow \{0,1\}\\ b^* \leftarrow \mathcal{A}(\mathsf{Enc}(\mathsf{ek},m_b))\end{array} : b = b^*\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the probability is taken over the uniform random coins of the adversary, the key generation, and the encryption algorithm.

The input to our distributed shuffling algorithm will be a vector of ciphertexts, where each one is encrypted under a different public key. To be able to meaningfully shuffle this vector, we require that ciphertexts under different keys are indistinguishable from each other. This notion of key privacy was first considered by Bellare et al. [3]. We use slightly weaker formalization of key privacy that is sufficient for our purposes.

▶ **Definition 4** (Key Privacy)**.** *We say a semantically secure encryption scheme* $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is key private, if for any PPT adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{r}(\mathsf{ek}_0,\mathsf{dk}_0) \leftarrow \mathsf{Gen}(1^\lambda)\\ (\mathsf{ek}_1,\mathsf{dk}_1) \leftarrow \mathsf{Gen}(1^\lambda)\\ m \leftarrow \mathcal{A}(\mathsf{ek}_0,\mathsf{ek}_1)\\ b \leftarrow \{0,1\}\\ b^* \leftarrow \mathcal{A}(\mathsf{Enc}(\mathsf{ek}_b,m))\end{array} : b = b^*\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the probability is taken over the uniform random coins of the adversary, the key generation, and the encryption algorithms.

One possible instantiation of an encryption scheme with the desired properties is the ElGamal cryptosystem [13].

2.3 Local Shuffling Algorithms

The focus of our work lies in answering how to shuffle the elements of a vector of length n through the use of shuffle operations that permute k many elements at a time. To abstract away the concrete shuffling procedure that is used by any shuffler locally, we define an idealized function `Shuffle` that takes k ciphertexts as input and produces a fresh list of k ciphertexts that commit to the same multi-set of messages. In practice, the local shuffling procedure would be realized by combining a re-randomizable encryption or commitment scheme with an appropriate non-interactive zero-knowledge proof that attests the correctness of the performed shuffle. If the list were to contain ElGamal ciphertexts, then efficient shuffling arguments of Bayer and Groth [2] or Bünz et al. [7] could be used. If the list were to contain pedersen commitments [24], then efficient shuffling arguments of Bünz et al. [7] or Hoffmann et al. [19] could be used.

3 Model

In this section, we define the formal model within which we will present and analyze our distributed shuffling protocol. In our setting, we have a public bulletin board, where parties can post authenticated messages that are visible to all other parties. The messages are authenticated in the sense that each message on the bulletin board can be traced to its sender. In the beginning, the only thing written on the message board are ciphertxts $c_1, \ldots, c_n$, where $c_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, m_i)$ for some m_i for $i \in [n]$. The parties $P_1, \ldots, P_T$, also known as the shufflers, speak one after another by posting messages on the bulletin board. To compute their messages, each shuffler reads at most k ciphertexts, locally shuffles them using the `Shuffle` procedure, and writes the permuted vector of k ciphertexts (along with possibly auxiliary information) back on the bulletin board. At the end of the protocol execution, after all T shufflers have spoken, ciphertexts $\tilde{c}_1, \ldots, \tilde{c}_n$, which encrypt the same multiset as the input vector, should be written on the bulletin board. We call such a protocol Π a (T, n, k)-shuffle.

3.1 Corruptions

The shuffling protocol runs in the presence of adversarial behavior. The PPT adversary $\mathcal{A}$ can see who posts which messages on the bulletin board and in addition can perform two types of corruptions. At the beginning of a protocol execution, the adversary is corrupting α ciphertexts. For each corrupted ciphertext, the adversary learns the corresponding decryption key and can thus "track their positions" throughout the shuffling procedure. Additionally, the adversary is allowed to corrupt β shufflers in a fully adaptive manner, meaning that at the start of every round $i \in [T]$, the adversary is allowed to decide whether or not to corrupt shuffler P_i on the fly, as long as the total number of corrupted shufflers is at most β.

A corrupt shuffler can perform an arbitrary chosen, but valid permutation on an arbitrary choice of at most k ciphertexts. In principle, we do not need to assume that the adversary honestly permutes k ciphertexts or that she would even honestly report, which ciphertexts she touched. Both of these issues are easily resolved via standard non-interactive zero-knowledge arguments attesting the correctness of the shuffle. To avoid explicitly talking about such arguments, we simply restrict the adversary in her behaviour.

3.2 Definitions

For a shuffling protocol Π with input vector $\vec{c}$ and an adversary $\mathcal{A}$, we write $(z, \pi) \leftarrow \langle \Pi(\vec{c}; r), \mathcal{A}(\vec{c}; \tilde{r}) \rangle$ to denote the execution Π with random coins r in the presence of $\mathcal{A}$ with random coins $\tilde{r}$, where z is the adversary's output and π is the permutation on domain $[n]$, i.e. between the input and output ciphertexts' values. If at the end of a protocol execution the values inside the output ciphertexts are not a permutation of the input ciphertexts' values, then we write $\pi = \perp$.

▶ **Definition 5** (Correctness). *We say that an (T, n, k)-shuffle Π is correct in the presence of an adversary $\mathcal{A}$, if*

$$\Pr\left[\begin{array}{r} (\mathsf{ek}_i, \mathsf{dk}_i) \leftarrow \mathsf{Gen}(1^\lambda)\ \forall i \in [n] \\ c_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, i)\ \forall i \in [n] \\ \vec{c} := (c_1, \ldots, c_n) \\ (z, \pi) \leftarrow \langle \Pi(\vec{c}; r), \mathcal{A}(\vec{c}; \tilde{r}) \rangle \end{array} : \pi \neq \perp \right] = 1,$$

where the probability is taken over the random coins r and $\tilde{r}$.

$\Pi(c_1, \ldots, c_n)$

for $t \in [T]$:

S_t picks random $\{i_1, \ldots, i_k\} \subset [n]$

S_t computes $(\tilde{c}_{i_1}, \ldots, \tilde{c}_{i_k}) \leftarrow \texttt{Shuffle}(c_{i_1}, \ldots, c_{i_k})$

S_t publishes $(\tilde{c}_{i_1}, \ldots, \tilde{c}_{i_k})$

Figure 1 Distributed shuffling protocol.

▶ **Definition 6** (Security). *Let $\mathcal{A}$ be a PPT adversary that corrupts at most β shufflers. We say that an (T, n, k)-shuffle Π is (ϵ, δ)-secure in the presence of an (α, β)-adversary $\mathcal{A}$, if for all $I \subset [n]$ with $|I| \leq \alpha$, it holds that with probability at least $1 - \delta$ we have*

$$\Pr\left[\begin{array}{c} (\mathsf{ek}_i, \mathsf{dk}_i) \leftarrow \mathsf{Gen}(1^\lambda)\ \forall i \in [n] \\ c_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, i)\ \forall i \in [n] \\ \vec{c} := (c_1, \ldots, c_n) : \pi(i) = j \wedge i \notin I \\ \vec{dk} := \{\mathsf{dk}_i \mid i \in I\} \\ ((i,j), \pi) \leftarrow \langle \Pi(\vec{c}; r), \mathcal{A}(\vec{c}, \vec{dk}; \tilde{r}) \rangle \end{array}\right] \leq \epsilon,$$

where the randomness is taken over the random coins r and $\tilde{r}$.

In the definition above, there exists a naive attacking strategy. The adversary could just guess a random pair (i, j) of indices with $i, j \notin I$, which means that the best security we can hope for is $\epsilon = 1/(n - |I|)$. If on the other hand, we achieve $\epsilon \leq C/(n - |I|)$ for some constant C, then this translates into the intuitive guarantee that any element in the output vector comes from at least $(n - |I|)/C$ possible locations in the input vector.

4 Construction

Our distributed shuffling protocol is conceptually very simple. Each round, a shuffler picks a random subset of k ciphertexts and permutes those. The main technical challenge is to prove that after a not too large number of rounds, this procedure will shuffle the input vector sufficiently well.

We note that all shufflers in our protocol act independently and do not coordinate who will shuffle which entries in the vector. For this reason, even a powerful adaptive adversary cannot do anything better than corrupting an arbitrary subset of β shufflers. Thus, the question of how big the number of rounds T has to be set to tolerate an adversary that corrupts β shufflers, effectively reduces to the question of how well the input vector is shuffled in $T - \beta$ rounds in the presence of an adversary that can corrupt no shufflers at all.

The formal protocol description is given in Figure 1 and we prove the following theorem.

▶ **Theorem 7.** *Let $\mathcal{A}$ be a PPT adversary that corrupts at most β shufflers. Let $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a semantically secure and key private encryption scheme. For any $0 < \delta < 1/3$, if $T \geq 20(n/k)\ln(n/\delta) + \beta$ and $k \geq 256\ln^2(n/\delta)(1 - \alpha/n)^{-2}$, then the protocol in Figure 1 is a (ϵ, δ)-secure (T, n, k)-shuffle in the presence of a (α, β)-adversary, where $\epsilon = 2/(n - \alpha) + \mathsf{negl}(\lambda)$.*

Proof. Let $I \subset [n]$ with $|I| = \alpha$ be an arbitrary, but fixed subset of indices belonging to ciphertexts that are corrupted by the adversary. Let $H := [n] \setminus I$ be the indices of uncorrupted ciphertexts. Let hybrid hybrid_0 be the security game as stated in Definition 6. We consider hybrid hybrid_1, which is identical to hybrid_0 with the exception that $(\mathsf{ek}_i, \mathsf{dk}_i) := (\mathsf{ek}_1, \mathsf{dk}_1)$ for all $i \in H$. Indistinguishability of hybrid_0 and hybrid_1 follows from the key privacy of the underlying encryption scheme. In hybrid_2 we set $c_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, 1)$ for all $i \in H$. Indistinguishability of hybrid_1 and hybrid_2 follows from the semantic security of the underlying encryption scheme. At this point, we observe that in each invocation of `Shuffle` by an honest shuffler the adversary learns *nothing* about how the honest ciphertexts were permuted. To see this, we note that each honest ciphertext returned by `Shuffle` is identically distributed, encrypted under the same key, encrypting the same message.

Next, we observe that an adaptive adversary can not do anything better than corrupting an arbitrary set of β shufflers. To see this, observe that each shuffler chooses its subset of k ciphertexts independently, thus the distribution of permutations between input and output vector that is produced by our protocol is independent of which shufflers are corrupted by $\mathcal{A}$. For the remainder of the proof we determine the number T_H of honest shuffles that need to be performed, such that every ciphertext's location is hidden sufficiently well. Our protocol can then be run for $T \geq T_H + \beta$ rounds to be secure against β corrupt shufflers.

We now view the ciphertexts as a set of n cups, denoted $c_1, \ldots, c_n$. Of these n cups, the last α are *idle* and the first $n - \alpha$ are *active*. The cups may contain a non-negative amount of water.

Let $k \geq 2$. A T_H step k-way *mixing* consists of repeatedly selecting k cups uniformly at random (without replacement). If B denotes the set of selected cups, we then gather all water in active cups $c_i \in B$. The collected water is then distributed evenly among the active cups. This process is repeated for T_H steps. We call one such step a *mixing step*.

We say that a T_H step k-way mixing is successful if, for any c_i among the active cups, if we had placed 1 unit of water in c_i and 0 in all remaining cups, then at the end of the mixing, no cup contains more than $2/(n - \alpha)$ water. That is, regardless of which active cup we choose put 1 unit of water in, at the end of shuffling, no cup contains more than a factor 2 more water than if we had distributed all water uniformly among active cups.

▶ **Lemma 8.** *For any $0 < \delta < 1/3$, if $T_H \geq 20(n/k)\ln(n/\delta)$ and $k \geq 256\ln^2(n/\delta)(1-\alpha/n)^{-2}$, then a T_H step k-way mixing with α idle cups is successful with probability at least $1 - \delta$.*

Observe first that if a T_H step k-way mixing is successful, then if we perform another mixing step, the mixing remains successful. This is because the maximum amount of water in a cup cannot increase in a mixing step. Hence we prove the lemma for $T_H = 20(n/k)\ln(n/\delta)$ and note that it also implies the result for larger T_H.

In our proof, we first show that if c_1 has 1 unit of water and the remaining have 0, then with probability at least $1 - \delta/n$, it holds that after T steps that there is no cup with more than $2/(n - \alpha)$ units of water. A union bound over all $n - \alpha$ active cups that may contain the initial 1 unit of water completes the proof.

So consider the setup where c_1 has 1 unit of water and the remaining have 0. We define two undesirable events, such that if none of these events occur, the mixing is successful. To define the first of these events, let B_t be the indices of the cups selected for mixing in the t'th step.

Consider an execution of a T_H step k-way mixing. A *back-tracking from cup c_i* is a sequence of indices $i_1, \ldots, i_r \in [k]$, possibly with repetitions, such that the following holds: Initialize $b = i$, $j = 0$ and $t = T_H$. Repeat until $t = 0$: If cup c_b was selected for mixing in

step t, increment j and set b to be the index of the i_j'th cup in B_t (for some arbitrary but fixed ordering on cups). Decrement t and repeat (regardless of whether cup c_b was selected for mixing in step t).

A back-tracking thus specifies a "path" that starts with c_i and as we go backwards through the T_H mixing steps, whenever the current cup c_b is selected for mixing, the path proceeds to trace the next cup in the list. When j reaches r in the back-tracking, it must be the case that the currently traced cup c_b is not selected in any further mixing steps while decrementing t. The first undesirable event says that there is a short back-tracking:

- Event E_1: There is a back-tracking $i_1, \ldots, i_r$ with $r \leq 4 \lg_k n$.

To define the second event, let w_i^t denote the amount of water in cup c_i after t steps of mixing. We have $w_1^0 = 1$ and $w_i^0 = 0$ for $i \neq 1$. Also, let $A_t \subseteq B_t$ denote the indices of the active cups among B_t. Finally, let $W_t = \sum_{i \in A_t} w_i^{t-1}/|A_t|$ denote average amount of water in the cups selected in step t. By definition, we have $w_i^t = W_t$ for every $i \in A_t$. With these definitions in place, the second undesirable events says that we in some step perform a mixing that results in much water on average, yet none of the involved cups had significantly more water than the average:

- Event E_2: There is a step t where $W_t \geq 2/(n-\alpha)$ but $\max_{i \in A_t} w_i^{t-1} \leq k^{1/4} W_t$.

Success when none of E_1 and E_2 occur. We first show that a T_H step mixing is successful when none of the events E_1 and E_2 occur. For this, consider an unsuccessful mixing where E_2 did not occur. We claim that this implies that E_1 occurred. We thus need to show that an unsuccessful mix together with the fact that E_2 does not occur implies a short back-tracking. For this, let c_{i^*} be a cup such that $w_{i^*}^T > 2/(n-\alpha)$. Such a cup exists since the mixing is unsuccessful. We will now back-track from that cup. So let $i = i^*$, $b = i$ and initialize $t = T_H$. Also, let ω^t denote the amount of water in the cup c_b traced in step t. We thus have $\omega^{T_H} = w_{i^*}^{T_H} > 2/(n-\alpha)$. We will guarantee that the values ω^t are non-decreasing when we decrement t from T_H towards 0. For $t = T_H$ down to 0, if cup c_b is selected for mixing in step t, we know that $W_t = \omega^t \geq \omega^{T_H} > 2/(n-\alpha)$. Since E_2 did not occur, it must be the case that $\max_{h \in A_t} w_h^{t-1} > k^{1/4} W_t$. Let h^* be the index into B_t of the h obtaining this maximum water in step $t-1$. We append h^* to the constructed list of indices $i_1, \ldots, i_r$ in the back-tracking as well as append the step t to the list of steps $t_1, \ldots, t_r$. We then update b to h, set ω^{t-1} to $w_h^{t-1} \geq k^{1/4} \omega^t$ and decrement t. If c_b was not selected for mixing, we simply decrement t.

Since ω increases by a factor at least $k^{1/4}$ each time the traced cup c_b is selected for mixing, it must be the case that $\omega^0 \geq 2k^{r/4}/(n-\alpha)$ if the produced back-tracking has length r. Since no cup ever contains more than 1 unit of water, this implies $2k^{r/4}/(n-\alpha) \leq 1 \Rightarrow r < 4 \lg_k n$. This implies that the event E_1 occurs.

Probability of success. In the following two paragraphs, we will show that $\Pr[E_1] \leq 2\delta^{10}/n$ and $\Pr[E_2] \leq \delta^2/n$. A union bound and the fact that $\delta < 1/3$ implies that a T_H step k-way mixing is successful with probability at least $1 - \delta/n$ when c_1 has 1 unit of water. As mentioned earier, a union bound over all $n - \alpha$ choices of the cup with 1 unit of water completes the proof. What remains is thus to bound the probability of E_1 and E_2.

There is a short back-tracking (Event E_1). To rule out the existence of a short back-tracking, consider a fixed value of $r \leq 4 \lg_k n$. For any such r, there are no more than $k^r \leq n^4$ choices for $i_1, \ldots, i_r$ and n choices for i. For any such choice, there are no more than $\binom{T_H}{r} \leq T_H^r$ choices for the steps $t_1, \ldots, t_r$ where j is decremented (the traced cup

is selected for mixing). Fix any such r, $i_1, \ldots, i_r$ and $t_1, \ldots, t_r$. For this to be a valid back-tracking, it must hold for all steps $t \notin \{t_1, \ldots, t_r\}$ that the cup c_b traced in that step is not selected for mixing. Since the mixing steps are independent, this happens with probability precisely $(1 - k/n)$ independently of the random choices in steps $t+1, \ldots, T_H$. For all steps $t \in \{t_1, \ldots, t_r\}$, it must be the case that the cup c_b traced in that step is selected for mixing. Again by independence, this happens with probability precisely k/n independently of the random choices in steps $t+1, \ldots, T_H$. For the fixed choice of i, r, $i_1, \ldots, i_r$ and $t_1, \ldots, t_r$, the probability that these form a valid back-tracking is thus no more than $(1-k/n)^{T_H - r}(k/n)^r \leq \exp(-(T_H - r)k/n)(k/n)^r$. We have $T_H = 20(n/k)\ln(n/\delta)$ and $k \leq n$, thus $r = 4\lg_k n \leq T_H/2$ and the probability is no more than $\exp(-T_H k/(2n))(k/n)^r = \exp(-10\ln(n/\delta))(k/n)^r = (k/n)^r(\delta/n)^{10}$. A union bound over all possible back-trackings of length $r \leq 4\lg_k n$ shows that

$$\begin{aligned}
\Pr[E_1] &\leq \sum_{r=0}^{4\lg_k n} n^5 T_H^r (k/n)^r (\delta/n)^{10} \\
&= \sum_{r=0}^{4\lg_k n} n^5 (20(n/k)\ln(n/\delta))^r (k/n)^r (\delta/n)^{10} \\
&= \sum_{r=0}^{4\lg_k n} n^5 (20\ln(n/\delta))^r (\delta/n)^{10}.
\end{aligned}$$

For $k \geq 40\ln(n/\delta)$, this is no more than

$$\begin{aligned}
\sum_{r=0}^{4\lg_k n} n^5 (20\ln(n/\delta))^r (\delta/n)^{10} &\leq \\
\sum_{r=0}^{4\lg_k n} n^5 (k/2)^r (\delta/n)^{10} &\leq \\
\sum_{r=0}^{4\lg_k n} 2^{-r} n^5 k^{4\lg_k n} (\delta/n)^{10} &= \\
2\delta^{10}/n.
\end{aligned}$$

A mix with much water, but no full cup (Event E_2). Let us first consider a fixed step t and condition on a fixed cardinality a of A_t and an arbitrary execution of the first $t-1$ steps. If we let $E'_{2,t}$ denote the event that $\max_{i \in A_t} w_i^{t-1} \leq k^{1/4} W_t$ and $E''_{2,t}$ the event that $W_t \geq 2/(n-\alpha)$. We now wish to bound $\Pr[E'_{2,t} \cap E''_{2,t} \mid |A_t| = a]$. For this, we further split $E'_{2,t}$ and $E''_{2,t}$ into smaller events. Let $E'_{2,t,\xi}$ denote the event that $\max_{i \in A_t} w_i^{t-1} \leq 2k^{1/4}\xi$ and $E''_{2,t,\xi}$ the event $W_t \geq \xi$ and consider values of $\xi = 2^i/(n-\alpha)$ for $i = 1, \ldots, \lg_2 n$. We claim that

$$\Pr[E'_{2,t} \cap E''_{2,t} \mid |A_t| = a] \leq \Pr\left[\cup_{i=1}^{\lg_2 n} (E'_{2,t,2^i/(n-\alpha)} \cap E''_{2,t,2^i/(n-\alpha)}) \mid |A_t| = a\right].$$

To see this, note that when $E''_{2,t}$ occurs, there is a maximal $1 \leq i < \lg_2 n$ for which $2^i/(n-\alpha) \leq W_t \leq 2^{i+1}/(n-\alpha)$. When $E'_{2,t}$ also occurs, this further implies $\max_{i \in A_t} w_i^{t-1} \leq 2^{i+1}k^{1/4}/(n-\alpha)$. That is, both of the events $E'_{2,t,2^i/(n-\alpha)}$ and $E''_{2,t,2^i/(n-\alpha)}$ occur. By a union bound, we thus have

$$\begin{aligned}
\Pr[E'_{2,t} \cap E''_{2,t}, |A_t| = a] &\leq \\
\sum_{i=1}^{\log_2 n} \Pr[E''_{2,t,2^i/(n-\alpha)} \cap E'_{2,t,2^i/(n-\alpha)} \mid |A_t| = a] &\leq \\
\sum_{i=1}^{\log_2 n} \Pr[E''_{2,t,2^i/(n-\alpha)} \mid E'_{2,t,2^i/(n-\alpha)}, |A_t| = a]
\end{aligned}$$

Next, we recall that each shuffler picks a uniformly random subset of cups to mix. If we condition this choice on $E'_{2,t,\xi}$ and $|A_t| = a$, then A_t is distributed as a uniform sample of a elements without replacement from the set of active cups c_i where $w_i^{t-1} \leq 2k^{1/4}\xi$. Furthermore, recall that we started with one cup containing one unit of water and all other cups being empty. If we were to sample a cup uniformly at random among all active cups, then the expected amount of water in a sampled cup would be precisely $1/(n-\alpha)$. Conditioning on $E'_{2,t,\xi}$ removes the most full cups and hence the expected amount of water in each sampled cup may only decrease when conditioning on $E'_{2,t,\xi}$. It follows from Hoeffding's inequality for sampling without replacement that for any $\xi \geq 2/(n-\alpha)$, we have

$$\begin{aligned}
\Pr[E''_{2,t,\xi} \mid E'_{2,t,\xi}, |A_t| = a] &= \\
\Pr[|W_t| \geq \xi \mid E'_{2,t,\xi}, |A_t| = a] &\leq \\
\Pr[|W_t - \mathbb{E}[W_t]| \geq \xi - 1/(n-\alpha) \mid E'_{2,t,\xi}, |A_t| = a] &\leq \\
\Pr[|W_t - \mathbb{E}[W_t]| \geq \xi/2 \mid E'_{2,t,\xi}, |A_t| = a] &\leq \\
2\exp\left(-\frac{2(a\xi 2/2)^2}{a(2k^{1/4}\xi)^2}\right) &= \\
2\exp\left(-a/(8\sqrt{k})\right).
\end{aligned}$$

Thus

$$\Pr[E'_{2,t} \cap E''_{2,t} \mid |A_t| = a] \leq 2\lg_2(n)\exp(-a/(8\sqrt{k})).$$

Using this inequality, we then observe that

$$\begin{aligned}
&\Pr\left[E'_{2,t} \cap E''_{2,t}\right] \\
=&\sum_{a=0}^{k} \Pr\left[E'_{2,t} \cap E''_{2,t} \mid |A_t| = a\right] \Pr\left[|A_t| = a\right] \\
=&\sum_{a=0}^{\frac{k(1-\alpha/n)-2}{2}} \Pr\left[E'_{2,t} \cap E''_{2,t} \mid |A_t| = a\right] \Pr\left[|A_t| = a\right] \\
&+\sum_{a=\frac{k(1-\alpha/n)}{2}}^{k} \Pr\left[E'_{2,t} \cap E''_{2,t} \mid |A_t| = a\right] \Pr\left[|A_t| = a\right] \\
\leq&\sum_{a=0}^{\frac{k(1-\alpha/n)-2}{2}} \Pr\left[|A_t| = a\right] \\
&+\sum_{a=\frac{k(1-\alpha/n)}{2}}^{k} 2\lg_2(n)\exp\left(-(k(1-\alpha/n)/2)/(8\sqrt{k})\right) \Pr\left[|A_t| = a\right] \\
\leq&\Pr\left[|A_t| \leq \frac{k(1-\alpha/n)}{2}\right] + 2\lg_2(n)\exp\left(-(k(1-\alpha/n)/2)/(8\sqrt{k})\right)
\end{aligned}$$

To conclude the proof, we would now like to argue that both of the terms in the last inequality above are small. We observe that B_t is a uniform sample without replacement from the n cups and thus we have that $\mathbb{E}[|A_t|] = k(1-\alpha/n)$. Using the Chernoff bound for sampling without replacement and assuming $k \geq 16\ln(n/\delta)(1-\alpha/n)^{-1}$, we get

$$\Pr[|A_t| \leq (1/2)k(1-\alpha/n)] \leq \exp(-k(1-\alpha/n)/8) \leq (\delta/n)^2.$$

Similarly, for $k \geq 256\ln^2(\delta/n)(1-\alpha/n)^{-2}$, we have that

$$\begin{aligned} &2\lg_2(n)\exp(-(1/2)k(1-\alpha/n)/(8\sqrt{k})) \\ =&2\lg_2(n)\exp(-\sqrt{k}(1-\alpha/n)/16) \\ \leq&2\lg_2(n)(\delta/n)^2 \end{aligned}$$

Thus for $k \geq 256\ln^2(\delta/n)(1-\alpha/n)^{-2}$, we have

$$\Pr[E'_{2,t} \cap E''_{2,t}] \leq 3\lg_2(n)(\delta/n)^2.$$

A union bound over all T_H then implies

$$\Pr[E_2] \leq 3T_H\lg_2(n)(\delta/n)^2.$$

There are $T_H = 20(n/k)\ln(n/\delta)$ choices for t and for $k \geq 256\ln^2(n/\delta)$, we have that

$$\Pr[E_2] \leq \delta^2/n,$$

which concludes the proof. ◀

5 Ethereum's Block Proposer Elections

One particular real-world application that can benefit from our shuffling protocol, is Ethereum's block proposer election. In the following, we provide a high-level idea of this election process and we refer the interested reader to the current proposal [14] for more details. In this setting, we have commitments[1] $(c_1, \ldots, c_n)$, where $n = 2^{14}$ and where c_i belongs to some identity i, who is the only entity that can open the commitment. These identities need to be arranged in a random secret order. Once this is done, the first γ owners of the commitments reveal themselves in order of the output list and perform some consensus related action that is not relevant for us. That is, the first identity in the output list is the first block proposer, the second identity the second proposer and so on. From a security perspective, one would like to ensure that an adversary that corrupts α identities, β of the shufflers, and gets to see some of the proposers that already revealed themselves, cannot guess the identity of the next honest block proposer.

In order to obtain the random secret ordering, in the current proposal, a sequence of shufflers are effectively executing Håstad's square shuffle [17, 18], i.e. there $k = \sqrt{n}$, interspersed with some additional public permutation steps. The current proposal is purely heuristic, is only described in an informal model, and does not have a security proof. It is not secure against an adversary that can corrupt shufflers adaptively.

[1] We note again that all of our results work equally well for vectors of commitments.

Our approach can be used to obtain a secret random ordering of the block proposers with stronger security guarantees and, in particular, with provable security guarantees. We note, however, that in our model we do not consider parts of the performed permutation to be revealed once the shuffling protocol is finished. Luckily our analysis can easily be amended to account for this.

In the proof of Theorem 7, we assumed that a fixed number of cups, denoted α, were idle. We will now generalize the results to the following setup: Before the random shuffling process begins, we have two phases. In the first phase, we have a fixed set of α marked cups. In the second phase, we choose a uniform random subset of γ of the cups and mark them. If a cup was marked either during the first or second phase, it becomes idle and otherwise it is active. Notice that this corresponds to first corrupting α ciphertexts and then revealing γ random ciphertexts at the end. Let η be the number of idle cups.

Once the idle and active cups have been chosen, we run the water mixing process as in the proof of Theorem 7. We now bound the probability of seeing an active cup with more than $2/(n-\eta)$ units of water after T steps of mixing. We first bound the probability of seeing many idle cups. For this, notice that the first phase marks precisely α cups. For the second phase, the number of newly marked cups can be bounded by observing that the γ samples without replacement each picks a cup already marked in the first phase with probability precisely α/n (when looking at the marginal distribution of the cup). It follows by a Hoeffding bound for sampling without replacement that the number of newly marked cups in the second phase, denoted ζ, satisfies:

$$\Pr[\zeta - (1-\alpha/n)\gamma > \ell] < \exp(-2\ell^2/\gamma).$$

Setting $\ell = \sqrt{\gamma \ln(n/\delta)}$ bounds the above by δ^2/n^2. Thus with probability at least $1-\delta^2/n^2$, we have

$$\begin{aligned}\eta &\leq \alpha + \zeta \leq \alpha + \ell + (1-\alpha/n)\gamma \\ &= \alpha + \gamma - \alpha\gamma/n + \sqrt{\gamma \ln(n/\delta)}.\end{aligned}$$

A union bound together with Lemma 8 invoked with $\delta' = \delta/(2n)$ gives us that with probability at least $1 - \delta^2/n^2 - \delta/n$, there is no index i with $w_i^T \geq 2/(n-\eta)$. Note that the above analysis above is for a fixed number γ of revealed output locations. Doing a union bound over all $\gamma' \leq \gamma$ shows that the probability that throughout the revealing any of γ additional locations, that there is ever an input cup z whose output destination can be predicted with probability greater than $2/(n-\eta)$ is at most $n \cdot (\delta/(2n) + \delta^2/n^2) \leq \delta$.

6 Experiments

In this section, we perform numerical experiments to precisely determine the practical constants in our distributed shuffling protocol. We consider different sets of parameters. Since adversarially corrupt shufflers in our protocol are as bad as just no shuffle being performed, we simply measure the number of required honest shuffles, until the desired security guarantees are achieved. More precisely, if T_H honest shuffles are sufficient, then running our protocol for T rounds is secure against $\beta = T - T_H$ many corrupted shufflers.

In each experiment, we run the water mixing process from the proof of Lemma 8 with varying values for n, k, and α. For each fixed set of parameters the benchmark is repeated 100 times. In every round of an experimental run, we check whether any cup has too much water. If it does, then this run of the experiment for this round is considered to be failing. The

Table 1 Results of our numerical experiments for determining the number $T - \beta$ of honest shuffles that is needed for successfully shuffling with different sets of parameters.

#	n	k	α/n						
1	2^{14}	128	1/4	δ	0.8	0.6	0.4	0.2	0
				$T-\beta$	713	839	927	988	1804
2	2^{14}	256	1/4	δ	0.8	0.6	0.4	0.2	0
				$T-\beta$	337	398	452	502	627
3	2^{14}	512	1/4	δ	0.8	0.6	0.4	0.2	0
				$T-\beta$	199	229	254	278	438
4	2^{14}	128	1/2	δ	0.8	0.6	0.4	0.2	0
				$T-\beta$	874	955	1080	1204	1853

fraction of failing simulations in a given round, denoted by δ, is an unbiased estimate of the true probability that the adversary can determine the position of a uncorrupted ciphertext with probability greater than $2/(n-\alpha)$ in that round.

The result of our benchmarks are summarized in Figure 1. Even in a highly adversarial setting, where 1/2 of all elements in the vector are corrupted and the local shuffle size is as small as $k = 128$, our protocol successfully distributes the water after less than 2000 rounds for a vector of length $n = 2^{14}$. In the context of Ethereum's block proposer elections, we have $T = 2^{13}$ time slots for one election and thus one can tolerate a fraction of around 3/4 of corrupted shufflers.

References

1 Dave Bayer and Persi Diaconis. Trailing the dovetail shuffle to its lair. *The Annals of Applied Probability*, pages 294–313, 1992.

2 Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 263–280. Springer, 2012.

3 Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 566–582. Springer, 2001.

4 Dan Boneh, Saba Eskandarian, Lucjan Hanzlik, and Nicola Greco. Single secret leader election. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 12–24, 2020.

5 Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *Annual International Cryptology Conference*, pages 565–596. Springer, 2018.

6 Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A Kroll, and Edward W Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In *International Conference on Financial Cryptography and Data Security*, pages 486–504. Springer, 2014.

7 Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.

8 Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE, 2001.

9 Dario Catalano, Dario Fiore, and Emanuele Giunta. Efficient and universally composable single secret leader election from pairings. Cryptology ePrint Archive, Paper 2021/344, 2021. URL: `https://eprint.iacr.org/2021/344`.

10 Dario Catalano, Dario Fiore, and Emanuele Giunta. Adaptively secure single secret leader election from ddh. In *Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing*, 2022.

11 David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

12 Persi Diaconis and Mehrdad Shahshahani. Generating a random permutation with random transpositions. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(2):159–179, 1981.

13 Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

14 Ethereum. Whisk: A practical shuffle-based ssle protocol for ethereum. *Accessed 09/09/2022*, 2022.

15 Jun Furukawa and Kazue Sako. An efficient scheme for proving a shuffle. In *Annual International Cryptology Conference*, pages 368–387. Springer, 2001.

16 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 40–49, 2013.

17 Johan Håstad. The square lattice shuffle. *Random Structures and Algorithms*, 29(4):466–474, 2006.

18 Johan Håstad. The square lattice shuffle, correction. *Random Structures and Algorithms*, 48(1):213, 2016.

19 Max Hoffmann, Michael Klooß, and Andy Rupp. Efficient zero-knowledge arguments in the discrete log setting, revisited. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2093–2110, 2019.

20 Markus Jakobsson, Ari Juels, and Ronald L Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *11th USENIX Security Symposium (USENIX Security 02)*, 2002.

21 Gregory Maxwell. Coinjoin: Bitcoin privacy for the real world. *Accessed 09/09/2022*, 2013.

22 Ben Morris and Phillip Rogaway. Sometimes-recurse shuffle. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 311–326. Springer, 2014.

23 C Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125, 2001.

24 Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

25 Thomas Ristenpart and Scott Yilek. The mix-and-cut shuffle: small-domain encryption secure against n queries. In *Annual Cryptology Conference*, pages 392–409. Springer, 2013.

26 Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 393–403. Springer, 1995.

27 Edward O Thorp. Nonrandom shuffling with applications to the game of faro. *Journal of the American Statistical Association*, 68(344):842–847, 1973.

MPC with Low Bottleneck-Complexity: Information-Theoretic Security and More

Hannah Keller
Aarhus University, Denmark

Claudio Orlandi
Aarhus University, Denmark

Anat Paskin-Cherniavsky
Ariel University, Israel

Divya Ravi
Aarhus University, Denmark

Abstract

The bottleneck-complexity ($\mathcal{BC}$) of secure multiparty computation (MPC) protocols is a measure of the maximum number of bits which are sent and received by any party in protocol. As the name suggests, the goal of studying $\mathcal{BC}$-efficient protocols is to increase overall efficiency by making sure that the workload in the protocol is somehow "amortized" by the protocol participants.

Orlandi et al. [28] initiated the study of $\mathcal{BC}$-efficient protocols from simple assumptions in the correlated randomness model and for semi-honest adversaries. In this work, we extend the study of [28] in two primary directions: **(a)** to a larger and more general class of functions and **(b)** to the information-theoretic setting.

In particular, we offer semi-honest secure protocols for the useful function classes of abelian programs, "read-k" non-abelian programs, and "read-k" generalized formulas.

Our constructions use a novel abstraction, called *incremental function secret-sharing* (IFSS), that can be instantiated with unconditional security or from one-way functions (with different efficiency trade-offs).

2012 ACM Subject Classification Theory of computation → Cryptographic protocols

Keywords and phrases Secure Multiparty Computation, Bottleneck Complexity, Information-theoretic

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.11

Related Version *Full Version*: `https://eprint.iacr.org/2023/683`

Funding *Hannah Keller*: European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC).
Claudio Orlandi: Concordium Blockhain Research Center, Aarhus University, Denmark; the Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC).
Divya Ravi: European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC).

1 Introduction

Secure Multi-party Computation (MPC) [31, 19, 6, 11], allows a set of mutually distrusting parties to perform a joint computation of their private inputs in a secure way, which essentially means that no adversary corrupting a subset of parties can learn more information than the output of the joint computation (*privacy*), nor can they affect the *correctness* of the output (other than by choosing their own inputs).


© Hannah Keller, Claudio Orlandi, Anat Paskin-Cherniavsky, and Divya Ravi;
licensed under Creative Commons License CC-BY 4.0

The complexity of MPC protocols is most commonly analyzed in terms of three fundamental metrics, namely communication complexity (that measures the total number of bits communicated in the protocol), round complexity (number of sequential interactions in the protocol) and computation complexity (that captures the computational resources parties need to execute the protocol steps). In this paper, we focus on a more fine-grained, comparitively less-explored metric called *bottleneck complexity ($\mathcal{BC}$)* which was introduced by Boyle et al. [9]. This metric, which can be informally defined as the *maximum communication complexity of any party* captures the load-balancing aspect of MPC protocols – for e.g. a protocol where everyone sends a bit to a central party would have $O(n)$ $\mathcal{BC}$ (incurred by the central party, where n denotes the number of parties) as opposed to a protocol where everyone sends a single bit to its neighbour in a chain-like fashion, which has $O(1)$ $\mathcal{BC}$. Notably, both these protocols have the same communication complexity but the communication in the latter is more balanced among the parties as captured by its lower bottleneck complexity.

The works of [9, 28] focused on designing MPC protocols with bottleneck complexity sublinear in the number of parties, which is particularly interesting for large-scale settings where n is huge. [9] presented a FHE-based compiler that transforms insecure protocols into secure protocols while preserving the bottleneck complexity. However, FHE is still relatively inefficient, and is only known under a more limited set of assumptions - roughly, variants of LWE. In light of this, [28] initiated the study of designing protocols with low bottleneck complexity in the preprocessing model, under minimal computational assumptions (such as one-way functions and linearly homomorphic encryption, which can in turn be based on traditional assumptions such as discrete logarithm and factoring).

In this work, we extend the study of [28] in two primary directions: **(a)** to a larger and more general class of functions and **(b)** to the information-theoretic setting. We additionally consider a more extended notion of g-$\mathcal{BC}$-efficiency to capture protocols which have $\mathcal{BC}$ of $\mathsf{poly}(g(n), \lambda)$, where λ denotes the security parameter[1]. More specifically, [28] focused on protocols with $O(1)$-$\mathcal{BC}$-efficiency (i.e. with $\mathsf{poly}(\lambda)$ $\mathcal{BC}$, independent of n) and log-$\mathcal{BC}$-efficiency (i.e. with $\mathsf{poly}(\log(n), \lambda)$ $\mathcal{BC}$); while we consider a more general notion of g-$\mathcal{BC}$ efficiency, where g is any *sublinear* function. This allows us to work with a somewhat extended parameter setting – Consider a function $f(x_1, ..., x_n)$ where each x_i has ℓ bits, and the (common) output is z bits. The notions of $\mathcal{BC}$-efficiency become meaningful only if these parameters ℓ and z are typically small. In the prior work of [28], these are assumed to be constant or polylogarithmic in n. In this work we extend our quest to settings where ℓ and z are sublinear in size ($o(n)$), as this would still allow for constructions satisfying the extended notion of $\mathcal{BC}$-efficiency. Moreover, the constructions of [9, 28] have $\mathcal{BC}$ that scales with the security parameter λ (where λ is typically $\omega(\log(n))$) in computational settings), which is avoided by our information-theoretic constructions.

Related Work. The most relevant work to ours is [9, 28] (whose results we discuss above). There are several works in the MPC literature that focus on optimizing communication complexity, some of which we mention below. The works of [12, 15, 24] focus on designing communication-efficient protocols in the information-theoretic setting with correlated randomness. Interestingly, the notion of bottleneck complexity and communication complexity are the same for the two-party setting. The work of [27] presented a compiler that transforms an insecure protocol to secure one while preserving communication complexity. [13, 14, 29, 1] focus on optimizing communication complexity related to the circuit size.

[1] For information-theoretic protocols with perfect security where there is no dependency on λ, g-$\mathcal{BC}$ efficiency refers to $\mathcal{BC}$ of $\mathsf{poly}(g(n))$.

The constructions of [21, 23, 20] involve a chain-like interaction pattern (similar to our constructions). However, these constructions achieve a weaker notion of security (namely, residual security) as they are restricted to a single chain traversal (unlike our constructions which typically involve multiple traversals over chain). The efficient non-interactive multiparty computation (NIMPC) constructions in [22, 16, 4] also achieve this weaker security.

Our goal of minimizing bottleneck complexity is somewhat similar in spirit to the massive parallel computation model of [18, 17] which focuses on minimizing the storage and communication of servers. The works of [8] and [25] design protocols that optimize metrics that are closely related to bottleneck complexity (namely, communication locality and message complexity).

For further related work, we refer to references therein.

1.1 Our Contribution

Our main contribution is constructing $\mathcal{BC}$-efficient protocols in the correlated randomness model for various interesting function classes. Further, we introduce a new primitive, namely *Incremental Function Secret Sharing* (IFSS), which not only serves as a neat abstraction of $\mathcal{BC}$-efficient computation, but also allows us to cast our constructions in a generalized framework that captures both computational and information-theoretic variants.

All our constructions are secure against a semi-honest (passive) adversary who can corrupt up to $n-1$ among the n parties. Our computational constructions are based on garbling schemes, which rely on one-way functions. Our information-theoretic constructions (satisfying perfect security) are a $\mathcal{BC}$-friendly extension of the OTTT (one-time truth-tables) construction from [24]. We elaborate on our contributions below.

New primitive: Incremental Function Secret Sharing (IFSS). In Section 4, we introduce a new primitive, namely, Incremental Function Secret Sharing (IFSS), which essentially allows a set of parties to evaluate a hidden function on a joint public input. This tool is a clean abstraction of the core ideas of $\mathcal{BC}$-efficient evaluation in our constructions.

At a high-level, this primitive can be viewed as a variant of function secret sharing (which additively shares a function among a set of evaluators, enabling them to compute output shares which can be aggregated to obtain the output), with the difference that the output shares are aggregated incrementally on a chain, in a $\mathcal{BC}$-efficient manner. IFSS can be instantiated with garbled circuits or one-time truth tables (OTTT), enabling us to unify our computational and information-theoretic variants. We believe this primitive to be of independent interest and a useful building block for $\mathcal{BC}$-efficient protocols.

Abelian Programs. Recall that an abelian program h can be expressed as $h(X_1, \ldots, X_n) = f(\sum_{i=1}^{n} X_i)$ for some $f : G \to \{0,1\}$, where G denotes an abelian group. In Section 5, we use our IFSS primitive to generalize the approach of [28] that constructs $\mathcal{BC}$-efficient (computational) protocols for abelian programs. Plugging in the information-theoretic OTTT-based instantiation of IFSS yields an information-theoretic $\mathcal{BC}$-efficient protocol for abelian programs with $\mathcal{BC}$ of $O(\log |G|)$. For completeness, we additionally demonstrate how using the garbled-circuit based instantiation of IFSS results in the computational protocol of [28].

As an interesting application of $\mathcal{BC}$-efficient abelian programs, we demonstrate how it could be used to compute the maximum among n values as $y = \max(X_1, X_2, \ldots, X_n)$ in a $\mathcal{BC}$-efficient manner. This can be extended to compute $f(\max(X_1, X_2, \ldots, X_n))$, where f is any arbitrary function.

"Read-k" Non-Abelian Programs. Briefly, a non-abelian program extends the notion of an abelian program to non-abelian groups. Here, $h(x_1, \ldots, x_n) : \{D\}^n \to \{0,1\}$ is represented as $h(x) = g(\pi_{1,x_{i_1}} \cdot \ldots \cdot \pi_{t,x_{i_t}})$, where each $\pi_{j,x_{i_j}}$ is an element of a group G that depends only on j, x_{i_j} (where $i_j \in \{1, \ldots, n\}$), and $f : G \to \{0,1\}$[2]. In a read-k program, group elements depending on some x_i appear up to k times in the above representation.

We present a $\mathcal{BC}$-efficient protocol for any function f, that can be represented as a read-k non-abelian program over a group G. The computational and information-theoretic variants of these constructions incur a $\mathcal{BC}$ of $O(\log|G|(\lambda + k))$ and $O(k \log|G|)$ respectively and will therefore have sublinear $\mathcal{BC}$ as long as each of the parameters k and $\log|G|$ are "sufficiently small". More specifically, we can allow $k \log|G|$ to be of size $o(n)$. Even for e.g., $|G| = 2^{n^\epsilon}, k = O(1)$ for $\epsilon \in (0,1)$, we obtain sublinear $\mathcal{BC}$. The details of our results for non-abelian programs appear in the full version [26].

"Tree-based" read-k generalized formulas. In Section 6, we present a $\mathcal{BC}$-efficient protocol for any function f that can be represented as a "tree-like" formula, which may have multi-input (and output) gates. More concretely, nodes in this formula are either inputs X_i (which may belong to some finite group, not necessarily boolean domain), and gates with 2 inputs that output a single output [3]. The inputs, intermediate outputs (which are inputs to other gates) and the output are assumed to be bounded by ℓ bits (where $\ell = \log|G|$ in our constructions). For a formula that is read-k (i.e. each input variable appears at most k times) and has depth d [4], our computational and information-theoretic variants result in $\mathcal{BC}$ of $O(k \cdot d \cdot \ell \cdot \lambda)$ and $O(k \cdot d \cdot \ell)$ respectively.

As long as the above parameters of k, d, and ℓ are "sufficiently small" (i.e. $k \cdot d \cdot \ell$ is of size $o(n)$), the protocols remain $\mathcal{BC}$-efficient. Even with these restrictions, such formulas are quite expressive. Notably, we allow for "generalized" gates in terms of the functions they compute – the above restrictions are only with respect to the size of the inputs and outputs of these gates but the structure of the functions computed by these gates may be quite complex. For example, consider a generalized formula of depth $O(\log(n))$, and $k = \ell = n^{0.4}$, where each gate (having two inputs and an output of length $\ell = n^{0.4}$), evaluates a function with circuit complexity of $\Omega(2^{n^{0.5}})$. The $\mathcal{BC}$ complexity of the protocol above only would be $\tilde{O}(n^{0.8})$ [5] for our information theoretic implementation.

Lastly, we point out that while our information-theoretic constructions have better $\mathcal{BC}$ than the computational variants, the size of the correlated randomness for our information-theoretic constructions grows exponentially with the number of parties (due to the OTTT approach). However, the $\mathcal{BC}$ still remains sublinear in the number of parties for all our constructions.

In Section 2, we compare the expressiveness of the above function classes of abelian, non-abelian programs and tree-based formulas.

Open Problems. It remains an open question to determine the complete characterization of functions for which $\mathcal{BC}$-efficient protocols exist. In fact, since $\mathcal{BC}$-efficient protocols are known to be impossible for general functions even when no security is required [9], it would also be interesting to understand which functions can be computed in the clear with low bottleneck complexity.

---

[2] Note that unlike abelian programs, some π_{x_i} depending on x_i may crucially appear more than once, as the group G is not commutative.

[3] The construction is actually more general, and could allow for "generalized gates" with larger fan-in, but we stick with 2 for simplicity.

[4] Note that for balanced trees, $d = \log_2(k \cdot n)$, which is sublinear in n.

[5] $\tilde{O}$ ignores logarithmic factors.

1.2 Technical Overview

Our constructions have the following common two-step approach: **(1)** First, the private inputs are aggregated in a $\mathcal{BC}$-efficient way to obtain a joint common public input that "hides" the private inputs. **(2)** Next, we consider an augmented version of the function f, say f' (that may be required to be kept secret) such that evaluating f' on the common public input essentially corresponds to an evaluation of f. The evaluation of f' is carried out via IFSS.

Overview of IFSS. Before describing details of our protocols, we give a high-level overview of the IFSS primitive. In a nutshell, IFSS allows a set of parties to evaluate a hidden function f on a common public input x such that nothing beyond $f(x)$ is revealed (as long as one of the evaluators is honest). This evaluation is done in an incremental manner, where each party computes its "share" and these shares are aggregated over a chain. In the garbled-circuit based instantiation, these "shares" are additive shares of the label corresponding to the common input x. Once this label is reconstructed (via aggregation over chain), the garbled circuit computing f (given as part of the setup) is evaluated to compute the output. This is the crux of $\mathcal{BC}$-efficient evaluation in the constructions of [28], which satisfy computational security.

For the information-theoretic instantiation, we use an approach based on secret-sharing the truth-table inspired by the one-time truth table (OTTT) protocol of [24]. As already noted by [7], this leads to information-theoretic FSS. We detail the construction and show how it fits the IFSS framework.

In this protocol, parties are given an additive sharing of the (permuted) truth table of the function being evaluated, as a part of the correlated randomness setup. Roughly speaking, the parties first identify the relevant entry of the truth table (using their input and correlated randomness) i.e. the one that corresponds to the correct output. The pointer to this entry can be interpreted as the common input of the IFSS. Now, the evaluation is nothing but aggregating the additive shares of the relevant entry (determined by this common input), which can be done in a chain-like fashion to maintain $\mathcal{BC}$-efficiency. This is the main idea of the information-theoretic instantiation of IFSS.

Next, we describe our constructions. Note that for protocols to be $\mathcal{BC}$-efficient, the interaction involved in the above outlined common two-step approach must satisfy the following two properties: **(a)** Each intermediate value that is communicated must be "small". **(b)** Privacy of the inputs must be maintained.

Abelian Programs. The structure of abelian programs (say $h(x_1, \ldots, x_n) = f(\sum_{i=1}^{n} x_i)$) is such that it naturally supports property **(a)**.

This is because the sum of inputs can be computed incrementally in a chain-like fashion with the property that the size of the intermediate sums does not blow up. However, to satisfy **(b)**, the protocol of [28] makes parties aggregate their masked inputs instead (using masks received as part of setup) to compute a masked sum (say $z = y + R$, where y denotes the sum of inputs and R denotes the mask). Generalizing their construction, we view this "masked sum" as the common input, and use IFSS for evaluation. More specifically, we consider a (private) augmented function $f'_R(z) = f(z - R)$ (with secret R hard-coded), which first unmasks this masked sum to retrieve the sum y, upon which h is computed. We use IFSS to compute f', yielding computational and information-theoretic $\mathcal{BC}$-efficient protocols (depending on whether the IFSS is instantiated using the garbling-based or OTTT based approach).

"read-k" Non-Abelian Programs. Similar to abelian programs, non-abelian programs (say, $h(x) = f(\pi_{1,x_{i_1}} \cdot \ldots \cdot \pi_{t,x_{i_t}})$) support property **(a)** as the input value to f (i.e. $\pi_{1,x_{i_1}} \cdot \ldots \cdot \pi_{t,x_{i_t}}$) can be computed incrementally in a chain-like fashion (where party i_1 forwards $\pi_{1,x_{i_1}}$ to i_2 who computes $\pi_{1,x_{i_1}} \cdot \pi_{2,x_{i_2}}$ and forwards this value to i_3 and so on) while making sure that the intermediate values remain "small". To maintain property **(b)**, the aggregation could be done over masked inputs instead. However, unlike the case of abelian groups (which is commutative), we need to be slightly more careful in case of non-abelian groups (which may be non-commutative) to ensure that this aggregation of masked inputs happens in a specific order. In our protocol, the aggregated common input corresponds to $\big(\mathsf{r}_t \cdot \ldots (\mathsf{r}_2 \cdot (\mathsf{r}_1 \cdot \pi_{1,x_{i_1}}) \cdot \pi_{2,x_{i_2}}) \cdot \ldots \cdot \pi_{t,x_{i_t}}\big)$; accordingly each party i_j must compute its intermediate value by using its random value as a prefix and $\pi_{j,x_{i_j}}$ as a suffix to the intermediate value received from its neighbour on the chain. Once, parties have computed this common public input, we use IFSS to compute the augmented function f', where f' first uses a (secret hard-coded) prefix $\mathsf{r}_1^{-1} \cdot \mathsf{r}_2^{-1} \ldots \cdot \mathsf{r}_t^{-1}$ to unmask this common input and then compute f.

"Tree-based" read-k generalized formulas. Next, consider the case of "tree-like" formulas. Consider one of the "generalized gates" say $f(x_1, \ldots, x_{\mathsf{m}})$ (whose number of inputs and output size is "sufficiently small" [6] but could have any arbitrarily complicated structure). Unlike the previous cases, we cannot exploit the structure of the function to support incremental aggregation (that supports property **(a)**). Instead, every party involved in f must compute its masked input (using a random value given as part of the setup) and communicate it in a chain-like fashion *without* any incremental computation. These masked inputs are simply appended (therefore the size of the intermediate values grows in this case) and this set of masked values forms the aggregated common input. Note that the size of this aggregated common input grows with the number of inputs to this "generalized" gate, which brings in the need for restricting the size and number of inputs to these gates to be "small" (i.e. $o(n)$) for sublinear $\mathcal{BC}$.

Once the common input is determined, we proceed to evaluation. For this, we consider an instance of IFSS for each "generalized" gate and combine the intermediate outputs in a tree-like fashion. For simplicity, consider a gate at the first level, computing $f_{(1,2)}$ that takes two leaf nodes (corresponding to inputs x_1 and x_2) as input. As mentioned previously, the aggregate common input corresponds to $z = z_1 || z_2$, where $z_i = x_i + \mathsf{r}_i$ for $i \in [2]$, where r_i are random masks given during setup. An instance of IFSS with hidden function $f'_{(1,2)}$ (that has r_i values as hard-coded inputs) and single common input z is initiated, that first unmasks the random values from the set of masked inputs z and then computes the output of $f_{(1,2)}$. This instance involves only the subset of parties holding one of these inputs x_1 or x_2 as evaluators, not the set of all parties. Note that this does not violate security because even if one of the parties contributing an input to $f_{(1,2)}$ is honest, by IFSS security, the parties will not learn anything except of the output of $f_{(1,2)}$. Otherwise, if all parties are corrupted, they know all their inputs anyway, so there is nothing to hide. More generally, the IFSS instance corresponding to a root of a subtree involves only the parties whose input is one of the leaf nodes of this subtree. This is crucial to maintain $\mathcal{BC}$-efficiency.

However, this approach would result in parties learning the output of $f_{(1,2)}$ which may not necessarily be leaked by the output of f. Therefore, instead of computing the original $f_{(1,2)}$, we compute the modified function $f'_{(1,2)}$ that "masks" the output of $f_{(1,2)}$ with a random mask $\mathsf{r}_{(1,2)}$ chosen during setup. We ensure correctness of computation by defining

[6] More specifically, these parameters are bounded by ℓ such that $k \cdot d \cdot \ell$ is of size $o(n)$.

the functions at the levels above accordingly – for instance, consider another function $f_{(3,4)}$ at level 1 (with its similarly defined $f'_{(3,4)}$ that "masks" the output of $f_{(3,4)}$ with random mask $\mathsf{r}_{(3,4)}$). Suppose the "tree-like" formula had a function $f_{(1,4)}$ at level 2 that is supposed to take as input $y_{(1,2)}$ and $y_{(3,4)}$ (respectively the outputs of $f_{(1,2)}$ and $f_{(3,4)}$). We now define $f'_{(1,4)}$ as the function with hard-coded masks $\mathsf{r}_{(1,2)}$,$\mathsf{r}_{(3,4)}$ and $\mathsf{r}_{(1,4)}$ that receives instead masked inputs $(y_{(1,2)} + \mathsf{r}_{(1,2)})$ and $(y_{(3,4)} + \mathsf{r}_{(3,4)})$, unmasks them, evaluates $f_{(1,4)}$, and finally masks the output with $\mathsf{r}_{(1,4)}$ (unless this node corresponds to the root). The evaluations are done level-by-level in an upward fashion until the function corresponding to the root of the tree is computed. This demonstrates how the outputs of various instances of IFSS computations are combined; completing the high-level description of this construction.

Lastly, we point that in our constructions, the common input used in the IFSS could be dictated by the adversary and still "unknown" to the honest party when the evaluation of IFSS begins. For e.g. consider the case of abelian programs, where the common input is the sum of masked inputs and computed over a forward-pass of the chain. Suppose the adversary corrupts a set of parties at the end of the chain and the evaluation of IFSS is executed over the subsequent backward pass of the chain. In such a case, this common input is in some sense still "uncommitted" (as the adversary can consider various versions of the common input and try to recompute the IFSS incremental evaluations in her head). Even though the adversary is passive, she can try to learn more information by trying to obtain multiple evaluations of IFSS corresponding to different common inputs (referred to as a residual function attack); which would breach security [7]. Security of IFSS does not help in this case as it holds only if all evaluators agree on the common input. However, our constructions ensure that the common input gets "committed" as soon as the backward pass reaches the first honest evaluator. The incremental computation by this honest evaluator would "fix" the common input (in a way that it is not possible to recompute evaluations on other common inputs any further), which creates the effect of "fixing" the corrupt evaluators' inputs. We refer to respective technical sections for further details.

2 Comparison of the function classes

In this section, we discuss what kind of functions are captured by the function classes considered in this work.

Abelian versus Non-Abelian Programs. We observe that indeed non-abelian programs appear to be more expressive than abelian programs within our $\mathcal{BC}$ constraints. As a nice simple example, fix the regular language L accepted by a "permutation" DFA (deterministic finite automaton) that has $\{0,1\}$ as the set of input symbols, $\{q_0, q_1, q_2\}$ as the set of states (with q_0 as the start state and q_2 as the accepting state) and δ as the transition function specified by $\delta(q_i, 1) = q_{i+1 \mod 3}, \delta(q_0, 0) = q_1, \delta(q_1, 0) = q_0, \delta(q_2, 0) = q_2$.

Consider a function $h(x_1, \ldots, x_m)$ (where each x_j is a bit, where $j \in \{1, \ldots, m\}$) that outputs 1 if $x \in L$, and 0 otherwise. Assume each x_j is assigned to some party and each party is assigned at most $k = o(n)$ bits at fixed (not necessarily consecutive) positions. To evaluate this function, one can devise a simple non-abelian program $h(x_1, \ldots, x_m) = f(\pi_{1,x_{i_1}} \cdot \ldots \cdot \pi_{t,x_{i_t}})$,

[7] Note that even if the IFSS computes a "masked" output (like in the case of a non-root gate in the construction for tree-like formulas), this would still violate security as the adversary could learn additional information about private inputs of honest parties involved in this gate just by comparing these multiple masked outputs corresponding to different common inputs.

where all π_{t,x_t}'s are in the group S_3 (where S_3 denotes a permutation group, whose elements are permutations of a set $M = \{1, 2, 3\}$, and the group operation is the composition of permutations), and for each t, $\pi_{t,0} = (1, 2)(3)$ and $\pi_{t,1} = (1, 2, 3)$ [8]. The function $f(\pi)$ outputs 1 if and only if $\pi(1) = 3$ [9]. Note that this non-abelian program is over a "small" group. However, it is not clear how to devise an abelian program that works with "small" groups of similar size for this function.

We point that it is always the case that a function $h : D^m \to \{0, 1\}$ can be expressed as an abelian program that works over the group $\mathbb{Z}_{|D|}^m$ – Each party P_i simply computes $x_i\vec{e}_i$, where $\vec{e}_i$ is the i'th vector in the standard basis, which can be aggregated to compute the sum, denoting the entire input. The problem is that this group is too large, and it is not clear (to us) how to do much better with abelian programs for the above DFA example.

Non-Abelian Programs versus "Tree"-based Formulas. In terms of feasibility (within the $o(n)$ domain), the formula-based construction is more expressive, within our $\mathcal{BC}$ constrains. This holds since we can simulate a non-abelian program involving $k \cdot n$ terms via a (nearly) balanced tree of depth $d = \log(k) + \log(n)$, using associativity of multiplication in the group. So, a read-k program would result in a read-k formula. Plugging in our constructions using the two approaches would result in their $\mathcal{BC}$ being very close, with only polylogarithmic overhead for the formula-based construction.

Next, we discuss whether there also exists a transformation in the other direction i.e. from formula to non-abelian programs. Given a generalized read-k formula, it is not always clear how to devise a (non)-abelian program with small overhead as above. In particular, the generic transformation due to Barrington [2] transforming a formula into a BP results in a BP of length in quadratic in formula *size*, and constant width, which is already $\Omega(n^2)$ for non-trivial formulas (with size at least n), and is prohibitively expensive for $\mathcal{BC}$. In fact, the resulting BP is already a permutation BP, but this does not help us, due to the large k (size of formula $\Omega(n^2)$ implies that k must be $\Omega(n)$). Despite the fact that non-abelian programs allow computing an arbitrarily complicated function f after a sequence of compositions on non-abelian group elements (where the sequence can be visualized as a permutation BP), it is not clear how this would help in the above transformation.

Next, we observe that the formula-based solution works for branching programs [10] (BP's). This is because the formulas with the "generalized gates" can support arbitrary transformations induced by inputs between the BP layers, which can be composed due to associativity of the function. As long as the width and parameter k of a "k-read BP' is "small", the formula-based approach would be $\mathcal{BC}$-efficient.

The above raises a question regarding if non-abelian programs support BPs. We observe that they would support a special kind of BPs, namely, permutation BPs [11]. In such a program of width w, the transition from root forward can be viewed as a composition of

---

[8] We use the cycle notation to express permutations. E.g. $\pi = (12)(3)$ denotes the permutation where $\pi(1) = 2, \pi(2) = 1$ and $\pi(3) = 3$ as $(1, 2)$ denotes the cyclic permutation and 3 is left unchanged.

[9] For e.g. consider the $x = x_1, \ldots, x_6 = 001011$, where each P_i ($i \in \{1, 2, 3\}$) holds x_i and x_{i+3}. One can check that $x \in L$ and $g(\pi_{1,0} \cdot \pi_{2,0} \cdot \pi_{3,1} \cdot \pi_{4,0} \cdot \pi_{5,1} \cdot \pi_{6,1}) = 1$.

[10] A directed acyclic graph in which the nodes are labeled by input variables and every nonterminal node has two outgoing edges, labeled by 0 and 1

[11] In a nutshell, these are layered branching programs, where every level's transitions, for each input value $x_i = b$ constitute a permutation $\pi_{i,b}$ [3]. Another difference between it and standard BP's is the way acceptance is defined. There is no root and accept/reject nodes, but rather a single resulting composed permutations, and acceptance/rejection is defined by belonging to one of two sets of output permutations, partitioning S_w. The width of such a program is the number of nodes, w, in each layer except for the first and last ones.

permutations in S_w, which defines a non-abelian program. The output is then determined based on whether the composed permutation maps 1 to an accept node (where the first and last transition are adapted to be permutations in a natural way). If $w = n$, the resulting group elements are too large for sublinear $\mathcal{BC}$ (even when transferring a single element). If all permutations actually fall in a subgroup of S_w, we could work in that subgroup and hope to obtain efficiency. Notably, non-abelian programs would not support general branching BP, as it may not be possible to map the transformations between layers in a general BP to a group structure (as it may not have an inverse or identity element). Lastly, we point that permutation BPs are somewhat restricted, and moving from (regular) BP to a permutation BP may have large costs in terms of w.

The above discussion argues that formulas are generally more expressive. However, there are still situations where using the non-abelian program approach is more useful. For instance, consider functions for which the resulting $\mathcal{BC}$ for the non-abelian program based protocol is constant (as in the permutation DFA example above). In such a case, moving from the non-abelian program construction to the formula-based construction, incurs a super-constant overhead. In particular, a $O(k \cdot d)/O(k) = \log(n)$ overhead for the information-theoretic construction is incurred. So, the former construction would still be preferred if one wishes to achieve the "ideal" best possible notion of $\mathcal{BC}$-efficiency, namely constant $\mathcal{BC}$.

3 Preliminaries

Notation. The cryptographic security parameter will be denoted by λ. The n parties $\{P_1, \ldots, P_n\}$ are pair-wise connected by secure and authentic channels, where n is polynomially bounded. We operate with semi-honest security and assume that any adversary can passively corrupt up to $n - 1$ parties.

We evaluate functions $f : \mathcal{X} \to \mathcal{Y}$ from a function class $\mathcal{F}$. We often assume that $\mathcal{X}$ and $\mathcal{Y}$ are groups endowed with an operation. We consider both abelian and non-abelian groups.

Security Model. We prove the security of our protocols based on the standard real/ideal world paradigm and refer to Section A for the details.

3.1 Definitions

▶ **Definition 1** (Bottleneck Complexity of a Protocol). *Let $\mathcal{CC}_i(\Pi)$ denote the expected number of bits sent or received by P_i in an execution of Π, with worst case inputs. The bottleneck complexity of an n-party protocol Π is defined as $\mathcal{BC}(\Pi) = \max_{i\in[n]} \mathcal{CC}_i$.*

We use the formal definition of [9] for bottleneck complexity. Informally, the bottleneck complexity of a protocol is the maximum communication complexity required by any party in the protocol execution. We consider a protocol Π to be $\mathcal{BC}$-efficient if the $\mathcal{BC}$ is sublinear in the number of total parties.

▶ **Definition 2** (Abelian Programs). *Let G be an abelian group, $S_1, \ldots, S_n$ be subsets of G, and $\mathcal{H}^G_{S_1,\ldots,S_n}$ be the set of functions $h : S_1 \times \cdots \times S_n \to \{0, 1\}$ of the form $h(x_1, \ldots, x_n) = f(\Sigma_{i=1}^n x_i)$, for some $f : G \to \{0, 1\}$. We call such functions h abelian programs.*

▶ **Definition 3** (Non-Abelian Programs). *Let $(G, \cdot)$ be a non-abelian group, $x_1, \ldots, x_n$ be inputs from domain D, and $\mathcal{H}^G_D$ be the set of functions $h : \{D\}^n \to \{0, 1\}$ of the form $h(x_1, \ldots, x_n) = f(\pi_{1,x_{i_1}} \cdot \ldots \cdot \pi_{t,x_{i_t}})$, where each $\pi_{j,x_{i_j}}$ is an element of a group G that depends only on j, x_{i_j} (where $i_j \in \{1, \ldots, n\}$) for some $f : G \to \{0, 1\}$. We call such functions h non-abelian programs.*

3.2 Primitives

Garbled Circuits. A garbling scheme, introduced by Yao [30] and formalized by Bellare et al. [5], enables a party to "encrypt" or "garble" a circuit in such a way that it can be evaluated on inputs – given tokens or "labels" corresponding to those inputs – without revealing what the inputs are.

▶ **Definition 4** (Garbling Scheme). *A projective garbling scheme is a tuple of efficient algorithms* $\mathtt{GC} = (\mathtt{garble}, \mathtt{eval})$ *defined as follows.*

$\mathtt{garble}(1^\lambda, \mathtt{C}) \to (\mathtt{GC}, \mathbf{K})$*: The garbling algorithm* garble *takes as input the security parameter* λ *and a boolean circuit* $\mathtt{C} : \{0,1\}^\ell \to \{0,1\}^m$*, and outputs a garbled circuit* GC *and* ℓ *pairs of garbled labels* $\mathbf{K} = (K_1^0, K_1^1, \ldots, K_\ell^0, K_\ell^1)$*. For simplicity we assume that for every* $i \in [\ell]$ *and* $b \in \{0,1\}$ *it holds that* $K_\ell^b \in \{0,1\}^\lambda$*.*

$\mathtt{eval}(\mathtt{GC}, K_1, \ldots, K_\ell) \to y$*: The evaluation algorithm* eval *takes as input the garbled circuit* GC *and* ℓ *garbled labels* $K_1, \ldots, K_\ell$*, and outputs a value* $y \in \{0,1\}^m$*.*

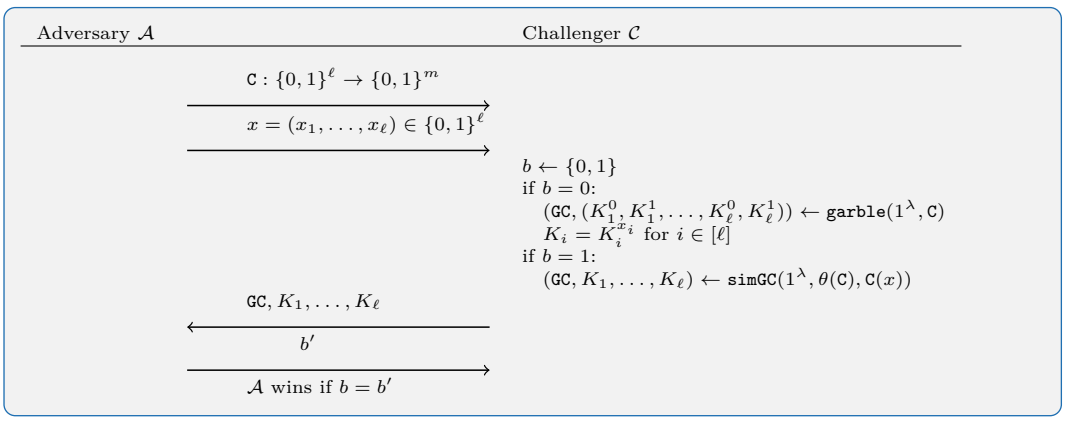
We require the following properties of a projective garbling scheme:

Correctness. We say GC satisfies *correctness* if for any boolean circuit $\mathtt{C} : \{0,1\}^\ell \to \{0,1\}^m$ and $x = (x_1, \ldots, x_\ell)$ it holds that $\Pr[\mathtt{eval}(\mathtt{GC}, \mathbf{K}[x]) \neq \mathtt{C}(x)] = negl(\lambda)$ where $(\mathtt{GC}, \mathbf{K}) \leftarrow \mathtt{garble}(1^\lambda, \mathtt{C})$ with $\mathbf{K} = (K_1^0, K_1^1, \ldots, K_\ell^0, K_\ell^1)$, and $\mathbf{K}[x] = (K_1^{x_1}, \ldots, K_\ell^{x_\ell})$.

Next, we formally define the security notions we require for a garbling scheme. When garbled circuits are used in such a way that decoding information is used separately, *obliviousness* requires that a garbled circuit together with a set of labels reveals nothing about the input the labels correspond to, and *privacy* requires that the additional knowledge of the decoding information reveals only the appropriate output. In our work, we do not consider decoding information separately (but rather, consider it to be included in the garbled circuit), so we do not need obliviousness.

Privacy. Informally, privacy requires that a garbled circuit together with a set of labels reveal nothing about the input the labels correspond to (beyond the appropriate output and the side-information). For our constructions, we assume the side-information to be the topology of the circuit, denoted as $\theta(C)$.

More formally, we say that GC satisfies *privacy* if there exists a simulator simGC such that for every PPT adversary $\mathcal{A}$, it holds that $\Pr[\mathcal{A} \text{ wins}] \leq \frac{1}{2} + negl(\lambda)$ in the following experiment:



4 Incremental Function Secret-Sharing

We begin by defining *incremental function secret-sharing (*IFSS*)*, which allows a set of parties to evaluate a hidden function on a joint input. Diverging from the original definition of function secret sharing [7], IFSS requires shares to be aggregated incrementally on a chain, in a $\mathcal{BC}$-efficient communication pattern. IFSS can be instantiated with garbled circuits or one-time truth tables and can be used in $\mathcal{BC}$-efficient protocols for particular function classes.

▶ **Definition 5** (Incremental Function Secret-Sharing). *An n-party incremental function secret-sharing (IFSS) scheme for a function class $\mathcal{F}$ is a pair of PPT algorithms* $(\texttt{Gen}, \texttt{Eval})$ *with the following syntax:*

- $\texttt{Gen}(1^\lambda, f)$*: On input security parameter 1^λ and function description $f \in \mathcal{F}$,* Gen *outputs keys $(k_1, \ldots, k_n)$;*
- $\texttt{Eval}(i, k_i, x, y_{i+1})$*: On input party index i, a key k_i, input string x, and the output of the next party y_{i+1}, the algorithm* Eval *outputs a value y_i;*

We require the following:

Correctness: For all $(f : \mathcal{X} \to \mathcal{Y}) \in \mathcal{F}, x \in \mathcal{X}$ we require that the following holds except with negligible probability:

$$\texttt{Eval}(1, k_1, x, \texttt{Eval}(2, k_2, x, \ldots, \texttt{Eval}(n, k_n, x, \bot))) = f(x)$$

Privacy: Let $\mathcal{H}$ be the set of honest parties. Then if $\vec{k} \leftarrow \texttt{Gen}(1^\lambda, f)$, we define $\vec{k}_{-\mathcal{H}}$ to be $\vec{k}$ where we replace, for all $i \in \mathcal{H}$, k_i with $\bot$. We also define $\texttt{Eval}^{\mathcal{H}}(\vec{k}, x)$ to compute, for $i = n, \ldots, 1$, $y_i = \texttt{Eval}(i, k_i, x, y_{i+1})$ (with $y_{n+1} = \bot$), and then output y_i for all $i \in \mathcal{H}$. We say that an IFSS satisfies privacy if, there exists a PPT simulator Sim such that for all $f \in \mathcal{F}, \mathcal{H} \subset [n], x \in \mathcal{X}$:

$$\{k_{-\mathcal{H}}, \texttt{Eval}^{\mathcal{H}}(\vec{k}, x) : \vec{k} \leftarrow \texttt{Gen}(1^\lambda, f)\}_{\lambda, f, x}, \text{ and } \{\texttt{Sim}(1^\lambda, \mathcal{H}, x, f(x))\}_{\lambda, f, x}$$

are (unconditionally or computationally) indistinguishable.

Bottleneck Complexity: We define the bottleneck complexity $\mathcal{BC}$ of an IFSS for $\mathcal{F}$ as the expected size of the largest y_i, for all $i \in [n], f \in \mathcal{F}, x \in \mathcal{X}$.

4.1 Instantiating IFSS

We show two instantiations of IFSS, one based on one-way functions and one with unconditional security.

With Unconditional Security. IFSS can be implemented with information theoretic security using an approach similar to the OTTT protocol [24] (as observed in [7]). The construction is as follows: $\texttt{Gen}(1^\lambda, f)$ chooses random vectors $T_1, \ldots, T_n$ whose dimensionality is $|\mathcal{X}_f|$, the size of the input domain of f, such that for all possible inputs $x \in \mathcal{X}_f$, $\sum_i T_i[x] = f(x)$. Gen then outputs $k_i = T_i$. The evaluation algorithm $\texttt{Eval}(i, k_i, x, y_{i+1})$ outputs $y_i = T_i[x] + y_{i+1}$ (for $y_{i+1} \neq \bot$ and $y_i = T_i[x]$ otherwise).

The protocol satisfies correctness since by construction $y_1 = \sum_i T_i[x] = f(x)$. It also satisfies unconditional privacy: The simulator $\texttt{Sim}(1^\lambda, \mathcal{H}, x, f(x))$ samples $k_{-\mathcal{H}} = \{T_i\}_{i \notin \mathcal{H}}$ as a set of uniform random strings of length $|\mathcal{X}_f|$, and random $\{z_i\}_{i \in \mathcal{H}}$ from $\mathcal{Y}_f$. Then it simulates the outputs of the Eval function as follows: it sets $y_1 = f(x)$, and $y_{i+1} = y_i - ((i \in$

$\mathcal{H})$? $z_i : T_i[x]$) [12] for all $i < n$, and finally outputs $(k_{-\mathcal{H}}, \{y_i\}_{i\in\mathcal{H}})$. Indistinguishability follows since in the simulation, like in the real protocol, the corrupt parties receive uniformly random k_i, and the y_i values are uniformly distributed under the constraint that y_1 is the result of the computation.

Note that since the constructions leaks $\mathcal{X}, \mathcal{Y}$, we assume that for all $f \in \mathcal{F}$ $\mathcal{X}_f = \mathcal{X}, \mathcal{Y}_f = \mathcal{Y}$. For this IFSS, $\mathcal{BC} = \mathcal{O}(\log|\mathcal{Y}|)$. (Note that the size of the keys can be exponential in the input size, namely $\mathcal{O}(\log|\mathcal{Y}| \cdot |\mathcal{X}|)$, like the original OTTT protocol).

From One-Way Functions. IFSS can be implemented from garbled circuits (which in turn can be implemented from one-way functions) by abstracting the "Phase 2" step of the protocol for abelian programs presented in [28]. The construction is as follows: the algorithm $\texttt{Gen}(1^\lambda, f)$ runs $(\texttt{GC}, \mathbf{K}) \leftarrow \texttt{garble}(1^\lambda, f)$. Then it picks uniformly random $\{\mathbf{K}_i\}_{i\in[n]}$ under the constraint that $\sum_i \mathbf{K}_i = \mathbf{K}$. Finally it outputs $k_i = \mathbf{K}_i$ for all $1 \neq i \in [n]$ and $k_1 = (\texttt{GC}, \mathbf{K}_1)$. The evaluation algorithm $\texttt{Eval}(i, k_i, x, y_{i+1})$, for all $i \neq 1$, selects the shares of the encoding information of $k_i = \mathbf{K}_i$ that correspond to x i.e., $\mathbf{K}_i[x] = ((K_i)_1^{x_1}, \ldots, (K_i)_\ell^{x_\ell})$ where $\ell = \lceil \log|\mathcal{X}| \rceil$, and finally outputs $y_i = \mathbf{K}_i[x] + y_{i+1}$ (for $y_{i+1} \neq \perp$ and $y_i = \mathbf{K}_i[x]$ otherwise). For $i = 1$ the Eval algorithm follows the instructions above to produce y_1, and finally outputs $\texttt{eval}(\texttt{GC}, y_1)$.

The protocol satisfies correctness since by construction $y_1 = \sum_i \mathbf{K}_i[x] = \mathbf{K}(x)$, and by correctness of the garbling scheme $\texttt{eval}(\texttt{GC}, \mathbf{K}[x]) = f(x)$ except with negligible probability.

It also satisfies computational privacy: The simulator $\texttt{Sim}(1^\lambda, \mathcal{H}, x, f(x))$ runs the simulator for the garbled circuits $(\texttt{GC}, \mathbf{Y}) \leftarrow \texttt{simGC}(1^\lambda, \theta(\mathcal{F}), f(x))$, where $\mathbf{Y}$ is the set of ℓ labels that make the simulated garbled circuit GC output $f(x)$. The simulator then picks $\{\mathbf{K}_i\}_{i\notin\mathcal{H}}$, a set of uniform random strings of the same length as $\mathbf{K}$, and random strings $\{z_i\}_{i\in\mathcal{H}}$ of the same length as $\mathbf{K}[x]$. Then it simulates the outputs of the Eval function as follows: it sets $y_1 = \mathbf{Y}$ and $y_{i+1} = y_i - ((i \in \mathcal{H})$? $z_i : \mathbf{K}_i[x])$ for all $i < n$, and finally outputs $(k_{-\mathcal{H}}, \{y_i\}_{i\in\mathcal{H}})$.

An adversary $\mathcal{A}$ that can distinguish between the real and simulated distribution can easily be used to break the privacy property of the underlying garbling scheme. The reduction $\mathcal{B}$ queries the GC challenger $\mathcal{C}$ on input f, x and receives $(\texttt{GC}, \mathbf{Y})$ in return. It then picks random $\mathbf{K}_i$ for all $i \notin [n]$ and computes the y_i as the simulator described above. The resulting distribution corresponds to the real protocol execution if the GC challenger sampled $b = 0$, or the simulated one if $b = 1$, therefore the reduction $\mathcal{B}$ wins in the GC privacy game with the same advantage as the IFSS adversary $\mathcal{A}$ distingushes between the real and simulated view.

Note that privacy of the GC scheme leaks some information $\theta(f)$ about the function f, therefore we assume for simplicity that for all $f \in \mathcal{F}$, $\theta(f) = \theta(\mathcal{F})$. However, given an upper bound of the size of $f \in \mathcal{F}$ it is possible to remove this requirement using universal circuits, albeit with an efficiency loss, and this is reflected in the Lemma below. For the GC based IFSS, $\mathcal{BC} = \mathcal{O}(\lambda \cdot \log|\mathcal{X}|)$. Note that the size of the keys is polynomial in the input size for this instantiation, namely $\mathcal{O}(\lambda \cdot (\log|\mathcal{X}| + |f|))$ for the first party and $\mathcal{O}(\lambda \cdot \log|\mathcal{X}|)$ for the others.

The discussion in this subsection can be summarized in the following:

▶ **Lemma 6.** *Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \mathcal{Y}$, λ the security parameter. Then for all n:*

- *It is possible to implement* IFSS *for $\mathcal{F}$ with $\mathcal{BC} = \mathcal{O}(\log|\mathcal{Y}|)$ with unconditional security.*
- *If one-way functions exist, it is possible to implement* IFSS *for $\mathcal{F}$ with $\mathcal{BC} = \mathcal{O}(\lambda \cdot \log|\mathcal{X}|)$*

[12] Here, a ? $b : c$ is used to denote the folllowing – if a holds, then b; else c.

In particular, the $\mathcal{BC}$ complexity in both cases is independent of n. Note however that the size of the correlated randomness in the variant with unconditional security is exponential in the input size.

Using IFSS in the compiler of [9]. The work of [9] presents a compiler that transforms an insecure protocol to secure protocol while preserving $\mathcal{BC}$. The compiler is based on the tool of "incremental FHE", which is similar to FHE except that its "joint" public key and decryption of ciphertext can be computed by incrementally combining shares provided by different parties. The main idea of the compiler is to execute the insecure protocol under the hood of (incremental) FHE to compute the encryption of the output (say ciphertext ct). Next, the parties combine their partial decryptions (computed locally by each party using its share of secret key) corresponding to ct in an incremental manner to reconstruct the final decrypted output.

We analyze whether this compiler can be viewed in terms of the common two-step approach of our constructions (elaborated in the technical overview, where the first step is to compute a "masked" aggregate common input and the second step is to use IFSS to evaluate a hidden function on this input). Recall that in our constructions, the first step involves masking the inputs using random values from setup (and either aggregating them by incremental computation or concatenating them) and the second step involves using IFSS to carry out the unmasking and compute the relevant function. On the other hand, in the above compiler, the first step uses FHE to compute the encryption of output ct directly. We observe that considering ct to be the common joint input, now IFSS can in fact be used to carry out the "decryption" function of FHE. In some more detail, IFSS could be used to evaluate the "hidden" function which has the secret decryption keys hardcoded and computes the decryption of the ciphertext ct. The above approach would result in making the compiler rely on correlated randomness, but would allow to instantiate it using any (non-incremental) FHE scheme. This shows that IFSS can serve as a general useful building block in $\mathcal{BC}$-efficient constructions.

5 Low $\mathcal{BC}$-complexity for Abelian Programs from IFSS

In this section, we generalize the results of [28] using the IFSS primitive defined above. Doing so allows us to achieve an information-theoretic $\mathcal{BC}$-efficient protocol for abelian programs.

Note that we can't use IFSS directly in MPC protocols for two reasons: first, all parties in IFSS would need to know all the inputs $\vec{X}$. This could be fixed by introducing a mask $\vec{R}$, reveal $\vec{X} + \vec{R}$ to all parties, and then modify the function so that it removes the mask securely inside the IFSS. The second issue is that revealing (even a potentially masked) $\vec{X}$ to all parties would lead to high $\mathcal{BC}$ since $\mathcal{O}(|\vec{X}|) = \mathcal{O}(n)$.

Recall that an abelian program h can be expressed as $h(\vec{X}) = f(\sum_{i=1}^{n} X_i)$ for some $f : G \to \{0, 1\}$, where G denotes an abelian group. We observe that the specific function class of abelian programs has a special structure that allows us to view it as a single-input function rather than an n-input function h. Exploiting this observation, we fix the above problems as follows: first, the trusted dealer picks a (single) random $R \in G$, defines the function $f'_R(Z) = f(Z - R)$, and then gives all parties P_i an additive share r_i of R, which they can use to mask their inputs, and an IFSS key k_i for the function f'_R.

In the protocol, the parties securely compute a masked sum of their inputs (say $Z = X + R$, where R denotes the mask and X denotes the sum of inputs) in a $\mathcal{BC}$-efficient way over a chain (similar to the protocol of [28]). The parties mask their inputs X_i with r_i as $Z_i = X_i + \mathsf{r}_i$, add

it to $\sum_{j=1}^{i-1} Z_j$, which they received from the previous party P_{i-1}, and send the result to P_{i+1}. The last party in the chain then can recover $Z = X + R$, and begins the second phase where each party P_i sends Z together with $y_i = \texttt{Eval}(i, k_i, Z, y_{i+1})$ to P_{i-1}, so that P_1 can finally retrieve $f'_R(Z) = f(Z - R) = f(X)$ Intuitively, security follows from the use of the masks and privacy of IFSS which hides which function (and therefore mask) was used. The $\mathcal{BC}$ of the protocol is inherited from the IFSS plus the size of elements in G, both independent of n.

The formal protocol appears in Figure 1 and we state the theorem below (whose proof appears in Section B).

Protocol $\Pi_{\texttt{abl}}$

Private input. Each party P_i has input $X_i \in G$ from group G.
Output. $y = f(\sum_{i=1}^{n} X_i)$, where the output is a single bit.
Correlated Randomness Setup. The setup involves the following:

1. For each $i \in [n]$, sample $\mathsf{r}_i \in G$, such that $\sum_{i=1}^{n} \mathsf{r}_i = R$.
2. Define the function $f'_R(Z)$ that computes $f(Z - R)$ on the input Z. Use an IFSS scheme to compute $(k_1, \ldots, k_n) \leftarrow \texttt{Gen}(1^\lambda, f'_R)$.
3. Output (r_i, k_i) to P_i for each $i \in [n]$.

The Protocol. The following steps are run in the online phase:
Phase 1 (Round 1 to Round n). *(Input Masking)* In round i, P_i does the following:

- If $i = 1$, let $V_i = \mathsf{r}_i + X_i$.
- If $i \neq 1$, let V_{i-1} denote the message received from P_{i-1} during the previous round. Compute $V_i \leftarrow V_{i-1} + X_i + \mathsf{r}_i$.
- If $i < n$, send V_i to P_{i+1}.
- If $i = n$, set $Z = V_n$.

Phase 2 (Round $n+1$ to Round $2n$) *(IFSS Evaluation)* Each P_i does the following in sequence, starting from $i = n$ to 1:

- If $i = n$, set $y_i = \texttt{Eval}(i, k_i, Z, \perp)$.
- If $i \neq n$, parse the message received from P_{i+1} in the previous round as (Z, y_{i+1}). Compute $y_i = \texttt{Eval}(i, k_i, Z, y_{i+1})$.
- If $i \neq 1$ Send (Z, y_i) to P_{i-1}.

Output Computation. P_1 sets the output $y = y_1$.
Phase 3 (Round $2n+1$ to $3n$) *(Output Transfer)* For i starting from 1 to n, each P_i does the following in sequence:

- If $i \neq 1$, let y denote the output received from P_{i-1} in previous round.
- If $i \neq n$, send y to P_{i+1}.
- Output y.

Figure 1 $\mathcal{BC}$-efficient protocol for Abelian Programs.

▶ **Theorem 7.** *Protocol $\Pi_{\texttt{abl}}$ securely computes the abelian program h against a semi-honest adversary corrupting upto $n-1$ parties. The $\mathcal{BC}$ of $\Pi_{\texttt{abl}}$ is $O(\log |G|)$ and $O(\lambda \log |G|)$ for the information-theoretic and the computational variant respectively.*

Lastly, we refer to the full version [26] for several protocols, which directly use $\Pi_{\mathtt{abl}}$ as a subprotocol in a $\mathcal{BC}$-efficient way. In particular, we present protocols to compute maximum among a set of values and any function of the maximum.

6 $\mathcal{BC}$-efficient MPC for tree-structured circuits

In this section, we build on the ideas of our generalized protocol for abelian programs in Figure 1 to formulate $\mathcal{BC}$-efficient protocols for additional classes of functions.

We present a protocol that using IFSS allows to evaluate, in a $\mathcal{BC}$-efficient way, functions that can be expressed as a tree of sub-functions, each taking inputs only from a subset of parties. This can be visualized as a tree with the leaves representing inputs and the non-leaf nodes representing functions. More specifically, the root of a sub-tree would represent the sub-function involving the values in the sub-tree. For our construction to be $\mathcal{BC}$ efficient, we require that each sub-function involves at most $o(n)$ inputs [13]. Further, the resulting tree depth $d = \log_{\mathsf{m}}(n)$ must also be sublinear in n. As a concrete example, the function $g = F\big(f_1(x_1^1, \ldots, x_{\mathsf{m}}^1), \ldots, f_{\mathsf{B}}(x_1^{\mathsf{B}}, \ldots, x_{\mathsf{m}}^{\mathsf{B}})\big)$, can be expressed as 2-level tree with $\mathsf{m} = \sqrt{n}$, where the f_i's denote the sub-functions at level 1 and F denotes the function (represented by the root) that aggregates the outputs of these sub-functions.

The main idea is that the subset of parties involved in computation of a single sub-function first interact among themselves to compute the outputs of this sub-function, which are later aggregated to compute the final output. In order to "hide" the outputs of these sub-functions (since they may not necessarily be leaked by the output), the sub-functions are tweaked to compute a "masked" output instead, using a mask chosen by the setup. When multiple "masked" outputs are taken as inputs to another sub-function (at a higher-level of the tree), the sub-function is further tweaked to unmask these values and then compute the function.

We observe that the above approach for an m-ary tree of depth $d = \log_{\mathsf{m}} n$ would result in $\mathcal{BC}$ of $\Omega(\mathsf{m} \log_{\mathsf{m}} n)$ Note that this term $\mathsf{m} \cdot \log_{\mathsf{m}} n$ incurs the least communication when $\mathsf{m} = 2$. Since choosing $\mathsf{m} = 2$ results in better $\mathcal{BC}$-efficiency, we focus only on binary trees in our formal protocol specification.

For ease of exposition, we use a slightly different naming convention and let the n parties be $P_0, \ldots, P_{n-1}$, with $n = 2^d$, and we consider a function $f(x_0, \ldots, x_{n-1})$ that can be decomposed with a binary tree of binary functions as explained below. Note that this can be easily generalized to the k-read setting by letting each party "control" k different inputs of the functions, but doing in the protocol description below would introduce unnecessarily cumbersome notation.

We start by introducing some useful notation to explain our protocol: We label the leaves of the binary tree with the bitstrings corresponding to the indices of the parties i.e., $0^d, 0^{d-1}1, \ldots, 1^d$, and we assign the input of each party to its corresponding leaf. (We will use integers and strings representing them interchangeably in the protocol description i.e., $P_0 = P_{0^d}, P_1 = P_{0^{d-1}1}, \ldots$). The internal nodes of the tree correspond to the functions into which f can be decomposed. To label the internal nodes/functions, we introduce the wildchar $*$, and we label the $n/2$ parents of the leaf nodes as $0^{d-1}*, 0^{d-2}1*, \ldots, 1^{d-1}*$, assigning one function to each such node. We continue introducing an extra wildchar $*$ every time we

[13] However, if there is a $\mathcal{BC}$-efficient protocol independent of the number of inputs (such as our protocol for abelian programs) that can be used to compute the sub-function, then our construction does not require the number of inputs to this sub-function to be sublinear in n.

climb a layer of the tree until we reach the root that gets labeled as $*^d$, corresponding to the function f_{*^d}. For simplicity, we assume that all the inputs and the outputs of all the functions in the tree are elements of the same group G.

We also introduce some notation to deal with strings with wildchars: We say a string $s \in \{0,1,*\}^d$ is valid if the wildchar $*$ is only followed by other $*$ wildchars (e.g., $0*$ is valid while $*0$ is not). Then, given a valid string s, we denote by $s|b$ the (valid) string s where the first wildchar $*$ is replaced by the bit b. Finally, given a valid string s we define $[s] \subseteq \{0,1\}^d$ to be the set of all strings that can be obtained when replacing the wildchars $*$ in s with bits.

We can now conveniently describe how to decompose the function $f(x_1,\ldots,x_n)$: for all valid strings $s \in \{0,1,*\}^d$ (starting with the parents of the leaves) we compute $x_s = f_s(x_{s|0}, x_{s|1})$, and finally we let the output be $f(x_1,\ldots,x_n) = f_{*^d}(x_{*^{d-1}0}, x_{*^{d-1}1}) = x_{*^d}$. In other words, we begin by pairing the leaf inputs two-by-two, then combine the results of these computations two-by-two climbing the tree until we reach the root.

We now need to address two issues in order to evaluate such functions *securely* and in a $\mathcal{BC}$*-efficient* way. First, we need to make sure that no intermediate values are leaked. This can be solved by assigning a mask $\mathsf{r}_{s|b}$ on each edge of the tree, such that the child function $f_{s|b}$ will mask its output with $\mathsf{r}_{s|b}$, and its parent function will de-mask the inputs before evaluating the function. That is, instead of evaluating $f_s(x_{s|0}, x_{s|1})$ we will evaluate using an IFSS scheme $f'_s(z_{s|0}, z_{s|1}) = f_s(z_{s|0} - \mathsf{r}_{s|0}, z_{s|1} - \mathsf{r}_{s|1}) + \mathsf{r}_s$ (where the root has no mask i.e., $\mathsf{r}_{*^d} = 0$). Second, to make sure that the overall protocol is $\mathcal{BC}$-efficient, we will only let the parties P_i with $i \in [s]$ participate in the secure evaluation of f'_s. Intuitively, this is fine since if all parties $i \in [s]$ are corrupt then they would already be able to compute all inputs and outputs in the subtree of the function f_s, thus it does not matter if those masks leak due to the fact that all parties involved in those IFSS computations are corrupt.

The formal description and details of our protocol appear in Section C.

References

1 Prabhanjan Ananth, Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. From FE combiners to secure MPC and back. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 199–228. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-36030-6_9.

2 David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc(1). In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 1–5. ACM, 1986. doi:10.1145/12130.12131.

3 David Arno Barrington. *Width-3 permutation branching programs.* Laboratory for Computer Science, Massachusetts Institute of Technology, 1985.

4 Amos Beimel, Ariel Gabizon, Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, and Anat Paskin-Cherniavsky. Non-interactive secure multiparty computation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 387–404. Springer, Heidelberg, August 2014. doi:10.1007/978-3-662-44381-1_22.

5 Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012. doi:10.1145/2382196.2382279.

6 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988. doi:10.1145/62212.62213.

7 Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1292–1303. ACM Press, October 2016. doi:10.1145/2976749.2978429.

8 Elette Boyle, Shafi Goldwasser, and Stefano Tessaro. Communication locality in secure multi-party computation - how to run sublinear algorithms in a distributed setting. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 356–376. Springer, Heidelberg, March 2013. doi:10.1007/978-3-642-36594-2_21.

9 Elette Boyle, Abhishek Jain, Manoj Prabhakaran, and Ching-Hua Yu. The bottleneck complexity of secure multiparty computation. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 24:1–24:16. Schloss Dagstuhl, July 2018. doi:10.4230/LIPIcs.ICALP.2018.24.

10 Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000. doi:10.1007/s001459910006.

11 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988. doi:10.1145/62212.62214.

12 Geoffroy Couteau. A note on the communication complexity of multiparty computation in the correlated randomness model. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 473–503. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17656-3_17.

13 Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 501–520. Springer, Heidelberg, August 2006. doi:10.1007/11818175_30.

14 Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 241–261. Springer, Heidelberg, August 2008. doi:10.1007/978-3-540-85174-5_14.

15 Ivan Damgård, Jesper Buus Nielsen, Antigoni Polychroniadou, and Michael Raskin. On the communication required for unconditionally secure multiplication. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 459–488. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53008-5_16.

16 Reo Eriguchi, Kazuma Ohara, Shota Yamada, and Koji Nuida. Non-interactive secure multiparty computation for symmetric functions, revisited: More efficient constructions and extensions. In Tal Malkin and Chris Peikert, editors, *CRYPTO*, 2021.

17 Rex Fernando, Yuval Gelles, Ilan Komargodski, and Elaine Shi. Maliciously secure massively parallel computation for all-but-one corruptions. In *CRYPTO 2022*, 2022.

18 Rex Fernando, Ilan Komargodski, Yanyi Liu, and Elaine Shi. Secure massively parallel computation for dishonest majority. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 379–409. Springer, Heidelberg, November 2020. doi:10.1007/978-3-030-64378-2_14.

19 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.

20 S. Dov Gordon, Tal Malkin, Mike Rosulek, and Hoeteck Wee. Multi-party computation of polynomials and branching programs without simultaneous interaction. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 575–591. Springer, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_34.

21 Shai Halevi, Yuval Ishai, Abhishek Jain, Eyal Kushilevitz, and Tal Rabin. Secure multiparty computation with general interaction patterns. In Madhu Sudan, editor, *ITCS 2016*, pages 157–168. ACM, January 2016. doi:10.1145/2840728.2840760.

22 Shai Halevi, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. Best possible information-theoretic MPC. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 255–281. Springer, Heidelberg, November 2018. doi:10.1007/978-3-030-03810-6_10.

23 Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 132–150. Springer, Heidelberg, August 2011. doi:10.1007/978-3-642-22792-9_8.

24 Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 600–620. Springer, Heidelberg, March 2013. doi:10.1007/978-3-642-36594-2_34.

25 Yuval Ishai, Manika Mittal, and Rafail Ostrovsky. On the message complexity of secure multiparty computation. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 698–711. Springer, Heidelberg, March 2018. doi:10.1007/978-3-319-76578-5_24.

26 Hannah Keller, Claudio Orlandi, Anat Paskin-Cherniavsky, and Divya Ravi. Mpc with low bottleneck-complexity: Information-theoretic security and more. Cryptology ePrint Archive, Paper 2023/683, 2023. URL: https://eprint.iacr.org/2023/683.

27 Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *33rd ACM STOC*, pages 590–599. ACM Press, July 2001. doi:10.1145/380752.380855.

28 Claudio Orlandi, Divya Ravi, and Peter Scholl. On the bottleneck complexity of mpc with correlated randomness. *International Conference on Practice and Theory of Public-Key Cryptography*, 2022.

29 Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018. doi:10.1109/FOCS.2018.00086.

30 Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982. doi:10.1109/SFCS.1982.38.

31 Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. doi:10.1109/SFCS.1986.25.

A Security Model

We prove the security of our protocols based on the standard real/ideal world paradigm. Essentially, the security of a protocol is analyzed by comparing what an adversary can do in the real execution of the protocol to what it can do in an ideal execution, that is considered secure by definition (in the presence of an incorruptible trusted party). In an ideal execution, each party sends its input to the trusted party over a perfectly secure channel, the trusted party computes the function based on these inputs and sends to each party its respective output. Informally, a protocol is secure if whatever an adversary can do in the real protocol (where no trusted party exists) can be done in the above described ideal computation. In this work, the adversary is assumed to be *passive* (alternately, referred to as being semi-honest) – the corrupt parties must follow the protocol specifications. However, the adversary attempts to learn private information by observing the view of the passively corrupt parties. We refer to [10] for further details regarding the security model.

In more detail, let Π be a protocol and $\mathcal{F}$ be a functionality. Let $\mathcal{I}$ denote the set of parties that are corrupt (of size at most $n-1$). The "ideal" world execution involves parties $\{P_1, \ldots, P_n\}$, an ideal adversary $\mathcal{S}$ who controls the parties in $\mathcal{I}$. The "real" world execution involves the PPT parties $\{P_1, \ldots, P_n\}$, and a real world adversary $\mathcal{A}$ who corrupts the parties in $\mathcal{I}$ passively. The *view* of a party in the real world is defined to be its random tape, together with all messages received during the execution of the protocol. In the ideal world, the

simulator $\mathcal{S}$ is given as input nothing but the corrupt parties' inputs sent to the trusted party and the outputs they receive from the trusted party. If $\mathcal{S}$ is able to "simulate" the real-world view with just this information, intuitively, security must hold. This is formalized below.

We define the following distributions of random variables.

$\texttt{REAL}_{\Pi}(1^{\lambda}, \mathcal{I}; x_1, \ldots, x_n)$: suppose Π is run with security parameter λ where each party P_i runs the protocol honestly using private input x_i. Let V_i denote the view of party P_i at the end of the protocol execution and let y_i denote the output of P_i. Output $\big(\{V_i\}_{i\in\mathcal{I}}, (y_1, \ldots, y_n)\big)$.

$\texttt{IDEAL}_{\mathcal{F},\mathcal{S}}(1^{\lambda}, \mathcal{I}; x_1, \ldots, x_n)$: Let $(y_1, \ldots, y_n) \leftarrow \mathcal{F}(x_1, \ldots, x_n)$. Output $\big(\mathcal{S}(I, \{x_i, y_i\}_{i\in\mathcal{I}}), (y_1, \ldots, y_n)\big)$

A protocol is secure against passive adversaries if the corrupted parties in the real world have views that are indistinguishable from their views in the ideal world.

▶ **Definition 8.** *A protocol Π securely realizes $\mathcal{F}$ if there exists a PPT ideal world adversary $\mathcal{S}$, such that for every subset of corrupt parties $\mathcal{I}$ and all inputs $x_1, \ldots, x_n$, the following two distributions are computationally indistinguishable:*

$$\texttt{REAL}_{\Pi}(1^{\lambda}, \mathcal{I}; x_1, \ldots, x_n) \overset{c}{\approx} \texttt{IDEAL}_{\mathcal{F},\mathcal{S}}(1^{\lambda}, \mathcal{I}; x_1, \ldots, x_n)$$

B Proof of Theorem 7

Let $\mathcal{I}$ and $\mathcal{H} = \mathcal{P}\backslash\mathcal{I}$ denote the set of indices corresponding to corrupt and honest parties respectively. Since we are running a protocol on a chain, it is useful to be able to talk about corrupt parties who receive messages from honest parties, and we therefore define $\mathcal{I}_L$ (resp. $\mathcal{I}_R$) to be the sets of all $i \in \mathcal{I}$ such that $i-1 \in \mathcal{H}$ (resp. $i+1 \in \mathcal{H}$).

To prove security, we define a simulator $\mathcal{S}$ that simulates the real-world view of the corrupt parties. Recall that $\mathcal{S}$ is given $(\mathcal{I}, \{x_i\}_{i\in\mathcal{I}}, y)$.

Setup simulation. Run $(\{k_i\}_{i\in\mathcal{I}}, \{y_i\}_{i\in\mathcal{H}}) \leftarrow \mathsf{simIFSS}(1^{\lambda}, \mathcal{H}, Z', y)$, where $\mathsf{simIFSS}$ denotes the simulator of the IFSS scheme's $\texttt{Gen}$ and $\texttt{Eval}$ functionality for a function class $\mathcal{F}$ computing $f'_R(Z) = f(Z-R)$ (note that the function class is independent of the value R in the function). Z' is chosen uniformly at random from the elements of G.

Additionally, for each $i \in \mathcal{I}$, sample r_i uniformly in G, and include (k_i, r_i) in the view of P_i.

Phase 1 Simulation. We need to simulate V_{i-1} for all $i \in \mathcal{I}_L$. We do so by choosing uniformly random V_{i-1} from G for all such $i \in \mathcal{I}_L$, except the largest one, which we denote by $\tilde{i}$, which we simulate by computing $V_{\tilde{i}-1} = Z' - \sum_{j\geq\tilde{i}}(X_j + \mathsf{r}_j)$ (In other words we define the message sent by the honest party with the largest index, to be consistent with the Z' which was chosen when simulating the IFSS, the input of the corrupt parties and their shares of R which were already defined during setup).

Phase 2 Simulation. We include in the view of all P_i with $i \in \mathcal{I}_R$ the tuple (Z', y_{i+1}), where y_{i+1} was received from the IFSS simulator.

Phase 3 Simulation. We include in the view of all P_i with $i \in \mathcal{I}_L$ the result y.

Below, we argue that the views of corrupt parties in the real and ideal world are indistinguishable via a series of intermediate hybrids:

- Hyb_0 : Same as the real-world execution.
- Hyb_1 : Same as Hyb_0, except that the values k_i for all $i \in \mathcal{I}$ and all y_{i+1} for $i \in \mathcal{I}_R$ are computed using the IFSS simulator on input $(\mathcal{H}, Z, f(x))$.
  This is in contrast to the previous hybrid, where the true IFSS evaluation is used instead of a simulator, changing the k_i of corrupt parties and y_i of honest parties. Indistinguishability follows from the privacy of IFSS.
- Hyb_2 : Same as Hyb_1, except that R is not used anymore to define the $\mathsf{r}_i : i \in \mathcal{I}$, which are instead just chosen at random from G. Since the r_i of the honest parties are not part of the view the two distributions are identically distributed.
- Hyb_3 : Same as Hyb_2, except that a random Z' is input to the IFSS simulator, and $V_i : i \in \mathcal{I}$ are simulated as decribed in "Phase 1 Simulation". This is in contrast to the previous hybrid, where Z is computed from the V_i values, and V_i are computed based on the parties' inputs. Since the r_i of the honest parties are not part of the view the two distributions are identically distributed. Note that in this hybrid we do not use the inputs of the honest parties anymore.

Since Hyb_3 corresponds to the simulated execution and each pair of consecutive hybrids are indistinguishable, this completes the proof that the views of corrupt parties in the real and ideal worlds are indistinguishable.

$\mathcal{BC}$-Analysis. We note that in Phase 1 and 3, the maximum communication complexity incurred by a party is $\log|G|$. In Phase 2, a party incurs the $\mathcal{BC}$ of the IFSS instance (in addition to $|Z| = \log|G|$), which is $O(1)$ for the information-theoretic instantiation and $O(\lambda \log|G|)$ for the computational instantiation. We can thus conclude that the resulting $\mathcal{BC}$ of the information-theoretic protocol for abelian programs is $O(\log|G|)$. The computational variant (which is the same as the construction in [28]) has a $\mathcal{BC}$ of $O(\lambda \log|G|)$.

C $\mathcal{BC}$-efficient protocol for tree-structured circuits

We present the formal description of the protocol (Figure 2). For convenience, we enhance the notation of the IFSS generation algorithm $\mathtt{Gen}$ to include an extra parameter $S \subseteq [n]$, which indicates which subset of parties should receive keys i.e., running $(k_0, \ldots, k_{n-1}) \leftarrow \mathtt{Gen}(1^\lambda, S, f)$ returns $|S|$ IFSS keys k_i for $i \in S$ and sets $k_i = \perp$ for $i \notin S$.

Correctness. Thanks to the correctness of the IFSS scheme the output of each node in the tree is computed correctly, meaning that the input masks are removed by f'_s before evaluating f_s and adding the output mask. Finally, since the mask of the root r_{*^*} is 0, the output of the final computation z_{*^*} is equal to $f(x_0, \ldots, x_{n-1})$.

$\mathcal{BC}$-Analysis. First, we note that to transfer the masked inputs, a party sends messages of size at most $\mathcal{O}(\log|G|)$. Next, consider evaluation of a specific sub-function. Here, transferring masked outputs would require a party to send messages of size at most $\mathcal{O}(\log|G|)$ along a chain. Next, the steps using IFSS incur communication of size at most $\mathcal{O}(\lambda \log|G|)$ for GC-based instantiation and $\mathcal{O}(\log|G|)$ for the OTTT-based instantiation. Since the above occurs for each level and there are $\log(n)$ levels, we can conclude that the overall $\mathcal{BC}$ of the protocol is $\mathcal{O}(\lambda \cdot \log|G| \cdot \log(n))$ for the computational variant and $\mathcal{O}(\log|G| \cdot \log(n))$ for the information-theoretic variant. The above discussion assumes balanced trees. If this is not the case, more generally, for depth d, the $\mathcal{BC}$ is $\mathcal{O}(\lambda \cdot \log|G| \cdot d)$ for the computational variant and $\mathcal{O}(\log|G| \cdot d)$ for the information-theoretic variant.

Protocol $\Pi_{\mathtt{tree}}$

Private input. There are $n = 2^d$ parties. Each party P_i, with $i = 0, \ldots, n-1$ has input x_i. We assume all inputs are from some goup G.

Correlated Randomness Setup.

1. For each valid string $s \in \{0,1,*\}^d$ choose a uniform random mask r_s from G, except for r_{*^d} which is set to 0.
2. For each valid string $s \in \{0,1,*\}^d \setminus \{0,1\}^d$ (e.g., for all nodes except the leaves) run the IFSS setup

$$(k_0^s, \ldots, k_{n-1}^s) \leftarrow \mathtt{Gen}(1^\lambda, [s], f'_s)$$

(remember that $k_i^s = \perp$ for all $i \notin [s]$) where f'_s is defined as

$$f'_s(z_{s|0}, z_{s|1}) = f_s(z_{s|0} - \mathsf{r}_{s|0}, z_{s|1} - \mathsf{r}_{s|1}) + \mathsf{r}_s$$

Finally, send to each party P_i their mask r_i and the keys k_i^s for all s such that $i \in [s]$.

The Protocol. The following steps are run in the online phase of the protocol:

1. *Transferring Masked Inputs for Leaf Nodes.*
   Each P_i sets $z_i = x_i + \mathsf{r}_i$ and sends it to their "sibling" party i.e., if $i = s|b$ send z_i to $P_{s|(1-b)}$.
2. *Climbing the Tree.*
   For all valid strings $s \in \{0,1,*\}^d \setminus \{0,1\}^d$ (e.g., for all intermediate nodes, starting with the parents of the leaves):
   a. *Evaluating the IFSS.*
      Let all parties P_i with $i \in [s]$ run the IFSS evaluation on inputs $z_{s|0}, z_{s|1}$ e.g., starting from the party with the highest index $i \in [s]$ and going backwards run:

$$y_i^s = \mathtt{Eval}(i, k_i^s, (z_{s|0}, z_{s|1}), y_{i+1}^s)$$

      (where as usual $y_j^s = \perp$ if j is "out of bounds").
   b. *Transfering Masked Outputs.*
      Let ι be the smallest index in $[s]$. Let all parties P_i with $i \in [s]$ learn the output $z_s = y_\iota^s$. E.g., all parties in $[s]$, starting from P_ι, send z_s to the next party in $[s]$.
   c. *Transfering Masked Inputs for Subtrees.* Each P_i with $i \in [s]$ sends z_s to one party in the "sibling" sub-tree i.e., if $i = s|b_1, \ldots, b_h$ (with h representing the height we have reached in the tree), then P_i sends z_s to P_j with $j = i = s|(1-b_1), \ldots, b_h$.

Figure 2 $\mathcal{BC}$-efficient protocol for tree-based formulas.

For a read-k tree-like structure (where a party's input could correspond to at most k leaves), the number of leaves is at most kn and the depth for a balanced tree is $\log(kn)$. This results in $\mathcal{BC}$ of $\mathcal{O}(k \cdot \lambda \cdot \log|G| \cdot \log(kn))$ for the computational variant and $\mathcal{O}(k \cdot \log|G| \cdot \log(kn))$ for the information-theoretic variant.

Privacy. Proving privacy of the tree-based construction requires building a simulator that can simulate the view of an adversary corrupting up to $n-1$ parties in the protocol. This can be done following the blueprint of the simulator of the protocol $\Pi_{\mathtt{abl}}$. That is, the simulator receives as input the output of the computation y as well as the inputs of the malicious parties. The simulator will pick random values for all edges on the tree and simulate the setup phase by running the simulator of the IFSS on those random inputs/outputs. Then, the simulator will provide masks to the adversary which are consistent with these random inputs.

In the online phase, the simulator will simulate the transfer of masked inputs of leaf nodes using the random values already chosen during setup. Then, the simulator includes in the view of the corrupted parties the values y_i^s provided by the IFSS simulator. Indistinguishability between the real protocol and the simulated execution can be argued by replacing, one by one, each real execution of IFSS with a simulated one. Indistinguishability in this first series of hybrids follows from the privacy guarantees of IFSS. In the next series of hybrids, we replace the masked inputs/outputs learned by the adversary in the protocol execution with uniformly random values from G. Since in this hybrid the masks of the honest parties are not used anymore (as the IFSS is simulated), this new series of hybrids are all unconditionally indistinguishable from their previous one. As the final hybrid of this series corresponds to the simulator, this concludes the argument.

The discussion above therefore leads to the following:

▶ **Theorem 9.** *Protocol $\Pi_{\mathtt{tree}}$ securely computes the aggregated function f against a semi-honest adversary corrupting upto $n-1$. The $\mathcal{BC}$ of $\Pi_{\mathtt{tree}}$ is $\mathcal{O}(k \cdot \log|G| \cdot \log(kn))$ and $\mathcal{O}(k \cdot \lambda \cdot \log|G| \cdot \log(kn))$ for the information-theoretic and computational variant respectively.*

Randomness Recoverable Secret Sharing Schemes

Mohammad Hajiabadi
Cheriton School of Computer Science, University of Waterloo, Canada

Shahram Khazaei
Department of Mathematical Sciences, Sharif University of Technology, Tehran, Iran

Behzad Vahdani
Department of Mathematical Sciences, Sharif University of Technology, Tehran, Iran

Abstract

It is well-known that randomness is essential for secure cryptography. The randomness used in cryptographic primitives is not necessarily recoverable even by the party who can, e.g., decrypt or recover the underlying secret/message. Several cryptographic primitives that support randomness recovery have turned out useful in various applications. In this paper, we study *randomness recoverable secret sharing schemes* (RR-SSS), in both information-theoretic and computational settings and provide two results. First, we show that while every access structure admits a perfect RR-SSS, there are very simple access structures (e.g., in monotone AC^0) that do not admit efficient perfect (or even statistical) RR-SSS. Second, we show that the existence of efficient computational RR-SSS for certain access structures in monotone AC^0 implies the existence of one-way functions. This stands in sharp contrast to (non-RR) SSS schemes for which no such results are known.

RR-SSS plays a key role in making advanced attributed-based encryption schemes randomness recoverable, which in turn have applications in the context of designated-verifier non-interactive zero knowledge.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques; Security and privacy → Mathematical foundations of cryptography

Keywords and phrases Secret sharing, Randomness recovery

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.12

Acknowledgements We would like to thank Sorush Bahariyan for bringing Fact 1 to our attention and Motahareh Gharahi for fruitful discussions on the proof of Theorem 20.

1 Introduction

Without randomness, secure cryptography is unachievable. The randomness used in cryptographic primitives is not necessarily, efficiently and even sometimes information-theoretically, recoverable. For example, the randomness used for an ElGamal ciphertext is not efficiently recoverable even by a party holding the secret key. On the other hand, several well-known constructions for PKE, such as the OAEP [5] and its variants[35, 8, 32] are randomness recoverable (RR). Another notable RR-PKE construction is Yao's construction [37] based on injective trapdoor functions (TDF).

RR-PKE schemes have found applications in constructing optimistic fair exchange protocols [30], signcryption schemes [28], proofs of correct decryptions in electronic-voting applications in [24] (to avoid heavy zero-knowledge proofs) and recently in CCA-secure PKE in [14].

In addition to PKE, RR variants of symmetric encryption schemes (SKE), attribute-based encryption (ABE) and garbled circuits (GC) have been studied in the literature [25, 13].


© Mohammad Hajiabadi, Shahram Khazaei, and Behzad Vahdani;
licensed under Creative Commons License CC-BY 4.0

1.1 RR secret sharing and motivations

In this paper, we initiate the study of secret sharing schemes (SSS) [34, 7] from a randomness recovery point of view. In addition to being an interesting notion on its own, it has applications in settings such as designated-verifier non-interactive zero-knowledge (DV-NIZK) for NP [29, 13], as we will discuss later.

Main results. We take the first steps toward delineating the notion of RR-SSS from both information-theoretic and computational perspectives. First, we show that while every access structure admits a perfect RR-SSS, there are very simple access structures (e.g., in AC^0) that do not admit efficient perfect RR-SSS. Our result also applies to the weaker security notions including *statistical security*. Second, we show that the existence of efficient computational RR-SSS for certain access structures in AC^0 implies one-way functions (OWF). Our second result provides strong evidence that realizing RR-SSS for AC^0 from assumptions not currently known to imply OWFs (e.g., worst-case complexity-type assumptions) may be impossible.

Applications of RR-SSS and motivations. Assuming the existence of RR-PKE, RR-SSS for access structures in NC^1 seems to be an important step towards single-key RR-ABE for circuits in P (see Section 1.5). Single-key RR-ABE for P, in turn, is sufficient for DV-NIZK for all NP[29, 13][1]. Currently, it is known how to base RR-ABE and DV-NIZK on CDH and LWE [29, 13] but it is still open whether they can be achieved using weaker primitives such as TDFs (which by [13] is implied by RR-PKE and hinting PRG [27]).

Moreover, RR-SSS can be useful in applications in which proofs of well-formedness are needed for recovered shares. This motivates the study of RR-SSS as an independent primitive.

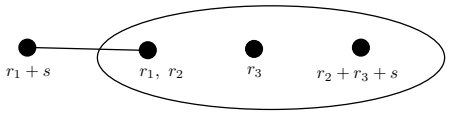
1.2 A perfect RR-SSS for every access structure

Let us first recall what an SSS is. In an SSS, a secret is shared among a set of participants by giving a share to each one. The shares are computed by applying a public rule on the secret and randomness. Only certain pre-specified subsets of participants are *qualified* to recover the secret and the secret must remain hidden from every other subset of participants. These requirements are called *correctness* and *privacy*, respectively, and can be defined either in the computational or information-theoretic setting. The set of all qualified subsets is called the *access structure* [18].

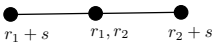
In an RR-SSS, we additionally require that every qualified set, in addition to the secret, is also able to recover the randomness.

The most well-known SSS, *Shamir's threshold scheme*, is RR. In Shamir's scheme, a secret $s \in \mathbb{F}$ is shared among a set of n participants as follows ($\mathbb{F}$ is a finite field with at least $n+1$ elements). The randomness $(r_1, \ldots, r_{t-1}) \in \mathbb{F}^{t-1}$ is chosen ($1 \leq t \leq n$), the polynomial $f(x) = s + r_1 x + r_2 x^2 + \ldots + r_{t-1} x^{t-1}$ is constructed, and the share $s_i = f(x_i)$ is given to participant $i \in \{1, \ldots, n\}$, where $x_1, \cdots, x_n$ are some distinct public elements of $\mathbb{F}$. It is easy to verify that only a subset A of size at least t is qualified to recover the secret using the shares $\{s_i\}_{i \in A}$. The corresponding access structure is called the (n, t)-*threshold access structure*. It is also easy to see that in Shamir's scheme, a qualified set recovers the polynomial $f(x)$, and hence, the randomness.

[1] Lombardi *et al.* [29] showed how to generically construct DV-NIZK from single-key weak function-hiding ABE. These sorts of ABE can be constructed from single-key RR-ABE [13].



(a) A non-RR-SSS.



(b) A RR-SSS.

Figure 1 Access structure 1a contains a minimal set of size 2 and a minimal set of size 3. Access structure 1b contains two minimal sets of size 2. The secret is a single bit s and r_is are the randomness. The value written under each party, indicates that party's share.

Not every SSS is RR. For example, consider the well-known Ito-Saito-Nishizeki construction in [18] for a general access structure, which we refer to as the *ISN1*. The secret is a single bit $s \in \mathbb{F}_2$ and the randomness is

$$\mathcal{R} = \{r_{A,i} \mid A \text{ is a minimal qualified set and } i \in A\},$$

where a qualified set is called *minimal* if none of its proper subsets are qualified. The $r_{A,i}$'s are randomly chosen bits subject to $\sum_{i \in A} r_{A,i} = s$. The share of a participant i is

$$s_i = \{r_{A,i} \mid \text{there exists a minimal qualified set } A \text{ such that } i \in A\} .$$

It is easy to verify that the construction is information-theoretically both correct and private. However, as shown in Figure 1, the ISN1 construction is not RR in general.

A perfect RR-SSS construction. A natural question to ask is whether every access structure admits a perfect (i.e., information-theoretically secure) RR-SSS. The answer to this question is not entirely trivial, but in the following, we show that another general construction, also introduced by Ito-Saito-Nishizeki in [17] which we refer to as the *ISN2*, is RR.

The secret is again a single bit $s \in \mathbb{F}_2$ and the randomness is

$$\mathcal{R} = \{r_B \mid B \text{ is a maximal unqualified set}\},$$

where an unqualified set is called *maximal* if every proper superset of it is qualified. The r_B's are randomly chosen bits. The share of a participant i is

$$s_i = \left(s + \sum_B r_B, \{r_B \mid B \text{ is a maximal unqualified set and } i \notin B\}\right) .$$

It is easy to verify that the construction is both perfectly correct and perfectly private. Also, a minimal qualified set recovers the whole randomness.

▶ **Fact 1.** *The ISN2 construction [17] is RR.*

1.3 Results on perfect RR-SSS

We study the RR variant of some questions that have been extensively studied for (standard) perfect SSSs.

On Beimel's conjecture for RR-SSSs. The *information ratio*, defined to be the ratio between the largest share size and the secret size, is an important parameter that measures the efficiency of a SSS. Both ISN1 and ISN2 constructions have exponential information ratios in the number of participants. A long-standing open problem in the theory of secret sharing is to answer whether exponential upper bound is inevitable. Beimel [3] has conjectured that this is the case.

▶ **Conjecture 2** (Beimel). *There exists an $\varepsilon > 0$ such that, for every integer n, there is an access structure with n participants such that every perfect SSS that realizes it has information ratio $2^{\Omega(n^{\varepsilon})}$.*

Surprisingly, the best-known lower bound, due to Csirmaz [10], is $\Omega(n/\log n)$. We prove that an exponential lower bound holds for perfect RR-SSSs.

▶ **Theorem 3** (Exponential lower bound for perfect RR-SSS). *For every integer n, there is an access structure with n participants such that every perfect RR-SSS that realizes it has information ratio $2^{\Omega(n)}$.*

We prove the theorem for an access structure on n participants, which is the union of $n/3$ disjoint $(3,3)$-threshold access structures (see Figure 2); but the result holds in general, i.e., for the union of n/k disjoint (k,k)-thresholds for every $k \geq 2$. Similarly to Csirmaz, we use the so-called *Shannon-type information inequalities* to prove an exponential lower bound on the information ratio of this access structure for perfect RR-SSSs.

On weaker security notions. Several non-perfect security notions for secret sharing have been proposed in the literature. It is well-known [20, Theorem 36] that any lower bound derived using information inequalities applies not only to perfect security but also to standard relaxations such as quasi-perfect [20, Chapter 5], almost-perfect [21, 11], and statistical security. The exponential lower bound of Theorem 3 is also valid for these relaxations because we only use (Shannon-type) information inequalities in the proof.

Ruling out the existence of efficient perfect RR-SSS for mAC^0. Access structures are in 1-1 correspondence with monotone circuits. The mAC^0 class consists of all monotone circuits of depth $O(1)$ and polynomial size, with AND/OR gates with unbounded fan-in. Unfortunately, the above result shows that we cannot have efficient perfect RR-SSS for access structures even in mAC^0.

On contrary, the class of access structures admitting efficient perfect (standard) SSSs is much richer. In particular, it contains mNC^1, the class of monotone circuits of depth $O(\log n)$ and polynomial size with AND/OR gates with a maximum fan-in of 2, which is known to strictly contain mAC^0. We refer to [4] for further discussion on the class of efficient perfect SSSs. It is open whether every access structure in mP, the class of monotone circuits of polynomial size with AND/OR gates with unbounded fan-in, admits an efficient perfect SSS.

1.4 Results on computational RR-SSS

In a computational SSS [33], we require that the sharing and reconstruction algorithms be polynomial-time in the security parameter and the number of participants. Furthermore, we require that a polynomial-time adversary cannot distinguish between the shares of an unqualified set for every pair of secrets.

An unpublished result by Yao shows that assuming the existence of *one-way functions*, every access structure in mP admits an efficient computational SSS. The construction is a generalization of the results of Benaloh and Leichter [6] that constructs a perfect SSS for polynomial-size monotone formulae. We refer to [36] for details of the construction. It is open whether (efficient) computational SSS for any class of access structures implies OWFs. Assuming the existence of OWFs, an unpublished result of Rudich shows that computational SSS for mNP implies oblivious transfer; see [3, 26].

OWFs from RR-SSS for AC^0. As we mentioned above, it is still open whether computational (standard) SSS for any class of access structures implies OWFs. One main obstacle to proving this possibly true statement is that the existence of efficient perfect SSS for every access structure has not yet been (unconditionally) ruled out, even though it is generally believed not to be the case, as it has been manifested in Beimel's conjecture (Conjecture 2). However, by our result on the exponential lower bound for RR-SSS (Theorem 3), the situation for RR-SSS is different. We use the method developed by Impagliazzo and Luby in [16], together with a variant of Csirmaz's framework [10] for lower bounding the information ratio of perfect SSSs adapted for the computational setting, to prove that existence of computational RR-SSS for certain access structures in AC^0 implies the existence of OWFs.

Construction of computational RR-SSS. A perfect linear SSS can be converted into a computational RR-SSS using a one-time KDM-secure SKE naturally and straightforwardly. For the sake of completeness, in Section 5, we state this formally. In that section, we introduce a type of PRG with a KDM-like security which turns out convenient in constructing a simple computational RR-SSS from a perfect linear SSS with the same access structure.

1.5 Applications of RR-ABE

The notion of RR-SSS was implicitly used as a key tool to obtain randomness recoverable single-key attribute-based public-key encryption schemes [29, 13], which in turn imply DV-NIZK for all NP [29]. Let us recall the definition of ABE. We have a master public key mpk and a master secret key msk. For any attribute string x, we have an attribute secret key sk_x, obtained as $\mathsf{KGen}(msk, x)$, where KGen is the key generation algorithm of the ABE. We encrypt a message m under mpk and a given circuit C to get a ciphertext ct. Now someone who has sk_x can decrypt ct to get m iff $C(x) = 1$.

We say that the ABE is RR if when $C(x) = 1$, then sk_x not only recovers m, but also all the randomness used by the encryption algorithm.

In the single-key security notion, an adversary can ask for only one attribute secret key sk_x, and has to win in an indistinguishability sense against a challenger who encrypts with respect to some circuit C where $C(x) = 0$.

A standard way to build single-key RR-ABE is as follows: if $|x| = n$, then the master secret key has n PKE secret keys $(sk_1, ..., sk_n)$ and mpk contains the corresponding public keys $(pk_1, ..., pk_n)$. An attribute secret key for x contains those sk_i where $x_i = 1$. To encrypt m under mpk and C, we share m according to C to get the shares. We then encrypt each share under pk_i, and return all the ciphertexts. The notion of RR-SSS is a key tool in realizing randomness recoverability for the above single-key ABE scheme, as it allows us to recover the randomness used by sharing process, a major source of the overall randomness.

2 Preliminaries

In this section, we present the necessary background.

2.1 Random variables

We denote random variables (RV) by boldface characters and use $\mathsf{supp}(\boldsymbol{X})$ to denote the support of RV $\boldsymbol{X}$. We use the terms RV and distribution interchangeably throughout the paper. The Shannon entropy of $\boldsymbol{X}$ is denoted by $\mathrm{H}(\boldsymbol{X})$. The entropy of $\boldsymbol{X}$ conditioned on RV $\boldsymbol{Y}$ is denoted and defined by $\mathrm{H}(\boldsymbol{X}|\boldsymbol{Y}) := \mathrm{H}(\boldsymbol{X}, \boldsymbol{Y}) - \mathrm{H}(\boldsymbol{Y})$. The mutual information between $\boldsymbol{X}, \boldsymbol{Y}$ is defined and denoted by $\mathrm{I}(\boldsymbol{X} : \boldsymbol{Y}) := \mathrm{H}(\boldsymbol{X}) - \mathrm{H}(\boldsymbol{X}|\boldsymbol{Y})$.

Let us also recall the *functional representation lemma* [12, page 626], a well-known lemma in information theory, that will be used in this paper. We use the notation $\boldsymbol{X} \equiv \boldsymbol{Y}$ for identically distributed RVs.

▶ **Lemma 4** (Functional representation lemma [12]). *For every pair of jointly distributed RVs $(\boldsymbol{X}, \boldsymbol{Y})$, there exists a RV $\boldsymbol{R}$, independent of $\boldsymbol{X}$, and a mapping μ such that $(\boldsymbol{X}, \boldsymbol{Y}) \equiv \big(\boldsymbol{X}, \mu(\boldsymbol{X}, \boldsymbol{R})\big)$*

▶ Remark 5. Throughout the paper, we will consider a *non-uniform* model of computation, however, our results hold true for the *uniform* model.

We call the family $\boldsymbol{X} = \{\boldsymbol{X}_\lambda\}_{\lambda\in\mathbb{N}}$ of RVs *efficiently sampleable* if there exists a family of polynomial-time algorithms $\mathsf{Sample} = \{\mathsf{Sample}_\lambda\}_{\lambda\in\mathbb{N}}$ such that $\mathsf{Sample}_\lambda(1^\lambda) \equiv \boldsymbol{X}_\lambda$. We call λ the *security parameter* and refer to $\boldsymbol{X}$ as a *family of RVs*, or simply an RV, indexed by the security parameter. We recall that a function $\varepsilon : \mathbb{N} \to \mathbb{R}^{\geq 0}$ is called *negligible* if for every $d > 0$ there exists some λ_0 such that for every $\lambda > \lambda_0$ it holds that $\varepsilon(\lambda) < \frac{1}{\lambda^d}$.

▶ **Definition 6** (Computational indistinguishablity). *Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be efficiently sampleable distributions indexed by the security parameter λ. We say that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are* computationally indistinguishable *and write $\boldsymbol{X}_\lambda \overset{c}{\equiv} \boldsymbol{Y}_\lambda$ if for every family of polynomial-time size circuits $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ (i.e., $\mathcal{D}_\lambda$ has polynomially many gates in the security parameter), there exists a* negligible function ε *such that*

$$|\Pr[\mathcal{D}_\lambda(\boldsymbol{X}_\lambda) = 1] - \Pr[\mathcal{D}_\lambda(\boldsymbol{Y}_\lambda) = 1]| \leq \varepsilon(\lambda) \ .$$

We usually drop the security parameter and write $\boldsymbol{X} \overset{c}{\equiv} \boldsymbol{Y}$ for $\boldsymbol{X}_\lambda \overset{c}{\equiv} \boldsymbol{Y}_\lambda$, and $\mathcal{D}(\boldsymbol{X}_\lambda)$ or $\mathcal{D}(\boldsymbol{X})$ instead of $\mathcal{D}_\lambda(\boldsymbol{X}_\lambda)$.

We will also face functions of the form $\varepsilon(n, \lambda)$, indexed by two parameters, which we require them to be polynomial in n and negligible in λ (e.g., to be of the form $\mathsf{poly}(n)\mathsf{negl}(\lambda)$), where n will be the number of participants in secret sharing schemes. To remove any confusion, we make the definition precise.

2.2 One-way function

▶ **Definition 7** (OWF). *A function $f : \{0,1\}^\star \to \{0,1\}^\star$ is called a one-way function (OWF) if the following two conditions hold:*

1. *There is a polynomial-time algorithm that on input x outputs $f(x)$.*
2. *For every polynomial-size circuit family $\{\mathcal{C}_\lambda\}_\lambda$, the following probability is negligible:*

$$\Pr[f(\mathcal{C}_\lambda(f(\boldsymbol{U}_\lambda))) = f(\boldsymbol{U}_\lambda)].$$

The following lemma is due to Impagliazzo, Levin, and Luby [15]. It was used by Impagliazzo and Luby in [16] to prove that short-key SKE implies OWF. In Section 4, we use this lemma, in a similar manner, to prove that computational RR-SSS for AC^0 implies the existence of OWF.

▶ **Lemma 8** ([15]). *If there is a polynomial-time computable function $f : \{0,1\}^\lambda \to \{0,1\}^{l(\lambda)}$, a polynomial-time samplable distribution $\boldsymbol{D} = \{\boldsymbol{D}_\lambda\}_\lambda$ and a constant $d > 0$ such that $f(\boldsymbol{U}_\lambda) \overset{c}{\equiv} \boldsymbol{D}_\lambda$ and for large enough λ, $H(\boldsymbol{D}_\lambda) \geq H(f(\boldsymbol{U}_\lambda)) + 1/\lambda^d$, then there is a OWF.*

2.3 Access structure

In the secret sharing context, there is *set of participants*, which we denote by P, and a distinguished participant called the *dealer*, which we denote by $0 \notin P$.

▶ **Definition 9** (Access structure). *A non-empty subset $\Gamma \subseteq 2^P$, with $\emptyset \notin \Gamma$, is called an* access structure *on P if it is* monotone*; that is, $A \subseteq B \subseteq P$ and $A \in \Gamma$ imply that that $B \in \Gamma$. A subset $A \subseteq P$ is called* qualified *if $A \in \Gamma$; otherwise, it is called* unqualified*. A qualified subset is called* minimal *if none of its proper subsets is qualified. An unqualified subset is called* maximal *if every proper superset of it is qualified.*

There is a natural one-to-one correspondence between access structures with n participants and monotone Boolean functions with n variables.

2.4 Secret sharing

A secret sharing scheme (SSS) can be defined in the following two equivalent ways. The first definition is more useful for working in the information-theoretic setting, while the second one is more useful in the computational setting.

▶ **Definition 10** (SSS in terms of jointly distributed RVs). *A tuple $\left(\boldsymbol{S}_i\right)_{i\in P\cup\{0\}}$ of jointly distributed RVs is called a* SSS *on the set of participants P when $|\mathsf{supp}(\boldsymbol{S}_0)| \geq 2$. The RV $\boldsymbol{S}_0$ is called the* secret *RV and its support is called the* secret space*. The RV $\boldsymbol{S}_i$ is called the* share *RV of the participant $i \in P$ and its support is called his* share space*.*

▶ **Definition 11** (SSS in terms of sharing map). *Let $\mu : \mathcal{S}_0 \times \mathcal{R} \to \left(\mathcal{S}_i\right)_{i\in P}$ be a mapping and $\boldsymbol{R}$ be a distribution on $\mathcal{R}$, called the* randomness RV*. We refer to $\Pi = (\boldsymbol{R}, \mu)$ as a SSS if $|\mathcal{S}_0| \geq 2$. We call μ the* sharing map *and $\mathcal{R}$ the* randomness space*. Also, $\mathcal{S}_0$ is called the secret space and $\mathcal{S}_i$ is called the share space of participant i.*

The equivalence between these two definitions follows by the functional representation lemma (Lemma 4).

The following notation will be used throughout the paper.

▶ **Notation 12.** *For a SSS $\Pi = \left(\boldsymbol{S}_i\right)_{i\in P\cup\{0\}}$ and a subset $A \subseteq P$, we use the notation $\boldsymbol{S}_A$ for the projection of Π on the components in A; i.e., $\boldsymbol{S}_A := \left(\boldsymbol{S}_i\right)_{i\in A}$. Also, for a sharing map $\mu : \mathcal{S}_0 \times \mathcal{R} \to \left(\mathcal{S}_i\right)_{i\in P}$, μ_A stands for the projection of μ on the components in A. That is, if $(s_i)_{i\in P} = \mu(s, r)$, then $\mu_A(s, r) := (s_i)_{i\in A}$.*

Linear SSS. We call a SSS with sharing map $\mu : \mathcal{S}_0 \times \mathcal{R} \to \left(\mathcal{S}_i\right)_{i\in P}$ and randomness $\boldsymbol{R}$ *linear* when $\mathcal{R}$ and all $\mathcal{S}_i$'s, $i \in P \cup \{0\}$, are vector spaces over a common finite field, μ is a linear map and $\boldsymbol{R}$ is uniformly distributed over $\mathcal{R}$. Throughout the paper, for simplicity, we assume that he underline finite field is the binary field.

2.5 Security definitions for SSSs

The security of a SSS can be defined both in information-theoretic and computational settings.

▶ **Definition 13** (Perfect security). *We say that $\Pi = \left(\boldsymbol{S}_i\right)_{i\in P\cup\{0\}}$ is a* perfect SSS *for an access structure Γ, if the following two conditions hold:*

- ***Perfect correctness:*** $\mathrm{H}(\boldsymbol{S}_0|\boldsymbol{S}_A) = 0$ *for every qualified set $A \in \Gamma$.*
- ***Perfect privacy:*** $\mathrm{I}(\boldsymbol{S}_0 : \boldsymbol{S}_B) = 0$ *for every unqualified set $B \notin \Gamma$.*

If Π is a perfect SSS for Γ, we also say that Π *realizes* Γ perfectly or Γ *admits* Π perfectly.

Computational secret sharing is defined to realize a family $\Gamma = \{\Gamma_n\}_{n\in\mathbb{N}}$ of access structures, where Γ_n is an access structure with n participants with participants set P_n. A computational SSS for Γ is a tuple $\Pi = \big(\boldsymbol{R}, \mu\big)$ with

$$\begin{aligned} \boldsymbol{R} &= \{\boldsymbol{R}_{\lambda,n}\}_{n,\lambda\in\mathbb{N}}\ , \\ \mu &= \{\mu_{\lambda,n} : \mathcal{S}_{0,\lambda,n} \times \mathcal{R}_{\lambda,n} \to \big(\mathcal{S}_{i,\lambda,n}\big)_{i\in P_n}\}_{\lambda,n\in\mathbb{N}}\ , \end{aligned}$$

where for every $\lambda, n \in \mathbb{N}$, the tuple $\big(\boldsymbol{R}_{\lambda,n}, \mu_{\lambda,n}\big)$ is a secret sharing scheme with participant set P_n.

▶ **Definition 14** (Computational security)**.** *Let $\Gamma = \{\Gamma_n\}_{n\in\mathbb{N}}$ be a collection of access structures and $\Pi = \big(\boldsymbol{R}, \mu\big)$ with $\boldsymbol{R} = \{\boldsymbol{R}_{\lambda,n}\}_{\lambda,n}$ and $\mu = \{\mu_{\lambda,n} : \mathcal{S}_{0,\lambda,n} \times \mathcal{R}_{\lambda,n} \to \big(\mathcal{S}_{i,\lambda,n}\big)_{i\in P_n}\}_{\lambda,n\in\mathbb{N}}$ be a family of SSSs indexed by the security parameter λ and n. We say that Π is a* computational SSS *for Γ if the following conditions hold:*

- ***Efficient randomness sampling:*** *The RV $\boldsymbol{R}$ is polynomial-time sampleable in λ and n.*
- ***Polynomial secret length:*** *$\log|\boldsymbol{S}_{0,\lambda,n}|$ is polynomial in λ and n.*
- ***Efficient sharing:*** *The sharing map $\mu_{\lambda,n}$ is polynomial-time computable in λ and n.*
- ***Efficient secret reconstruction:*** *There exists a polynomial-time algorithm* Recon *in λ and n such that for every polynomial $n = n(\lambda)$ there exists a negligible function* negl *such that for every sequence of qualified sets $\{A_\lambda \in \Gamma_{n(\lambda)}\}_\lambda$ and every sequence of secrets $\{s_\lambda \in \mathcal{S}_{0,\lambda,n(\lambda)}\}_\lambda$ one has*

$$\Pr[\mathsf{Recon}(\mu_{A_\lambda}(s_\lambda, \boldsymbol{R}_{\lambda,n(\lambda)})) \neq s_\lambda] \leq \mathsf{negl}(\lambda). \tag{1}$$

- ***Computational privacy:*** *For every polynomial $n = n(\lambda)$, every sequence of unqualified sets $\{B_\lambda \notin \Gamma_{n(\lambda)}\}_\lambda$ and every pair of secret sequences $\{s_\lambda \in \boldsymbol{S}_{0,\lambda,n(\lambda)}\}_\lambda$ and $\{s'_\lambda \in \boldsymbol{S}_{0,\lambda,n(\lambda)}\}_\lambda$, one has*

$$\mu_{B_\lambda}(s_\lambda, \boldsymbol{R}_{\lambda,n(\lambda)}) \overset{\mathrm{c}}{\equiv} \mu_{B_\lambda}(s'_\lambda, \boldsymbol{R}_{\lambda,n(\lambda)}). \tag{2}$$

If Π is a computational SSS for Γ we say that Π *realizes* Γ computationally or Γ *admits* Π computationally.

▶ Remark 15. In the rest of the paper, in the computational setting, we implicitly take the access structure, qualified sets and unqualified sets to be parameterized by n, and take the secret space, share space, randomness space, and RVs over these spaces to be parameterized by n and λ. In particular, we drop the indices in (1) and (2) and simply write: $\Pr[\mathsf{Recon}(\mu_A(s, \boldsymbol{R})) \neq s]$ is negligible and $\mu_B(s, \boldsymbol{R}) \overset{\mathrm{c}}{\equiv} \mu_B(s', \boldsymbol{R})$. This simplifies the notation and allows us to state some of the properties of the computational and perfect SSS in a unified manner. Additionally, in the rest of paper, when we consider aysmptotic properties of the scheme, we implicitly assume that n is a polynomial in λ.

The following lemma will be used later in the paper. We refer to Appendix A for the proof.

▶ **Lemma 16.** *Let $\Pi = (\mu, \boldsymbol{R})$ be a computational SSS for Γ with t-bit secrets and let $\boldsymbol{S}$ be an RV independent of $\boldsymbol{R}$ over the secret space. Then, for every $B \notin \Gamma$,*

$$(\boldsymbol{S}, \mu_B(\boldsymbol{S}, \boldsymbol{R})) \overset{\mathrm{c}}{\equiv} (\boldsymbol{S}, \mu_B(0^t, \boldsymbol{R})).$$

2.6 Information ratio

The efficiency of SSSs is usually measured using a parameter called *information ratio*. The information ratio of an SSS with participants set P, secret space $\mathcal{S}_0$ and share space $\mathcal{S}_i$ for participant $i \in P$, is defined to be $\max_{i \in P} \frac{\log |\mathcal{S}_i|}{\log |\mathcal{S}_0|}$.

The *perfect information ratio*, or simply information ratio, of an access structure is defined to be the infimum of all information ratios of all SSSs that perfectly realize it.

Beimel [3] has conjectured that there are families of access structures with exponential information ratio in the number of participants; see Conjecture 2.

▶ Remark 17. Beimel has also stated the conjecture in terms of *share size* instead of information ratio in [3]; this corresponds to the case where the secret is a single bit. There are access structures whose information ratio for exponentially-long secrets (in the number of participants) may be significantly better than the information ratio achievable for short secrets [2]. Nevertheless, it is widely believed that the stronger conjecture (i.e., for information ratio) holds true.

Csirmaz framework for lower bounding information ratio. Following [23, 9], Csirmaz proposed a framework in [10] to prove lower bounds on the information ratio of perfect SSSs. His framework is captured in the following lemma which is based on the properties of the entropy function as well as the correctness and privacy properties of perfect SSSs.

▶ **Lemma 18** (Csirmaz/Perfect). *Let $\Pi = (\boldsymbol{S}_i)_{i \in P \cup \{0\}}$ be a perfect SSS for an access structure Γ. For every subset $A \subseteq P \cup \{0\}$, let $f(A) = \frac{\mathrm{H}(\boldsymbol{S}_A)}{\mathrm{H}(\boldsymbol{S}_0)}$. Then, the following holds:*

1. **Non-negativity.** *$f(A) \geq 0$ for every $A \subseteq P \cup \{0\}$.*
2. **Monotonicity.** *$f(A) \geq f(B)$ for every $B \subseteq A \subseteq P \cup \{0\}$.*
3. **Submodularity.** *$f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ for every $A, B \subseteq P \cup \{0, \}$.*
4. **Strong monotonicity.** *$f(A) \geq f(B) + 1$ for every $A \in \Gamma$ and $B \subseteq A$ such that $B \notin \Gamma$.*
5. **Strong submodularity.** *$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) + 1$ for every $A, B \in \Gamma$ such that $A \cap B \notin \Gamma$.*

If, using the inequalities (1)–(5), one can prove that for some participant $i \in P$, it holds that $f(\{i\}) \geq \sigma$, then σ will be a lower bound on the information ratio of the underlying access structure.

2.7 Randomness recoverable SSS

We call a SSS $\Pi = (\boldsymbol{R}, \mu)$ *randomness recoverable (RR)* if qualified sets, in addition to the secret, can also recover the randomness; that is, there exists a function $\mathsf{RNDrecover}$ such that for every qualified set A, $\Pr[\mathsf{RNDrecover}(\mu_A(\boldsymbol{R}, s)) = \boldsymbol{R}] = 1$ for every secret s. When Π is a computational SSS, we require that $\mathsf{RNDrecover}$ be a polynomial-time algorithm, in the security parameter and n, the number of participants; we also allow a negligible amount of error; i.e., when n is a polynomial in λ, then $\Pr[\mathsf{RNDrecover}(\mu_A(s, \boldsymbol{R})) \neq \boldsymbol{R}]$ is negligible in the security parameter.

The following claim will be used in Section 3 and Section 4.

▷ Claim 19. If $\Pi = (\boldsymbol{S}_i)_{i \in P \cup \{0\}}$ is an RR-SSS with perfect correctness (i.e., zero reconstruction error probability), then for every pair of qualified sets A, B, we have $\mathrm{H}(\boldsymbol{S}_A) = \mathrm{H}(\boldsymbol{S}_B)$, or equivalently $f(A) = f(B)$, using the notation of Lemma 18.

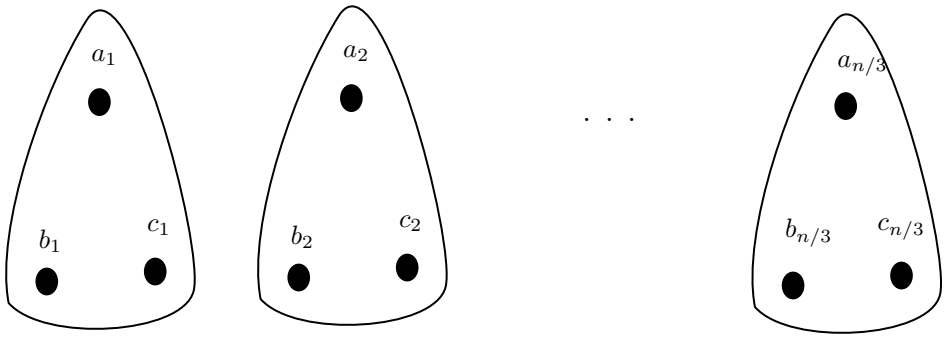


Figure 2 The Moon-Moser access structure.

Proof. Denote the support of $\boldsymbol{S}_i$ by $\mathcal{S}_i$, for $i \in P \cup \{0\}$. Let $(\boldsymbol{R}, \mu)$, with $\mu : \mathcal{S}_0 \times \mathcal{R} \to (\mathcal{S}_i)_{i \in P}$, be the equivalent SSS in terms of Definition 11, which exists by the functional representation lemma (Lemma 4); that is, $(\boldsymbol{S}_0, (\boldsymbol{S}_i)_{i \in P}) \equiv (\boldsymbol{S}_0, \mu(\boldsymbol{S}_0, \boldsymbol{R}))$. For simplicity, let us assume that $(\boldsymbol{S}_i)_{i \in P} = \mu(\boldsymbol{S}_0, \boldsymbol{R})$. Since $\boldsymbol{S}_P$ is a function of the secret and randomness and every qualified set can recover both of them, it follows that $\mathrm{H}(\boldsymbol{S}_P | \boldsymbol{S}_A) = 0$, or equivalently $\mathrm{H}(\boldsymbol{S}_A) = \mathrm{H}(\boldsymbol{S}_P)$, for every qualified set A. The claim then follows. ◁

3 Exponential lower bound for perfect RR-SSS

In this section, we show that the Moon-Moser access structure, to be defined below, has an exponential information ratio for every perfect RR-SSS that realizes it. The result also applies to weaker security notions such as statistical security as will be discussed at the end of this section.

The Moon-Moser access structure. Due to an old result by Moon and Moser [31], any graph with n vertices has at most $3^{n/3}$ maximal independent sets. A graph with exactly $3^{n/3}$ maximal independent sets is easy to construct: simply take the disjoint union of $n/3$ triangle graphs. Motivated by this example, we consider the access structure in Figure 2, which is the union of $n/3$ $(3,3)$-threshold access structures, and refer to it as the Moon-Moser access structure. Clearly, this access structure lies in AC^0.

▶ **Theorem 20.** *For every n, there is an access structure in AC^0 such that every perfect RR-SSS that realizes it has information ratio $2^{\Omega(n)}$.*

We first present a notation and a claim and then prove the theorem.

Notation. Denote the set of participants of the Moon-Moser access structure, with n participants, by $P = \{a_1, b_1, c_1, \ldots, a_{n/3}, b_{n/3}, c_{n/3}\}$ and let $\{a_i, b_i, c_i\}$ be a minimal qualified set for every $i = 1, \ldots, n/3$ (see Figure 2). Let $\Pi = (\boldsymbol{S}_i)_{i \in P \cup \{0\}}$ be a perfect RR-SSS for this access structure and f be as in Lemma 18. For a participant $p_i \in \{a_i, b_i, c_i\}$, we define p_i' and p_i'' to be the cyclic rotations of p_i by one and two positions, respectively; i.e., $a_i'' = b_i' = c_i$, $b_i'' = c_i' = a_i$ and $c_i'' = a_i' = b_i$. Also, we denote a set $\{p_{i_1}, \ldots, p_{i_k}\}$ simply by $p_{i_1} \cdots p_{i_k}$.

▷ Claim 21. For every qualified set A, every $k = 0, 1, \ldots, n/3$, and all choices for $p_1, \ldots, p_k$ with $p_i \in \{a_i, b_i, c_i\}$, the following inequality holds:

$$f(A) \quad \geq \quad f(p_1 p'_1 \ldots p_k p'_k) + 3^{n/3-k} \ . \tag{3}$$

Proof of Claim 21. First, let us show that the following inequality is implied by Inequality (3):

$$f(A) \quad \geq \quad f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p_k) + 2 \times 3^{n/3-k} \ . \tag{4}$$

By Inequality (3) we have:

$$\begin{array}{lll} f(A) & \geq & f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p_k p'_k) + 3^{n/3-k} \ , \\ f(A) & \geq & f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p''_k p_k) + 3^{n/3-k} \ . \end{array}$$

Also by the monotonicity property, we have

$$\begin{array}{l} f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p_k p'_k) + f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p''_k p_k) \geq \\ \qquad\qquad f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p_k p'_k p''_k) + \quad f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p_k) \ . \end{array}$$

Notice that $p_1 p'_1 \ldots p_{k-1} p'_{k-1} p_k p'_k p''_k$ is qualified and, hence, by Claim 19 we have

$$f(A) = f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p_k p'_k p''_k).$$

Therefore, Inequality (4) follows by adding the above three inequalities.

Now, we prove Inequality (3) by backward induction on k.

Base. Denote $m = n/3$. For $k = m$, by strong submodularity property, we have:

$$\begin{array}{l} f(p''_1 p_1 p'_1 \ldots p_m p'_m) + f(p''_2 p_1 p'_1 \ldots p_m p'_m) \geq \\ \qquad\qquad f(p''_1 p''_2 p_1 p'_1 \ldots p_m p'_m) + \quad f(p_1 p'_1 \ldots p_m p'_m) + 1 \ . \end{array}$$

Since the sets $p''_1 p_1 p'_1 \ldots p_m p'_m$, $p''_2 p_1 p'_1 \ldots p_m p'_m$ and $p''_1 p''_2 p_1 p'_1 \ldots p_m p'_m$ are all qualified, for every qualified set A, by Claim 19, we have:

$$f(A) = f(p''_1 p_1 p'_1 \ldots p_m p'_m) = f(p''_2 p_1 p'_1 \ldots p_m p'_m) = f(p''_1 p''_2 p_1 p'_1 \ldots p_m p'_m) \ .$$

Therefore,

$$f(A) \geq f(p_1 p'_1 \ldots p_m p'_m) + 1 \ ;$$

that is, Inequality (3) holds for $k = n/3$.

Induction. Now suppose that by the induction hypothesis

$$f(A) \geq f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p_k p'_k) + 3^{n/3-k} \ .$$

By Inequality (4), we also have:

$$f(A) \geq f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p''_k) + 2 \times 3^{n/3-k} \ .$$

By the monotonicity property, we have

$$\begin{array}{l} f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p_k p'_k) + f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p''_k) \geq \\ \qquad\qquad f(p_1 p'_1 \ldots p_{k-1} p'_{k-1} p_k p'_k p''_k) + \quad f(p_1 p'_1 \ldots p_{k-1} p'_{k-1}) \ . \end{array}$$

By adding the above three inequalities, noticing that $p_1 p'_1 \ldots p_{k-1} p'_{k-1} p_k p'_k p''_k$ is qualified, and using Claim 19, we get:

$$f(A) \geq f(p_1 p'_1 \ldots p_{k-1} p'_{k-1}) + 3^{n/3-(k-1)} \ ;$$

that is, Inequality (3) holds for $k - 1$. This completes the proof of Claim 21. ◁

Proof of Theorem 20. Let $p_i \in \{a_1, b_1, c_1, \ldots, a_{n/3}, b_{n/3}, c_{n/3}\}$. By letting $k = 0$ and $A = \{p_i, p'_i, p''_i\}$ in Inequality (3), we have:

$$f(p_i p'_i p''_i) \geq 3^{n/3} \ .$$

Also, $f(p_i) + f(p'_i) + f(p''_i) \geq f(p_i p'_i p''_i)$. Therefore, for every $i \in \{1, \ldots, n/3\}$, for at least one $p \in \{a_i, b_i, c_i\}$, we have

$$f(p) \geq 3^{n/3-1} \ .$$ ◀

▶ Remark 22. The above proof can be converted, in a straightforward manner, to a proof for the case of an access structure that is the union of n/k disjoint (k, k)-thresholds. Stated explicitly, every perfect RR-SSS that realizes the access structure that has

$$\{a_{1,1}, a_{1,2}, \cdots, a_{1,k}\}, \{a_{2,1}, a_{2,2}, \cdots, a_{2,k}\}, \cdots, \{a_{n/k,1}, a_{n/k,2}, \cdots, a_{n/k,k}\}$$

as its minimal qualified sets has information-ratio $2^{\Omega(n \log k/k)}$. The best exponent is achieved for $k = 3$, which justifies our choice for the Moon-Moser access structure in this section.

Exponential lower bound for non-perfect RR-SSSs. Besides perfect and computational security, several non-perfect security notions for secret sharing have appeared in the literature, including *almost-perfect*, *quasi-perfect*, and *statistical*. We refer to [19] for a comprehensive study of these security notions. Kaced [20, Theorem 36] has shown that any lower bound derived on the information ratio of (standard) SSSs using information inequalities applies not only to perfect security but also to quasi-perfect security (which can be shown to apply to almost-perfect and statistical security too). His result can also be extended to the case of RR-SSSs. Since, we only used (Shannon-type) information inequalities to derive our exponential lower bound on perfect RR-SSS, it also holds for all mentioned non-perfect security notions.

4 Computational RR-SSS for AC^0 implies OWF

In this section, we show that the existence of computational RR-SSS for some access structures in AC^0 implies the existence of OWFs. Our method is similar to Impagliazzo and Levin's method for proving that short-key SKE implies OWFs [16]. The idea is as follows: if $\Pi = (\mu, \boldsymbol{R})$ is a SSS for an access structure where B is unqualified, then $\boldsymbol{S}||\mu_B(\boldsymbol{S}, \boldsymbol{R})$ and $\boldsymbol{S}'||\mu_B(\boldsymbol{S}, \boldsymbol{R})$ are computationally indistinguishable, where $\boldsymbol{S}$ and $\boldsymbol{S}'$ are independent uniform RVs over the secret space. Indeed, when the SSS is perfect, $\mu_B(\boldsymbol{S}, \boldsymbol{R})$ reveals no information about $\boldsymbol{S}$ and so the two distributions are information-theoretically indistinguishable. But when the SSS is computational, $\mu_B(\boldsymbol{S}, \boldsymbol{R})$ reveals some information about $\boldsymbol{S}$. If this amount is not negligible, then we have two distributions that are computationally indistinguishable but statistically distinguishable and we can apply Lemma 8 to deduce the existence of OWF.

In Section 3, it was shown that there are access structures in AC^0 that do not admit efficient perfect RR-SSSs. In other words, an RR-SSS for such an access structure, that perfectly hides the secret from unqualified sets, has to have shares with exponential length. Hence intuitively, in a computational RR-SSS for such an access structure (because shares are of polynomial length), there are unqualified sets that obtain a considerable amount of *information* about the secret. This intuition is exactly phrased and proved in this section.

For simplicity, we first study the simpler case where in the definition of computational SSS (Definition 14), we require the reconstruction error probability to be equal to zero.

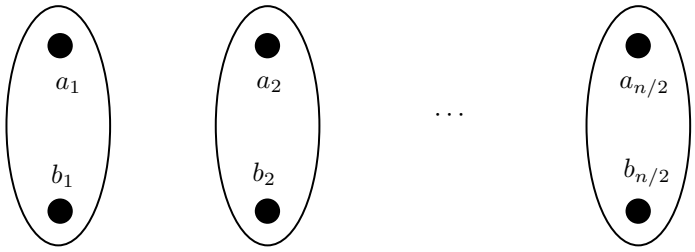


Figure 3 Union of $(2,2)$-threshholds.

4.1 Zero reconstruction error

In this subsection, we present a lemma, a claim, and a corollary for computational SSSs with zero reconstruction errors. These results are modified in Subsection 4.2 to consider non-zero reconstruction error and will be used in Subsection 4.3 to prove the main result of this section.

A variant of Csirmaz's framework (see lemma 18) adapted to the computational setting with perfect correctness (i.e., zero reconstruction error) is needed. The following lemma states this variant.

▶ **Lemma 23** (Csirmaz/Computational/Perfect correctness). *Let $\Pi = (\boldsymbol{S}_i)_{i\in P\cup\{0\}}$ be a computational SSS with perfect correctness for an access structure Γ. For $A, B \subseteq P \cup \{0\}$, denote $H(\boldsymbol{S}_A)$ with $H(A)$ and $H(\boldsymbol{S}_A|\boldsymbol{S}_B)$ with $H(A|B)$, respectively. Then, the non-negativity, monotonicity, and submodularity properties hold as in Lemma 18 and, one has the following modified formulation of strong monotonicity and strong submodularity:*

1. **Strong monotonicity.** *$H(A) \geq H(B) + H(0|B)$ for every $A \in \Gamma$ and $B \subset A$ such that $B \notin \Gamma$.*
2. **Strong submodularity.** *$H(A)+H(B) \geq H(A\cup B)+H(A\cap B)+H(0|A\cap B)$ for every $A, B \in \Gamma$ such that $A \cap B \notin \Gamma$.*

Proof. Inequality (1) holds because A is qualified and due to the monotonicity property:

$$H(A) = H(\{0\} \cup A) \geq H(\{0\} \cup B) = H(B) + H(0|B).$$

Inequality (2) follows from the following relations:

$$\begin{aligned} H(A) + H(B) &= H(\{0\} \cup A) + H(\{0\} \cup B) \\ &\geq H(\{0\} \cup A \cup B) + H(\{0\} \cup (A \cap B)) \\ &\geq H(A \cup B) + H(A \cap B) + H(0|A \cap B). \end{aligned}$$

In the first equality, we have used the fact that A and B are qualified. The first and second inequalities follow by the submodularity and monotonicity properties, respectively. ◀

Notation. In what follows, let $P = \{a_1, b_1, a_2, b_2, \cdots, a_{n/2}, b_{n/2}\}$ and Γ be an access structure with minimal qualified sets $\{a_1, b_1\}, \cdots, \{a_{n/2}, b_{n/2}\}$ (see Figure 3). Note that this access structure lies in AC^0. According to Remark 22, Γ's information ratio is $2^{\Omega(n)}$. For $p_i \in \{a_i, b_i\}$, let p'_i be the other element of $\{a_i, b_i\}$; i.e., if $p_i = a_i$ then $p'_i = b_i$ and if $p_i = b_i$ then $p'_i = a_i$. Also denote $\{p_1, p_2, \cdots, p_k\}$ with $p_1 p_2 \cdots p_k$ and use the notation in Lemma 23 for entropies.

▷ Claim 24. Let Π be a computational RR-SSS with perfect correctness for Γ and A be a qualified set in Γ with $H(A) \leq c$. Then for all $k = 0, 1, \cdots, n/2$,

$$H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_k) + \frac{c}{2^k} \geq H(A),$$

where $\boldsymbol{p}_i$ is a uniform RV over $\{a_i, b_i\}$ and $\boldsymbol{p}_i$'s are independent.

Proof. We prove the claim by induction on k.

Base. The base ($k = 0$) holds by the assumption.

Induction. Suppose that by the induction hypothesis we have:

$$H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_k) + \frac{c}{2^k} \geq H(A), \tag{5}$$

where $k < n/2$. By the submodularity property

$$H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_k a_{k+1}) + H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_k b_{k+1}) \geq H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_k a_{k+1}b_{k+1}) + H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_k). \tag{6}$$

Since $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_k, a_{k+1}, b_{k+1}\}$ is qualified, then according to Claim 19, we have:

$$H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_k a_{k+1}b_{k+1}) = H(A). \tag{7}$$

Summing up relations (5),(6) and (7), we get:

$$H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_k a_{k+1}) + H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_k b_{k+1}) + \frac{c}{2^k} \geq 2H(A).$$

So:

$$H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_k\boldsymbol{p}_{k+1}) + \frac{c}{2^{k+1}} = \frac{1}{2}\big(H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_k a_{k+1}) + H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_k b_{k+1})\big) + \frac{c}{2^{k+1}} \geq H(A)\ .$$ ◁

The following corollary could be considered as a quantitative contrapositive for Theorem 20.

▶ **Corollary 25.** *Let Π be a computational RR-SSS for Γ with perfect correctness and m-bit secrets and let n be a polynomial in λ. Then for large enough λ:*

$$\frac{m}{2} \geq H(0|\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_{n/2}),$$

where $\boldsymbol{p}_i$ is a uniform RV over $\{a_i, b_i\}$ and $\boldsymbol{p}_i$'s are independent.

Proof. Sharing algorithm's running time and m are polynomials, so for large enough λ we have $2^{\frac{n}{2}-1}m \geq H(A)$, where A is an arbitrary qualified set. Applying Claim 24 to this inequality, if follows that

$$H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_{n/2}) + \frac{m}{2} \geq H(A). \tag{8}$$

On the other hand, $\{\boldsymbol{p}'_1, \boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_{n/2}\}$ and $\{\boldsymbol{p}'_2, \boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_{n/2}\}$ are qualified sets, while $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_{n/2}\}$ is not. So, according to the (computational) strong submodularity property,

$$H(\boldsymbol{p}'_1\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_{n/2}) + H(\boldsymbol{p}'_2\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_{n/2}) \geq H(\boldsymbol{p}'_1\boldsymbol{p}'_2\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_{n/2}) + H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_{n/2}) + H(0|\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_{n/2}).$$

Applying Claim 19, we get

$$H(A) \geq H(\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_{n/2}) + H(0|\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_{n/2}).$$

Summing up the above inequality and Inequality (8), one gets the desired result. ◀

4.2 Non-zero reconstruction error

In this subsection, we provide variants of Lemma 23, Claim 24, and Corollary 25 that do not assume zero reconstruction error.

When the reconstruction error is zero, the entropy of the secret conditioned on the share of a qualified set is zero, because in this case, the secret is determined by the qualified set's share. When we allow the reconstruction algorithm to fail with some bounded probability, this property no longer holds. The following is a variant of Fano's inequality that we will use to prove that in this case, conditioned on the share of a qualified set, the entropy of the secret is $o(1)$.

▶ **Lemma 26.** *Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be families of RVs such that $\boldsymbol{Y}$ has polynomial length and f be a function such that $\Pr[\boldsymbol{Y} \neq f(\boldsymbol{X})]$ is negligible. Then $H(\boldsymbol{Y}|\boldsymbol{X})$ is $o(1)$.*

Proof. Define the indicator RV $\boldsymbol{Z}$ as follows:

$$\boldsymbol{Z} = \begin{cases} 1 \text{ if } \boldsymbol{Y} = f(\boldsymbol{X}) \\ 0 \text{ if } \boldsymbol{Y} \neq f(\boldsymbol{X}) \end{cases}.$$

Since $H(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{Y}) = 0$, we have:

$$\begin{aligned}
H(\boldsymbol{Y}|\boldsymbol{X}) &= H(\boldsymbol{Y}|\boldsymbol{X}) + H(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{Y}) \\
&= H(\boldsymbol{Y},\boldsymbol{Z}|\boldsymbol{X}) \\
&= H(\boldsymbol{Z}|\boldsymbol{X}) + H(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{Z}) \\
&\leq H(\boldsymbol{Z}) + \sum_{x\in\mathsf{Supp}(\boldsymbol{X})} \big(\Pr[\boldsymbol{X}=x,\boldsymbol{Z}=0]H(\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{Z}=0) \\
&\quad + \Pr[\boldsymbol{X}=x,\boldsymbol{Z}=1]H(\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{Z}=1)\big) \\
&= o(1) + \sum_{x\in\mathsf{Supp}(\boldsymbol{X})} \Pr[\boldsymbol{X}=x,\boldsymbol{Z}=0]H(\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{Z}=0) \qquad (9) \\
&\leq o(1) + \left(\sum_{x\in\mathsf{Supp}(\boldsymbol{X})} \Pr[\boldsymbol{X}=x,\boldsymbol{Z}=0]\right)\log(|\mathsf{Supp}(\boldsymbol{Y})|) \qquad (10) \\
&= o(1) + \Pr[\boldsymbol{Z}=0]\log(|\mathsf{Supp}(\boldsymbol{Y})|) \\
&= o(1). \qquad (11)
\end{aligned}$$

Equation (9) holds for two reasons: First, $\boldsymbol{Z}$ is a Bernouli RV with $\Pr[\boldsymbol{Z}=0] = o(1)$ (indeed, this probability is negligable), so $H(\boldsymbol{Z}) = o(1)$; Second, when $\boldsymbol{Z}=1$, $\boldsymbol{Y}$ is determined by $\boldsymbol{X}$; therefore, $H(\boldsymbol{Y}|\boldsymbol{X}=x,\boldsymbol{Z}=1)=0$. Inequality (10) holds because $H(\boldsymbol{Y}) \leq \log(|\mathsf{Supp}(\boldsymbol{Y})|)$. Equality (11) holds because $\Pr[\boldsymbol{Z}=0]$ is negligable and $\boldsymbol{Y}$ has polynomial length. ◀

▶ **Lemma 27.** *Let $\Pi = (\mu, \boldsymbol{R})$ be a computational SSS and $\boldsymbol{S}_0$ be an RV over the secret space. Then for every qualified set A, $H(\boldsymbol{S}_0|\mu_A(\boldsymbol{S}_0,\boldsymbol{R})) = o(1)$.*

Proof. Let Recon be the reconstruction algorithm. Then $\Pr[\mathsf{Recon}(\mu_A(\boldsymbol{S}_0,\boldsymbol{R})) \neq \boldsymbol{S}_0]$ is negligible in the security parameter. Also, the length of $\boldsymbol{S}_0$ is polynomial in the security parameter. Therefore, according to Lemma 26, $H(\boldsymbol{S}_0|\mu_A(\boldsymbol{S}_0,\boldsymbol{R})) = o(1)$. ◀

The following is a variant of Claim 19 that does not assume zero reconstruction error.

▷ Claim 28. Let $\Pi = (\mu, \boldsymbol{R})$ be a computational RR-SSS, n be a polynomial in λ and $\boldsymbol{S}_0$ be an RV over the secret space. Then for any two qualified sets A and B, $|H(\mu_A(\boldsymbol{S}_0, \boldsymbol{R})) - H(\mu_B(\boldsymbol{S}_0, \boldsymbol{R}))| = o(1)$.

Proof. By Lemma 27, $H(\boldsymbol{S}_0|\mu_A(\boldsymbol{S}_0, \boldsymbol{R})) = o(1)$. Because Π is RR, it can be proved that similarly

$$H(\boldsymbol{R}|\mu_A(\boldsymbol{S}_0, \boldsymbol{R})) = o(1).$$

Therefore, $H(\boldsymbol{S}_0, \boldsymbol{R}|\mu_A(\boldsymbol{S}_0, \boldsymbol{R})) = o(1)$ and, hence, $H(\boldsymbol{S}_0, \boldsymbol{R}) \leq H(\mu_A(\boldsymbol{S}_0, \boldsymbol{R})) + o(1)$. On the other hand, $\mu_A(\boldsymbol{S}_0, \boldsymbol{R})$ is determined by $\boldsymbol{S}_0$ and $\boldsymbol{R}$; thus $H(\mu_A(\boldsymbol{S}_0, \boldsymbol{R})) \leq H(\boldsymbol{S}_0, \boldsymbol{R})$. Similar bounds hold for $\mu_B(\boldsymbol{S}_0, \boldsymbol{R})$. The claim follows from these bounds. ◁

The following is a variant of Csirmaz's computational framework (23) stated for the case of the non-zero reconstruction error.

▶ **Lemma 29** (Csirmaz/Computational). *Let $\Pi = (\boldsymbol{S}_i)_{i \in P \cup \{0\}}$ be a computational SSS for an access structure Γ. Then, the non-negativity, monotonicity, and submodularity properties hold as in Lemma 18 and, one has the following modified formulation of strong monotonicity and strong submodularity:*

1. **Strong monotonicity.** *$H(A) + o(1) \geq H(B) + H(0|B)$ for every $A \in \Gamma$ and $B \subset A$ such that $B \notin \Gamma$.*
2. **Strong submodularity.** *$H(A) + H(B) + o(1) \geq H(A \cup B) + H(A \cap B) + H(0|A \cap B)$ for every $A, B \in \Gamma$ such that $A \cap B \notin \Gamma$.*

Proof. Inequality (1) follows from the following relations:

$$H(A) + o(1) = H(\{0\} \cup A) \geq H(\{0\} \cup B) = H(B) + H(0|B).$$

The left-hand side equality follows from Lemma 28. The rest is as in the proof of Lemma 23. Inequality (2) follows from the following relations:

$$\begin{aligned} H(A) + H(B) + o(1) &= H(\{0\} \cup A) + H(\{0\} \cup B) \\ &\geq H(\{0\} \cup A \cup B) + H(\{0\} \cup (A \cap B)) \\ &\geq H(A \cup B) + H(A \cap B) + H(0|A \cap B). \end{aligned}$$

The equality follows from Lemma 28. The rest is as in the proof of Lemma 23. ◀

Below is a modification of Claim 24 stated for the case of the non-zero reconstruction error.

▷ Claim 30. Let Π be a computational RR-SSS for Γ and A be a qualified set such that $H(A) \leq c$. Then for $k = 0, 1, \cdots, n/2$ one has:

$$H(\boldsymbol{p}_1 \boldsymbol{p}_2 \cdots \boldsymbol{p}_k) + \frac{c}{2^k} + o(1) \geq H(A),$$

where $\boldsymbol{p}_i$ is a uniform RV over $\{a_i, b_i\}$ and $\boldsymbol{p}_i$'s are independent.

Proof. Proof of this claim is achieved by applying appropriate and straightforward modifications to the proof of Claim 24. Explicitly, Claim 19 is used there to deduce $H(\boldsymbol{p}_1 \boldsymbol{p}_2 \cdots \boldsymbol{p}_k a_{k+1} b_{k+1}) = H(A)$. Instead, we apply Claim 28 to deduce

$$H(\boldsymbol{p}_1 \boldsymbol{p}_2 \cdots \boldsymbol{p}_k a_{k+1} b_{k+1}) + o(1) \geq H(A).$$

Also, the induction hypothesis should be modified to include the term $o(1)$. ◁

Finally, we state a variant of Corollary 25 that does not assume zero reconstruction error.

▶ **Corollary 31.** *Let* Π *be a computational RR-SSS for* Γ *with* m*-bit secrets. Then*

$$\frac{m}{2} + o(1) \geq H(0|\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_{n/2}),$$

where $\boldsymbol{p}_i$ *is a uniform RV over* $\{a_i, b_i\}$ *and* $\boldsymbol{p}_i$*'s are independent.*

Proof. The proof is the same as the proof of Corollary 25 with the following exceptions: Usages of Claim 24 and Claim 19 are replaced with those of Claim 30 and Claim 28, respectively. Indeed, these replacements substitute each claim with a corresponding variant that is adapted to the case of the non-zero reconstruction error. Also, the variant of strong submodularity that is stated in Lemma 29 should be used. ◀

4.3 Main result

▶ **Theorem 32.** *Let* Γ *be the union of* $n/2$ *disjoint* $(2,2)$*-thresholds (see Figure 3). If* Γ *has a computational RR-SSS, then there exists an OWF.*

Proof. As in the previous subsections, assume that $\{a_i, b_i\}$, $1 \leq i \leq n/2$, are the minimal qualified sets. Let $\Pi = (\mu, \boldsymbol{R})$ be a computational RR-SSS for Γ with m-bit secrets and $n = \mathsf{poly}(\lambda)$. For $0 \leq i \leq n/2$, take $\boldsymbol{p}_i$ to be a uniform RV over $\{a_i, b_i\}$ and set $\boldsymbol{B} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_{n/2}\}$.
According to Corollary 31 we have,

$$\frac{m}{2} + o(1) \geq H(0|\boldsymbol{p}_1\boldsymbol{p}_2\cdots\boldsymbol{p}_{n/2}).$$

So if we let $\boldsymbol{S}_0$ be a uniform RV over the secret space, then

$$\frac{m}{2} + o(1) + H(\mu_{\boldsymbol{B}}(\boldsymbol{S}_0, \boldsymbol{R})) \geq H(\boldsymbol{S}_0||\mu_{\boldsymbol{B}}(\boldsymbol{S}_0, \boldsymbol{R})).$$

Let $\boldsymbol{S}'_0$ be a uniform secret independent of $\boldsymbol{S}_0$ and $\boldsymbol{R}$. Then

$$H(\boldsymbol{S}'_0||\mu_{\boldsymbol{B}}(\boldsymbol{S}_0, \boldsymbol{R})) = m + H(\mu_{\boldsymbol{B}}(\boldsymbol{S}_0, \boldsymbol{R})).$$

These together imply that

$$H(\boldsymbol{S}'_0||\mu_{\boldsymbol{B}}(\boldsymbol{S}_0, \boldsymbol{R})) + o(1) \geq H(\boldsymbol{S}_0||\mu_{\boldsymbol{B}}(\boldsymbol{S}_0, \boldsymbol{R})) + \frac{m}{2}. \tag{12}$$

On the other hand,

$$\boldsymbol{S}'_0||\mu_{\boldsymbol{B}}(\boldsymbol{S}_0, \boldsymbol{R}) \stackrel{c}{\equiv} \boldsymbol{S}_0||\mu_{\boldsymbol{B}}(\boldsymbol{S}_0, \boldsymbol{R}). \tag{13}$$

Applying Lemma 8 to (12) and (13) (with $\boldsymbol{D}_\lambda = \boldsymbol{S}'_0||\mu_{\boldsymbol{B}}(\boldsymbol{S}_0, \boldsymbol{R})$ and $f(\boldsymbol{S}_0||\boldsymbol{R}||\boldsymbol{B}) = \boldsymbol{S}_0||\mu_{\boldsymbol{B}}(\boldsymbol{S}_0, \boldsymbol{R})$), we get the desired result. ◀

5 Construction of computational RR-SSS

In this section, we observe that computational RR-SSS for NC^1 can be based on simple minicrypt primitives that have some kind of one-time KDM-like security. In particular, we first observe that an efficient linear SSS (and generally, an efficient SSS with a property that we call *randomness simulatability*) can be converted into a computational RR-SSS assuming the existence of one-time KDM-secure RR-SKE. Next we introduce the notion of *linear-resistant* PRG. Then, we see how an efficient perfect linear SSS can be converted into an efficient computational RR-SSS, using a linear-resistant PRG.

5.1 RR-SKE and KDM security

First, we recall the definition of (RR-)SKE and (one-time) KDM-security.

▶ **Definition 33** (SKE/RR-SKE). *Let $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of message spaces and $\Sigma = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a tuple of probabilistic polynomial-time algorithms where*

- Gen*, called key-generation algorithm, on input 1^λ returns a key k,*
- Enc*, called encryption algorithm, gets a message m and a key k as input and returns a ciphertext ct,*
- Dec*, called decryption algorithm, gets a ciphertext ct and a key k as input and returns a message m or $\perp$.*

Σ is called a symmetric-key encryption (SKE) for $\mathcal{M}$ if for every $m \in \mathcal{M}_\lambda$:

$$\Pr[k \leftarrow \mathsf{Gen}(1^\lambda); ct \leftarrow \mathsf{Enc}_k(m) : \mathsf{Dec}_k(ct) = m] = 1 \ .$$

We call Σ randomness recoverable SKE (RR-SKE) if additionally there exists a polynomial-time algorithm $\mathsf{Recover}$ *such that:*

$$\Pr[k \leftarrow \mathsf{Gen}(1^\lambda); ct \leftarrow \mathsf{Enc}_k(m; \boldsymbol{R}) : \mathsf{Recover}_k(ct) = \boldsymbol{R}] = 1 \ ,$$

where $\boldsymbol{R}$ is the randomness used in the encryption algorithm.

▶ **Definition 34.** *Let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an SKE with key-space $\mathcal{K}$ and message-space $\mathcal{M}$. We say that Π is one-time KDM-secure, if for each efficiently computable function $f : \mathcal{K} \to \mathcal{M}$,*

$$\{k \leftarrow \mathsf{Gen}(1^\lambda) : \mathsf{Enc}_k(f(k)))\} \stackrel{c}{\equiv} \{k \leftarrow \mathsf{Gen}(1^\lambda) : \mathsf{Enc}_k(0^{|f(k)|})\}.$$

▶ **Lemma 35.** *Assume that $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a one-time KDM-secure SKE and $g : \{0,1\}^{l_1+l_2+l_3} \to \{0,1\}^l$ is an effieciently computable function. Then one has*

$$(\mathbf{x}, \mathsf{Enc}_{\boldsymbol{k}}(g(\boldsymbol{k}, \mathbf{x}, \mathbf{y}))) \stackrel{c}{\equiv} (\mathbf{x}, \mathsf{Enc}_{\boldsymbol{k}}(0^l))$$

where $\boldsymbol{k}$ is Π's key and has length l_1 and $(\mathbf{x}, \mathbf{y})$ are jointy distributed RVs over $\{0,1\}^{l_2} \times \{0,1\}^{l_3}$ and independent of $\boldsymbol{k}$.

We refer to Appendix B for the proof.

5.2 RR-SSS from randomness simulatable SSS and one-time KDM-secure RR-SKE

Consider this simple construction for a computational RR-SSS using a general (i.e., not necessarily perfect or linear) efficient standard SSS (which is known to exist for access structures in mP, assuming OWF) and an RR-SKE with one-time KDM-security. The construction is as follows. First, use the SSS to share $s||k$ with randomness r to compute the shares for the secret s, where k is the key of the SKE. Then, encrypt r under the secret key k using the SKE and append the ciphertext to the shares. The correctness and randomness recoverability requirements are trivial. Privacy follows from the KDM-security of the SKE. However, in order for the proof to go through, we require a property of the original SSS that we refer to as the *randomness simulatability*. Every linear SSS has this property but it remains open whether every access structure in mP admits a randomness simulatable SSS.

In the following, we first define the notion of *randomness simulatable* SSS. Then, we present a theorem that formalizes the above construction.

▶ **Definition 36** (Randomness simulatable SSS). *Let $\Pi = (\boldsymbol{R}, \mu)$ be a perfect or computational SSS for an access structure. We say that SSS Π is* randomness simulatable*, if for each RV $\boldsymbol{S}$ over the secret space and each unqualified set B there exists an efficiently computable function g and an efficiently sampleable RV $\widehat{\boldsymbol{R}}$ independent of $(\boldsymbol{S}, \boldsymbol{R})$ such that*

$$(\boldsymbol{S}, \boldsymbol{\mu}_B, \boldsymbol{R}) \stackrel{c}{\equiv} (\boldsymbol{S}, \boldsymbol{\mu}_B, g(\boldsymbol{S}, \boldsymbol{\mu}_B, \widehat{\boldsymbol{R}})) \ ,$$

where $\boldsymbol{\mu}_B = \mu_B(\boldsymbol{S}, \boldsymbol{R})$ denotes the share of the unqualified set B.

Notice that, ignoring the efficient computability of g and efficient sampleability of $\widehat{\boldsymbol{R}}$, the existence of g and $\widehat{\boldsymbol{R}}$ is always guaranteed by the functional representation lemma (Lemma 4). Also, in particular, linear SSSs are randomness simulatable. It is unclear to us whether every access structure in mP – which is known to admit an efficient computational SSS [37] – admits a randomness simulatable scheme.

▶ **Theorem 37.** *Let Π be a one-time KDM-secure RR-SKE with ℓ-bit keys. Let μ_i be the sharing map of the i'th participant in a perfect/computational randomness simulatable SSS for an access structure with t-bit secret, $t > l$, and ρ-bit randomness (i.e., the share of participant i is $\mu_i(s, r)$, where s is the secret and r is the randomness). Then, the SSS defined below is a computational RR-SSS for the same access structure.*

> *Given a secret $s \in \{0,1\}^{t-\ell}$ and a randomness $r \in \{0,1\}^{\rho}$:*
> - *generate a key $k \leftarrow \mathsf{Gen}(1^{\lambda})$,*
> - *let $ct \leftarrow \mathsf{Enc}_k(r)$,*
> - *let $\mu_i(s||k, r)||ct$ be the share of i'th participant.*

Proof. Correctness and randomness recoverability trivially hold. We prove privacy. Let $s \in \{0,1\}^{t-\ell}$ be an arbitrary secret and let B be an unqualified set in the access structure. Let $\boldsymbol{R}$ be SSS's randomness, $\boldsymbol{k}$ denote $\mathsf{Gen}(1^{\lambda})$ and $\boldsymbol{\mu}_B$ denote $\mu_B(s||\boldsymbol{k}, \boldsymbol{R})$. For ease of notation, we simply denote the share of B for the secret s by $\boldsymbol{\mu}_B||\mathsf{Enc}_{\boldsymbol{k}}(\boldsymbol{R})$ (i.e., we ignore the repetitions of $\mathsf{Enc}_{\boldsymbol{k}}(\boldsymbol{R})$). Based on the randomness simulatability of the SSS, there exists an efficiently computable function g and an efficiently sampleable RV $\widehat{\boldsymbol{R}}$ independent of $(\boldsymbol{k}, \boldsymbol{R})$ such that

$$(s||\boldsymbol{k}, \boldsymbol{\mu}_B, \boldsymbol{R}) \stackrel{c}{\equiv} (s||\boldsymbol{k}, \boldsymbol{\mu}_B, g(s||\boldsymbol{k}, \boldsymbol{\mu}_B, \widehat{\boldsymbol{R}}))$$

Therefore, one has the following indistinguishability:

$$\boldsymbol{\mu}_B||\mathsf{Enc}_{\boldsymbol{k}}(\boldsymbol{R}) \stackrel{c}{\equiv} \boldsymbol{\mu}_B||\mathsf{Enc}_{\boldsymbol{k}}(g(s||\boldsymbol{k}, \boldsymbol{\mu}_B, \widehat{\boldsymbol{R}})) \tag{14}$$

According to Lemma 16, one has

$$(\boldsymbol{k}, \mu_B(s||\boldsymbol{k}, \boldsymbol{R})) \stackrel{c}{\equiv} (\boldsymbol{k}, \mu_B(0^t, \boldsymbol{R})) \ .$$

In other words, $(\boldsymbol{k}, \boldsymbol{\mu}_B) \stackrel{c}{\equiv} (\boldsymbol{k}, \boldsymbol{\mu}'_B)$, where $\boldsymbol{\mu}'_B = \mu_B(0^t, \boldsymbol{R})$. Because g is efficiently computable and $\widehat{\boldsymbol{R}}$ is efficiently sampleable and independent of $(\boldsymbol{k}, \boldsymbol{R})$, we have

$$\boldsymbol{\mu}_B||\mathsf{Enc}_{\boldsymbol{k}}(g(s||\boldsymbol{k}, \boldsymbol{\mu}_B, \widehat{\boldsymbol{R}})) \stackrel{c}{\equiv} \boldsymbol{\mu}'_B||\mathsf{Enc}_{\boldsymbol{k}}(g(s||\boldsymbol{k}, \boldsymbol{\mu}'_B, \widehat{\boldsymbol{R}})). \tag{15}$$

On the other hand, because $(\boldsymbol{\mu}'_B, \widehat{\boldsymbol{R}})$ is independent of $\boldsymbol{k}$, by Lemma 35, we have:

$$\boldsymbol{\mu}'_B||\mathsf{Enc}_{\boldsymbol{k}}(g(s||\boldsymbol{k}, \boldsymbol{\mu}'_B, \widehat{\boldsymbol{R}})) \stackrel{c}{\equiv} \boldsymbol{\mu}'_B||\mathsf{Enc}_{\boldsymbol{k}}(0^{\rho}). \tag{16}$$

Equations (14), (15) and (16) then imply that

$$\boldsymbol{\mu}_B||\mathsf{Enc}_{\boldsymbol{k}}(\boldsymbol{R}) \stackrel{c}{\equiv} \boldsymbol{\mu}'_B||\mathsf{Enc}_{\boldsymbol{k}}(0^{\rho}) \ .$$

Because $\boldsymbol{\mu}'_B||\mathsf{Enc}_{\boldsymbol{k}}(0^{\rho})$ hides the secret s, privacy follows. ◀

5.3 Linear-resistant PRG

In this section, we present a variant of pseudo-random generators (PRG), with a KDM-like security for the class of linear functions.

Recall that a polynomial-time deterministic algorithm, $G : \{0,1\}^\star \to \{0,1\}^\star$ that maps λ-bit strings to $\ell(\lambda)$-bit strings is said to be PRG if $\ell(\lambda) > \lambda$ and $G(\boldsymbol{U}_\lambda) \stackrel{c}{\equiv} \boldsymbol{U}_{\ell(\lambda)}$.

In the following definition, $\{0,1\}$ is identified with $\mathbb{F}_2$, the finitie field with two elements, and $+$ stands for the addition in the field or bitwise-XOR; that is, for $x = x_1, \ldots, x_\ell$ and $y = y_1, \ldots, y_\ell$, $x + y = (x_1 \oplus y_1)|| \cdots ||(x_\ell \oplus y_\ell)$.

▶ **Definition 38.** *Let $G : \{0,1\}^\lambda \to \{0,1\}^\ell$ be a polynomial-time deterministic algorithm with $\ell := \ell(\lambda) > \lambda$. We call G a* linear-resistant *PRG if for every $\mathbb{F}_2$-linear function $L : \{0,1\}^\lambda \to \{0,1\}^\ell$, $G(\boldsymbol{U}_\lambda) + L(\boldsymbol{U}_\lambda) \stackrel{c}{\equiv} \boldsymbol{U}_\ell$.*

Clearly, every linear-resistant PRG is also a PRG. However, the converse is not necessarily correct. For example, if $G : \{0,1\}^{\lambda-1} \to \{0,1\}^{\ell-1}$ is a PRG, then so is $G' : \{0,1\}^\lambda \to \{0,1\}^\ell$ defined as $G'(s_1 \cdots s_\lambda) = s_1 || G(s_2 \cdots s_\lambda)$. It is clear that G' is not linear-resistant.

It is easy to see that linear-resistant PRG implies one-time KDM-secure SKE against the class of all affine functions: simply consider the standard one-time-pad encryption scheme $\mathsf{Enc}_k(m) = G(k) + m$. More precisely, if the input and output lengths of the linear-resistant PRG G are λ and ℓ, the key and message spaces of the constructed scheme are $\mathcal{K} = \mathbb{F}_2^\lambda$ and $\mathcal{M} = \mathbb{F}_2^\ell$, respectively, and it has KDM-security against all affine functions from $\mathbb{F}_2^\lambda$ to $\mathbb{F}_2^\ell$.

In particular, since this scheme is deterministic, the resulting SKE is RR. Another variant of PRG that has a KDM-like property is the hinting PRG which can be used to achieve one-time KDM-secure SKE against any class of functions that can be computed in fixed polynomial time [25, Appendix B]. Also, note that both of these primitives can be instantiated using a random oracle. Despite the similarity between linear-resistant PRG and hinting PRG, the relationship between these primitives remains open, as is the (im)possibility of constructing linear-resistant PRG from OWF. In contrast, black-box separation between hinting PRG and PKE is known [1].

5.4 RR-SSS from linear perfect SSS and linear-resistant PRG

Consider the following simple construction for a computational RR-SSS using an efficient (standard) linear perfect SSS and a linear-resistant PRG G. To share a secret s, use the linear SSS to share $s||r$ with randomness $G(r)$ to compute the shares, where r is the randomness. It is clear that every qualified set can recover not only s but also r. Privacy follows from the linear-resistance security of the PRG. Notice that the class of access structures that admit efficient linear SSS is equivalent to the class of monotone boolean functions that admit efficient MSP (monotone-span programs [22]) which includes NC^1 (e.g., using the Benaloh-Leichter [6] construction).

We state the above construction in a theorem:

▶ **Theorem 39.** *Let μ be the sharing map of a perfect linear SSS for an access structure with $k\lambda$-bit secrets, $k > 1$, and ℓ-bit randomness (i.e., the shares of participants are the outputs of $\mu(s, r)$, where s is the secret and r is the randomness). Let $G : \{0,1\}^\lambda \to \{0,1\}^\ell$ be a linear-resistant PRG. Then, the SSS defined by the sharing map $\mu'(s, r) = \mu(s||r, G(r))$ is a computational RR-SSS for the same access structure, where $s \in \{0,1\}^{(k-1)\lambda}$ is the secret and $r \in \{0,1\}^\lambda$ is the randomness with uniform distribution.*

Proof. Correctness and randomness recoverability trivially hold. We prove privacy. Let B be an unqualified set and let $\mu_B(s_1||s_2, r) = L_1(s_1) + L_2(s_2) + L_3(r)$ be the share of B for the secret $s_1||s_2$ and randomness $r \in \{0,1\}^\ell$ in the perfect linear scheme, where L_i's are linear functions, $s_1 \in \{0,1\}^{(k-1)\lambda}$ and $s_2 \in \{0,1\}^\lambda$.

By perfect privacy of the linear scheme, for any $s \in \{0,1\}^{(k-1)\lambda}$, the RVs $L_2(s)+L_3(\boldsymbol{r})$ and $L_3(\boldsymbol{r})$ have the same distributions, where $\boldsymbol{r}$ is a uniform RVs on ℓ-bit strings (they correspond to the shares of the secrets $s||0^\lambda$ and $0^{k\lambda}$, respectively). Therefore $\mathsf{supp}(L_2(s) + L_3(\boldsymbol{r})) = \mathsf{supp}(L_3(\boldsymbol{r}))$ which implies that $L_2(s) + \mathsf{range}(L_3) = \mathsf{range}(L_3)$. As a result $L_2(s) \in \mathsf{range}(L_3)$ and because s is arbitrary, we have $\mathsf{range}(L_2) \subseteq \mathsf{range}(L_3)$. If f and g are linear functions from V to W such that range of g is a subspace of the range of f, then for a suitable linear function h over V one has $g = f \circ h$. By this fact, there is a linear function L such that $L_2 = L_3 \circ L$.

Let $s, s' \in \{0,1\}^{(k-1)\lambda}$ be two arbitrary secrets and $\boldsymbol{r}$ be as before. Again, by perfect privacy of the linear scheme, $L_1(s) + L_3(\boldsymbol{r})$ and $L_1(s') + L_3(\boldsymbol{r})$ have the same distributions (they correspond to the shares of the secrets $s||0^\lambda$ and $s'||0^\lambda$, respectively). Since G is linear-resistant, by a standard reduction argument, $L_1(s) + L_3(G(\boldsymbol{r}) + L(\boldsymbol{r}))$ and $L_1(s') + L_3(G(\boldsymbol{r}) + L(\boldsymbol{r}))$ are computationally indistinguishable where $\boldsymbol{r}$ is a uniform RV on λ-bit strings. Therefore, $\mu'_B(s, \boldsymbol{r}) = L_1(s) + L_2(\boldsymbol{r}) + L_3(G(\boldsymbol{r}))$ and $\mu'_B(s', \boldsymbol{r}) = L_1(s') + L_2(\boldsymbol{r}) + L_3(G(\boldsymbol{r}))$ are computationally indistinguishable, which is the desired result. ◀

We conclude this section with the following remark that relates the observations of this section and the previous ones.

▶ Remark 40. Notice that in the proof of Theorem 37, we do not require that the SKE be KDM-secure against the whole class of efficiently computable functions. Indeed, security against all the functions g for all unqualified sets is sufficient. Since the class of linear SSSs is randomness simulatable with linear g's, and one-time secure RR-SKE against the class of linear functions is implied by linear-resistant PRG, Theorem 39 follows by Theorem 37, via a simpler construction though.

6 Conclusion

We initiated the study of SSS from the viewpoint of randomness recovery. By proving an exponential lower bound for the information ratio of an RR-SSS that realizes some very simple access structure in monotone AC^0, we showed that the situation is very different for RR-SSS, compared to the standard SSS, for which the best-known lower bound is sub-linear. We also managed to shed some light on the complexity of the computational RR-SSS, by proving that computational RR-SSS for certain access structures in monotone AC^0 implies OWF. This computational result is essentially a consequence of our information-theoretic lower bound; This can be justified by the very general idea that an algorithm that hides the secret from a bounded adversary but is unable to do so against an unbounded adversary implies OWF.

In the final section, we observed that an efficient perfect linear SSS can be converted into a computational RR-SSS for the same access structure using a type of PRG that we called linear-resistant PRG. We also noted that using a one-time KDM-secure RR-SKE, one can convert an efficient perfect/computational SSS into an RR-SSS, assuming that the SSS has the extra property of randomness simulatability.

References

1 Navid Alamati and Sikhar Patranabis. Cryptographic primitives with hinting property. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022 – 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 33–62. Springer, 2022. doi:10.1007/978-3-031-22963-3_2.

2 Benny Applebaum and Barak Arkis. On the power of amortization in secret sharing: d-uniform secret sharing and CDS with constant information rate. *ACM Trans. Comput. Theory*, 12(4):24:1–24:21, 2020. doi:10.1145/3417756.

3 Amos Beimel. Secret-sharing schemes: A survey. In *Coding and Cryptology – Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*, pages 11–46, 2011. doi:10.1007/978-3-642-20901-7_2.

4 Amos Beimel and Yuval Ishai. On the power of nonlinear secret-sharing. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 188–202, 2001. doi:10.1109/CCC.2001.933886.

5 Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994. doi:10.1007/BFb0053428.

6 Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *Advances in Cryptology – CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 27–35, 1988. doi:10.1007/0-387-34799-2_3.

7 George Robert Blakley. Safeguarding cryptographic keys. *Proceedings of the 1979 AFIPS National Computer Conference*, 48:313–317, 1979.

8 Dan Boneh. Simplified OAEP for the RSA and rabin functions. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 275–291. Springer, 2001. doi:10.1007/3-540-44647-8_17.

9 Renato M. Capocelli, Alfredo De Santis, Luisa Gargano, and Ugo Vaccaro. On the size of shares for secret sharing schemes. *J. Cryptology*, 6(3):157–167, 1993. doi:10.1007/BF00198463.

10 László Csirmaz. The size of a share must be large. *J. Cryptology*, 10(4):223–231, 1997. doi:10.1007/s001459900029.

11 László Csirmaz. Secret sharing and duality. *J. Math. Cryptol.*, 15(1):157–173, 2020. doi:10.1515/jmc-2019-0045.

12 Abbas El Gamal and Young-Han Kim. *Network Information Theory*. Cambridge University Press, 2011. doi:10.1017/CBO9781139030687.

13 Sanjam Garg, Mohammad Hajiabadi, Giulio Malavolta, and Rafail Ostrovsky. How to build a trapdoor function from an encryption scheme. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021 – 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part III*, volume 13092 of *Lecture Notes in Computer Science*, pages 220–249. Springer, 2021. doi:10.1007/978-3-030-92078-4_8.

14 Susan Hohenberger, Venkata Koppula, and Brent Waters. Chosen ciphertext security from injective trapdoor functions. 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 I*, volume 12170 of *Lecture Notes in Computer Science*, pages 836–866. Springer, 2020. doi:10.1007/978-3-030-56784-2_28.

15 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.

16 Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography. In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235, 1989. doi:10.1109/SFCS.1989.63483.

17 Mitsuru Ito, Akira Saio, and Takao Nishizeki. Multiple assignment scheme for sharing secret. *J. Cryptol.*, 6(1):15–20, 1993. doi:10.1007/BF02620229.

18 Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72(9):56–64, 1989.

19 Amir Jafari and Shahram Khazaei. Partial secret sharing schemes. *IACR Cryptol. ePrint Arch.*, 2020:448, 2020. URL: https://eprint.iacr.org/2020/448.

20 Tarik Kaced. Almost-perfect secret sharing. In *2011 IEEE International Symposium on Information Theory Proceedings, ISIT 2011, St. Petersburg, Russia, July 31 – August 5, 2011*, pages 1603–1607, 2011. doi:10.1109/ISIT.2011.6033816.

21 Tarik Kaced. Information inequalities are not closed under polymatroid duality. *IEEE Trans. Information Theory*, 64(6):4379–4381, 2018. doi:10.1109/TIT.2018.2823328.

22 Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 102–111, 1993. doi:10.1109/SCT.1993.336536.

23 Ehud D. Karnin, J. W. Greene, and Martin E. Hellman. On secret sharing systems. *IEEE Trans. Information Theory*, 29(1):35–41, 1983. doi:10.1109/TIT.1983.1056621.

24 Shahram Khazaei, Tal Moran, and Douglas Wikström. A mix-net from any CCA2 secure cryptosystem. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012 – 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 607–625. Springer, 2012. doi:10.1007/978-3-642-34961-4_37.

25 Fuyuki Kitagawa, Takahiro Matsuda, and Keisuke Tanaka. CCA security and trapdoor functions via key-dependent-message security. *J. Cryptol.*, 35(2):9, 2022. doi:10.1007/s00145-022-09420-8.

26 Ilan Komargodski, Moni Naor, and Eylon Yogev. Secret-sharing for NP. *J. Cryptol.*, 30(2):444–469, 2017. doi:10.1007/s00145-015-9226-0.

27 Venkata Koppula and Brent Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. 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 671–700. Springer, 2019. doi:10.1007/978-3-030-26951-7_23.

28 Chung Ki Li and Duncan S. Wong. Signcryption from randomness recoverable public key encryption. *Inf. Sci.*, 180(4):549–559, 2010. doi:10.1016/j.ins.2009.10.015.

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 Silvio Micali. Simple and fast optimistic protocols for fair electronic exchange. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003, Boston, Massachusetts, USA, July 13-16, 2003*, pages 12–19. ACM, 2003. doi:10.1145/872035.872038.

31 John W Moon and Leo Moser. On cliques in graphs. *Israel journal of Mathematics*, 3(1):23–28, 1965.

32 Duong Hieu Phan and David Pointcheval. Chosen-ciphertext security without redundancy. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 – December 4, 2003, Proceedings*, volume 2894 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003. doi:10.1007/978-3-540-40061-5_1.

33 Phillip Rogaway and Mihir Bellare. Robust computational secret sharing and a unified account of classical secret-sharing goals. In *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 172–184, 2007. doi:10.1145/1315245.1315268.

34 Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979. doi:10.1145/359168.359176.

35 Victor Shoup. OAEP reconsidered. *J. Cryptol.*, 15(4):223–249, 2002. doi:10.1007/s00145-002-0133-9.

36 Vinod Vaikuntanathan, Arvind Narayanan, K. Srinathan, C. Pandu Rangan, and Kwangjo Kim. On the power of computational secret sharing. In Thomas Johansson and Subhamoy Maitra, editors, *Progress in Cryptology – INDOCRYPT 2003, 4th International Conference on Cryptology in India, New Delhi, India, December 8-10, 2003, Proceedings*, volume 2904 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 2003. doi:10.1007/978-3-540-24582-7_12.

37 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.

A Proof of Lemma 16

Assume that Π is a computational SSS. Let $\Gamma = \{\Gamma_n\}_n$, $\boldsymbol{R} = \{\boldsymbol{R}_{\lambda,n}\}_{\lambda,n}$, $\boldsymbol{S} = \{\boldsymbol{S}_{\lambda,n}\}_{\lambda,n}$, $t = t(\lambda, n)$, $B = \{B_n\}_n$ and $n = n(\lambda)$ be a polynomial. We should prove that

$$(\boldsymbol{S}_{\lambda,n(\lambda)}, \mu_{B_{n(\lambda)}}(\boldsymbol{S}_{\lambda,n(\lambda)}, \boldsymbol{R}_{\lambda,n(\lambda)})) \stackrel{c}{\equiv} (\boldsymbol{S}_{\lambda,n(\lambda)}, \mu_{B_{n(\lambda)}}(0^{t(\lambda,n(\lambda))}, \boldsymbol{R}_{\lambda,n(\lambda)})).$$

For contradiction, let poly be a polynomial and $\mathcal{D} = \{\mathcal{D}_\lambda\}_\lambda$ be a family of polynomial-size distinguishers such that for infinitely many λ,

$$\begin{aligned}|\Pr[\mathcal{D}_\lambda(\boldsymbol{S}_{\lambda,n(\lambda)}, \mu_{B_{n(\lambda)}}(\boldsymbol{S}_{\lambda,n(\lambda)}, \boldsymbol{R}_{\lambda,n(\lambda)})) = 1] \\ - \Pr[\mathcal{D}_\lambda(\boldsymbol{S}_{\lambda,n(\lambda)}, \mu_{B_{n(\lambda)}}(0^{t(\lambda,n(\lambda))}, \boldsymbol{R}_{\lambda,n(\lambda)})) = 1]| \geq \frac{1}{\mathsf{poly}(\lambda)}.\end{aligned}$$

Therefore, according to the independence of $\boldsymbol{S}$ and $\boldsymbol{R}$, for each such λ, there is $s_\lambda \in \mathsf{supp}(\boldsymbol{S}_{\lambda,n(\lambda)})$ such that

$$|\Pr[\mathcal{D}_\lambda(s_\lambda, \mu_{B_{n(\lambda)}}(s_\lambda, \boldsymbol{R}_{\lambda,n(\lambda)})) = 1] - \Pr[\mathcal{D}_\lambda(s_\lambda, \mu_{B_{n(\lambda)}}(0^{t(\lambda,n(\lambda))}, \boldsymbol{R}_{\lambda,n(\lambda)})) = 1]| \geq \frac{1}{\mathsf{poly}(\lambda)}.$$

Therefore, for $\mathcal{C}_\lambda(\cdot) = \mathcal{D}_\lambda(s_\lambda, \cdot)$ one has

$$|\Pr[\mathcal{C}_\lambda(\mu_{B_{n(\lambda)}}(s_\lambda, \boldsymbol{R}_{\lambda,n(\lambda)})) = 1] - \Pr[\mathcal{C}_\lambda(\mu_{B_{n(\lambda)}}(0^{t(\lambda,n(\lambda))}, \boldsymbol{R}_{\lambda,n(\lambda)})) = 1]| \geq \frac{1}{\mathsf{poly}(\lambda)},$$

which contradicts the computational privacy of the SSS.

B Proof of Lemma 35

Let $\boldsymbol{k} = \{\boldsymbol{k}_\lambda\}_\lambda$ where $\boldsymbol{k}_\lambda = \mathsf{Gen}(1^\lambda)$, $\mathbf{x} = \{\mathbf{x}_\lambda\}_\lambda$, $\mathbf{y} = \{\mathbf{y}_\lambda\}_\lambda$ and $g = \{g_\lambda\}_\lambda$. Assume that the assertion is false and there is a polynomial poly and a polynomial-size distinguisher $D = \{\mathcal{D}_\lambda\}_\lambda$ and infinitely many λ for which:

$$|\Pr[\mathcal{D}_\lambda(\mathbf{x}_\lambda, \mathsf{Enc}_{\boldsymbol{k}_\lambda}(g_\lambda(\boldsymbol{k}_\lambda, \mathbf{x}_\lambda, \mathbf{y}_\lambda))) = 1] - \Pr[\mathcal{D}_\lambda(\mathbf{x}_\lambda, \mathsf{Enc}_{\boldsymbol{k}_\lambda}(0^{l(\lambda)})) = 1]| \geq \frac{1}{\mathsf{poly}(\lambda)}.$$

Because $\boldsymbol{k}$ is independent of $\mathbf{x}$ and $\mathbf{y}$, for each such λ, there is $(x_\lambda, y_\lambda) \in \mathsf{supp}(\mathbf{x}_\lambda) \times \mathsf{supp}(\mathbf{y}_\lambda)$ such that:

$$|\Pr[\mathcal{D}_\lambda(x_\lambda, \mathsf{Enc}_{\boldsymbol{k}_\lambda}(g_\lambda(\boldsymbol{k}_\lambda, x_\lambda, y_\lambda))) = 1] - \Pr[\mathcal{D}_\lambda(x_\lambda, \mathsf{Enc}_{\boldsymbol{k}_\lambda}(0^{l(\lambda)})) = 1]| \geq \frac{1}{\mathsf{poly}(\lambda)}.$$

Letting $\mathcal{C}_\lambda(\cdot) = \mathcal{D}_\lambda(x_\lambda, \cdot)$ and $f_\lambda(\cdot) = g_\lambda(\cdot, x_\lambda, y_\lambda)$, we have

$$|\Pr[\mathcal{C}_\lambda(\mathsf{Enc}_{\boldsymbol{k}_\lambda}(f_\lambda(\boldsymbol{k}_\lambda))) = 1] - \Pr[\mathcal{C}_\lambda(\mathsf{Enc}_{\boldsymbol{k}_\lambda}(0^{l(\lambda)})) = 1]| \geq \frac{1}{\mathsf{poly}(\lambda)},$$

which contradicts the KDM-security of Π.

Secure Communication in Dynamic Incomplete Networks

Ivan Damgård
Aarhus University, Denmark

Divya Ravi
Aarhus University, Denmark

Daniel Tschudi
Concordium, Zürich, Switzerland

Sophia Yakoubov
Aarhus University, Denmark

Abstract

In this paper, we explore the feasibility of reliable and private communication in *dynamic* networks, where in each round the adversary can choose which direct peer-to-peer links are available in the network graph, under the sole condition that the graph is k-connected at each round (for some k).

We show that reliable communication is possible in such a dynamic network if and only if $k > 2t$. We also show that if $k = cn > 2t$ for a constant c, we can achieve reliable communication with polynomial round and communication complexity.

For unconditionally private communication, we show that for a passive adversary, $k > t$ is sufficient (and clearly necessary). For an active adversary, we show that $k > 2t$ is sufficient for statistical security (and clearly necessary), while $k > 3t$ is sufficient for perfect security. We conjecture that, in contrast to the static case, $k > 2t$ is not enough for perfect security, and we give evidence that the conjecture is true.

Once we have reliable and private communication between each pair of parties, we can emulate a complete network with secure channels, and we can use known protocols to do secure computation.

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

Keywords and phrases Secure Communication, Dynamic Incomplete Network, Information-theoretic

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.13

Related Version *Full Version*: `https://eprint.iacr.org/2023/529`

Funding *Ivan Damgård*: The SecureDNA project, The Villum Foundation.
Divya Ravi: The European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC).
Sophia Yakoubov: The Danish Independent Research Council under Grant-ID DFF-2064-00016B (YOSO).

1 Introduction

In this paper, we study the feasibility of unconditionally secure communication (and hence multiparty computation) when parties communicate over an incomplete and dynamic network. More precisely, we assume a synchronous network with secure point-to-point channels where, in each round, only some of the point-to-point connections work, so the network is incomplete. Furthermore, the adversary can decide to change the set of active connections from one round to the next. We call this a *dynamic* incomplete network, in contrast to a *static* incomplete network, where the graph describing the active connections stays the same throughout the protocol.

© Ivan Damgård, Divya Ravi, Daniel Tschudi, and Sophia Yakoubov;
licensed under Creative Commons License CC-BY 4.0

The study of reliable communication on an incomplete network starts with the work of Dolev [2], who showed that, for a static incomplete network where t of the n parties are malicious, one can do secure broadcast if and only if the network is at least $2t+1$-connected, and $3t < n$. (A network is k-connected if it remains connected when one removes any set of less than k vertices. By Menger's theorem, this is equivalent to requiring that any pair of distinct nodes are connected by at least k disjoint paths.)

Later, Dolev et al. [3] showed that any two parties in a static incomplete network can communicate with perfect security (privacy and reliability) if and only if the network is $2t+1$-connected. Using the protocols from that work, one can emulate a complete network with secure point-to-point channels, so combining this with well-known feasibility results for MPC, one can conclude that, in a static network, $2t+1$-connectivity is necessary and sufficient for unconditionally secure MPC to be possible.

However, as mentioned, we are interested in what happens if the network is dynamic. To the best of our knowledge, very little work has been done in this direction. The problem was first considered in Mauer *et al.* [5], who define a notion called *Dynamic Min-Cut* for two nodes a, b, denoted by $DynMinCut(a, b)$. They show that reliable (non-private) communication in a dynamic network from a to b is possible if and only if $DynMinCut(a, b) > 2t$. Intuitively, this condition makes a statement on how the network evolves over time, and says that it does so in a way that is "kind enough" to allow a message to travel from p to q on at least $2t+1$ disjoint paths. It is not surprising that this is the condition for reliable communication, given what was known about static networks. However, dynamic min-cut makes a rather complicated statement about the entire sequence of network graphs. It is natural to ask if there is a simple property we can require for each individual network graph that would then imply the $DynMinCut$-condition. This is a natural question from a theory point of view, but also a question one would ask in order to decide if secure communication is feasible in a given application scenario. Finally, the work of Mauer *et al.* did not consider unconditionally private communication.

1.1 Our Contribution

In this paper, we aim to fill the gaps pointed out above. We introduce a model where an adversary chooses, in each round, a network graph to be used, under the sole condition that it is k-connected, for some k. This is a natural generalization of the static case and also a model that might be relevant in practice – consider, for instance, a mobile network where connections come and go, but where each possible connection is active with some probability p. Known results on random graphs indicate that if p is large enough, then the resulting graph is highly connected.

For a static network, it seems reasonable to assume that the parties know the topology of the network, while this is quite unrealistic for a dynamic network, where the absence on connections may be caused by movement of mobile devices, or equipment crashing. These are events that a party cannot predict locally. We therefore assume that honest parties do not know the network topology. In one version of the model, parties know their immediate neighbourhood, but our protocols work in the worst case where parties do not know which connections they have.

We show that reliable communication is possible in a dynamic network if and only if $k > 2t$ (note that we inherit impossibility results from the static case since the adversary could choose to keep the network graph constant). For unconditionally private communication, we show that for a passive adversary, $k > t$ is sufficient (and clearly necessary). For an active adversary, we show that $k > 2t$ is sufficient for statistical security (and clearly necessary),

while $k > 3t$ is sufficient for perfect security. We conjecture that, in contrast to the static case, $k > 2t$ is not enough for perfect security, and we give evidence that the conjecture is true. As mentioned above, once we can emulate a complete network with secure channels, we can use known protocols to do secure computation.

Even though we can provide secure communication on dynamic networks, it is natural to expect that there is a performance penalty in going from static to dynamic networks: in the static case we can use a fixed set of paths for communication, while this clearly will not work in the dynamic case, as the adversary could block these paths. Intuition clearly suggests that one needs to try a large number of paths from sender to receiver to make sure something gets through.

We study a class of protocols where the main step is that the sender S tries to send data along a set of paths Paths, and where the protocol will be successful if the receiver R receives something on at least k disjoint paths. All the protocols we construct are in this class. We will say that Paths *is k-connected with respect to a family of graphs $\mathcal{G}$* if, for each graph $\mathsf{G} \in \mathcal{G}$, for any pair (S, R), there exist k disjoint paths between S and R, such that these paths are contained in both G and in Paths. Here, one should think of $\mathcal{G}$ as a sequence of network graphs chosen by the adversary under the constraints in our model, so we assume throughout that all graphs in $\mathcal{G}$ are k-connected. As short hand, we say that Paths *is k-connected* if it is k-connected with respect to *all* graph families (within our model). Paths being k-connected can be thought of as a condition saying that Paths is large enough, in comparison to the set of connections that adversary allows at any one time.

We show that if Paths is k-connected (with respect to the family of graphs chosen by the adversary), then our protocols terminate correctly in at most $L|\mathsf{Paths}|$ rounds, where L is the maximal length of paths in Paths and $|\mathsf{Paths}|$ is the cardinality of Paths. We can make sure that Paths is always k-connected by simply choosing Paths to be the set of all possible paths, which unfortunately is exponentially large in n.

However, if k is $\Omega(n)$, we can do much better: we show that in this case, among the k disjoint paths that exist in every round, there must be a large number of short (constant length) paths. More precisely, if $k = dn$ for a constant d, and we let Paths_c the set of paths of length c for some constant c, then Paths_c is $k' = d'n$-connected for $d' < d$, and by choosing c large enough, we can make d' be arbitrarily close to d. Moreover, $|\mathsf{Paths}_c|$ is polynomial in n for any constant c, so if we run our protocols with Paths_c as the target set of paths, their complexity will be polynomial in n.

This efficient version is almost as robust against corruptions as is possible: from the discussion above, it follows that the maximum number of corruptions that can be tolerated with connectivity k is roughly $(k-1)/2$. So if k is a constant times n, it is also the case that $t = \alpha n$ for a constant α. Since we can get connectivity almost k with constant length paths, our efficient version can be designed to tolerate βn corruptions, for any $\beta < \alpha$.

Note that if we want to do MPC, one usually wants to tolerate $\Omega(n)$ corruptions, which implies that k must also be $\Omega(n)$, even for a static network.

1.2 Technical overview

The case of sending a public message reliably from S to R is relatively straightforward: S can send the same message on all paths in the pathset used. Here, as in all of our protocols, each copy of the message will be accompanied by metadata that specifies on which path the message travelled. Once R has received something on sufficiently many disjoint paths, she can determine what the correct message is. This is basically the same protocol that was considered in [7] for a static and asynchronous network. However, the proof that it terminates correctly is completely different in our case.

Table 1 Reliable Communication Protocols for a message m, over k-connected path set Paths of cardinality M and with maximal path length L. The communication complexity is given per party per round. Paths' denotes a k-connected path set having paths of length at most c, where c is a constant.

	Corruption		Graph	Complexity	
Scheme	Type	Threshold	Connectivity	Rounds	Communication
Protocol 1	passive	$t < n$	1	n	$\mathcal{O}(n\|m\|)$
Protocol 2	active	$t < \frac{n}{2}$	$k > 2t$	LM	$\mathcal{O}(n^2 2^n \|m\|)$
Protocol 2 with constant-length paths	active	$t < \frac{n}{2}$	$k = cn > 2t$	$c\|\mathsf{Paths}'\| = \mathsf{poly}(n)$	$\mathsf{poly}(n)\|m\|$

Table 2 Private Communication Protocols: The communication complexity is given per party per round.

		Corruption		Graph	Complexity	
Scheme	Security	Type	Threshold	Connectivity	Rounds	Communication
Protocol 3	perfect	passive	$t < n$	$k > t$	$LM + 2n$	$\mathcal{O}(Mn^2(\log M + \|m\|))$
Protocol 5	perfect	active	$t < \frac{n}{3}$	$k > 3t$	$3LM$	$\mathcal{O}(M^n n(\log M + \|m\|))$
Protocol 5 communication efficient variant	perfect	active	$t < \frac{n}{3}$	$k > 3t$	$\mathsf{poly}(LM)$	Per attempt same as Protocol 4. In the worst case overall same as Protocol 5.
Protocol 6	perfect	active	$t < \frac{n}{4}$	$k > 4t$	$3LM$	$\mathcal{O}(Mn^3 2^n(\log M + \|m\|))$
Protocol 4	statistical	active	$t < \frac{n}{2}$	$k > 2t$	$3LM$	$\mathcal{O}(n^3 2^n(\|m\| + \log M) + M(\|m\| + \lambda \cdot M))$

Sending unconditionally private messages comes with new and bigger challenges. For the static case, Dolev *et al.* [3] designated a fixed set of k disjoint paths to be used for communicating between S and R. After this, they could abstract away the network and simply assume that S and R are connected by k channels where t of these are controlled by the adversary. This is the problem of *secure message transmission* that has been studied in many subsequent works. However, this abstraction cannot be used for a dynamic network: we cannot predict on which paths R will receive something. Moreover, corrupt parties may claim that they heard something on a path, even if the adversary's choice of network graphs actually did not allow transmission via that path. R has no obvious way to tell that such a claim is false – after all, with a different scheduling of network graphs, the claim might have been true.

We therefore need a new approach to sending private messages. For simplicity, we first sketch the idea for a passive adversary: we let S send independent randomness on sufficiently many different paths, and once R has received something on k disjoint paths, she can report to S the identity of these paths (but not the randomness she received). The reporting can be done using the protocol we already have for reliable public communication. If $t < k$, at least one random value that made it to R is unknown to the adversary, so S can derive a key (by XORing together all of the random values that made it), use that key to one-time pad encrypt the message, and send the resulting ciphertext in public.

This will not work for an active adversary, as we need to make sure the correct message is received. Our high-level strategy to solve this is as follows: if S sends data on a k-connected pathset, R will eventually receive data on $k-t$ disjoint paths (but not necessarily more, as t paths can be blocked by corrupt players). However, due to misreporting by corrupt players, R may have data delivered that claim to come from a large set of paths, containing many sets of $k-t$ disjoints paths. The idea is now to use reliable public communication to identify a sufficiently large set C of disjoint paths where R received *correct* data on *all* paths in C. We call such a set a *good* set. It turns out that if $k \geq 3t+1$ we can identify a good path set of size $2t+1$ with zero error probability by exploiting the fact that any set of at least $2t+1$ disjoint paths must contain a majority of paths with only honest players. This requires a lot of work, but if $k \geq 4t+1$, it can be done much more efficiently using error correction. If $k \geq 2t+1$ we can identify a good set of size $t+1$, except with negligible error probability, by using unconditionally secure authentication[1]. Now, from the data received via paths in C we can extract a value that can be used to one-time pad encrypt the secret message, as in the passively secure solution. We therefore get statistical security for $k \geq 2t+1$ and perfect security for $k \geq 3t+1$.

However, if the goal is perfect security and we only assume $k \geq 2t+1$, there are serious problems, and we conjecture that in fact perfect security cannot be achieved in this case. We give evidence for the conjecture: For connectivity $2t+1$, we construct an example scenario where the set of paths delivered to R contains several maximal sets of disjoint paths of size $k-t = t+1$. One of these sets contains only honest paths, but the others have the same number of honest and corrupt paths and may contain incorrect data. We show that R cannot perfectly decide which one is the all-honest set: for each choice there exists an adversarial strategy that would be consistent with that choice[2]. Now, consider any protocol which (as we do) would select one of these path sets and try to use the data received there and public communication to get a message across with perfect security. Any such protocol will fail with non-zero probability: if the wrong set is selected, we may have a situation where half the received values are known and/or manipulated by the adversary. Intuitively, if the receiver's output depends on only half the values (and the public communication) this may be the half the adversary knows and the output is not secret. If it depends on more than half the values, it must depend on values the adversary may have changed, and the output is not correct. We therefore conjecture that, in contrast to the static network case, $k > 3t$ is needed for perfect security.

1.3 Related Work

Maurer *et al.* [5] consider the setting closest to our own. They identify the minimal and necessary condition for authenticated peer-to-peer communication; that is, there should exist $2t+1$ (or $t+1$, assuming digital signatures) disjoint paths from the sender to the receiver *over time*. We would like to point that if the conditions required in our model are satisfied, this would imply the conditions are also satisfied over time. Conversely, there could be settings where conditions from Maurer *et al.* [5] are satisfied, while ours are not. For instance, the condition of Maurer *et al.* [5] could be satisfied even if the graphs at every point in time

[1] It may seem that authentication cannot be used here, as S needs to send a key to R, and this key must be sent privately and correctly, just like the message. We solve this apparent circularity by observing that S could send many keys on different paths and it is sufficient that one key makes it to R unseen by the adversary. See more details within.

[2] note that unconditionally secure authentication does not give perfect security and so does not help.

Table 3 Related Work.

Construction	Assumptions	Corruption	Guarantees on Graph
auth. P2P [5]	-	$t < \frac{n}{2}$	dynamic, dynamic min cut $> 2t$
auth. P2P [5]	signatures	$t < \frac{n}{2}$	dynamic, dynamic min cut $> t$
auth. P2P [7]	-	$t < \frac{n}{3}$	static, connectivity $k > 2t$
broadcast [7]	-	$t < \frac{n}{3}$	static, connectivity $k > 2t$

are not even connected. However for this to happen, the network graphs would have to be "engineered" in a way that allows the message to progress one step at a time; which seems quite unrealistic.

Wang and Wattenhofer [7] consider *static* incomplete networks (and additionally consider *asynchrony*, where messages can take arbitrarily long to traverse a link in the graph). We build on one of their protocols, adapting it to our dynamic, synchronous setting.

1.4 Future Directions

Our work leaves open the question of proving / disproving our conjecture regarding whether perfect security is possible to achieve when $3t > k > 2t$. Another interesting open question is whether the upper bound on the round complexity of our constructions can be improved.

2 Preliminaries

2.1 Model

We consider a set $\mathcal{P} = \{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$ of n parties.

Corruption. A central adversary corrupts at most t parties. The corruption is *static*, i.e., the adversary is required to select the set of corrupted parties before the protocol execution. We distinguish between *passive* corruption where the adversary can access the internal state of corrupted parties and *active* corruption where the adversary has full control over the behavior of corrupted parties.

Communication network. Parties communicate over a *dynamic incomplete network* of secure (private and authentic) synchronous channels. In each round r (a.k.a. time-step), parties can communicate over the network graph G_r that has been selected by the adversary from a publicly-known family of graphs $\mathcal{G}$. The graph family $\mathcal{G}$ models the guarantees for honest parties, e.g., with respect to graph connectivity. There are three possibilities for modelling the network adversary:

- A *static* adversary is required to decide / commit to the set of graphs beforehand i.e. before the protocol begins.
- An (non-rushing) *adaptive* adversary can choose the graph for round $r + 1$ at the end of round r.
- A *rushing adaptive* adversary can first wait to see through what network edges the messages were attempted to be sent during round r, before determining the graph for round r.

In this paper, we consider a *rushing adaptive* network adversary.

We assume that honest parties are *oblivious* on the communication graph of a given round. That is, they only know the overall family $\mathcal{G}$, but not the actual G_r. So, in a protocol they may *attempt* to use any channel in their neighborhood of $\bar{\mathsf{G}} = \bigcup_{\mathsf{G}\in\mathcal{G}} \mathsf{G}$ whereas only channels in the unknown G_r will actually transmit. Honest parties do not learn which of their outgoing transmissions were successful.

Communication Complexity. In the setting with oblivious honest parties, every *attempted* use of a communication channel will count towards the communication complexity even if the actual transmission fails. For example, if an honest party tries to send a bit to every other party, this will count as n bits of communication irrespective of outcome.

Future Work: Other Models. For static or adaptive adversaries we could also consider *aware* honest parties, who are given G_r at the beginning of the round. We therefore assume that they will only use channels within G_r which are guaranteed to work. Aware honest parties will not attempt to send messages on channels which are not available, leading to better communication complextiy.

In the rushing adaptive setting, we could also consider looking at *retroactive awareness*, where honest parties are notified about successful transmissions.

2.2 Building Blocks

In this section we define the building blocks necessary for our protocols.

2.2.1 Threshold Secret Sharing Scheme

A t-out-of-n secret sharing scheme allows a party to "split" a secret into n shares that can be distributed among different parties. To reconstruct the original secret x at least $t+1$ shares need to be used.

▶ **Definition 1** (Secret Sharing). *A t-out-of-n secret sharing scheme is a tuple of efficient algorithms* $(\texttt{share}, \texttt{reconstruct})$ *defined as follows.*

- *The randomized algorithm* `share` *takes as input a secret $x \in \mathbb{F}$ and outputs a set of n share, i.e.,*

$$(s_1, \ldots, s_n) \in \mathbb{F}^n \xleftarrow{\$} \texttt{share}(x).$$

- *The reconstruct algorithm* `reconstruct` *takes as input a vector of at least $t+1$ shares and outputs either the secret x, or outputs $\perp$, i.e.,*

$$\{x, \perp\} \leftarrow \texttt{reconstruct}\left(\{s_i\}_{i\in S\subseteq[n],|S|>t}\right).$$

We require the following properties of a t-out-of-n secret sharing scheme:

Perfect Correctness. *The perfect correctness property requires that the shares of a secret x should always reconstruct to x. More formally, a secret sharing scheme is* perfectly correct *if for any secret x, for any subset $S \subseteq [n], |S| > t$,*

$$\mathsf{P}\left[x = x' \;:\; \begin{array}{c}(s_1, \ldots, s_n) \xleftarrow{\$} \texttt{share}(x),\\ x' \leftarrow \texttt{reconstruct}(\{s_i\}_{i\in S})\end{array}\right] = 1,$$

where the probability is taken over the random coins of `share`*. Moreover, if a negligible error probability is allowed, we simply say that the scheme is correct.*

Protocol 1: $\mathsf{Flood}(\mathcal{G}, \mathsf{S}, \mathsf{R}, m)$

Let $\bar{\mathsf{G}} = \bigcup_{\mathsf{G}\in\mathcal{G}} \mathsf{G}$. The sender S has message m as input. The protocol runs for n rounds:

- In each round the sender S sends m to all neighbors in $\bar{\mathsf{G}}$.
- Once a party $\mathsf{P} \in \mathcal{P} \setminus \{\mathsf{S}\}$ receives the first message m' from any neighbor, it will send m' to all neighbors in $\bar{\mathsf{G}}$ in all subsequent rounds.
- The receiver will output the first received message m' at the end of the protocol.

Figure 1 Simple flooding protocol, secure against $t < n$ passive corruptions in a connected network.

Privacy: *The privacy property requires that any combination of up to t shares should leak no information about the secret x. More formally, we say that a secret sharing scheme is* private *if for any x, and for any set $\mathbb{A} \subseteq \{1, \ldots, n\}$, $|\mathbb{A}| \leq t$ the distribution of the set $\{s_i\}_{i\in\mathbb{A}}$ is statistically independent of x.*

2.2.2 Instantiation

In our constructions, we use the Shamir's threshold secret sharing scheme [6]. We give a brief description of this scheme below. Informally, the shares output by the algorithm `share` correspond to the set of evaluations on n different points of a t-degree polynomial (whose coefficients are chosen at random from a finite field $\mathbb{F}$) with constant term as its secret. The algorithm `reconstruct` uses Lagrange interpolation to identify the t-degree polynomial that is consistent with the set of shares in order to return the constant term as the secret ($\perp$ is returned if no such polynomial exists).

3 Reliable Communication

In this section we describe protocols for reliable communication in our model, which we summarize in Table 1.

3.1 Passive Corruptions

In the case of passive corruption and a family $\mathcal{G}$ of connected graphs, a simple flooding protocol where parties echo the first received message can be used.

▶ **Lemma 2.** *Given a family $\mathcal{G}$ of connected graphs, Protocol 1 allows S to reliably send a message m to R in the presence of a rushing adaptive network adversary that passively corrupts $t < n$ parties. The protocol runs for n rounds and in each round a party sends $\mathcal{O}(n|m|)$ bits.*

Proof.

Correctness. Corruption is passive, so all parties which received a message actually get the sender's message m. Let H denote the set of nodes who have already heard the message, and D denote the set that have not. Since the actual communication graph chosen by the adversary is connected, there must be at least one edge between H and D. As any party in

Protocol 2: AuthenticatedP2P(Paths, S, R, m)

The set of possible paths Paths from S to R is public knowledge. The sender S has message m as input.
The protocol runs for LM rounds:

- In each round S sends $(m, \{\mathsf{S}\})$ to each neighbor that is on a path in Paths.
- Upon receiving (m', π) from a neighbor P_j, party $\mathsf{P}_i \in \mathcal{P} \setminus \{\mathsf{S}, \mathsf{R}\}$ does the following:
  1. If $i \in \pi$ or $j \notin \pi$, P_i ignores the message.
  2. Otherwise the party will in all subsequent rounds send $(m', \pi \cup \{i\})$ to all neighbors that are next nodes on paths in Paths for which $\pi \cup \{i\}$ forms a prefix.
- Upon receiving (m', π) from a neighbor P_j, R stores (m', π) unless $\mathsf{R} \in \pi$ or $j \notin \pi$.

At the protocol end, if party R has more than t stored values (m', π) with disjoint paths in Paths, R outputs m'.

Figure 2 The modified authenticated P2P protocol of Wang and Wattenhofer [7].

H will try to send it to all potential neighbors in every round, at least one party in D will learn the message in each round. By induction the message must reach the receiver after at most n rounds.

Complexity. In each round a party sends at most one message of size $|m|$ to at most n. The round complexity follows from the protocol description. ◀

3.2 Active Corruptions

For the active corruption setting, we assume that the honest parties' knowledge about $\mathcal{G}$ comes in the form of a set Paths of possible paths between sender S and receiver R. We assume that the paths have length of at most L and $|\mathsf{Paths}| = M$.

In Protocol 2 we adapt the peer-to-peer protocol from [7] to allow for reliable communication in the presence of a rushing adaptive adversary.

▶ **Theorem 3.** *The authenticated P2P protocol of Wang and Wattenhofer [7]* modified to only accept messages received over $t + 1$ disjoint paths *as described in Protocol 2 achieves reliable communication between* S *and* R *in the presence of a rushing adaptive network adversary that actively corrupts at most t parties as long as the network has connectivity $k > 2t$ in every round. The protocol runs for LM rounds.*

We approach Theorem 3 by first introducing Lemma 4 and Lemma 5.

▶ **Lemma 4.** *If the network has connectivity k in every round, then there are k disjoint paths from the sender to the receiver* over time.

Proof. Since there are a finite number of sets of k disjoint paths from sender S to receiver R, and the adversary has to choose one such path set in every round, it follows that after a finite number of rounds one path set will have been chosen sufficiently many times that the message had the opportunity to traverse all of its paths. ◀

The following lemma is useful for analyzing the round complexity of our constructions.

▶ **Lemma 5.** *Consider a path set* Paths *which is k-connected and a sender* S *who sends a message along each path of* Paths*. Then, the receiver* R *would receive the messages from a set of k disjoint paths in at most LM rounds, where L is the maximal length of paths in* Paths *and M is the cardinality of* Paths*.*

Proof. Let $\mathcal{C}_r$ be the set of paths along which S's message has reached R at round r, and let Paths_r be the set of paths between S and R chosen in that round. In the first round, the message advances along at least k edges. In every subsequent round, if $\mathcal{C}_r$ contains a set of k disjoint paths, then we are done. Otherwise, it must be that $|\mathsf{Paths}_r \setminus \mathcal{C}_r| > 0$, so the message must advance along at least one edge it has not advanced along before. Since the total number of edges across all paths in Paths is at most LM, we can be certain that we are done once the message advances along all of them, we must be done after at most LM rounds. ◀

We can now give the proof for Theorem 3.

Proof.

Correctness. Assume S and R are honest.

First consider the network without the corrupted nodes. This network is guaranteed to be $t+1$ connected in every round (the overall network is at least $2t+1$ connected). It follows from Lemma 4 and Lemma 5 that the message m sent by the sender will arrive at the receiver R in at most LM rounds via at least $t+1$ disjoint paths. This makes m a valid output.

Next, consider any message $m' \neq m$. If R gets a tuple (m', π') there must be a corrupt P_j that sent (m', π'') for $\pi'' \subset \pi'$ to an honest party. Honest parties only forward a tuple if the sender of the tuple is in the path information. This implies that $j \in \pi''$ and thus $j \in \pi'$. There are at most t corrupted parties, so R will receive m' on at most t disjoint paths. Hence m' can never be a valid output. This means m is the unique output for R after LM rounds.

Round complexity. Follows from the protocol description. ◀

Analyzing the communication complexity of Protocol 2

We observe that the maximum communication complexity of any party in a round r is $\mathcal{O}(2^n n^2 |m|)$. This is because a party in round $r-1$ may have received messages from various paths represented by different sets (which we refer to as metadata). Since there could be 2^{n-1} such sets (all possible sets that exclude this party), we can infer that a party has to communicate a message of size $|m|$ and metadata of size at most $\mathcal{O}(n2^n)$ (n bits are sufficient to represent one set) to each of her neighbours (which are at most $\mathcal{O}(n)$), which adds up to a communication complexity of $\mathcal{O}(2^n n(n + |m|))$.

Lastly, we point that the computation done by the receiver R to check if she has received values along $t+1$ disjoint paths would involve $\mathcal{O}(LMn)$ computation per round. This can be done by the receiver R as follows: consider the graph $\mathcal{C}_r$ formed by the set of paths containing the same message along which S's message has reached R at round r. Apply the Ford-Fulkerson algorithm [4] (with complexity $O(|E|n)$, where $|E|$ denotes the number of edges and n denotes the number of nodes) on $\mathcal{C}_r$ to check if it is $t+1$ connected. If yes, output this common message. Else, try with set of paths containing a different message. If none of them succeed, try again in the next round.

Efficiency

In this section, we discuss a special case when this authenticated communication protocol becomes *efficient* (has complexity polynomial in n). For this, suppose there are k *short* paths from sender to receiver at each round. We let c denote the upper bound on the length of such a "short" path. First, we analyze the case where c is any constant (Lemma 6). Finally, we argue that for sufficiently large k, we are guaranteed to have many short paths (Lemma 7).

▶ **Lemma 6.** *If* S *is connected to* R *via* k *disjoint paths of length at most* c *in every round (for a constant* c*), then the authenticated P2P protocol of Wang and Wattenhofer [7] described in Figure 2 runs in at most a polynomial (in* n*) number of rounds.*

Proof. Let Paths' be a set of paths with length at most c. Protocol 2 on Paths' runs in $c|\mathsf{Paths}'|$ rounds as shown in *Theorem* 3. So it remains to analyze the size of $|\mathsf{Paths}'|$. Let $|\mathsf{Paths}_\ell|$ denote how many paths of length at most ℓ exist from S to R in Paths'. We have $|\mathsf{Paths}_1| \leq 1$ and for $\ell > 1$, $|\mathsf{Paths}_\ell| \leq |\mathsf{Paths}_{\ell-1}| + \frac{(n-2)!}{((n-2)-(\ell-1))!}$[3]. We have $|\mathsf{Paths}'| = |\mathsf{Paths}_c|$ which is polynomial in n for constant c. We can thus conclude that the protocol terminates in at most a polynomial (in n) number of rounds. ◀

Ensuring Enough Paths of Constant Length. We have shown that if we have k disjoint paths of constant length c at every timestep, S's message will reach R along k disjoint paths in polynomial time. Of course, k-connectivity *over constant length paths* is in general a much stronger assumption than k-connectivity. However, we show here that (loosely speaking) if we have many disjoint paths, this implies that at least some of them must be short, and in particular, if we have k-connectivity for k linear in n, this implies we have k'-connectivity over constant length paths, where k' can be very close to k.

▶ **Lemma 7.** *Say we have* k *disjoint paths in a graph on* n *nodes. For any* L*, at least* $k - \frac{n}{L+1}$ *of these have length at most* L*.*

Proof. Let $k_{\leq L}$ and $k_{>L}$ be the number of paths of length at most L and greater than L, respectively, among the k given ones. The subset of paths of length greater than L contain at least $(L+1)k_{>L}$ distinct nodes, so we have $(L+1)k_{>L} \leq n$, implying $k_{>L} \leq \frac{n}{L+1}$. Since clearly $k = k_{\leq L} + k_{>L}$, the lemma follows. ◀

An immediate consequence of this is that if $k = dn$ for a constant fraction d, we are guaranteed to have at least $n(d - \frac{1}{L+1})$ disjoint paths of length L. By choosing a large enough but constant L, we can have $\Omega(n)$-connectivity over constant length paths; in fact that underlying constant can be chosen arbitrarily close to d.

In particular, assume we want to tolerate a constant fraction of corrupted players, as is standard in MPC. We know that for t active corruptions and even for a static network, we must always have connectivity at least $2t + 1 = cn$ (for a constant c); otherwise broadcast is impossible. Therefore, in the dynamic case, if we ask for slightly larger connectivity, namely $k = dn$ for any $d > c$, the above lemma allows us to assume $2t + 1$-connectivity over constant length paths, implying that our protocols will run in polynomial time.

We state the formal theorem below.

[3] where the latter term is the number of ways of choosing $\ell - 1$ intermediate nodes among $n - 2$ nodes (excluding S and R).

▶ **Theorem 8.** *The modified authenticated P2P protocol of Wang and Wattenhofer [7] as described in Protocol 2 is an* efficient reliable communication protocol *(i.e. runs in polynomial time and with polynomial complexity) in the presence of t active corruptions as long as the network has connectivity $k = dn$ for any $d > c$ (where c is a constant) in every round (using the set of all paths of length at most c as* Paths*).*

4 Private Communication

In this section, we look at the feasibility of establishing a secure channel between sender S and receiver R. The knowledge on $\mathcal{G}$ is given as a set Paths of bidirectional paths between S and R. The set is of size M and paths in the set are of length of at most L.

As a starting point, assume that S and R somehow have shared secret randomness o. Given the results from the previous section, they could establish a reliable channel to securely transmit message m as $c = m + o$. This reduces the problem of secure communication to establishing shared randomness between S and R. At a first glance this seems as difficult as the original problem. However, we note that there is a slight difference i.e. this value o (unlike m) need not be a "fixed" value pre-determined by S but can be dynamically determined during the protocol.

This is exactly what we exploit in our upper bounds that have the following common approach: S chooses a set of random values, one for each path in Paths. Next, R upon receiving "sufficiently many" random values reports back to S which paths she received information from. For this, R acts as a sender and can rely on a reliable communication protocol. This is because while we may want to hide the random values along paths that the adversary does not have access to (i.e. the paths that comprise of only honest nodes), there is no harm in revealing to the adversary the identity of the paths R received information from (as these paths were in fact determined by the dynamic adversary). Given this path information both S and R can compute o from the randomness sent along those paths. Finally, S can mask the actual m with o and send it over a reliable channel.

For simplicity, we assume in the following that $m \in \mathbb{F}$.

4.1 Passive Corruptions

In this section, we present a protocol that constructs a secure channel given that the graph is at least $t + 1$-connected.

Consider plugging in the above common approach in a network with connectivity $k > t$ where the dynamic adversary corrupts up to t nodes passively. We are guaranteed that R would receive random values from $(t + 1)$ disjoint paths, among which the adversary has access to at most t of them (because in the worst case, there could be one corrupt node in each of the t disjoint paths). These $(t + 1)$ random values could simply be viewed as an additive sharing of the shared randomness o that remains private from this passive adversary. Tying this with the above outlined approach, R would reliably communicate the identity of these $(t + 1)$ disjoint paths to S, allowing S to compute o and reliably communicate the masked secret $c = m + o$. The formal description of the protocol is given as Protocol 3.

▶ **Theorem 9.** *Protocol 3 is perfectly secure protocol that allows* S *to securely send m to* R *in the presence of a rushing adaptive adversary that passively corrupts at most t parties, as long as the network has connectivity $k > t$. The protocol runs for $LM + 2n$ rounds and in each round a party sends $\mathcal{O}(Mn^2(\log M + |m|))$ bits.*

Protocol 3: $\Pi^{\mathsf{prv}}_{\mathsf{perf},\mathsf{sh}}(\mathsf{Paths}, \mathsf{S}, \mathsf{R}, m)$

The set of possible paths Paths is public knowledge. The sender S has message m as private input.

Randomness Generation For ML rounds the parties do the following:

- For each path $p \in \mathsf{Paths}$ the sender S:
  1. Sample randomness $r_p \in \mathbb{F}$.
  2. In each time step send $m_p = (p, r_p)$ to the first node on the path p until the phase is complete.
- Once intermediate node P_i receives the first message $m_p = (p, r_p)$ on path p, it will echo m_p to the next node on the path in each of the subsequent time steps until the phase is complete.
- The receiver node R initializes sets $\mathsf{RecPaths} = \emptyset$ and sets $\mathsf{RecPaths} = \mathsf{RecPaths} \cup \{p\}$ upon receiving $m_p = (p, r_p)$ for any $p \in \mathsf{Paths}$.

Afterwards, the receiver define $\mathsf{GoodPaths} \subseteq \mathsf{RecPaths}$ as a set of $t+1$ *disjoint* paths and send GoodPaths to S using an instance of Protocol 1. This concludes the randomness generation phase.

Secure Communication The parties do the following:

1. Both S and R (locally) compute $o = \Sigma_{p \in \mathsf{GoodPaths}} r_p$.
2. S sends $c = m + o$ to R using an instance of Protocol 1.
3. R outputs $m = c - o$.

Figure 3 Perfectly-secure private communication protocol against $t < n$ passive corruptions in a network with connectivity $k > t$.

Proof.

Correctness. As the graph is at least $t+1$ connected, receiver R will have received randomness over at least $t+1$ disjoint paths after LM rounds (cf. Lemma 5). This makes GoodPaths well defined. Correctness therefore follows by the correctness of Protocol 1 (cf. Lemma 2) and the correctness of one time pad encryption.

Privacy. To argue privacy, we note that the adversary has access to the random values corresponding to at most t paths among the $(t+1)$ disjoint paths constituting GoodPaths. The property of additive secret sharing guarantees that o remains perfectly hidden from the adversary. It now directly follows from the security of the one-time pad encryption the adversary does not learn m from c alone.

Complexity. Protocol 1 has a round complexity of n. This implies a round complexity of $LM + 2n$. Lastly, we analyze the communication complexity. While sending the randomness forward, the complexity per party in each round is $\mathcal{O}(M(\log M + |m|))$. Observe that GoodPaths can be encoded in $\mathcal{O}(n \log M)$ bits, so the per party per round complexity of both instances of Protocol 1 is bounded by $\mathcal{O}(n^2(\log M + |m|))$. This gives an overall (loose) bound of $\mathcal{O}(Mn^2(\log M + |m|))$. ◀

4.2 Active Corruptions

In this section, we provide protocols for secure communication in the presence of an adversary that actively corrupts parties.

4.2.1 Statistical Security with $k > 2t$

The above construction (Protocol 3) does not withstand active corruptions as the active adversary could tamper with the random values along the paths in GoodPaths where there is an actively corrupt node. This would lead to R determining an incorrect random one-time pad (o) i.e. different than the one computed by S; resulting in R obtaining the wrong secret. Further, the adversary could also tamper with the path information (i.e. the sequence of nodes forming the path); this may potentially lead R to wrongly believe that certain values are coming from "disjoint" paths when in fact they are not.

To detect cheating of the above type in the statistical setting, one could authenticate the random values (using information theoretic mac) and ensure that R verifies each potentially tampered random value accompanied by its mac using the corresponding verification key determined by the honest S. However, for this to work, we should make sure that the verification key remains unknown to the adversary and untampered until it reaches R! We resolve this seemingly circular issue in the following way: To authenticate a random value, say r_p along a path p, S generates a mac using a different verification key for every path disjoint from p. R accepts r_p only if its macs verify against at least t paths that are mutually disjoint and also disjoint to p. The idea is that at least one of these t paths would comprise of only honest nodes and the verification keys sent along this all-honest path would be untampered and unknown to the adversary. (Note that if the value along p is tampered, then there must be at least one corrupt node already in p; so there can be at most $t-1$ paths disjoint to p that could be influenced by the adversary.) This completes the high-level overview of the protocol, whose formal details are described in Protocol 4.

▶ **Theorem 10.** *Protocol 4 is a statistically-secure protocol that allows* S *to securely send* m *to* R *in the presence of a rushing adaptive adversary that actively corrupts at most* t *parties, as long as the network has connectivity* $k > 2t$*. The protocol runs for* $3LM$ *rounds and in each round a party sends* $\mathcal{O}(n^3 2^n(|m| + \log M) + M(|m| + \lambda \cdot M))$ *bits.*

Proof.

Correctness. By Theorem 3 sender and receiver will agree on GoodPaths and the receiver will receive the sender's c. So correctness follows if sender and receiver agree on o. This is the case if R actually got the sender's randomness for each path in GoodPaths. So assume there must exist at least one path, say p_i such that R obtained $r'_{p_i} \neq r_{p_i}$ but $p_i \in$ GoodPaths. This can occur only if p_i contains at least one corrupt node on behalf of which the adversary tampered with the random value and potentially its set of accompanying macs. However, since $p_i \in$ GoodPaths, it must hold that the macs verified against verification keys received along t other disjoint paths, which must include at least one verification key that was untampered and unknown to the adversary. This is because there can be at most $t-1$ corrupt nodes along these t disjoint paths; therefore there must be a path containing all honest nodes along which the verification key was correct (i.e. the same as chosen by S). It now follows directly from the unforgeability property of the mac that the (potentially) adversarial chosen mac can verify against this verification key successfully for $r'_{p_i} \neq r_{p_i}$ only with negligible probability. This implies statistical correctness.

Protocol 4: $\Pi^{\mathsf{prv}}_{\mathsf{stat,mal}}(\mathsf{Paths}, \mathsf{S}, \mathsf{R}, m)$

The set of possible paths Paths is public knowledge. The sender S has message m as private input.

Randomness Generation For ML rounds the parties do the following:

- For each path $p_i \in \mathsf{Paths}$ the sender S:
  1. Sample randomness $r_{p_i} \in \mathbb{F}$ and for each disjoint path p_j sample key o_{p_i,p_j} and corresponding mac mac_{p_i,p_j}. Let $\mathcal{K}_{p_i} = \{o_{p_j,p_i}\}$ and $\mathcal{M}_{p_i} = \{\mathsf{mac}_{p_i,p_j}\}$ the set of macs.
  2. In each time step send $m_{p_i} = (p_i, r_{p_i}, \mathcal{M}_{p_i}, \mathcal{K}_{p_i})$ to the first node on the path p_i until the phase is complete.
- Once intermediate node P receives the first message $m_p = (p, r_p, \mathcal{M}_p, \mathcal{K}_p)$ on path p, it will echo m_p to the next node on the path in each of the subsequent time steps until the phase is complete.
- The receiver node R initializes sets $\mathsf{RecPaths} = \emptyset$ and sets $\mathsf{RecPaths} = \mathsf{RecPaths} \cup \{p\}$ upon receiving $m_p = (p, r_p, \mathcal{M}_p, \mathcal{K}_p)$.

Afterwards the receiver initializes $\mathsf{GoodPaths} = \emptyset$ and does the following for each $p_i \in \mathsf{RecPaths}$:

1. Check if $\mathsf{RecPaths}$ contains a set Paths' of at least t path disjoint from p_i such that for each $p_j \in \mathsf{Paths}'$ the key o_{p_i,p_j} (received via p_j) confirms the mac mac_{p_i,p_j} on r_{p_i} (both received via p_i).
2. If the above check holds (and $|\mathsf{GoodPaths}| < t$) set $\mathsf{GoodPaths} = \mathsf{GoodPaths} \cup \{p\}$.

Then R sends $\mathsf{GoodPaths}$ to S using an instance of Protocol 2. This concludes the randomness generation phase.

Secure Communication As in Protocol 3 the message is sent one-time padded except that Protocol 2 is used for the reliable communication channel.

Figure 4 Statistically-secure private communication protocol against $t < n$ active corruptions in a network with connectivity $k > 2t$.

Privacy. Privacy follows by the same argument as in the proof of Theorem 9.

Complexity. Protocol 2 has a round complexity of LM. This implies a round complexity of $3LM$. Lastly, we analyze the communication complexity. The two instances of Protocol 2 have a communication complexity of $\mathcal{O}(n^3 2^n(|m| + \log M))$. The messages m_p have a complexity of $\mathcal{O}(|m| + \lambda \cdot M)$ as the mac and key sets are $\mathcal{O}(\lambda \cdot M)$. So in the first LM rounds a party communicates $\mathcal{O}(M(|m| + \lambda \cdot M))$ bits per round. This totals upto communication complexity of $\mathcal{O}(n^3 2^n(|m| + \log M) + M(|m| + \lambda \cdot M))$. ◀

4.2.2 Perfect Security with $k > 3t$

In the above construction, settling for statistical security allowed us to use authentication tools to detect cheating. We now analyze how to achieve perfect security, where it becomes more challenging to deal with the adversary tampering with the random values sent across the paths.

To handle this, we make S send redundant information in a way that allows R to detect such misbehaviour. Instead of using sum sharing, we rely on threshold sharing in the following way: We let R report back $(2t+1)$ disjoint paths (instead of $t+1$ as before). Next, we make S compute a threshold sharing (with threshold t) of the message m and mask the shares (instead of the message directly) using the $(2t+1)$ random values corresponding to paths that were reported by R. Once these $(2t+1)$ masked shares are reliably communicated to R, R can retrieve the shares and attempt to reconstruct the secret. The main idea is that, unlike before, R can now check that she has the "correct" secret by checking that all the $(2t+1)$ shares lie on a t-degree polynomial. This is because, the set of shares will comprise of $(t+1)$ untampered shares (that were masked with random values along the paths that comprised only of honest nodes) which suffice to uniquely determine the t degree polynomial. If the check fails, we make R retry with another set of $(2t+1)$ disjoint paths until she finds one that verifies. To accommodate for this, we assume larger connectivity i.e. $k > 3t$. This completes the high-level idea of the protocol, which is formally described in Protocol 5 (Section A). We state the formal theorem below (whose proof appears in Section A.1).

▶ **Theorem 11.** *Protocol 5 is a perfectly-secure protocol that allows* S *to securely send* m *to* R *in the presence of a rushing adaptive adversary that actively corrupts at most* t *parties, as long as the network has connectivity* $k > 3t$*. The protocol runs for* $3LM$ *rounds and in each round a party sends* $\mathcal{O}(nM^n(M+|m|))$ *bits.*

We refer to Section A for our other results related to perfect security.

References

1 E. R. Berlekamp and L. Welch. Error correction of algebraic block codes. *US Patent Number 4,633,470. Issued Dec.*, 1986.

2 Danny Dolev. The byzantine generals strike again. *Journal of algorithms*, 3(1):14–30, 1982.

3 Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *Journal of the ACM (JACM)*, 40(1):17–47, 1993.

4 L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

5 Alexandre Maurer, Sébastien Tixeuil, and Xavier Defago. Communicating reliably in multihop dynamic networks despite byzantine failures. In *2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS)*, pages 238–245, 2015. doi:10.1109/SRDS.2015.10.

6 Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

7 Ye Wang and Roger Wattenhofer. Asynchronous byzantine agreement in incomplete networks. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, AFT '20, pages 178–188, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3419614.3423250.

A Perfect Security

A.1 Perfect Security with $k > 3t$

In this section, we present the formal protocol (Protocol 5) and the security proof of Theorem 11.

Protocol 5: $\Pi^{\mathsf{prv}}_{\mathsf{perf,mal}}(\mathsf{Paths}, \mathsf{S}, \mathsf{R}, m)$

The set of possible paths Paths is public knowledge. The sender S has message m as private input.

Randomness Generation For ML rounds the parties do the following:

- The sender S does the following:
  1. For each set $\mathsf{Paths}' \subset \mathsf{Paths}$ such that the paths in Paths' are disjoint and $|\mathsf{Paths}'| = 2t+1$ the receiver samples for each $p \in \mathsf{Paths}'$ random value $r_p^{\mathsf{Paths}'\,a}$. For any $p \in \mathsf{Paths}$ denote by $r_p = \{r_p^{\mathsf{Paths}'}\}$ the set of all sampled random values.
  2. In each time step send $m_p = (p, r_p)$ to the first node on the path p until the phase is complete.
- Once intermediate node P_i receives the first message $m_p = (p, r_p)$ on path p, it will echo m_p to the next node on the path in each of the subsequent time steps until the phase is complete.
- The receiver node R initializes sets $\mathsf{RecPaths} = \emptyset$ and sets $\mathsf{RecPaths} = \mathsf{RecPaths} \cup \{p\}$ upon receiving $m_p = (p, r_p)$ for any $p \in \mathsf{Paths}$.

Afterwards, the receiver R sets $\mathsf{GoodPaths}$ to the maximal set of pairwise disjoint paths in $\mathsf{RecPaths}$. This can be done using the Ford-Fulkerson Algorithm. Then R sends $\mathsf{GoodPaths}$ to S using an instance of Protocol 2. This concludes the randomness generation phase.

Secure Communication The message m is sent as follows.

1. Let $\mathcal{W}$ denote the set of all subsets $\mathsf{Paths}' \subseteq \mathsf{GoodPaths}$ such that $|\mathsf{Paths}'| = 2t+1$ For each $\mathsf{Paths}' \in \mathcal{W}$:
   - The sender computes shamir-sharing of the message m with threshold t as $(s_1^{\mathsf{Paths}'}, \ldots, s_{2t+1}^{\mathsf{Paths}'}) \leftarrow \texttt{Shamir.share}(m)$.
   - Set $c_{p_i}^{\mathsf{Paths}'} = s_i^{\mathsf{Paths}'} + r_{p_i}^{\mathsf{Paths}'}$ for each $p_i \in \mathsf{Paths}'$, where $r_{p_i}^{\mathsf{Paths}'}$ is the appropriate random mask (i.e. the random value that was chosen for path p_i corresponding to set Paths').
2. Sender S sends $\{c_{p_i}^{\mathsf{Paths}'}\}_{p_i \in \mathsf{Paths}', \mathsf{Paths}' \in \mathcal{W}}$ to R using an instance of Protocol 2.
3. For each $\mathsf{Paths}' \in \mathcal{W}$ the receiver R:
   - Recovers the shares as $s_i = c_{p_i}^{\mathsf{Paths}'} - r_{p_i}^{\mathsf{Paths}'}$ for $p_i \in \mathsf{Paths}'$
   - Checks if $m' \leftarrow \texttt{reconstruct}(\{s_i\}_{i \in [2t+1]})$ results in $m' \neq \perp$ (this step essentially checks if all the shares lie on a t degree polynomial). If yes, the receiver outputs m'.

[a] We assume that all possible sets Paths' are lexicographically ordered. Abusing notation, when Paths' is used in superscript, it refers to the appropriate index of Paths' based on this ordering.

Figure 5 Perfectly-secure private communication protocol against $t < n$ active corruptions in a network with connectivity $k > 3t$.

Proof.

Correctness. By Theorem 3 sender and receiver will agree on $\mathsf{GoodPaths}$ and thus on $\mathcal{W}$. This also hold for the set of one-time padded shares $\{c_{p_i}^{\mathsf{Paths}'}\}$.

By Lemma 4 the set $\mathcal{W}$ must contain at least $2t+1$ disjoint paths that comprise of only honest nodes. So, there must exist at least one candidate set Paths' which will lead would lead to the correct output of m. On the other hand any set Paths' that leads R to output m' contains at least $t+1$ honest paths (This is because at most t paths in Paths' could be such that it has a corrupt node.). The adversary cannot tamper with the random values or

path information across these $(t+1)$ paths, therefore we are guaranteed that $(t+1)$ among the $(2t+1)$ shares retrieved by R must indeed be correct. Since these suffice to uniquely determine the t-degree polynomial that S used for Shamir sharing, it follows that the message reconstructed by R must be correct.

Privacy. Note that for each candidate set $\mathsf{Paths}' \subseteq \mathcal{W}$, the adversary knows at most t shares, namely, those that were masked using random values corresponding to the (at most) t paths in Paths' that the adversarial nodes were a part of. Privacy of the secret now follows directly from the privacy guarantee of threshold sharing and the security of one-time pad encryption (which holds as each random value is used for encryption at most once).

Complexity. The round complexity follows analogous to the proof of Theorem 10. Lastly, we analyze the communication complexity. The complexity is dominated by randomness generation phase. There are fewer than M^{2t+1} sets Paths' such that $|\mathsf{Paths}'| = 2t+1$. Since there are $2t+1$ paths p in each Paths', the sender picks fewer than $(2t+1)M^{2t+1}$ random values $r_p^{\mathsf{Paths}'}$. Each path can be represented using $\log M$ bits, and each random value $r_p^{\mathsf{Paths}'}$ can be represented using $|m|$ bits; so, the communication per round per party is $\mathcal{O}((2t+1)M^{2t+1}(\log M + |m|)) = \mathcal{O}(nM^n(\log M + |m|))$. ◀

Improving the expected complexity. In Protocol 5, the primary communication bottleneck is due to the fact that S has to account for all possible path sets of size $2t+1$. We propose the following modification to improve the expected communication complexity: Similar to Protocol 4, S could choose just one random value per path, and augment these values with information-theoretic MACs. Accordingly R could include a path p in GoodPaths only if t of the MACs have verified successfully using verification keys sent across t disjoint paths (that are disjoint from p as well). Once GoodPaths comprises of $2t+1$ disjoint paths, these are reported by R to S, who sends masked threshold shares corresponding to just this one subset. However, the check in the communication phase still remains the same i.e. R continues to check if *all* the shares lie on a t-degree polynomial (which maintains that the protocol is perfectly-secure). If this check fails, then we re-run the protocol beginning with the randomness generation phase.

While the communication complexity of the above modified protocol (which we refer to as the communication-efficient variant of Protocol 5) is same as Protocol 5 in the worst case (which occurs when the all-honest subset is the last one to be tried by R), the modified protocol has an expected running time of $\mathsf{poly}(ML)$. This is because the security of the MAC guarantees that the adversary can make an iteration fail only with negligible probability, say μ. The probability that the protocol does not terminate in γ iterations is therefore μ^γ.

We remark that the MACs help only to improve the expected communication complexity, but the protocol is still perfectly secure (due to the verification done by the receiver R before accepting the output).

A.2 Perfect Security with $k > 4t$

In the above construction (Protocol 5), we make S account for all possible disjoint path sets of size $2t+1$ as she does not know in advance which set of $2t+1$ all-honest disjoint paths will eventually reach R. We observe that this can be avoided by assuming a larger connectivity of $k > 4t$. Higher connectivity allows transferring more redundant information, enabling R to recover from the incorrect information rather than simply detect it. We tweak the above

construction to let R report back $(3t+1)$ disjoint paths instead. Now upon receiving $(3t+1)$ shares among which at most t could be incorrect, R can use error-correction techniques (such as Reed-Solomon error correction) to reconstruct the correct t-degree polynomial despite the errors. This completes the high-level description of the protocol, which is formally described in Protocol 6.

Protocol 6: $\Pi^{\mathsf{prv},k>4t}_{\mathsf{perf},\mathsf{mal}}(\mathsf{Paths},\mathsf{S},\mathsf{R},m)$

The set of possible paths Paths is public knowledge. The sender S has message m as private input.
We assume parties have access to decoding algorithm $\Pi_{\mathtt{RSDec}}(W,t)$ that takes as input a vector W of shamir shares with threshold t (viewed as a Reed-Solomon codeword) where some of these may be incorrect, and either removes the errors and returns the correct secret if there are at most $\frac{|W|-t-1}{2}$ of them, or produces $\perp$ if there are more than $\frac{|W|-t-1}{2}$ errors. This can be instantiated for instance by Berlekamp-Welch algorithm [1].

Randomness Generation Same as in Protocol 3 where S sends one random value per path $p \in \mathsf{Paths}$. The set GoodPaths is selected to contain at least $3t+1$ disjoint paths and sent back to S using Protocol 2.

Secure Communication The message m is sent as follows.

1. S computes a shamir-sharing of the message m with threshold t as $(s_1,\ldots,s_{k'}) \xleftarrow{\$} \mathtt{Shamir.share}(m)$, where $k' = |\mathsf{GoodPaths}|$.
2. The sender computes $c_{p_i} = s_i + r_{p_i}$ for each $p_i \in \mathsf{GoodPaths}$.
3. The sender S sends $\{c_{p_i}\}_{p_i \in \mathsf{GoodPaths}}$ to R using an instance of Protocol 2.
4. The receiver computes the shares as $s_i = c_{p_i} - r_{p_i}$ for $p_i \in \mathsf{GoodPaths}$.
5. The sender outputs $m \leftarrow \mathtt{RSDec}(\{s_i\})_{i\in[k']}, t)$.

Figure 6 Perfectly-secure private communication protocol against $t < n$ active corruptions in a network with connectivity $k > 4t$.

▶ **Theorem 12.** *Protocol 6 is a perfectly-secure protocol that allows* S *to securely send* m *to* R *in the presence of a rushing adaptive adversary that actively corrupts at most* t *parties, as long as the network has connectivity* $k > 4t$*. The protocol runs for* $3LM$ *rounds and in each round a party sends* $\mathcal{O}(Mn^3 2^n(|m| + \log M))$ *bits.*

Proof.

Correctness. First, we note that Lemma 4 and the presence of at most t active corruptions imply that GoodPaths must consist of at least $3t+1$ disjoint paths, therefore $|W| > 3t$ holds. Since the quantity $\frac{|W|-t-1}{2} \geq t$, error-correction would definitely be successful as long as there are at most t errors. This is indeed true in our case as only the shares that were masked using random values sent along paths in GoodPaths where there was at least one corrupt node could be incorrect; and there could be at most t such paths. Correctness now follows directly from the correctness of the error correction algorithm RSDec and the correctness of the reliable communication protocols.

Privacy. Privacy follows by similar argument as Theorem 11. Since the adversary knows at most t shares, (namely, those that were masked using random values corresponding to the (at most) t paths in GoodPaths that the adversarial nodes were a part of), privacy follows directly from the privacy guarantee of threshold sharing and the security of one-time pad encryption.

Complexity. The round complexity follows analogous to the proof of Theorem 10. Lastly, we analyze the communication complexity. The two instances of the reliable communication protocol Protocol 2 have a communication complexity of $\mathcal{O}(n^3 2^n(|m|+\log M))$. The messages m_p have a complexity of $M(|m| + \log M)$. This totals upto communication complexity of $\mathcal{O}(Mn^3 2^n(|m| + \log M))$. ◀

A.3 Perfect Security With $k > 2t$?

In this section, we elaborate on our conjecture that it is impossible to achieve perfectly private communication against an active adversary in a dynamic network with connectivity $k > 2t$. The guarantee in such a network is that the receiver R receives values sent via a set of at least $t+1$ disjoint paths (if she waits long enough, as shown in Lemma 5). Consider any protocol which, like ours, tries to identify such a set of paths where only correct values were received, it then uses the data received on these paths and public communication to send the secret message. Of course, if there is exactly one suitable set of paths, we would be done, as this would be the all-honest set of paths. However, as we now explain, there are cases where there are multiple sets and where the receiver cannot identify the right one with certainty.

For simplicity, consider $t = 1$ and $k = 2t + 1 = 3$. Say, the adversary chooses the same 3-connected graph, say $\mathcal{G}$ in all the rounds: $\mathcal{G}$ has three disjoint paths p_1, p_2 and p_3, where the corrupt node is the last node on path p_2 denoted as P_c, adjacent to the receiver P_r (see Figure 7). The adversary has the following strategy: She blocks the message along the path p_2 and sends two fabricated paths to the receiver node P_r instead. **(a)** p'_1 that intersects with p_1 but is disjoint from p_3. **(b)** p'_3 that intersects with p_3 but is disjoint from p_1. Note that the last node in each of these fabricated paths would be the corrupt node P_c who directly communicates them to the receiver node P_r. Now, from the perspective of the receiver, there are three disjoint sets of size 2, namely **(1)** $\{p_1, p_3\}$, **(1)** $\{p'_1, p_3\}$ and **(3)** $\{p_1, p'_3\}$. Assuming that the adversary tampers with the values associated with p'_1, p'_3, only the first set has correct values. However, there is no way for the receiver to identify $\{p_1, p_3\}$ to be the all-honest set. This is because the receiver could have the same identical view in the following different scenario – when the last node in p_2 was actually honest, the paths p'_1, p'_3 and p_3 existed and the values delivered for these paths were correct. While, on the other hand, the path p_1 and the value it carries was fabricated by a corrupt node. In this case, the receiver would have the same set of three candidate disjoint sets and values, but $\{p'_1, p_3\}$ is the correct set this time.

While authentication can be used to select the correct set with overwhelming probability as we showed earlier, this cannot give perfect security. Hence if the protocol is not allowed to abort but must continue with some set, there is a non-zero probability that a set will be chosen with one correct and one incorrect value. Intuitively, it is not possible to send a private message reliably based on such data and public communication: the output of the receiver must depend on the values received on both paths (as well as the public communication), if it only depends on one of them, this might be value the adversary knows and the message would not be private. But if it depends on both values and one is tampered with, the output will be incorrect.

This type of argument is of course not a real impossibility proof as it only considers one type of protocol, it should only be taken as evidence for our conjecture.

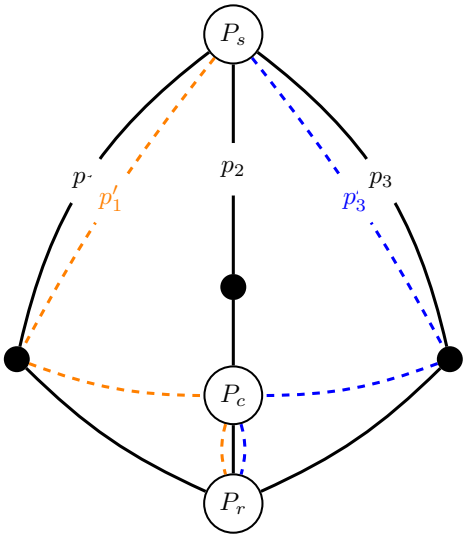


Figure 7 3-connected example graph.

Locally Covert Learning

Justin Holmgren
NTT Research, Sunnyvale, CA, USA

Ruta Jawale
University of Illinois at Urbana-Champaign, IL, USA

Abstract

The goal of a covert learning algorithm is to learn a function f by querying it, while ensuring that an adversary, who sees all queries and their responses, is unable to (efficiently) learn any more about f than they could learn from random input-output pairs. We focus on a relaxation that we call *local covertness*, in which queries are distributed across k servers and we only limit what is learnable by $k-1$ colluding servers.

For any constant k, we give a locally covert algorithm for efficiently learning any Fourier-sparse function (technically, our notion of learning is improper, agnostic, and with respect to the uniform distribution). Our result holds unconditionally and for computationally unbounded adversaries. Prior to our work, such an algorithm was known only for the special case of $O(\log n)$-juntas, and only with $k = 2$ servers [9].

Our main technical observation is that the original Goldreich-Levin algorithm only utilizes i.i.d. pairs of correlated queries, where each half of every pair is uniformly random. We give a simple generalization of this algorithm in which pairs are replaced by k-tuples in which any $k-1$ components are jointly uniform. The cost of this generalization is that the number of queries needed grows exponentially with k.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques; Theory of computation → Machine learning theory

Keywords and phrases learning theory, adversarial machine learning, zero knowledge, Fourier analysis of boolean functions, Goldreich-Levin algorithm, Kushilevitz-Mansour algorithm

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.14

1 Introduction

In CRYPTO 2019, [9] formulated a new learning task whose utility is illustrated by the following scenario. Company A wishes to outsource experiments to company B while deterring employees of B from selling the outcomes of these experiments to a competitor C. Motivated by this scenario and others like it, they asked the following question (parameterized by a function family $\mathcal{F}$):

> *Can one learn*[1] *a function $f \in \mathcal{F}$ with oracle queries such that an adversary, who sees all queries x along with the corresponding values $f(x)$, is unable to learn f?*

When might this be possible? The adversary necessarily learns some non-trivial information about f, namely pairs of the form $\big(x, f(x)\big)$, called *examples* of f. We need it to be computationally intractable to use this information to "learn" f, e.g. output a circuit that agrees with f on all but a small fraction of inputs. There are simple function families for which learning from (uniformly) random examples is believed to be intractable. Such families include noisy linear functions over finite fields (see the learning parity with noise (LPN) [3]

[1] We are deliberately vague for now about the precise meaning of "learn"; we defer to Section 2 a detailed specification of the learning model used in our results.


© Justin Holmgren and Ruta Jawale;
licensed under Creative Commons License CC-BY 4.0

and learning with errors (LWE) [13] assumptions), $O(\log n)$-juntas [2], and more generally[2] polynomial-size decision trees, DNFs, CNFs, and Fourier-sparse functions. In each of these cases, we emphasize that the complexity of learning f depends on the joint distribution of the examples, and is only conjectured to be difficult for examples that are independent and uniformly random.

On the other hand, the learner has the power to *choose* the values of x for which it sees $f(x)$. This power, known in the learning theory literature as the ability to make *membership queries*, enables learning all of the aforementioned function families in polynomial time. However, this power is a double-edged sword in our setting; if not wielded carefully, it provides exactly the same benefits to the adversary! In fact, a learning algorithm must incorporate at least a one-way function in order to have any advantage over the adversary: if the learner's queries are not a (distributional) one-way function [8] of the learner's randomness, then the adversary could emulate the learner with arbitrary inverse polynomial accuracy. This also shows that we can only hope for security against a computationally bounded adversary.

Following Canetti and Karchmer [4], we focus on preserving any advantage of membership queries over random examples. They defined a learning algorithm to be *covert* if its transcript[3] with the membership query oracle is *simulatable* from random examples. As with all simulation definitions of security, there are variants (e.g. with computational, statistical, and perfect security) corresponding to analogous degrees of indistinguishability of the simulator's output from reality.

Previous Covert Learning Algorithms

There are two main previous results on covert learning, which we now quickly summarize.

1. Prior to [4], Ishai, Kushilevitz, Ostrovsky, and Sahai [9] gave a simple algorithm for learning $O(\log n)$-juntas with a relaxed notion of covertness that we retroactively call 1-out-of-2 covertness. In this relaxation, the learning algorithm's membership queries are distributed across *two* oracles, and we only require that the transcript with any *one* of the oracles is simulatable given random examples.
   The [9] algorithm is non-trivial in the sense that $O(\log n)$-juntas are not known to be learnable in polynomial time from random examples. However, this non-triviality is quantitatively rather weak: r-juntas are learnable in time $O(n^r)$.
   The notion of "1-out-of-2 covertness" naturally generalizes to "t-out-of-k covertness" for any $t \leq k$. The single-oracle setting considered by Canetti and Karchmer is the special case obtained by setting $t = k = 1$. We call this case *globally* covert learning, and we call the $t < k$ case *locally* covert learning.
   The motivating scenario with which we began this introduction is nearly as relevant to locally covert learning as it is to globally covert learning. Only a small modification is required: the company A now outsources its experiments to *multiple* companies $B_1, \ldots, B_k$, and aims to deter employees from t of these companies from colluding and selling their (combined) experiment outcomes to a competitor C.
2. Canetti and Karchmer [4] devised a *globally* covert algorithm for learning (polynomial-size) decision trees. In fact, their algorithm achieves stronger guarantees known as *agnostic learning* – even if the target function f is only α-close to a decision tree, their algorithm

[2] It is easy to prove that any k-junta has a decision tree of depth k and size 2^k, and also has at most 2^k non-zero Fourier coefficients.

[3] The transcript lists the queries made to the oracle alongside the corresponding oracle replies.

can still produce a circuit that is $(\alpha + \epsilon)$-close to f for arbitrarily small ϵ. Compared to [9], Canetti and Karchmer learn a larger family of functions with stronger (agnostic) correctness guarantees, and they achieve global rather than local covertness, but they rely on the sub-exponential LPN assumption and only achieve the computational variant of covertness.

Their main lemma regards a task that we call Goldreich-Levin learning (after the classic paper [5], which gave the first algorithm for the problem): using membership queries, produce a list of all *parity* functions whose correlation with f is above a given threshold $\tau > 0$. Goldreich-Levin learning is complemented by the Kushilevitz-Mansour algorithm [10], which shows how to learn a Fourier-sparse function f from only random examples if one is additionally given such a list of parity functions. Thus there is a quite general reduction from covert learning to covert Goldreich-Levin learning.

Canetti and Karchmer's algorithm actually achieves only a weak variant of Goldreich-Levin learning; it does not output *all* parity functions that are correlated with f, only those that depend on $O(\log n)$ variables. Nevertheless, they show that this variant is sufficient for learning decision trees. We note that decision trees are also learnable with only random examples if the learner is allowed to run in quasi-polynomial time (and use quasi-polynomially many examples).

1.1 Our Contributions

For any constant k, we construct a polynomial-time algorithm for Goldreich-Levin learning that is (perfectly) $(k-1)$-out-of-k covert. Combined with the Kushilevitz-Mansour algorithm, this immediately implies a polynomial-time algorithm for $(k-1)$-out-of-k covertly and agnostically learning Fourier-sparse functions under the uniform distribution. In contrast to previous works, this learning task is probably not achievable in quasi-polynomial time given only random examples. Assuming the sub-exponential hardness of LPN (with constant noise rate), it is even impossible to agnostically learn *parities*, which are maximally Fourier-sparse, in *sub-exponential* time given only random examples.

Unlike [4], our algorithm achieves the full functionality of the Goldreich-Levin algorithm (outputting *all* parities that correlate with the target function). In fact, for $k = 2$ our algorithm is nearly identical to the original algorithm! We merely specify which queries go to which oracle, add some dummy queries, and observe that the resulting algorithm is perfectly 1-out-of-2 covert. Our generalization to $k > 2$ is just a simple tweak to this algorithm, the crux of which is our Proposition 11.

We believe that our observation, though straightforward, is only obvious with hindsight. In particular, the original Fourier-analytic Goldreich-Levin algorithm that we crucially rely on appears to have been largely forgotten within the cryptography community. It has been supplanted in every cryptography curriculum known to the authors (and also in [4, 7]) by a more elementary algorithm that is attributed (see e.g. [1]) to Charles Rackoff. However, Rackoff's algorithm does *not* suffice for us. It is not locally covert with any reasonable parameters (i.e. t-out-of-k covert for $t = \Omega(k)$) for the simple reason that its queries do not have enough entropy. We believe that the modern language of Fourier analysis makes it easier to appreciate the elegance of the original proof. In the course of explaining our observations we give a succinct exposition of that proof (heavily inspired by O'Donnell [12]), which we hope will help in that regard.

Interactive Proofs for Verifying Machine Learning [7]

We also mention the work of [7], which focused on a different but related problem that they call interactive proofs for PAC verification. In this problem, the goal is to verify that a given hypothesis $\tilde{h}$ is as accurate as it is purported to be, making use of both random examples and interaction with an untrusted prover. At first glance this appears easy even without a prover – one can easily see how well $\tilde{h}$ agrees with the target concept f by testing $\tilde{h}$ on random examples for f. The key requirement that makes their problem technically challenging is that they focus on agnostic learning, where $\tilde{h}$ is supposed to perform as well as the *best* function h^* in a function family $\mathcal{F}$, and the verifier does not know h^* or how well h^* agrees with f.

Covert learning turns out to be applicable to this problem, as explored in both [4] and [7]. The rough idea is that the verifier runs a covert learning algorithm $\mathcal{L}$, and whenever $\mathcal{L}$ makes a query q, the verifier requests $f(q)$ from the prover. To prevent the prover from lying with impunity, the verifier also requests $f(x)$ for different values of x for which the verifier already knows $f(x)$ – such x are readily available in the form of random examples for f. Covertness is used to ensure that the prover cannot distinguish these "dummy" queries (on which incorrect answers would be caught) from the real queries (where incorrect answers would be impactful). Since the prover is now forced to answer queries mostly correctly, the verifier is assured that the resulting output of $\mathcal{L}$ is good.

The connection between covertness and interactive PAC verifiability remains intact for locally covert learning. Specifically, a 1-out-of-k covert learning algorithm gives rise to a standard *multi-prover* interactive proof (MIP), where each prover's messages are a function only of the messages sent to that prover. Starting instead with a t-out-of-k covert learning algorithm for $t > 1$ gives soundness against a form of bounded prover collusion that to our knowledge has not been previously studied in the context of MIPs.

2 Locally Covert Learning

In this section, we spell out the details of our learning model. We first formalize what an adversary (a coalition of oracles) is able to see when one of our learning algorithms is executed.

▶ **Definition 1** (The Adversarial View)**.** *For any k-oracle algorithm $\mathcal{L}$, an input x, and a function f, we define*

$$\mathsf{View}^f\big(\mathcal{L}, x\big) := (\mathcal{T}_1, \ldots, \mathcal{T}_k),$$

where $\mathcal{T}_1, \ldots, \mathcal{T}_k$ are correlated random variables that are sampled as follows. Let r be uniformly sampled randomness for $\mathcal{L}$. Suppose that on input x and randomness r, and with each oracle implementing f, the algorithm $\mathcal{L}$'s queries to its i^{th} oracle are $q_1^{(i)}, \ldots, q_{m_i}^{(i)}$. Then $\mathcal{T}_i$ (the transcript with the i^{th} oracle) is defined as

$$\mathcal{T}_i := \Big(\big(q_1^{(i)}, f(q_1^{(i)})\big), \ldots, \big(q_{m_i}^{(i)}, f(q_{m_i}^{(i)})\big)\Big).$$

For any subset $S = \{i_1, \ldots, i_t\} \subseteq [k]$ with $i_1 < \cdots < i_t$, we also define $\mathsf{View}_S^f(\mathcal{L}, x) := \big(\mathcal{T}_{i_1}, \ldots, \mathcal{T}_{i_t}\big)$.

We are now ready to define locally covert algorithms, following the standard real/ideal simulation paradigm that dates back to [6].

▶ **Definition 2** (Local Covertness). *We say that a k-oracle learning algorithm $\mathcal{L}$ is* (perfectly) t-out-of-k covert modulo $\ell(\cdot)$ *if there exists a probabilistic polynomial-time algorithm* Sim *such that for every subset $S \subseteq [k]$ with $|S| \leq t$, every function f, and every input x, it holds that*

$$\mathsf{View}^f_S(\mathcal{L}, x) \equiv \mathsf{Sim}^{\mathsf{Ex}(f)}\big(S, \ell(x)\big), \tag{1}$$

where $\equiv$ denotes equality of distributions and $\mathsf{Ex}(f)$ denotes a probabilistic oracle that when queried, samples x uniformly from the domain of f and returns $\big(x, f(x)\big)$.

If "$\equiv$" in Equation (1) *is replaced by a form of computational (resp. statistical) indistinguishability, we say that $\mathcal{L}$ is computationally (resp. statistically) t-out-of-k covert.*

In this definition, the function $\ell(\cdot)$ serves to enumerate all leakage about $\mathcal{L}$'s input that we consider benign. For example, the input x to a learning algorithm often includes an accuracy parameter ϵ and a confidence parameter δ. Relaxing the desired guarantees on accuracy or error probability means that the algorithm can make fewer queries. Fewer queries is usually a *good* thing, but it is also clearly visible to the adversary, so we declare it benign by defining $\ell(x)$ to include ϵ and δ.

On the other hand, for some learning algorithms the input may explicitly include more sensitive information, such as auxiliary information z about the target function f. When this information is omitted from $\ell(x)$, covertness modulo $\ell(\cdot)$ guarantees that the algorithm's queries do not convey information about z.

3 Fourier Analysis Preliminaries

It is mathematically convenient in this paper to view Boolean functions as functions from $\mathbb{F}_2^n$ to $\{-1, 1\}$ (as opposed to functions from $\{0,1\}^n$ to $\{0,1\}$). In this setting, a parity function is also known as a *character*.

▶ **Definition 3.** *A* character *of $\mathbb{F}_2^n$ is a homomorphism from $\mathbb{F}_2^n$ to the multiplicative group $\{-1, 1\} \subseteq \mathbb{R}$. Characters are parameterized by a vector $\gamma \in \mathbb{F}_2^n$, with the character corresponding to γ mapping $x \mapsto (-1)^{\gamma \cdot x}$.*

The characters form a group under multiplication, which corresponds to addition of the vectors γ in $\mathbb{F}_2^n$. We will identify characters with their index, i.e. we will simply write $\gamma(x)$ rather than $(-1)^{\gamma \cdot x}$. To avoid resulting confusion, we will denote the set of characters by $\widehat{\mathbb{F}_2^n}$ instead of $\mathbb{F}_2^n$.

▶ **Theorem 4** (Fourier Expansion Theorem [12, Theorem 1.1]). *Every function $f : \mathbb{F}_2^n \to \mathbb{R}$ can be uniquely expressed as*

$$f(x) = \sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \hat{f}(\gamma) \cdot \gamma(x)$$

where each $\hat{f}(\gamma)$ is a real number called the Fourier coefficient of f on γ. *This expression is called the* Fourier expansion *of f.*

▶ **Proposition 5** (Fourier Coefficient Formula [12, Proposition 1.8]). *For $f : \mathbb{F}_2^n \to \mathbb{R}$ and $\gamma \in \widehat{\mathbb{F}_2^n}$, the Fourier coefficient $\hat{f}(\gamma)$ is given by the formula*

$$\widehat{f}(\gamma) = \mathop{\mathbb{E}}_{x \leftarrow \mathbb{F}_2^n}[f(x) \cdot \gamma(x)].$$

One can view $\hat{f}$ as a function from $\mathbb{F}_2^n$ to $\mathbb{R}$ (because characters γ correspond to elements of $\mathbb{F}_2^n$ as described in Definition 3). Then combining Theorem 4 and Proposition 5, we obtain the standard fact that two iterations of the Fourier transform applied to a function returns the same function scaled by a factor of 2^{-n}.

▶ **Theorem 6** (Fourier Duality). *For any $f : \mathbb{F}_2^n \to \mathbb{R}$, $\hat{\hat{f}} = 2^{-n} \cdot f$.*

▶ **Theorem 7** (Parseval's theorem). *For any function $f : \mathbb{F}_2^n \to \mathbb{R}$, it holds that*

$$\mathop{\mathbb{E}}_{x \leftarrow \mathbb{F}_2^n}[f(x)^2] = \sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \widehat{f}(\gamma)^2.$$

Recall that for a subset $A \subseteq X$, the *indicator function* of A in X, denoted by 1_A, maps $x \in X$ to 1 if $x \in A$ and to 0 otherwise.

▶ **Proposition 8** (Fourier Transform of Affine Subspaces [12, Proposition 3.12]). *If $A = V + x$ is an affine subspace of $\mathbb{F}_2^n$ with codimension k, then*

$$\widehat{1_A}(\gamma) = \begin{cases} 2^{-k} \cdot \gamma(x) & \text{if } \gamma \in V^\perp \\ 0 & \text{otherwise.} \end{cases}$$

Here $1_A : \mathbb{F}_2^n \to \{0,1\}$ denotes the indicator function for A in $\mathbb{F}_2^n$, and $V^\perp$ denotes the set of $\gamma \in \widehat{\mathbb{F}_2^n}$ for which $\gamma \cdot v = 0$ (i.e. $\gamma(v) = 1$) for every $v \in V$.

In the statement of Proposition 8, note that although the decomposition $A = V + x$ is not unique (A can also be written $A = V + x'$ for any $x' \in x + V$), the definition of $\widehat{1_A}$ is independent of the choice of decomposition. This is because for any $x' \in x + V$ and any $\gamma \in V^\perp$, we have $\gamma(x) = \gamma(x' + (x - x')) = \gamma(x') \cdot \gamma(x - x') = \gamma(x')$.

▶ **Corollary 9.** *If $A = V + \gamma^*$ is an affine subspace of $\widehat{\mathbb{F}_2^n}$ with dimension d, then for any $x \in \mathbb{F}_2^n$,*

$$\sum_{\gamma \in A} \gamma(x) = \begin{cases} 2^d \cdot \gamma^*(x) & \text{if } x \in V^\perp \\ 0 & \text{otherwise.} \end{cases}$$

Equivalently, the function $f : \mathbb{F}_2^n \to \mathbb{R}$ defined by $\hat{f} = 1_A : \widehat{\mathbb{F}_2^n} \to \{0,1\}$ is

$$f(x) = \begin{cases} 2^d \cdot \gamma^*(x) & \text{if } x \in V^\perp \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Follows from applying Fourier duality (Theorem 6) to Proposition 8. ◀

▶ **Corollary 10** (Generalization of Plancherel). *For any $f_1, \ldots, f_k : \mathbb{F}_2^n \to \mathbb{R}$, it holds that*

$$\sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \prod_{j=1}^{k} \hat{f}_j(\gamma) = \mathop{\mathbb{E}}_{\substack{x_1, \ldots, x_k \in \mathbb{F}_2^n \\ x_1 + \cdots + x_k = 0}} \left[\prod_{j=1}^{k} f_j(x_j) \right]. \tag{2}$$

Proof. We have

$$\begin{aligned}
&\sum_{\gamma\in\widehat{\mathbb{F}_2^n}} \prod_{j=1}^{k} \hat{f}_j(\gamma) \\
&= \sum_{\gamma} \prod_{j=1}^{k} \mathop{\mathbb{E}}_{x\leftarrow\mathbb{F}_2^n}[f_j(x)\cdot\gamma(x)] && \text{(Proposition 5)} \\
&= \sum_{\gamma} \mathop{\mathbb{E}}_{x_1,\ldots,x_k\leftarrow\mathbb{F}_2^n}\Big[\prod_{j=1}^{k} f_j(x_j)\cdot\gamma(x_j)\Big] && \text{(independence of } x_1,\ldots,x_k) \\
&= \sum_{\gamma} \mathop{\mathbb{E}}_{x_1,\ldots,x_k\leftarrow\mathbb{F}_2^n}\Big[\Big(\prod_{j=1}^{k} f_j(x_j)\Big)\cdot\gamma\Big(\sum_{j=1}^{k} x_j\Big)\Big] && \text{(characters are homomorphisms)} \\
&= \mathop{\mathbb{E}}_{x_1,\ldots,x_k\leftarrow\mathbb{F}_2^n}\Big[f_1(x_1)\cdots f_k(x_k)\cdot\sum_{\gamma}\gamma\Big(\sum_{j=1}^{k} x_j\Big)\Big] && \text{(linearity of expectation)}
\end{aligned}$$

But this is equal to

$$\mathop{\mathbb{E}}_{\substack{x_1,\ldots,x_k\in\mathbb{F}_2^n \\ x_1+\cdots+x_k=0}}\Big[\prod_{j=1}^{k} f_j(x_j)\Big]$$

because

$$\sum_{\gamma}\gamma\Big(\sum_{j=1}^{k} x_j\Big) = \begin{cases} 2^n & \text{if } x_1+\cdots+x_k=0 \\ 0 & \text{otherwise.} \end{cases}$$

◀

4 Covertly Measuring Fourier Weight on Affine Spaces

The following algorithm is the main subroutine in our exposition of the Goldreich-Levin algorithm.

▶ **Proposition 11.** *For all $k\in\mathbb{Z}^+$, there is a perfectly $(k-1)$-out-of-k covert algorithm (modulo the parameters n, ϵ and δ below) that takes as input:*

- *an affine subspace $A = V+\gamma^*$ of $\widehat{\mathbb{F}_2^n}$, where V is a vector subspace;*
- *an "accuracy" parameter $\epsilon>0$;*
- *a "confidence" parameter $\delta>0$; and*
- *oracle access to a function $f:\mathbb{F}_2^n\to[-1,1]$,*

runs in time $\mathsf{poly}\big(n,\frac{1}{\epsilon},\log(\frac{1}{\delta})\big)$ and, with all but δ probability, outputs a real number $\tilde{w}$ satisfying

$$\tilde{w}-\epsilon \le \sum_{\gamma\in A}\hat{f}(\gamma)^k \le \tilde{w}+\epsilon.$$

Moreover, the queries of this algorithm are non-adaptive and are computable in time $\mathsf{poly}\big(n,\frac{1}{\epsilon},\log(\frac{1}{\delta})\big)$ given V, ϵ, δ, and the algorithm's randomness (in particular there is no dependence on γ^*).

Proof. Let $A = V + \gamma^*$ be an affine subspace of $\widehat{\mathbb{F}_2^n}$ (here V is the vector space parallel to A, and $\gamma^* \in \widehat{\mathbb{F}_2^n}$ is the offset of A), and let d denote the dimension of A. We can write

$$\sum_{\gamma \in A} \hat{f}(\gamma)^k = \sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \hat{f}(\gamma)^k \cdot 1_A(\gamma).$$

By Corollary 9, the indicator function $1_A : \widehat{\mathbb{F}_2^n} \to \{0,1\}$ is the Fourier transform of the function

$$g : \mathbb{F}_2^n \to \mathbb{R}$$
$$g(x) = \begin{cases} 2^d \cdot \gamma^*(x) & \text{if } x \in V^\perp \\ 0 & \text{otherwise.} \end{cases}$$

Rewriting 1_A as $\hat{g}$, we obtain

$$\begin{aligned} \sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \hat{f}(\gamma)^k \cdot \hat{g}(\gamma) &= \mathop{\mathbb{E}}_{x_1,\ldots,x_k \leftarrow \mathbb{F}_2^n} \left[f(x_1) \cdots f(x_k) \cdot g(x_1 + \cdots + x_k) \right] && \text{(Corollary 10)} \\ &= \mathop{\mathbb{E}}_{\substack{x_1,\ldots,x_k \\ x_1+\cdots+x_k \in V^\perp}} \left[f(x_1) \cdots f(x_k) \cdot \gamma^*(x_1 + \cdots + x_k) \right] && (3) \end{aligned}$$

Our algorithm estimates Equation (3) by sampling $m = O(\log(1/\delta)/\epsilon^2)$ i.i.d. tuples $\left((x_1^{(i)}, \ldots, x_k^{(i)})\right)_{i \in [m]}$ sampled uniformly from $\binom{n}{2}^k$ conditioned on $x_1^{(i)} + \cdots + x_k^{(i)} \in V^\perp$. Then for each $j \in [k]$, the algorithm queries its j^{th} oracle to obtain $f(x_j^{(i)})$, and computes a "sample" $f(x_1^{(i)}) \cdots f(x_k^{(i)}) \cdot \gamma^*(x_1^{(i)} + \cdots + x_k^{(i)})$. Finally, it outputs the average of these samples. This output satisfies the desired accuracy guarantee by Hoeffding's inequality. The "moreover" of the theorem statement follows from the fact that the values $\left(x_j^{(i)}\right)$ were sampled from a distribution that depends only on V.

$(k-1)$-out-of-k covertness follows from the standard fact that for *any* v, when the queries $(x_1^{(i)}, \ldots, x_k^{(i)})$ are sampled uniformly from $\mathbb{F}_2^n$ conditioned on $x_1^{(i)} + \cdots + x_k^{(i)} = v$, the distribution of $x_S^{(i)}$ is uniform on $(\mathbb{F}_2^n)^{|S|}$ for any $S \subseteq [k]$ with $|S| < k$. ◀

5 The Goldreich-Levin Theorem

The seminal theorem of Goldreich and Levin [5] plays an important role in cryptography, learning theory, and this paper. Loosely speaking, the theorem says that it is possible to efficiently find all heavy Fourier coefficients of a function f using membership queries to f. We prove a $(k-1)$-out-of-k covert variant of this theorem for any constant k (the running time and number of queries grow exponentially in k).

▶ **Theorem 12** (Locally Covert Goldreich-Levin). *For every integer $k \geq 2$, there is an explicit algorithm that when given:*

- *an integer $n \in \mathbb{Z}^+$,*
- *a "threshold" parameter $0 < \tau \leq 1$,*
- *a "confidence" parameter $\delta > 0$, and*
- *access to k oracles all implementing the same function $f : \mathbb{F}_2^n \to [-1,1]$,*

the algorithm runs in time $\mathsf{poly}(n, \log(\frac{1}{\delta}), \frac{1}{\tau^k})$ and outputs a set $S \subseteq \widehat{\mathbb{F}_2^n}$ (of size $O(1/\tau^k)$) such that with all but δ probability,

$$S \text{ contains all } \gamma \in \widehat{\mathbb{F}_2^n} \text{ for which } |\hat{f}(\gamma)| \geq \tau. \tag{4}$$

Moreover, this algorithm is perfectly $(k-1)$-out-of-k covert modulo (n, τ, δ).

Proof. We assume without loss of generality that k is even (so we are looking for γ satisfying $\hat{f}(\gamma)^k \geq \tau^k$). This is without loss of generality because for odd k, we can emulate a k-out-of-$(k+1)$ covert algorithm $\mathcal{L}$. Whenever $\mathcal{L}$ makes a query q to its i^{th} oracle, we pass the query to our $(i \bmod k)^{th}$ oracle. With this query mapping, it is easy to see that the view of any $k-1$ of our oracles is simulatable from the view of k of $\mathcal{L}$'s oracles, which is in turn simulatable from random examples.

For a given f and τ, say that $\gamma \in \widehat{\mathbb{F}_2^n}$ is *heavy* if $|\hat{f}(\gamma)| \geq \tau$, and let H denote the set of heavy γ. Our algorithm maintains a set S of $O(1/\tau^k)$ disjoint affine subspaces that collectively cover H (i.e. $H \subseteq \cup_{A \in S} A$). The algorithm starts with the trivial covering $S = \{\widehat{\mathbb{F}_2^n}\}$. It then "refines" S until S contains only affine sets of dimension 0, i.e. singleton sets, at which point S (or more precisely $\cup_{A \in S} A$) is the desired output.

To refine S, the algorithm repeats the following two steps n times:

1. Replace each $A \in S$ by disjoint A_0 and A_1 such that $\dim A_0 = \dim A_1 = \dim A - 1$ and $A = A_0 \cup A_1$.
2. Use Proposition 11 to "filter" S, keeping all A for which $\sum_{\gamma \in A} \hat{f}(\gamma)^k \geq \tau^k$ (in particular this includes all A that contain a heavy γ) and removing all A for which $\sum_{\gamma \in A} \hat{f}(\gamma)^k \leq \tau^k/2$. This involves running the algorithm of Proposition 11 $|S|$ times, where recall $|S| \leq O(1/\tau^k)$. For covertness, we we also perform up to $O(1/\tau^k)$ "dummy" executions of the algorithm so that we do not leak information through $|S|$.

We remark that with a careful choice of decomposition in step 1, this algorithm can be made non-adaptive. Specifically, fix a basis $e_1, \ldots, e_n$ for $\mathbb{F}_2^n$, and in the i^{th} iteration choose $A_b := \{\gamma \in A : \gamma(e_i) = (-1)^b\}$. Then in the i^{th} iteration at step 2, each affine space $A \in S$ will be parallel to the fixed vector space $V_i = \{\gamma \in \widehat{\mathbb{F}_2^n} : \gamma(e_1) = \cdots = \gamma(e_i) = 1\}$. The "moreover" of Proposition 11 then implies that the queries can all be made nonadaptively.

To complete the description and analysis of the algorithm, it remains to show that this filtration process guarantees $|S| \leq O(1/\tau^k)$. Specifically we claim $|S| \leq 2/\tau^k$, for otherwise we would have

$$
\begin{aligned}
1 &< |S| \cdot \frac{\tau^k}{2} \\
&\leq \sum_{A \in S} \sum_{\gamma \in A} \hat{f}(\gamma)^k \\
&\leq \sum_{A \in S} \sum_{\gamma \in A} \hat{f}(\gamma)^2 && \text{(each } \hat{f}(\gamma) \text{ lies in } [-1,1]) \\
&\leq \sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \hat{f}(\gamma)^2 && \text{(the spaces } A \in S \text{ are disjoint)} \\
&\leq 1 && \text{(Parseval).}
\end{aligned}
$$

◀

References

1 Mihir Bellare. The goldreich-levin theorem, October 1999. Lecture notes, available at `https://cseweb.ucsd.edu/~mihir/papers/gl.pdf`.

2 Avrim Blum. Learning a function of r relevant variables. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, volume 2777 of *Lecture Notes in Computer Science*, pages 731–733. Springer, 2003. `doi:10.1007/978-3-540-45167-9_54`.

3 Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

4 Ran Canetti and Ari Karchmer. Covert learning: How to learn with an untrusted intermediary. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography – 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part III*, volume 13044 of *Lecture Notes in Computer Science*, pages 1–31. Springer, 2021. doi:10.1007/978-3-030-90456-2_1.

5 Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 25–32. ACM, 1989. doi:10.1145/73007.73010.

6 Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377. ACM, 1982.

7 Shafi Goldwasser, Guy N. Rothblum, Jonathan Shafer, and Amir Yehudayoff. Interactive proofs for verifying machine learning. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 41:1–41:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITCS.2021.41.

8 Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *FOCS*, pages 230–235. IEEE Computer Society, 1989.

9 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptographic sensing. In *CRYPTO (3)*, volume 11694 of *Lecture Notes in Computer Science*, pages 583–604. Springer, 2019.

10 Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM J. Comput.*, 22(6):1331–1348, 1993. doi:10.1137/0222080.

11 Yishay Mansour. Learning boolean functions via the fourier transform. *Theoretical advances in neural computation and learning*, pages 391–424, 1994.

12 Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014. Available online at arXiv:2105.10386.

13 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93. ACM, 2005.

A Agnostic Learning from Heavy Fourier Coefficients

In agnostic learning, the goal is to learn a hypothesis $h \in \mathcal{H}$ that is nearly as good as any other hypothesis $\mathcal{H}$. More precisely, we aim to (approximately) minimize a *loss function* $L(h, f)$.

We use the squared ℓ_2 loss function, i.e. $L(h, f) = \|h - f\|^2 := \mathbb{E}_{x \leftarrow_2^n}\left[\left(h(x) - f(x)\right)^2\right]$. This loss function possesses a number of appealing mathematical properties. When $h, f :_2^n \to \{0, 1\}$ are both Boolean functions, $L(h, f)$ is just the fraction of inputs on which h and f differ. On the other hand if $\tilde{h}$ is a *real-valued* function with $L(\tilde{h}, f) = \epsilon$, then one can efficiently obtain a *Boolean* function h with $L(h, f) \leq 4\epsilon$ by setting $h(x) = 0$ if $\tilde{h}(x) \leq 1/2$, and $h(x) = 1$ otherwise.

This loss function has two appealing properties that connect it to Boolean functions. When $h, f :_2^n \to \{-1, 1\}$ are both Boolean functions, $L(h, f)$ is exactly four times the fraction of inputs on which h and f differ. On the other hand if $\tilde{h}$ is a *real-valued* function with $L(\tilde{h}, f) = \alpha$, then one can efficiently obtain a *Boolean* function h with $L(h, f) \leq 4\alpha$ by setting $h(x) = \mathsf{sign}(\tilde{h}(x))$, i.e.

$$h(x) = \begin{cases} 1 & \text{if } \tilde{h}(x) \geq 0 \\ -1 & \text{otherwise.} \end{cases}$$

For completeness we include a proof of the following proposition, which was *already known* (see e.g. the survey of Mansour [11]).

▶ **Proposition 13.** *For every integer $k \geq 2$, there is an explicit algorithm that when given:*

- *an integer $n \in \mathbb{Z}^+$,*
- *a "sparsity" parameter $t \in \mathbb{Z}^+$,*
- *an "accuracy" parameter $\epsilon > 0$,*
- *a "confidence" parameter $\delta > 0$, and*
- *access to k oracles all implementing the same function $f : \{-1,1\}^n \to [-1,1]$,*

runs in time $\mathsf{poly}\left(n, \log(\frac{1}{\delta}), \left(\frac{t}{\epsilon}\right)^k\right)$ and outputs a t-sparse polynomial $\tilde{p} : \mathbb{R}^n \to \mathbb{R}$ (as a list of monomials and coefficients) such that with all but δ probability, $L(\tilde{p}, f) \leq L(p, f) + \epsilon$ for every *t-sparse polynomial p.*

Moreover, this algorithm is perfectly $(k-1)$-out-of-k covert modulo (n, t, ϵ, δ).

Proof. The algorithm executes the following steps:

1. Use the (locally covert) Goldreich-Levin algorithm of Theorem 12 to find a set $\tilde{S} \subseteq \widehat{\mathbb{F}_2^n}$ such that, for some parameter τ to be specified later, $\tilde{S}$ contains all γ for which $|\hat{f}(\gamma)| \geq \tau$ (unless a certain bad event B_1 occurs, which happens with probability at most $\delta/2$). Without loss of generality assume $|\tilde{S}| \geq t$ (arbitrary elements can be added to $\tilde{S}$ if necessary to ensure this).
2. Use random examples to obtain estimates $\tilde{f}(\gamma)$ of $\hat{f}(\gamma)$ such that
$$\left|\tilde{f}(\gamma) - \hat{f}(\gamma)\right| \leq \sqrt{\tau} \text{ for each } \gamma \in \tilde{S} \tag{5}$$
(unless a certain bad event B_2 occurs, which happens with probability at most $\delta/2$).
3. Sort the elements of $\tilde{S}$ as $\tilde{\gamma}_1, \ldots, \tilde{\gamma}_{|\tilde{S}|}$ such that $|\tilde{f}(\tilde{\gamma}_1)| \geq |\tilde{f}(\tilde{\gamma}_2)| \geq \cdots \geq \left|\tilde{f}(\tilde{\gamma}_{|\tilde{S}|})\right|$, and define $\tilde{S}_t = \{\tilde{\gamma}_1, \ldots, \tilde{\gamma}_t\}$.
4. Output $\tilde{p} = \sum_{\gamma \in \tilde{S}_t} \tilde{f}(\gamma) \cdot \gamma$.

Toward analyzing the correctness of this algorithm, enumerate the elements of $\widehat{\mathbb{F}_2^n}$ as $\gamma_1, \gamma_2, \ldots, \gamma_{2^n}$ such that $|\hat{f}(\gamma_1)| \geq \cdots \geq \left|\hat{f}(\gamma_{2^n})\right|$, let S_t denote the set $\{\gamma_1, \ldots, \gamma_t\}$, and define the function $p = \sum_{\gamma \in S_t} \hat{f}(\gamma) \cdot \gamma$. Note that this p is the t-sparse function that is closest to f, so we wish to compare $L(\tilde{p}, f)$ to $L(p, f)$.

$L(p, f)$ has a convenient formula in the Fourier domain:

$$L(p, f) = \sum_{\gamma \in \widehat{\mathbb{F}_2^n}} \left(\hat{p}(\gamma) - \hat{f}(\gamma)\right)^2 = \sum_{\gamma \notin S_t} \hat{f}(\gamma)^2. \tag{6}$$

Similarly

$$L(\tilde{p}, f) = \sum_{\gamma \notin \tilde{S}_t} \hat{f}(\gamma)^2 + \sum_{\gamma \in \tilde{S}_t} \left(\hat{f}(\gamma) - \tilde{f}(\gamma)\right)^2, \tag{7}$$

so we have

$$L(\tilde{p}, f) - L(p, f) = \sum_{\gamma \notin \tilde{S}_t} \hat{f}(\gamma)^2 - \sum_{\gamma \notin S_t} \hat{f}(\gamma)^2 + \underbrace{\sum_{\gamma \in \tilde{S}_t} \left(\hat{f}(\gamma) - \tilde{f}(\gamma)\right)^2}_{\leq t\tau \text{ by } (5)}. \tag{8}$$

It remains to bound

$$\sum_{\gamma \in \tilde{S}_t} \hat{f}(\gamma)^2 - \sum_{\gamma \in S_t} \hat{f}(\gamma)^2 = \sum_{i=1}^{t} \left(\hat{f}(\gamma_i)^2 - \hat{f}(\tilde{\gamma}_i)^2\right). \tag{9}$$

Recall that $\hat{f}(\gamma_i)$ can be obtained by sorting the 2^n Fourier coefficients of f and picking the i^{th} largest. In comparison, $\hat{f}(\tilde{\gamma}_i)$ is obtained by first *perturbing* each Fourier coefficient by at most τ, then sorting and taking the i^{th} largest, then *unperturbing* by at most τ. Sorting of real numbers is 1-Lipschitz with respect to the ℓ_∞ metric (this follows from the analogous fact for min and max), which implies that $|\hat{f}(\gamma_i) - \hat{f}(\tilde{\gamma}_i)| \leq 2\tau$ for every i. Since each $|\hat{f}(\gamma)|$ is at most 1, (9) is at most $4t\tau$, which finally implies that (8) is at most $5t\tau$.

Setting $\tau = \epsilon/5t$ then achieves the desired accuracy. ◀

Online Mergers and Applications to Registration-Based Encryption and Accumulators

Mohammad Mahmoody
University of Virginia, Charlottesville, VA, USA

Wei Qi
University of Virginia, Charlottesville, VA, USA

Abstract

In this work we study a new information theoretic problem, called *online merging*, that has direct applications for constructing public-state accumulators and registration-based encryption schemes. An online merger receives the sequence of sets $\{1\}, \{2\}, \ldots$ in an online way, and right after receiving $\{i\}$, it can re-partition the elements $1, \ldots, i$ into $T_1, \ldots, T_{m_i}$ by merging some of these sets. The goal of the merger is to balance the trade-off between the maximum number of sets $\mathsf{wid} = \max_{i \in [n]} m_i$ that co-exist at any moment, called the *width* of the scheme, with its *depth* $\mathsf{dep} = \max_{i \in [n]} d_i$, where d_i is the number of times that the sets that contain i get merged. An online merger can be used to maintain a set of Merkle trees that occasionally get merged.

An online merger can be directly used to obtain public-state accumulators (using collision-resistant hashing) and registration-based encryptions (relying on more assumptions). Doing so, the width of an online merger translates into the size of the public-parameter of the constructed scheme, and the depth of the online algorithm corresponds to the number of times that parties need to update their "witness" (for accumulators) or their decryption key (for RBE).

In this work, we construct online mergers with $\mathsf{poly}(\log n)$ width and $O(\log n / \log\log n)$ depth, which can be shown to be optimal for all schemes with $\mathsf{poly}(\log n)$ width. More generally, we show how to achieve optimal depth for a given fixed width and to achieve a 2-approximate optimal width for a given depth d that can possibly grow as a function of n (e.g., $d = 2$ or $d = \log n / \log\log n$). As applications, we obtain accumulators with $O(\log n / \log\log n)$ number of updates for parties' witnesses (which can be shown to be optimal for accumulator digests of length $\mathsf{poly}(\log n)$) as well as registration based encryptions that again have an optimal $O(\log n / \log\log n)$ number of decryption updates, resolving the open question of Mahmoody, Rahimi, Qi [TCC'22] who proved that $\Omega(\log n / \log\log n)$ number of decryption updates are necessary for any RBE (with public parameter of length $\mathsf{poly}(\log n)$). More generally, for any given number of decryption updates $d = d(n)$ (under believable computational assumptions) our online merger implies RBE schemes with public parameters of length that is optimal, up to a constant factor that depends on the security parameter. For example, for any constant number of updates d, we get RBE schemes with public parameters of length $O(n^{1/(d+1)})$.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Registration-based encryption, Accumulators, Merkle Trees

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.15

Funding *Mohammad Mahmoody*: Supported by NSF grants CCF-1910681 and CNS193679.
Wei Qi: Supported by NSF grants CCF-1910681 and CNS193679.

1 Introduction

Registration-based encryption [12] is a primitive that aims to offer what identity-based encryption [26, 5] offers (i.e., a compact public parameter that can be used to encrypt for all identities) but without the key-escrow problem (i.e., that the holder of the master secret key can decrypt all the messages). It was shown [12] that essentially two relaxations will


© Mohammad Mahmoody and Wei Qi;
licensed under Creative Commons License CC-BY 4.0

enable such a primitive. In particular, in RBE the parties generate their own public *and secret* keys, and then they register them to a transparent algorithm called the key curator (KC). However, this comes at the cost of an evolving (compact) public parameter and need for occasional decryption updates. Namely, firstly the public parameter pp_n is now possibly changing after the nth identity registers into the system, and secondly the parties might sometimes need to reach out to the KC for hints/updates so that they can complete their decryption tasks. It was shown in [12] that with a total of $O(\log n)$ number of decryption updates, one can keep the length of the public parameter $\mathsf{poly}(\log n)$ (with a constant that depends on the security parameter). The follow-up works on RBE [13, 16, 9] made progress in various aspects such as assumptions and concrete efficiency, but asymptotically they all required the same $\Theta(\log n)$ number of decryption updates.

How many decryption updates are needed? The above state of affairs left open the possibility that RBE schemes with *sub-logarithmic* $o(\log n)$ number of updates could be constructed. Recently, Qi, Rahimi, and Mahmoody [22] proved that $\Omega(\log n/\log\log n)$ many updates are necessary for any RBE schemes (with a public parameter of size $\mathsf{poly}(\log n)$, as required by the standard definition of RBE) regardless of the computational assumptions used for constructing them, so long as the updates arrived at fixed times. The latter property is known to hold for all constructions of RBE so far. It remained open to close the gap between the upper bound of $O(\log n)$ and the lower bound of $\Omega(\log n/\log\log n)$.

In this work, we further close this gap and show that the lower bound of $\Omega(\log n/\log\log n)$ on the number of decryption updates is optimal (up to a constant factor that depends on the security parameter). We do so by improving the core information theoretic object that is at the center of the original RBE scheme of [12] as well as the accumulators used in such RBE schemes. More specifically, the RBE scheme of [12] relies on a *transparent accumulator* (i.e., one that does not have a secret state) [4, 3, 6, 25] that accumulates all the public-keys tied with their corresponding identities. In such an accumulator, a short digest pp of all the accumulated strings $\{x_1, \ldots, x_n\}$ can be used to efficiently verify membership of, say, x_i in the collection, so long as this verifier is provided with a short witness w_i of the membership. Hence, there are clear similarities between what RBE does with the keys and what an accumulator does with the strings x_i, and so it is not surprising that accumulators are useful for building RBEs. Our main result is to identify a core problem that is also at the heart of transparent accumulators that in turn are used for building RBEs.

The most natural approach for building a transparent accumulator is to use Merkle trees, based on collision resistant hash functions. Namely, one can build a Merkle tree T over the leaves $\mathcal{S} = \{x_1, \ldots, x_n\}$, and publish the label of the root r as the public parameter. Then, to prove membership of x_i in the collection $\mathcal{S}$, one can provide the "Merkle opening" of x_i, which consists of the labels along the path from (the leaf) x_i to the root r as well as the labels of the neighbors of this path. Then the verifier will do the basic sanity checks to pass the verification, and the scheme will be sound so long as the compressing hash function used for building the tree is collision resistant. When we are in the *dynamic* setting and the elements $\{x_1, \ldots, x_n\}$ arrive one by one, we no longer can use a single Merkle tree for hashing them, at least as long as we do not want to change the "opening witnesses" frequently.

The work of Reyzin and Yakoubov [25] showed how to make use of a *collection* of Merkle trees in such a way that the opening of each x_i will needs to be updated only $O(\log n)$ times over the course of the n steps of the system. They called an accumulator with this feature an *asynchronous accumulator*, and their construction was also used in the RBE construction of [12]. The idea behind this accumulator is to keep the collection of trees $\mathcal{T}$ in a way that: (1) any tree $T \in \mathcal{T}$ is always a full binary tree with 2^i leaves, (2) every pair of different trees

$T_1, T_2 \in \mathcal{T}$ have *different* sizes, (3) there is a bijection between the leaves of the trees in $\mathcal{T}$ and the accumulated set of objects. Therefore, the sizes of the trees in $\mathcal{T}$ directly give the *binary representation* of n, where n is the total number of elements in the collection. When a new element x_n arrives, this accumulator first generates a set $\{x_n\}$, and then the accumulator keeps merging the trees of the same size till there is no such pair of trees. It can be seen that the maximum depth of any tree $T \in \mathcal{T}$ will upper bound the number of times that an opening witness of an element $x \in \{x_1, \ldots, x_n\}$ needs to be updated, and this number is $\Theta(\log n)$. This is the core reason that this accumulator and the RBE scheme of [12] require logarithmically many updates.

1.1 Online mergers

In this work, we revisit the way a collection of sets/trees are merged to maintain a collection of "Merkle trees", but we study the problem more abstractly and independently of the direct connection to Merkle trees. We ask how to do the merging, while we try to balance the number of trees versus their maximum depth. More formally, we ask the following question. Suppose the elements $\{x_1, \ldots, x_n\}$ arrive one by one in n rounds, and suppose at the beginning of each round $i+1$ we have a collection of sets $T_1^i, \ldots, T_{m_i}^i$ already that partitions $\{x_1, \ldots, x_i\}$ and the new single-element set $T_{m_i+1}^i = \{x_{i+1}\}$ gets added to the current collection of sets. Then, the job of an *online merger* is to choose how to merge some of the sets and shape the updated collection $T_1^{i+1}, \ldots, T_{m_{i+1}}^{i+1}$ that is a partitioning of $T_1^i, \ldots, T_{m_i}^i, \{x_{i+1}\}$ (and also of $\{x_1, \ldots, x_{i+1}\}$). Since the merger has to decide about these choices in *each* of the n rounds, we call it an *online* merger.[1] The two key parameters of interest for online mergers are the following.

- The **width** of an online merger is the maximum number of sets (i.e., maximum of m_i over all $i \in [n]$) that it ever maintains during the course of its execution in the n rounds. This parameter is important as it captures the size of the digest/public parameter if each of the sets T is a Merkle tree, because it simply counts the number of roots of the trees at its maximum peak.
- The **depth** of an online merger is the maximum of d_i for $i \in [n]$, where d_i is the number of times that the set containing x_i gets merged with other sets. If we choose to represent the sets as trees and merge them as such, the depth of a scheme is simply the maximum depth of the collected trees at the end. If x_i has depth d_i, then (in case the sets shape Merkle trees) the opening witness for the membership of x_i in the collection needs to be updated d_i times.

Key questions about online mergers. On the extreme points, one can achieve width 1 and depth $n-1$ by immediately merging any incoming set, and one can achieve depth 0 and width n by not merging any of the sets. Thus, the interesting question is to find the optimal *trade-off* between the width and depth of online mergers as a function of n. We can ask this question both for online mergers that know the set size n ahead of the time (called *bounded* online mergers) and for those that are "unbounded" and receive the incoming single-element sets without knowing the upper bound n on the final set size (called *unbounded* online mergers). More specifically, we are interested in finding the minimum width needed for mergers that are given an upper bound on their depth, and conversely we would like to find out the minimum depth needed for mergers that are given an upper bound on their

[1] In contrast, an offline merger gets all of $\{x_1, \ldots, x_n\}$ before deciding on how to partition them.

width. The given upper bounds (for width or depth) could be absolute constants or growing functions of n. On the one hand, both width and depth can be $\Theta(\log n)$ simultaneously, due to the accumulator of [25]. On the other hand, as we will show, the tools from [22] can be used to show that the depth of any online merger needs to be at least $\Omega(\log n/\log\log n)$ if the width needs to be $\mathsf{poly}(\log n)$ (which is the standard size of the public parameter for RBE [12]). Prior to our work, it was not known how many updates are necessary for achieving public parameters of length $\mathsf{poly}(\log n)$.

1.2 Our results

Our main result is to find the exact trade-off between the two key parameters (width and depth) of online mergers. As a corollary, we obtain transparent accumulators and RBE schemes that can take as input parameters the number of updates and produce public parameters that are optimal within a constant factor. As a special case, we also obtain accumulators and RBE schemes that have the optimal $O(\log n/\log\log n)$ number of witness/decryption updates, while their public parameter is assumed to be of length $O(\mathsf{poly}(\log n))$. Below, we explain these results in more detail.

Our results about online mergers. To have a reference to judge the optimality of a trade off between depth and width of an online merger, we start by proving a lower bound on this trade off. In particular, using a key combinatorial tool from [22], we first derive a lower bound on the trade-off between the depth d and width w of any online merger, and the lower bound holds even if the set of elements $[n] = \{1,\ldots,n\}$[2] is known to the online merger ahead of the time. In particular, we find lower bound functions $\mathsf{widthLB}(n,d)$ (for the with) and $\mathsf{depthLB}(n,w)$ (for the depth) when we are given the set size n and either of the width w or the depth d as inputs.

We remark that the mere fact that one can use the tools of [22] to obtain a lower bound for the trade off between depth and width of online mergers is not surprising, as online mergers are *one way* to obtain accumulators and RBEs and the lower bound of [22] applies to any RBE scheme. However, we emphasize that as our starting point in this work we obtain concrete lower bound functions $\mathsf{widthLB}(n,d)$ and $\mathsf{depthLB}(n,w)$ that do *not* hide any unknown constants that can depend, say, on the security parameter, as online mergers are a purely information theoretic object without security parameters. Hence, these lower bound functions allow us to prove *exact* bounds on the trade-off between depth and width of online mergers, which is what we do next. Having the reference lower bounds $\mathsf{widthLB}(n,d)$ and $\mathsf{depthLB}(n,w)$ we show how to achieve positive constructions that (sometimes approximately) match these lower bounds, as stated below.

▶ **Theorem 1** (Optimal bounded online mergers – informally stated)**.** *For any known set size n, there is an efficient construction of online merger MerWid_w of width w that achieves optimal depth $\mathsf{depthLB}(n,w)$, and there is an efficient construction of an online merger MerDep_d of depth d that achieves optimal width $\mathsf{widthLB}(n,d)$.*

See Theorem 13 and Proposition 14 for formalization of the above theorem.

We remark that the appendix in [22] included a graph construction that showed their lower-bound cannot be further improved. However, that graph construction only shows the limitation of the proof *approach* of [22] and does not uniquely determine a positive

[2] Note that even though we will use larger blocks of data in applications of online mergers (i.e., accumulators and RBE schemes) for simplicity we can pretend that the arriving sets are $\{1\},\ldots,\{n\}$.

construction. In fact, any positive construction (e.g., that of our Theorem 1) can be used to obtain such graphs, showing that the approach of [22] cannot lead to better lower bounds, but the reverse is not true.

▶ **Theorem 2** (Approximately optimal unbounded online mergers – informally stated)**.** *If we do* not *know n ahead of the time, and if $d(n)$ is a non-decreasing function of n (e.g., $d(n) = \log n / \log\log n$) that upper bounds the depth of the online merger that we would like to have for a set of n elements, then there is an unbounded online merger that achieves width at most $2 \cdot \mathsf{widthLB}(n, d(n))$.*

See Proposition 19 for a formal statement.

The cost of being unbounded. Theorem 2 only achieves a solution whose width is within 2 multiplicative approximation of the optimal solution. Hence, it brings up the question of whether the approximation factor 2 (or any other factor bigger than 1) is needed here. We prove that this is indeed the case, so long as we aim for *unbounded* online mergers. Namely, in Theorem 22 we show that for unbounded online mergers such overhead is necessary.

Implications to accumulators and RBEs. Theorems 1 and 2 can be directly used to construct accumulators and RBEs whose number of witness/decryption updates are bounded by the *depth* of the corresponding online merger, and whose public parameters are of the size $O_\kappa(w)$, where w is the width of the online merger and the constant in $O_\kappa(w)$ could depend on the security parameter κ of the accumulator/RBE scheme. In fact, Theorem 1 already suffices for obtaining accumulators and RBEs with *optimal* number of updates when we already know the final size of the set of elements/identities that will join the system over time. We then study *unbounded* online mergers who do *not* know the upper bound n on the population size, and obtain an almost tight solution.

The idea is quite straightforward: an online merger can be used to maintain a set of Merkle trees $\mathcal{T} = \{T_1, \ldots, T_m\}$ that would serve as an accumulator for the incoming objects $\{x_1, \ldots, x_n\}$, while the set of the *roots* of the trees $r_1, \ldots, r_m$ would serve as the digest/public parameter. To prove membership of x_i in the set, one has to prove the Merkle opening of x_i with respect to the tree $T \in \mathcal{T}$ that contains x_i as a leaf. Then, so long as the hash function used for constructing the Merkle trees in $\mathcal{T}$ is collision resistant, it would be computationally hard to prove membership of any $x \notin \{x_1, \ldots, x_n\}$ successfully. See Construction 44 for a formal description of this reduction. As formally stated in Proposition 45 the width and depth of the used online merger directly translate (in order) into the number of updates and length of the public parameter of the constructed accumulator. Finally, we observe that this construction of (transparent) accumulators is *tight* in its trade-off between the number of updates vs. the length of public parameter (up to a constant factor that can depend on the security parameter). The reason is that the same proof of the lower bound of [22] for RBEs can be directly adapted to transparent accumulators as well.

Finally, due to the fact that RBE schemes heavily rely on an internal accumulators for compressing the submitted public keys, using our optimal accumulator and extra assumptions (i.e., indistinguishability obfuscation [2, 11, 21] and somewhere-statistically binding hashing [20]) we can adapt the original construction of [12] to obtain an RBE scheme with an optimal $\log n / \log\log n$ number of updates, while all the other efficiency and compactness requirements of the scheme are as defined and required by [12].

▶ **Theorem 3** (RBE with optimal number of decryption updates – informal)**.** *Assuming standard computational hardness assumptions, there is an RBE scheme that has* $\log n / \log \log n$ *number of decryption updates.*

See Construction 28 for a formal adaptation of the construction of [12] to using an arbitrary online merger. The proof of security for this construction is identical to the (rather long) proof of the similar scheme in [12]. In fact, we obtain a more general result: using *any* given number of updates $d(n)$ as input parameter that can depend on n (e.g., $d(n) = \log(n)^{1/2}$) we can use our optimal online merger for that depth function $d(n)$ to obtain RBE schemes with *optimal* length for the public parameter for the given number of updates $d(n)$. An interesting corollary is that even using *just one update* allows us to obtain a scheme with a *sublinear* public parameter of length $O(\sqrt{n})$ and using a larger constant d will lead to significantly smaller public parameters of length $O(n^{1/(d+1)})$.

Other assumptions. We emphasize that our Theorem 3 above is merely to show that one can obtain the right number of updates for RBEs using *some* computational assumptions, and to show that we choose the simplest construction that is based on Indistinguishability obfuscation and Somewhere Statistically Binding Hashing [20]. However, as we explain in the full version of the paper, our main tool of this paper, namely online mergers, are versatile enough to be incorporated into other constructions of RBE based on standard assumptions [14, 16] as well.

1.3 Techniques

We now review some of the ideas used in the proof of our main results about online mergers.

Lower bound. To obtain the lower bound functions $\mathsf{widthLB}(n, d)$ (for the width) and $\mathsf{depthLB}(n, w)$ (for the depth) we take the following steps.

1. We first show how to derive a DAG of out-degree at most d from any merger of depth d, by connecting i to j if the set containing i gets merged in round j (see Definition 8).
2. We then use a result from [22] showing that DAGs with small out-degrees have a substructure, called *skipping sequence* (see Definition 33 and Theorem 34).
3. Finally, we observe that skipping sequences directly imply lower bounds on the width.

Optimal depth for a given constant width. Our starting point is an extremely simple scheme that obtains *optimal depth* (i.e., matching $\mathsf{depthLB}(n, w)$) when we are given any *fixed* width w as input. This scheme works *even if we do not know the set size* n in advance. The scheme can be described in one sentence: when a new set $T = \{n\}$ (as a single-node tree) is added to the collection of trees $\mathcal{T}$, if $\mathcal{T}$ has $w + 1$ trees, merge the newly arrived tree T with all the trees in $\mathcal{T}$ that have *minimum* depth (among the trees already in $\mathcal{T}$). Equivalently, as long as $|\mathcal{T}| = w + 1$, keep merging all the trees of minimum depth in $\mathcal{T}$. See Construction 11 and Theorem 13 for more details. One can interpret this scheme as "lazy merging" approach that does not do any merges when it does not have to, but when the merge is needed it only merges trees of minimal depth (with the incoming tree of depth 0). Interestingly, this simple scheme achieves a depth that is optimal *for every* n, as careful calculations can be used to show that its depth matches the lower bound $\mathsf{depthLB}(n, w)$ exactly.

Optimal width for a given depth and known set sizes. Once we have an online merger MerWid_w that achieves optimal depth for any given set size n and width w, we can turn this scheme around and switch the role of depth and width. Namely, for a given set size n and depth d, the new online merger MerDep_d can find the smallest w such that a construction on set $[n]$ with width w will have a depth at most d, and this would be an optimal choice of the width due to the optimality of the original algorithm MerWid_w. (See Proposition 14 for a formalization.) This finishes the proof of Theorem 1. Note that we lost something in this transformation: although our "width-based" merger MerWid_w did not need to know the set size n (and hence it was unbounded), the new "depth-based" merger MerDep_d needs to know n ahead of the time to find the optimal choice of width w. So, the new online merger is *not* unbounded anymore.

2-approximation for the width of unbounded online mergers. We finally describe how to achieve our online merger of Theorem 2. The key idea is to use our "width-based" merger MerWid_c for specifically chosen values of $n(c)$, by pretending that $n(c)$ is an upper bound on the total size of the streaming set. We will also increase c whenever $d(n)$ jumps by at least one (as it can gradually grow). Our careful choice of $n(c)$ and a "non-black-box" use of the analysis of our scheme MerDep_d allows us to prove that the resulting scheme will never use a width more than $2 \cdot \mathsf{widthLB}(n, d(n))$.

1.4 Related work

In addition to RBE, other works have also pursued paths to eliminate the key escrow problem from IBE. The work of [5] aimed to make the private-key generator decentralized. The works of [17, 18] proposed a notion of "accountablity" for PKG, by proposing how to catch an irresponsible PKG in case of breach. The works of [7, 8, 28] aimed to make it harder for the PKG to find out the receiver's identity. The works of [8, 10] studied interactive key generation that allows hiding user's identities. The work of [1] proposed to mix IBE and public-key encryption by constructing "Certificateless" Public Key Cryptography.

More recently, the work of [27] showed how to make RBE even more transparent by deploying it on blockchain. The two works of [15, 19] showed how to achieve *black-box* constructions of RBE based on assumptions on bilinear maps, while leveraging a polynomially large CRS that can grow with the number of parties. Further more, the work of [15] gave concrete implementations of RBE, and the work of [19] further generalized the notion of RBE to registered *attribute-based* encryption schemes that can handle attributes beyond identity (which is what RBE does).

The concept of cryptographic accumulators was first introduced by Benaloh and de Mare [4]. Using an accumulator, one can represent a set of values $\mathcal{S}$ by a short digest such that (1) there is a witness to prove membership for values in $\mathcal{S}$ and (2) it is infeasible to find such witness for values that are not in $\mathcal{S}$. Later, Barić and Pfitzmann [3] gave a more generalized definition called collision-resistant accumulators and a construction of such accumulators based on the strong RSA assumption. Both of the accumulators are static in the sense that the set of values $\mathcal{S}$ never changes after the digest has been generated. However, for many applications, the set of values can evolve with time. Observing this, Camenish and Lysyanskaya [6] introduced the concept of dynamic accumulators and provided a construction based on the strong RSA assumption. In a dynamic accumulator scheme, the set of values $\mathcal{S}$ can change. Namely, values can be added to or removed from the set. Therefore, both the digest and witnesses might be updated from time to time.

2 Online mergers and partitioners: constructions and lower bounds

In this section, we introduce the key information-theoretic problem of our work. To begin, we need to define merging operation on rooted trees.

▶ **Definition 4** (Merging operation and tree partitioning). *A rooted tree is either a single vertex* rt *(called root), or it has a root* rt *with* $i \geq 1$ *children* $u_1, \ldots, u_i$ *such that each* u_i *is the root of a rooted tree itself. A leaf is any vertex that has no children. (So a tree with a single vertex has a root that is also a leaf.) A merge operation takes a sequence* $(\mathsf{T}_1, \ldots, \mathsf{T}_j)$ *of* $j \geq 1$ *rooted trees and returns a single* T*, where* T *has a root* rt *with* j *children and the* i*th sub-tree of* rt *for* $i \in [j]$ *is* T_i*. For* $k \leq m$*, we say that a set of trees* $\{\mathsf{T}'_1, \ldots, \mathsf{T}'_k\}$ *can be obtained by a* single merge *operation from* $(\mathsf{T}_1, \ldots, \mathsf{T}_m)$*, if there is a subset* $\{\mathsf{T}''_1, \ldots, \mathsf{T}''_\ell\} \subseteq \{\mathsf{T}_1, \ldots, \mathsf{T}_m\}$ *that is merged into* T*, and* $\{\mathsf{T}'_1, \ldots, \mathsf{T}'_k\} = \{\mathsf{T}\} \cup \{\mathsf{T}_1, \ldots, \mathsf{T}_m\} \setminus \{\mathsf{T}''_1, \ldots, \mathsf{T}''_\ell\}$*. We also say that a set of trees* $\{\mathsf{T}'_1, \ldots, \mathsf{T}'_k\}$ *is a* merging *of* $\{\mathsf{T}_1, \ldots, \mathsf{T}_m\}$*, if one can obtain* $\{\mathsf{T}'_1, \ldots, \mathsf{T}'_m\}$ *from* $\{\mathsf{T}_1, \ldots, \mathsf{T}_m\}$ *by a series of (zero or more) single merge operations. A set of rooted trees* $\{\mathsf{T}_1, \ldots, \mathsf{T}_m\}$ *form a* tree partitioning *of* $\mathcal{S}$*, if they have* $|\mathcal{S}|$ *many leaves, and that each* $x \in \mathcal{S}$ *appears as a leaf in exactly one of* $\{\mathsf{T}_1, \ldots, \mathsf{T}_m\}$*.*

It is easy to see that if $\{\mathsf{T}_1, \ldots, \mathsf{T}_m\}$ form a tree partitioning of $\mathcal{S}$, then any merging of $\{\mathsf{T}_1, \ldots, \mathsf{T}_m\}$ will also be a tree partitioning of $\mathcal{S}$.

We now define the main object of our interest, namely an online merger on a set $[n]$.

▶ **Definition 5** (Online merger). *An* online merger M *for* $[n]$ *is a deterministic algorithm that works in* n *rounds as follows. Originally we have an empty set* $\mathcal{T}_0 = \emptyset$ *of trees. In round* i*, we start from* $\mathcal{T}_{i-1} \cup \{i\}$*; namely, a new tree with a single node labeled* i *gets added. Then, the algorithm* M *is allowed to apply any number of merge operations on* $\mathcal{T}_{i-1} \cup \{i\}$ *to reach* $\mathcal{T}_i$ *(at the end of round* i*). We call* $\mathsf{wid}_i = |\mathcal{T}_i|$ *the* width *at (the end of) round* i *and* $\mathsf{wid}_{[n]} = \max_{i \in [n]} |\mathcal{T}_i|$ *simply the width of* M *for* $[n]$*. At round* i *and* $j \in [i]$*, the* depth *of node* j *in round* i*, denoted by* dep_i^j *is the distance of the node* j *from the root in its rooted tree at the end of round* i*.*[3] *The depth of* $\mathcal{T}_i$ *is simply defined as* $\mathsf{dep}_i = \max_{j \in [i]} \mathsf{dep}_i^j$*, and the depth of* M *for* $[n]$ *is* $\mathsf{dep}_{[n]} = \max_{i \in [n]} \mathsf{dep}_i$*. An online merger for* all *the natural numbers* $\mathbb{N}$*, also called an* unbounded *online merger, informally, is one that keeps going forever. More specifically, an unbounded merger gets the inputs* $\{1\}, \{2\}, \ldots$*, and always maintains a tree partitioning over* $[n]$ *for all* $n \in \mathbb{N}$*. For an unbounded merger, we can still define the depths* $d_i, d_{[n]}$ *and widths* $\mathsf{wid}_i, \mathsf{wid}_{[n]}$ *of such mergers for all* $i, n \in \mathbb{N}$*. Any online merger* M *for* $[n]$ *(resp.* $\mathbb{N}$*) defines an online merger for all* $m \in [n]$ *(resp.* $[n], n \in \mathbb{N}$*).*

Extensions and generalizations. The definition above can be extended in multiple ways:

- **Randomness.** We could allow online mergers to be randomized algorithms, in which case the depth and width are defined with respect to a fixed randomness. However, in this work, we construct deterministic mergers, as our focus is on upper bounds, while our lower bounds also directly apply to randomized algorithms as well.
- **Other sets.** For simplicity, we defined mergers on $[n]$, while one can think of mergers who deal with *arbitrary* sets $\mathcal{S}$ of size n.
- **Partitioning.** One can generalize the notion of online mergers to algorithms that arbitrarily change the partitioning of the current set of elements. This class of algorithms are, e.g., useful to model algorithms that maintain a set of Merkle trees, but decide to break down those trees every now and then and reconstruct them from scratch.

[3] This is equal to the number of times that the trees containing j are merged, since $\{j\}$ was added and till the end of round i.

▶ **Definition 6** (Online partitioner). *An* online partitioner Part *for* $[n]$ *and an* unbounded *online partitioner for* $\mathbb{N}$ *are both defined similarly to their corresponding mergers with the difference that they can arbitrarily* re-partition *the current set of vertices rather than merely merging them. The notions of depth and width are defined similarly for partitioners, with the only difference that depth of a node* $j \in [n]$ *will denote the number of times that the set containing* j *has* changed.

Below, we first study lower bounds on depths and widths of bounded mergers. We then give *unbounded* online schemes that (closely) match these lower bounds.

2.1 Lower bounds for bounded partitioners

The goal of this subsection is to present functions $\mathsf{depthLB}(n, w)$ (resp. $\mathsf{widthLB}(n, d)$) that serve as lower bounds on the depth (resp. width) of any online bounded partitioner, assuming that the set size is n and the width is w (resp. depth is d).

We start by defining a DAG for any online partitioner or merger. Since partitioners generalize mergers, we only define these notions for partitioners.

▶ **Definition 7** (Forward DAGs of online partitioners and mergers). *Let* Part *be a deterministic online partitioner for* $[n]$*. Then,* Part *defines the following DAG* G_n *over* $[n]$*:* $(i, j) \in G$ *for* $i \le j$ *if the set* $\mathcal{S}$ *that contains* i *at the* beginning *of round* j *is different from the one that contains* i *at the end of the round* j*. For the special case of mergers,* $(i, j) \in G$ *means that the tree containing* i *at the beginning of round* j *is merged during round* j*.*

We now observe that skipping sequences in the DAGs of online partitioners imply a lower bound on their width.

▶ **Proposition 8** (Lower bound on the width from skipping sequences). *Let* Part *be an online partitioner for* $[n]$*. Let* G_n *be the forward DAG over* $[n]$ *defined by* Part*. Let* $\mathcal{S}$ *be a skipping sequence in* G_n*. Then, the width of* Part *is at least* $|\mathcal{S}|$*.*

Proof. Let $\mathcal{S} = \{s_1 < \cdots < s_\ell\}$. It suffices to show that s_i and s_j belong to different sets at the end of round ℓ for all $1 \le i < j \le \ell$. Assume, on the contrary, that s_i and s_j belong to the same set at the end of round ℓ. This means that at some time k such that $s_j \le k \le s_\ell$ the sets containing s_i, s_j change to include both of them, which implies the existence of the edge $(s_i, k) \in G_n$, where $s_i < s_j \le k \le s_\ell$. Such an edge, however, contradicts the definition of skipping sequences. ◀

If the depth of a partitioner or a merger is at most d, then by definition the out-degree of the vertices in the corresponding forward DAG G_n is at most d. Therefore, using Proposition 8 and Theorem 34 (proved in [22]) we obtain the following lower bound.

▶ **Theorem 9** (Lower bound on the width based on the depth). *Suppose* Part *is an online partitioner for* $[n]$*. Then the following hold for all* $d \ge 0, w \ge 1$*.*

1. *If* $n \ge \binom{w+d}{d+1}$ *and* $\mathsf{dep}_{[n]} \le d$*, then* $\mathsf{wid}_{[n]} \ge w$*.*
2. *If* $n \ge \binom{w+d}{d}$ *and* $\mathsf{wid}_{[n]} \le w$*, then* $\mathsf{dep}_{[n]} \ge d$*.*

Proof. The first part follows directly from Proposition 8 and Theorem 34. To prove the second part, suppose $\mathsf{wid}_{[n]} \le w$ and $\mathsf{dep}_{[n]} \le d-1$. Then, by applying the first part on depth $d-1$ (rather than d), we obtain that the depth should be at least $w+1$, which contradicts $\mathsf{wid}_{[n]} \le w$. ◀

We now present the depthLB and widthLB functions that serve as lower bounds on the depth and width of any online partitioner. Both of them can be expressed through the same function in the following corollary that follows directly from Theorem 9 above.

▶ **Corollary 10** (Lower-bound functions depthLB, widthLB for depth and width)**.** *For natural numbers x, y, let* $\mathsf{minBin}(x,y) = \min\{z \in \mathbb{N}^+ \mid \binom{y+z}{y} > x\}$*. Then, the following holds for any partitioner* Part *over the set* $[n]$*.*

1. *If* $\mathsf{wid}_{[n]} \leq w$*, then* $\mathsf{dep}_{[n]} \geq \mathsf{depthLB}(n,w)$ *for* $\mathsf{depthLB}(n,w) = \mathsf{minBin}(n,w) - 1$*.*
2. *If* $\mathsf{dep}_{[n]} \leq d$*, then* $\mathsf{wid}_{[n]} \geq \mathsf{widthLB}(n,d)$ *for* $\mathsf{widthLB}(n,d) = \mathsf{minBin}(n,d+1)$*.*

2.2 Optimal bounded mergers

In this subsection, we focus on bounded online mergers that know the set-size n in advance and aims to optimally balance the trade-off between the width and the depth. For this setting, we show that a simple construction optimally matches the bounds of Corollary 10. In fact, we show that the following extremely simple construction achieves optimal depth for *all* set sizes, so long as the width is upper bounded by a fixed given amount. In other words, our simple construction is even an *unbounded* online merger for any given bound on the width. As a corollary, when the set-size n and a given *depth* are both known in advance, one can use our simple construction using the width $w = \mathsf{widthLB}(n,d)$ and achieve optimal width for the given depth.

▶ **Construction 11** (Optimal unbounded online merger for fixed width)**.** *The online merger* MerWid_w *is parameterized by a positive integer w. At each round, while the total number of trees equals $w+1$,* MerWid_w *merges the subset of trees consisting of all trees with the minimum depth. (Note that if there is a unique tree of minimum depth, merging it with itself will simply increase its depth by one.)*

By definition the width (i.e., $\mathsf{wid}_i = |\mathcal{T}_i|$, where $\mathcal{T}_i$ is the collection of trees at the end of every round $i \in \mathbb{N}$) in Construction 11 it holds that $\mathsf{wid}_n \leq w$, and hence $\mathsf{wid}_{[n]} \leq w$ for all $n \in \mathbb{N}$. We now analyze the depth.

Alternative construction. Construction 11 sometimes merges a single tree, which simply increases its depth. This artificial merging can be avoided without increasing the depth or width. However, the redundant operations simplify some of the claims below about the analysis of the depth.

▶ **Lemma 12.** *In Construction 11, the following holds.*

1. *At the end of round* $\binom{d+w}{d} - 1$ *there are w trees of depth $d-1$.*
2. *Round* $\binom{d+w}{d}$ *is the first round, at the end of which there is a single tree of depth d.*

Proof. We use induction on d. The base case where $d = 1$ is true by inspection of the first $w+1$ rounds. Assume the claim is true for $d-1$. We show that it is true for d. By assumption we know at round $\binom{d-1+w}{d-1}$, there is exactly one tree of depth d. Reusing the assumption, we know at round $\binom{d+w-1}{d-1} + \binom{d+w-2}{d-1}$ there are 2 trees of depth d and no trees of other depth. Similarly, we know at round $\sum_{i=1}^{w} \binom{d+w-i}{d-1} = \binom{d+w}{d} - 1$ there are w trees of depth d and no trees of other depth. By inspection, we know at round $\binom{d+w}{d}$ there is exactly one tree of depth d and this is the first round where there is a tree of depth d. ◀

Using the lemma above, we can get an upper bound on the depth for Construction 11.

▶ **Theorem 13** (Construction 11 achieves optimal depth for all set sizes and widths)**.** *In Construction 11, the depth* $\mathsf{dep}_{[n]}$ *of* MerWid_w *for set size n is* $\mathsf{depthLB}(n,w) = \mathsf{minBin}(n,w) - 1$*, which is optimal.*

Proof. Using Lemma 12, we know for $n \in [\binom{d+w}{d}, \binom{d+w+1}{d+1} - 1]$, the depth of M is exactly d. For such choice of n, d, it holds that $\binom{w+d+1}{w} > n$ and $\binom{w+d}{w} \leq n$. Therefore, by definition, this means that $d+1 = \mathsf{minBin}(n, w)$, and hence $d = \mathsf{minBin}(n, w) - 1$. ◀

▶ **Proposition 14** (Achieving optimal width, given set size and depth)**.** *Let n, d be two positive integers. Using MerWid_w in Construction 11 with width $w = \mathsf{widthLB}(n, d) = \mathsf{minBin}(n, d+1)$ will have depth at most d. Consequently, the scheme will use optimal width.*

Proof. Using Lemma 12, we know round $\binom{d+1+w}{d+1}$ is the first round where there is a tree of depth $d+1$. However, by construction we know $\binom{d+1+w}{d+1} > n$. Thus, the depth will be at most d. ◀

2.3 Unbounded online mergers

In this section, we study unbounded online mergers that do *not* know the set size $[n]$ in advance. We first design unbounded mergers for a given constant depth and then will generalize our construction to the setting in which d is a growing function of n (e.g., $d = \log n$).

2.3.1 Unbounded mergers for a given fixed depth

The key idea of our extension to unbounded mergers is as follows. Even though unbounded mergers play in a game with an infinite number of rounds, one can divide the rounds into stages where each stage only has a finite number of consecutive rounds. We can then treat each stage as an independent known-size merging game, which allows us to use Construction 11.

▶ **Construction 15** (Unbounded online mergers for given fixed depth d)**.** *We construct a merger MerDep_d which uses MerWid_w in Construction 11 as a subroutine. We first partition the set of rounds into* stages *$\mathcal{S}_i$ that consist of consecutive rounds and that $|\mathcal{S}_i| = \binom{d+i}{d}$. In stage $\mathcal{S}_i$ (i.e., for round $k, k \in [\sum_{j=1}^{i-1} |\mathcal{S}_j| + 1, \sum_{j=1}^{i} |\mathcal{S}_j|]$) we use MerWid_w using width $w = i$ and treat the incoming sets $\{\ell\}, \ell \in \mathcal{I}$ as if it is $\{\ell - \sum_{j=1}^{i-1}\}$. While doing so, we ignore the trees that are constructed in the previous $i-1$ stages (i.e., keep them as part of the set of trees, without merging them).*

▶ **Proposition 16.** *In Construction 15, we have $\mathsf{wid}_{[n]} \leq 2 \cdot \mathsf{widthLB}(n, d) - 1$ for all $d \geq 0$.*

Proof. If $d = 0$, Construction 15 is trivially optimal in width. Below, assume $d > 0$. Uing Lemma 12, we know each completed stage ends up with exactly one tree of depth d.

Let k be the smallest positive integer such that $\binom{d+1}{d} + \binom{d+2}{d} + \cdots + \binom{d+k}{d} = \binom{d+1+k}{d+1} - 1 \geq n$. There are at least $k-1$ completed stage. If the last stage is also completed, there are in total k trees. Otherwise, since the width of the last stage is bounded by k, the total width is bounded by $2k-1$. Note that $\binom{d+1+k}{d+1} > n$ but $\binom{d+k}{d+1} \leq n$, which means $k = \mathsf{minBin}(n, d+1) = \mathsf{widthLB}(n, d)$. ◀

2.3.2 Unbounded online mergers for growing depths

Let $d(n)$ be a non-decreasing function from $\mathbb{N}$ to $\mathbb{N}$ modeling our desired upper bound on $\mathsf{dep}_{[n]}$. For example, we might want to have a scheme with depth $\log\log n$ or $\log n / \log\log n$ (rounded to integers). In this subsection, we show how to achieve online mergers that respect the upper bound $\mathsf{dep}_{[n]} \leq d(n)$, while achieving a width that is within 2 multiplicative factor of the optimal width, as it will hold that $\mathsf{wid}_{[n]} \leq 2\mathsf{widthLB}(n, d(n))$.

Before defining the construction, here we define *jumping points* of the function $d(n)$.

▶ **Definition 17** (Jumping points). *Suppose $d(n)\colon \mathbb{N} \to \mathbb{N}$ is non-decreasing. Define the set of* jumping points *of $d(n)$, $\{a_1, a_2, a_3, \dots\}$, as follows.*

1. $a_1 = 1$;
2. *For $i > 1$, a_i is the smallest positive integer satisfying $d(a_i) > d(a_{i-1})$.*

▶ **Construction 18** (Approximately optimal merger for a growing depth). *Suppose $\{a_1, a_2, a_3, \dots\}$ are the jumping points of the non-decreasing depth function $d(n)$. Our unbounded merger $\mathsf{MerDep}_{d(n)}$ uses an unbounded merger MerDep_c for* fixed *depth c as a subroutine. Originally, there is no trees, and the algorithm $\mathsf{MerDep}_{d(n)}$ works as follows at round n.*

- *If $n = a_i$ for $i \in \{1, 2, \dots\}$, then $\mathsf{MerDep}_{d(n)}$ will merge all the current trees (including the single-node tree $\{a_i\}$ that has just arrived) into one tree.*
- *For $a_i < n < a_{i+1}$, $\mathsf{MerDep}_{d(n)}$ ignores the single merged tree that was shaped in round a_i and will treat the arriving sets $\{\ell\}$ as $\{\ell - a_i\}$ and runs MerDep_c for $c = d(a_i) = d(n)$.*

The fact that the depth of the trees of Construction 18 satisfy $\mathsf{dep}_{[n]} \le d(n)$ is immediate by its definition and the fact that $d(n)$ is non-decreasing. Specifically, for $n = a_i$, assuming the depth was previously at most $d(a_i - 1)$, by doing the single merge, the single merged tree will have depth at most $1 + d(a_i - 1) \le d(a_i) = d(n)$. Hence, in the following we focus on analyzing its width.

▶ **Proposition 19.** *Suppose $\mathsf{wid}^c_{[n]}$ is the width of MerDep_c that is used inside Construction 18. Then the width of the final merger in Construction 18 is at most $\mathsf{wid}^{d(n)}_{[n]} + 1$.*

Proof. At every round a_i for $i \in \{1, 2, \dots\}$, there will be exactly one tree. At round n where $a_i < n < a_{i+1}$, the width of $\mathsf{MerDep}_{d(a_i)}$ is upper bounded by $\mathsf{wid}^{d(a_i)}_{[n-a_i]}$. Since $\mathsf{wid}^c_{[n]}$ is a non-decreasing function of n for any fixed c and that $d(a_i) = d(n)$, we have $\mathsf{wid}^{d(a_i)}_{[n-a_i]} \le \mathsf{wid}^{d(a_i)}_{[n]} = \mathsf{wid}^{d(n)}_{[n]}$. Also note that the tree built at a_i is not touched during the rounds n such that $a_i < n < a_{i+1}$. Therefore, the width $\mathsf{wid}_{[n]}$ of the merger in Construction 18 is upper bounded by $\mathsf{wid}^{d(n)}_{[n]} + 1$. ◀

The following corollary follows immediately from Propositions 16 and 19.

▶ **Corollary 20** (2-approximating the width for a given growing depth). *If $d(n)$ is a non-decreasing depth function of n, and if one uses Construction 15 as the subroutine in Construction 18, then the resulting construction will have width bounded as $\mathsf{wid}_{[n]} \le 2 \cdot \mathsf{widthLB}(n, d(n))$.*

How about unbounded online mergers for a growing width? The results above leave one case uncovered: what if we want to have an unbounded merger that satisfies width $\mathsf{wid}_{[n]} \le w(n)$ for a given non-decreasing function $w(n)$? For the constant $w(n) = w$, we already have an optimal solution (see Theorem 13). But, what if $w(n)$ is an increasing function of n? Here we argue that it is in fact impossible to find an unbounded merger that approximates the optimal depth within any constant factor, the way Corollary 20 does this for the width. The reason is that the width function $w(n)$ can remain small for too long (when n grows) and then suddenly jump significantly. For example, suppose $w(i) = 1$ for all $i < n$, and $w(n) = n$. Then, the depth is forced to grow linearly $d_{n'} = n' - 1$ for all $n' < n$, while after reaching round n, the width is suddenly allowed to be n, for which we do not need any depth more than 1. However, we have already paid the cost of having depth $d_n > n - 1$.

Examples of choices of depth functions. Here we demonstrate the bounds on the width that follow from choosing the depth in various ways in the construction of Corollary 20.

▶ **Corollary 21.** *In Construction 18, and for non-decreasing depth function $d(n)$ it holds that* $\mathsf{wid}_{[n]} = O(d \cdot n^{\frac{1}{d+1}})$*, where* $d = d(n)$*. In particular, if* $d(n) = c \cdot \log n / \log\log n$*, then* $\mathsf{wid}_{[n]} < \mathsf{poly}(\log n)$*.*

Proof. Let $w = \mathsf{widthLB}(n, d) = \mathsf{minBin}(n, d+1)$. Using the bound $\binom{m}{k} \geq (\frac{m}{k})^k$, we have $(\frac{w+d}{d+1})^{d+1} \leq \binom{w+d}{d+1} \leq n$. We can then bound $w \leq (d+1) \cdot n^{\frac{1}{d+1}} - d$. By Corollary 20, we know $\mathsf{wid}_{[n]} \leq 2 \cdot (d+1) \cdot n^{\frac{1}{d+1}} - 2d$.

Now, we prove the second part. By the first part, we know $\mathsf{wid}_{[n]} = O(d \cdot n^{\frac{1}{d+1}})$, where $d = d(n) = c \cdot \log n / \log\log n$. In addition, we have $n^{\frac{1}{d+1}} = n^{\frac{1}{c \cdot \log n / \log\log n + 1}} < (2^{\log n})^{\frac{\log\log n}{c \log n}} = \log^{1/c} n$. Therefore, $\mathsf{wid}_{[n]} = O(d \cdot n^{\frac{1}{d+1}}) = O(\log^{1+1/c} n / \log\log n)$. ◀

2.3.3 Stronger lower bounds for unbounded online mergers

In this section, we show that there is a real cost to pay when we aim for *unbounded* online mergers. Namely, we show that when the merger is *not* aware of the set size n, it *cannot* match the lower bound $\mathsf{widthLB}(n, d)$, even though we *could* match this bound knowing n ahead of the time. In particular, we prove the following theorem.

▶ **Theorem 22** (Stronger lower bound for depth-one unbounded online mergers)**.** *Let* M *be an unbounded online merger (for* $\mathbb{N}$*) whose depth is bounded by 1 (i.e., only one merge is allowed for each element's set). Then,* $\mathsf{wid}_{[n]} \not\leq (2\sqrt{2} - \Omega(1))\sqrt{n}$*.*

To appreciate the bound of Theorem 22 we observe that when we know n ahead of the time, we can *beat* this lower bound. As shown in Proposition 23, there is a merger satisfying $\mathsf{wid}_{[n]} \leq \sqrt{2n}$. So, the lower bound of Theorem 22 is strictly larger than the optimal bound for the setting of knowing set sizes.

▶ **Proposition 23.** *Let n be a given positive integer. There is an online merger for $[n]$ whose depth is bounded by 1 and width is bounded by $\sqrt{2n}$.*

Proof. From Proposition 14, we know MerWid_w in Construction 11 with width $w = \mathsf{minBin}(n, 2)$ is an online merger for $[n]$ whose depth is bounded by 1. By definition, we know $\binom{w+1}{2} \leq n$. We then have $w \leq \frac{\sqrt{8n+1}-1}{2} \leq \sqrt{2n}$. ◀

Moreover, as shown in Proposition 24, Construction 15 matches this bound up to an additive constant gap.

▶ **Proposition 24.** *The unbounded merger of Construction 15 has* $\mathsf{wid}_{[n]} \leq 2\sqrt{2n}$ *for* $d = 1$*.*

Proof. From Proposition 16, we know $\mathsf{wid}_{[n]} \leq 2 \cdot \mathsf{widthLB}(n, 1) = 2 \cdot \mathsf{minBin}(n, 2) \leq 2\sqrt{2n}$. ◀

The key tool we use is a special kind of depth one merger called 1-regular merger. First recall that a *stage* is a sequence of consecutive rounds. Intuitively, a merger is 1-regular if rounds can be divided into stages such that at the last round of every stage all trees added during this stage are merged into one tree. Note that since the merger has depth one, no trees are merged at times other than the last round of a stage.

▶ **Definition 25** (1-regular merger). *Let* M *be an unbounded online merger (for* $\mathbb{N}$*) whose depth is bounded by* 1*. Let* $\mathcal{T} = \{t_1 < t_2 < t_3 < \cdots\}$ *be the set of all rounds where the merge operation is performed. Let* $t_0 = 0$*. We say* M *is* 1-regular *if at every* $t_i \in \mathcal{T}$ *the set of single-node trees* $\{t_{i-1}+1, t_{i-1}+2, \cdots, t_i\}$ *are all merged into one set (i.e., tree of depth 1).*

We first show that given any M whose depth is bounded by 1, there is a 1-regular M′ that is at least as good as M with respect to width.

▶ **Lemma 26.** *Let* M *be a merger for* $\mathbb{N}$ *whose depth is bounded by* 1*. Then, there exists a 1-regular* M′ *for* $\mathbb{N}$ *such that the width of* M′ *is at most the width of* M *at every round* n*.*

Proof. If M is already 1-regular, then we are done. Otherwise, let $\mathcal{T} = \{t_1 < t_2 < t_3 < \cdots\}$ be the set of all rounds where the merge operation is performed by M. Let $t_0 = 0$. M′ simply merges $\{t_{i-1}+1, t_{i-1}+2, \cdots, t_i\}$ at t_i. At round $t_i \in \mathcal{T}$, M′ has only i trees while M has at least i trees since it has performed at least i merge operations, and due to the depth being 1, these merge operations are on separate nodes. At other times, since no merge operation is performed, M has at least as many trees as M′ does. ◀

The following lemma is also useful for the proof of Theorem 22.

▶ **Lemma 27.** *Let* $j > 2$ *be an integer, and* $d < 2$ *and* c *be two positive reals. Define the function* $f(i) := \frac{i \cdots j}{(i-d)\cdots(j-d)} \cdot c$ *for arbitrary positive integer* $i > j$*. Then,* $f(i) = o(i^2)$*.*

Proof. We first define the function $g(i) := \frac{i \cdots j}{(i-2)\cdots(j-2)} \cdot c$ for arbitrary positive integer $i > j$. Note that we have $g(i) = \frac{i \cdot (i-1)}{(j-1)\cdot(j-2)} \cdot c$. Therefore, we have $g(i) = \theta(i^2)$. Then, it suffices to prove that $f(i) = o(g(i))$.

Let γ be an arbitrary positive real number. We show that $g(i) > \gamma \cdot f(i)$ for sufficiently large i. Equivalently, we prove that $\ln g(i) - \ln f(i) > \ln \gamma$ for sufficiently large i. We have

$$\ln g(i) - \ln f(i) = \ln \frac{i-d}{i-2} + \cdots + \ln \frac{j-d}{j-2} = \sum_{k=j-2}^{i-2} \ln\left(1 + \frac{2-d}{k}\right).$$

It suffices to show that the series $\sum_{k=j-2}^{\infty} \ln(1 + \frac{2-d}{k})$ diverges. To this end, we use integral test to show that it diverges. Note that

$$\int \ln\left(1 + \frac{2-d}{x}\right) dx = (2-d) \cdot \ln|x + 2 - d| + x \cdot \ln\left(1 + \frac{2-d}{x}\right) + C.$$

Therefore, we have $\int_{j-2}^{\infty} \ln(1 + \frac{2-d}{x}) dx = \infty$, which implies that $\sum_{k=j-2}^{i-2} \ln(1 + \frac{2-d}{k}) > \ln \gamma$ for sufficiently large i. ◀

Proof of Theorem 22. By Lemma 26, it suffices to prove the theorem for 1-regular mergers. Therefore, we assume M is 1-regular. Let $\mathcal{T} = \{t_1 < t_2 < t_3 < \cdots\}$ be the set of all rounds where M performs a merge operation. Let $t_0 = 0$ and $s_i = t_i - t_{i-1}$ be the size of stage i.

We first consider the case where $\mathcal{T}$ is a finite set. Then, starting at some round, no merge operation is ever performed, which means the width will be $\Theta(n)$.

Let $d < 2$ be an arbitrary positive real number. We now consider the case where $\mathcal{T}$ is an infinite set and there are only finitely many i satisfying $s_i \geq d \cdot \frac{t_i}{i}$. Then, there is a j such that $s_i < d \cdot \frac{t_i}{i}$ for $i \geq j$. Note that $s_i = t_i - t_{i-1} < d \cdot \frac{t_i}{i}$, which implies $t_i < \frac{i}{i-d} \cdot t_{i-1}$. We then have $t_i < \frac{i \cdots j}{(i-d)\cdots(j-d)} \cdot t_{j-1}$. From Lemma 27, we know $t_i = o(i^2)$, which means $i > \omega(\sqrt{t_i})$. In this case, we know $\mathsf{wid}_{[t_i]} \geq \omega(\sqrt{t_i})$.

Finally, we consider the case where $\mathcal{T}$ is an infinite set and there are infinitely many i satisfying $s_i \geq d \cdot \frac{t_i}{i}$. Let's consider one such i. At round $t_i - 1$, the width is at least $i - 1 + s_i - 1 > i + d \cdot \frac{t_i - 1}{i} - 2 \geq 2 \cdot \sqrt{d \cdot (t_i - 1)} - 2$. Note that this holds for arbitrary $d < 2$, which means that $\mathsf{wid}_{[n]} \not\leq (2\sqrt{2} - \Omega(1))\sqrt{n}$. Otherwise, there would be a constant $d' < 2$ such that $\mathsf{wid}_{[n]} \leq 2\sqrt{d' \cdot n}$ for sufficiently large n, contradicting that $\mathsf{wid}_{[m]} \geq 2 \cdot \sqrt{d \cdot m} - 2$ for a constant $d > d'$ and $m = t_i - 1$, as proved above. ◀

3 RBE with optimal number of decryption updates

In this section, we observe that the same approach of using online merger for transparent additive accumulators in Section B extends to registration based encryption schemes (RBEs), while the length of the digests and number of witness updates would correspond to the length of the public parameter and the number of decryption updates. Hence, we can obtain RBEs with *optimal* number of decryption updates matching the lower bound of [22]. The basic definitions of RBE and the primitives used in our construction can be found in Section A. Roughly speaking, the construction is the same as the IO-based construction of [12], while we use our own updates-optimal accumulator instead of the one by [25].

▶ **Construction 28.** *We will use an IO scheme* $(\mathsf{Obf}, \mathsf{Eval})$*, and a SSB hash function system* $(\mathsf{Hash}, \mathsf{HGen})$ *and a PKE scheme* $(\mathsf{G}, \mathsf{E}, \mathsf{D})$*. Using them together with a merger* M*, we show how to implement the subroutines of RBE according to Definition 37.*

- $\mathsf{Stp}(1^\kappa) \to (\mathsf{pp}_0, \mathsf{aux}_0)$*: This algorithm outputs* $\mathsf{pp}_0 = \mathsf{hk}_2 \leftarrow \mathsf{HGen}(1^\kappa, 2, 0)$ *and* $\mathsf{aux} = \emptyset$ *is empty. Then, initialize* HMer *(see Definition 43) using* M *and* H*. H is defined such that* $H(\mathsf{Val}(\mathsf{x}_1), \ldots, \mathsf{Val}(\mathsf{x}_i)) = \mathsf{Hash}(\mathsf{hk}_i, (\mathsf{Val}(\mathsf{x}_1), \ldots, \mathsf{Val}(\mathsf{x}_i)))$*. Note that here we have only sampled* hk_2*. If another key* hk_i *is needed in* Reg *because* HMer *needs to merge more trees,* Reg *will generate the key on the fly by running* HGen*.*
- $\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{pp}_n, \mathsf{id}, \mathsf{pk}) \to \mathsf{pp}_{n+1}$*: First add a new tree whose root has value* $\mathsf{Hash}(\mathsf{hk}_2, (\mathsf{id}, \mathsf{pk}))$ *with* id *and* pk *as the children nodes. We then let* HMer *handles the merging of trees. If another key* hk_i *is needed, run* $\mathsf{HGen}(1^\kappa, i, 0)$ *to get the key. Then, output the list of pairs of root and depth of all trees* $((\mathsf{rt}_1, \mathsf{d}_1), \ldots, (\mathsf{rt}_\eta, \mathsf{d}_\eta))$ *together with all keys* hk_i *as* pp_{n+1}*.*
- $\mathsf{Enc}(\mathsf{pp}, \mathsf{id}, \mathsf{m}) \to \mathsf{ct}$*: First parse* pp *to get a list of pairs of root and depth of all trees* $((\mathsf{rt}_1, \mathsf{d}_1), \ldots, (\mathsf{rt}_\eta, \mathsf{d}_\eta))$*. Generate programs* $\mathsf{P}_1, \ldots, \mathsf{P}_\eta$ *where* P_i *works as follows:*
  Hardwired values: rt_i*,* d_i*,* $(\mathsf{hk}_1, \ldots, \mathsf{hk}_\kappa)$*,* m*,* id*,* r *(the randomness)*
  Input: pth
  1. *Parse* $\mathsf{pth} = ((\ell_{\mathsf{d}_i}, r_{\mathsf{d}_i}), \ldots, (\ell_1, r_1))$*, and if not possible, output* $\perp$*.*
  2. *If* $\mathsf{id} \neq \ell_{\mathsf{d}_i}$*, then output* $\perp$*.*
  3. *Compute* $\mathsf{tmp}_{j-1} = \mathsf{Hash}(\mathsf{hk}_{\mathsf{len}_j}, (\ell_j, \mathsf{tmp}_j, r_j))$ *for* $j = \mathsf{d}_i, \ldots, 1$ *where* $\mathsf{tmp}_{\mathsf{d}_i}$ *is the empty string and* len_j *is the length of* $(\ell_j, \mathsf{tmp}_j, r_j)$*. (Note that* ℓ_j *and* r_j *are tuples.) If* $\mathsf{tmp}_0 = \mathsf{rt}_i$*, then output* $\mathsf{E}(\mathsf{r}_{\mathsf{d}_i}, \mathsf{m}; \mathsf{r})$ *by using* $\mathsf{r}_{\mathsf{d}_i}$ *as the public key and* r *as the randomness, otherwise output* $\perp$*.*

  Then, output $\mathsf{ct} := (\mathsf{pp}, \mathsf{Obf}(\mathsf{P}_1), \ldots, \mathsf{Obf}(\mathsf{P}_\eta))$ *where* Obf *is IO obfuscation.*
- $\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}) \to \mathsf{u}$*: First locate the tree* T *having* id *as one of the leaves. If no such tree exists, halt. Otherwise, let* d *be the depth of* T *and* $(p_d, \ldots, p_0)$ *be the path from* p_d*, which is* id*, to the root* p_0*. Let* ℓ_i *(resp.* r_i*) be the tuple of values of left (resp. right) siblings of* p_i *for* $i \in [d-1]$*. Note that the sibling of* id *is* pk*. Let* $\ell_d = \mathsf{id}$ *and* $r_d = \mathsf{pk}$*. Output* $\mathsf{w} = ((\ell_d, r_d), \ldots, (\ell_1, r_1))$*.*
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct}) \to \mathsf{m}$*: Parse* $\mathsf{ct} = (\mathsf{pp}, \mathsf{Obf}(\bar{\mathsf{P}}_1), \ldots, \mathsf{Obf}(\bar{\mathsf{P}}_\eta))$*. Form* $\mathsf{m}_i = \mathsf{D}_{\mathsf{sk}}(\bar{\mathsf{P}}_i(\mathsf{u}))$ *for each program* $\bar{\mathsf{P}}_i$*. Output the first* $\mathsf{m}_i \neq \perp$*.*

Completeness of Construction 28 is straightforward.

▶ **Proposition 29** (Compactness and Efficiency of Construction 28)**.** *Construction 28 satisfies the compactness requirements of Definition 38.*

Proof. The length of public parameter is the sum of the length of keys, which is bounded by $\kappa \cdot \mathsf{wid}_{[n]}$, and roots, which is again bounded by $\kappa \cdot \mathsf{wid}_{[n]}$, and depth, which is bounded by $\mathsf{wid}_{[n]} \cdot \log \mathsf{dep}_{[n]}$. In total, it is bounded by $(2 \cdot \kappa + \log \mathsf{dep}_{[n]}) \cdot \mathsf{wid}_{[n]}$. The number of updates is bounded by $\mathsf{dep}_{[n]}$. The size of update for an accumulated value is the number of trees that are merged with the tree it belongs times the length of the roots. Since the depth is bounded by $\mathsf{dep}_{[n]}$, we know at most $\mathsf{dep}_{[n]}$ merges can happen. Since the width is bounded by $\mathsf{wid}_{[n]}$, we know at most $\mathsf{wid}_{[n]} \cdot \mathsf{dep}_{[n]}$ trees are merged. Therefore, the size of update is bounded by $\kappa \cdot \mathsf{wid}_{[n]} \cdot \mathsf{dep}_{[n]}$. For efficiency, note that the total number of merges during the addition of a value is bounded by $\mathsf{wid}_{[n]} \cdot \mathsf{dep}_{[n]}$. Also note that one can use an appropriate data structure that efficiently finds the tree an identity belongs. ◀

▶ Remark 30. By Corollary 21, we know that if we use the fully online merger in Construction 18 where $d(n) = \frac{\log n}{\log \log n}$, the number of updates is bounded by $\frac{\log n}{\log \log n}$ and the length of public parameter is $\mathsf{poly}(\kappa, \log(n))$, resolving the open question of [22].

▶ **Proposition 31** (Security of Construction 28)**.** *Construction 28 satisfies the soundness requirements of Definition 39.*

Proof (Sketch). The proof is almost identical to the proof in [12]. The key idea is that the index-hiding and somewhere statistically binding property forces all PPT algorithms to behave as if the public key is statistically binding to the root of the tree. Then, obfuscation and the semantic security of public key encryption guarantees that encryptions of different messages are indistinguishable. We refer readers to [12] for details. ◀

References

1 Sattam S Al-Riyami and Kenneth G Paterson. Certificateless public key cryptography. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 452–473. Springer, 2003.

2 Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-44647-8_1.

3 Niko Barić and Birgit Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT '97*, pages 480–494, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

4 Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93*, pages 274–285, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

5 Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-44647-8_13.

6 Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, pages 61–76, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

7 Zhaohui Cheng, Richard Comley, and Luminita Vasiu. Remove key escrow from the identity-based encryption system. In *Exploring New Frontiers of Theoretical Informatics*, pages 37–50. Springer, 2004.

8 Sherman SM Chow. Removing escrow from identity-based encryption. In *International Workshop on Public Key Cryptography*, pages 256–276. Springer, 2009.

9 Kelong Cong, Karim Eldefrawy, and Nigel P Smart. Optimizing registration based encryption. In *IMA International Conference on Cryptography and Coding*, pages 129–157. Springer, 2021.

10 Keita Emura, Shuichi Katsumata, and Yohei Watanabe. Identity-based encryption with security against the KGC: a formal model and its instantiation from lattices. In *European symposium on research in computer security*, pages 113–133. Springer, 2019.

11 Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29 2013. IEEE Computer Society Press. doi:10.1109/FOCS.2013.13.

12 Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from ibe. In *Theory of Cryptography Conference*, pages 689–718. Springer, 2018.

13 Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 63–93, Beijing, China, April 14–17 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17259-6_3.

14 Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography – PKC 2019*, pages 63–93, Cham, 2019. Springer International Publishing.

15 Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. Efficient registration-based encryption. Cryptology ePrint Archive, Paper 2022/1505, 2022. URL: https://eprint.iacr.org/2022/1505.

16 Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 621–651, Santa Barbara, CA, USA, August 17–21 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-56784-2_21.

17 Vipul Goyal. Reducing trust in the PKG in identity based cryptosystems. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 430–447, Santa Barbara, CA, USA, August 19–23 2007. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-74143-5_24.

18 Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. Black-box accountable authority identity-based encryption. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 427–436. ACM, 2008.

19 Susan Hohenberger, George Lu, Brent Waters, and David J. Wu. Registered attribute-based encryption. Cryptology ePrint Archive, Paper 2022/1500, 2022. URL: https://eprint.iacr.org/2022/1500.

20 Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science*, pages 163–172, Rehovot, Israel, January 11–13 2015. Association for Computing Machinery. doi:10.1145/2688073.2688105.

21 Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73, 2021.

22 Mohammad Mahmoody, Wei Qi, and Ahmadreza Rahimi. Lower bounds for the number of decryption updates in registration-based encryption. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography*, pages 559–587, Cham, 2022. Springer Nature Switzerland.

23 Tatsuaki Okamoto, Krzysztof Pietrzak, Brent Waters, and Daniel Wichs. New realizations of somewhere statistically binding hashing and positional accumulators. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015*, pages 121–145, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

24 Ilker Ozcelik, Sai Medury, Justin Broaddus, and Anthony Skjellum. An overview of cryptographic accumulators. In *Proceedings of the 7th International Conference on Information Systems Security and Privacy*. SCITEPRESS – Science and Technology Publications, 2021. doi:10.5220/0010337806610669.

25 Leonid Reyzin and Sophia Yakoubov. Efficient asynchronous accumulators for distributed pki. In *Security and Cryptography for Networks: 10th International Conference, SCN 2016, Amalfi, Italy, August 31–September 2, 2016, Proceedings 10*, pages 292–309. Springer, 2016.

26 Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, Santa Barbara, CA, USA, August 19–23 1984. Springer, Heidelberg, Germany.

27 Qin Wang, Rujia Li, David Galindo, Qi Wang, Shiping Chen, and Yang Xiang. Transparent registration-based encryption through blockchain. *Distributed Ledger Technologies: Research and Practice*, 2022.

28 Quanyun Wei, Fang Qi, and Zhe Tang. Remove key escrow from the BF and Gentry identity-based encryption with non-interactive key generation. *Telecommunication Systems*, pages 1–10, 2018.

A Preliminaries

▶ **Definition 32** (Forward DAGs). *Let $G = (\mathcal{V}_G, \mathcal{E}_G)$ be a directed acyclic graph (DAG) with vertices $\mathcal{V}_G = [n]$ (in case of being finite) or $\mathcal{V}_G = \mathbb{N}$ (in case of being infinite). We write $i \in G$ if $i \in \mathcal{V}_G$, ad we write $(i,j) \in G$ if $(i,j) \in \mathcal{E}_G$ (i.e., there is an edge from i to j in G). We call G a forward DAG, if for all $(i,j) \in G$, we have $i \leq j$. For any vertex u, by $\deg^+(u) = |\{v \mid (u,v) \in G\}|$ we denote the* out-degree *of u, and we let $\deg^+(G) = \max_{u \in G} \deg^+(u)$.*

▶ **Definition 33** (Skipping sequences [22]). *Let G be a forward DAG (see Definition 32). We call $\mathcal{S} = \{u_1 < u_2 < \cdots < u_k\} \subseteq \mathcal{V}_G$ a* skipping sequence *if for every $i \leq k-1$ and every edge $(u_i, v) \in G$, it holds that: either $v < u_{i+1}$ or $v > u_k$ (i.e., $v \notin \{u_{i+1}, u_{i+1}+1, \ldots, u_k\}$).*

▶ **Theorem 34** (Skipping sequences in low-degree DAGs [22]). *Let G be a forward DAG over vertices $[n]$, where $n \geq \binom{w+d}{d+1}$ for $w, d \in \mathbb{N}$, and that that $\deg^+(G) \leq d$. Then, there exists a skipping sequence in G of size at least w.*

▶ **Definition 35** (Indistinguishability obfuscation). *A uniform PPT algorithm* Obf *is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ (where each C_κ is a set indexed by a security parameter κ) if the following holds:*

- ***Completeness.*** *For all security parameters $\kappa \in \mathbb{N}$ and all circuits $C \in \mathcal{C}_\kappa$, we obtain an obfuscation with the same function:*

$$\Pr_{\mathsf{Obf}}[C' \equiv C : C' = \mathsf{Obf}(1^\kappa, C)] = 1.$$

- ***Security.*** *For any PPT distinguisher D, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\kappa \in \mathbb{N}$*, for all pairs of functionally equivalent circuits* $C_1 \equiv C_2$ *from the same family* $C_1, C_2 \in \mathcal{C}_\kappa$*,*

$$|\Pr_{\mathsf{Obf}}[D(1^\kappa, \mathsf{Obf}(1^\kappa, C_1)) = 1] - \Pr_{\mathsf{Obf}}[D(1^\kappa, \mathsf{Obf}(1^\kappa, C_2)) = 2]| = 1.$$

▶ **Definition 36** (SSB hash functions [23])**.** *A somewhere statistically binding hash system consists of two polynomial time algorithms* $\mathsf{HGen}, \mathsf{Hash}$*.*

- $\mathsf{HGen}(1^\kappa, 1^s, L, i) \to \mathsf{hk}$ *: This algorithm takes as input the security parameter* 1^κ*, a block size* s*, an input length* $L \leq 2^\kappa$ *and an index* $i \in \{0, \ldots, L-1\}$ *and outputs a hashing key* hk*. Without loss of generality, we assume that* $s = \kappa$*. Therefore, we will not write* s *explicitly.*
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{x}) \to \mathsf{y}$ *: This deterministic algorithm takes as input a hashing key* hk *and a value* $\mathsf{x} \in \{0,1\}^{s \cdot L}$ *and outputs a hash* $\mathsf{y} \in \{0,1\}^\kappa$*.*

We require the following properties:

- ***Index hiding.*** *We consider the following game between an attacker* $\mathcal{A}$ *and a challenger:*
  - *The attacker* $\mathcal{A}(1^\kappa)$ *chooses parameters* 1^s*,* L *and two indices* $i_0, i_1 \in \{0, \ldots, L-1\}$*.*
  - *The challenger chooses a bit* $b \leftarrow \{0,1\}$ *and sets* $\mathsf{hk} \leftarrow \mathsf{HGen}(1^\kappa, 1^s, L, i_b)$*.*
  - *The attacker* $\mathcal{A}$ *gets* hk *and outputs a bit* b'*.*

  We require that for any PPT attacker $\mathcal{A}$ *we have* $|\Pr[b = b'] - \frac{1}{2}| = \mathsf{negl}(\kappa)$ *in the above game.*
- ***Somewhere statistically binding.*** *Let* $\mathsf{x} \in \{0,1\}^{s \cdot L}$ *and* $i \in \{0, \ldots, L-1\}$*. By* $\mathsf{x}[i]$ *we denote the sub-string of* x *starting at* $s \cdot i + 1$ *and ending at* $s \cdot (i+1)$*. We say that* hk *is statistically binding for an index* $i \in \{0, \ldots, L-1\}$ *if there do not exist any values* $\mathsf{x}, \mathsf{x}' \in \{0,1\}^{s \cdot L}$ *with* $\mathsf{x}[i] \neq \mathsf{x}'[i]$ *such that* $\mathsf{Hash}(\mathsf{hk}, \mathsf{x}) = \mathsf{Hash}(\mathsf{hk}, \mathsf{x}')$*. We require that for any parameters* s*,* L *and any integer* $i \in \{0, \ldots, L-1\}$ *we have:*

$$\Pr_{\mathsf{HGen}}[\mathsf{hk} \text{ is statistically binding for index } i : \mathsf{hk} \leftarrow \mathsf{HGen}(1^\kappa, 1^s, L, i)] \geq 1 - \mathsf{negl}(\kappa).$$

A.1 Registration-Based Encryption

▶ **Definition 37** (Syntax of RBE)**.** *PPT algorithms* $(\mathsf{Gen}, \mathsf{Reg}, \mathsf{Enc}, \mathsf{Upd}, \mathsf{Dec})$ *form a* registration-based encryption *(RBE for short) if they work together as follows.*

- ***Generating CRS.*** *A common random string* crs *of length* $\mathsf{poly}(\kappa)$ *is publicly sampled at the beginning, for the security parameter* κ*.*
- ***Key Generation.*** $\mathsf{Gen}(1^\kappa) \to (\mathsf{pk}, \mathsf{sk})$*: The randomized algorithm* Gen *outputs a pair of public and secret keys* $(\mathsf{pk}, \mathsf{sk})$*. The key generation algorithm is run by any honest party locally who wants to register itself into the system.*
- ***Registration.*** $\mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \mathsf{pp}'$*: The deterministic algorithm* Reg *takes as input the CRS* crs*, current public parameter* pp*, a registering identity* id *and a public key* pk *(supposedly for the identity* id*), and it outputs* pp' *as the updated public parameters. The* Reg *algorithm uses* read *and* write *access to* auxiliary information aux *which will be updated into* aux' *during the process of registration and helps with the efficiency of the registration and updates (below). The system is initialized with* $\mathsf{pp}, \mathsf{aux} = \bot$*.*
- ***Encryption.*** $\mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{m}) \to \mathsf{ct}$*: The randomized algorithm* Enc *takes as input the CRS* crs*, a public parameter* pp*, a recipient identity* id*, and a plaintext message* m*, and it outputs a ciphertext* ct*.*

- ***Update.*** $\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}, \mathsf{pk}) \to \mathsf{u}$*: The deterministic algorithm* Upd *takes as input the current public parameter* pp*, an identity* id*, and a public key* pk*. It has* read only *oracle access to* aux *and generates an update information* u *that can help* id *to decrypt its messages.*
- ***Decryption.*** $\mathsf{Dec}(\mathsf{sk}, \mathsf{u}, \mathsf{ct}) \to \mathsf{m}$*: The deterministic decryption algorithm* Dec *takes as input a secret key* sk*, an update information* u*, and a ciphertext* ct*, and it outputs a message* $\mathsf{m} \in \{0,1\}^*$ *or in* $\{\perp, \mathsf{GetUpd}\}$*. The symbol* $\perp$ *indicates a syntax error while* GetUpd *indicates that more recent update information (than* u*) might be needed for decryption.*

The Reg *and* Upd *algorithms are performed by the party called* key curator*, which we call KC for short, and* aux *can be seen as the state held by the KC.*

▶ **Definition 38** (Completeness, compactness, and efficiency of RBE)**.** *Consider the following game* $\mathsf{Comp}_{\mathcal{A}}(\kappa)$ *between a challenger* $\mathcal{C}$ *and an interactive* computationally unbounded *adversary* $\mathcal{A}$ *who is yet limited to* $\mathsf{poly}(\kappa)$ *rounds of interaction.*

1. ***Initialization.*** *$\mathcal{C}$ sets* $\mathsf{pp} = \perp$, $\mathsf{aux} = \perp$, $\mathsf{u} = \perp$, $\mathcal{D} = \emptyset$, $\mathsf{id}^* = \perp$, $t = 0$, *and* $\mathsf{crs} \leftarrow U_{\mathsf{poly}(\kappa)}$*, and sends the sampled* crs *to $\mathcal{A}$.*
2. *Till $\mathcal{A}$ continues (which is at most* $\mathsf{poly}(\kappa)$ *steps), proceed as follows. At every iteration, $\mathcal{A}$ chooses exactly one of the actions below to perform.*
   a. ***Registering a corrupted (non-target) identity.*** *$\mathcal{A}$ sends some* $\mathsf{id} \notin \mathcal{D}$ *and* pk *to $\mathcal{C}$. $\mathcal{C}$ registers* $(\mathsf{id}, \mathsf{pk})$ *by letting* $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ *and* $\mathcal{D} := \mathcal{D} \cup \{\mathsf{id}\}$.
   b. ***Registering the (uncorrupted) target identity.*** *This step is allowed only if* $\mathsf{id}^* = \perp$*. In that case, $\mathcal{A}$ sends some* $\mathsf{id}^* \notin \mathcal{D}$ *to $\mathcal{C}$. $\mathcal{C}$ then samples* $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}(1^\kappa)$*, updates* $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$ *and* $\mathcal{D} := \mathcal{D} \cup \{\mathsf{id}^*\}$*, and sends* pk^* *to $\mathcal{A}$.*
   c. ***Encrypting for the target identity.*** *This step is allowed only if* $\mathsf{id}^* \neq \perp$*. In that case, $\mathcal{C}$ sets* $t = t + 1$*. $\mathcal{A}$ sends* $\mathsf{m}_t \in \{0,1\}^*$ *to $\mathcal{C}$ who then sets* $\mathsf{m}'_t := \mathsf{m}_t$ *and sends back a corresponding ciphertext* $\mathsf{ct}_t \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{m}_t)$ *to $\mathcal{A}$.*
   d. ***Decryption for the target identity.*** *$\mathcal{A}$ sends a* $j \in [t]$ *to $\mathcal{C}$. $\mathcal{C}$ then lets* $\mathsf{m}'_j = \mathsf{Dec}(\mathsf{sk}^*, \mathsf{u}, \mathsf{ct}_j)$*. If* $\mathsf{m}'_j = \mathsf{GetUpd}$*, $\mathcal{C}$ gets* $\mathsf{u} = \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{id}^*)$ *and then* $\mathsf{m}'_j = \mathsf{Dec}(\mathsf{sk}^*, \mathsf{u}, \mathsf{ct}_j)$.

Let $n = |\mathcal{D}|$ *be the number of identities registered till a specific moment. We require the following properties to hold for all such $\mathcal{A}$ (as specified above) and for all the moments during the game* $\mathsf{Comp}_{\mathcal{A}}(\kappa)$.

- ***Completeness.*** *The adversary $\mathcal{A}$ wins, if there is some* $j \in [t]$ *for which* $\mathsf{m}'_j \neq \mathsf{m}_j$*. We require that* $\Pr[\mathcal{A} \text{ wins } \mathsf{Comp}_{\mathcal{A}}(\kappa)] = \mathsf{negl}(\kappa)$.
- ***Parameterized compactness and efficiency.*** [4]
  - ***Size of public parameter.*** $|\mathsf{pp}| = \mathsf{w}(\kappa, n)$.
  - ***Number of updates.*** *The total number of invocations of* Upd *for identity* id^* *in Step 2(d) of the game* $\mathsf{Comp}_{\mathcal{A}}(\kappa)$ *is at most* $\mathsf{d}(\kappa, n)$.
  - ***Size of update.*** $|\mathsf{u}| \leq \mathsf{poly}(\kappa, \mathsf{w}, \mathsf{d})$.
  - ***Runtime of registration and update.*** *The running time of each invocation of* Reg *and* Upd *is at most* $\mathsf{poly}(\kappa, \mathsf{w}, \mathsf{d})$.

▶ **Definition 39** (Security of RBE)**.** *For any interactive PPT adversary $\mathcal{A}$, consider the following game* $\mathsf{Sec}_{\mathcal{A}}(\kappa)$ *between $\mathcal{A}$ and a challenger $\mathcal{C}$.*

[4] If both $\mathsf{w}(\kappa, n)$ and $\mathsf{d}(\kappa, n)$ are $\mathsf{poly}(\kappa, \log n)$, then this becomes the standard definition.

1. ***Initialization.*** *$\mathcal{C}$ sets* $\mathsf{pp} = \bot$, $\mathsf{aux} = \bot$, $\mathcal{D} = \emptyset$, $\mathsf{id}^* = \bot$, $\mathsf{crs} \leftarrow U_{\mathsf{poly}(\kappa)}$ *and sends the sampled* crs *to* $\mathcal{A}$.
2. *Till $\mathcal{A}$ continues (which is at most* $\mathsf{poly}(\kappa)$ *steps), proceed as follows. At every iteration, $\mathcal{A}$ chooses exactly one of the actions below to perform.*
   a. ***Registering non-target identity.*** *$\mathcal{A}$ sends some* $\mathsf{id} \notin \mathcal{D}$ *and* pk *to $\mathcal{C}$. $\mathcal{C}$ registers* $(\mathsf{id}, \mathsf{pk})$ *by* $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}, \mathsf{pk})$ *and* $\mathcal{D} := \mathcal{D} \cup \{\mathsf{id}\}$.
   b. ***Registering the target identity.*** *This step can be run only if* $\mathsf{id}^* = \bot$. *$\mathcal{A}$ sends some* $\mathsf{id}^* \notin \mathcal{D}$ *to $\mathcal{C}$. $\mathcal{C}$ then samples* $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{Gen}(1^\kappa)$, *updates* $\mathsf{pp} := \mathsf{Reg}^{[\mathsf{aux}]}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, \mathsf{pk}^*)$, $\mathcal{D} := \mathcal{D} \cup \{\mathsf{id}^*\}$, *and sends* pk^* *to $\mathcal{A}$.*
3. ***Encrypting for the target identity.*** *If* $\mathsf{id}^* = \bot$, *then $\mathcal{A}$ first sends some* $\mathsf{id}^* \notin \mathcal{D}$ *to $\mathcal{C}$ (this is for modeling encryptions for non-registered target identities.) Next $\mathcal{A}$ sends two messages m_0, m_1 of the same length to C. Next, $\mathcal{C}$ generates* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{crs}, \mathsf{pp}, \mathsf{id}^*, m_b)$, *where* $b \leftarrow \{0,1\}$ *is a random bit, and sends* ct *to* $\mathcal{A}$.
4. *The adversary $\mathcal{A}$ outputs a bit b' and wins the game if $b = b'$.*

An RBE scheme is secure if for all PPT $\mathcal{A}$, $\Pr[\mathcal{A} \text{ wins } \mathsf{Sec}_{\mathcal{A}}(\kappa)] < \frac{1}{2} + \mathsf{negl}(\kappa)$.

B Accumulators with optimal number of witness updates

An accumulator is a primitive in which a sequence of inputs $x_1, \ldots, x_n$ are added gradually to a pool, while (1) there is a compact digest/public parameter that is the representative of the added inputs $\{x_1, \ldots, x_n\}$, and (2) using the right witness, and the digest, one can prove the membership of x_i to the set. Such accumulators are called *additive* accumulators [24]. In this section, we use our results about online mergers to study the relation between the length of the digest and the number of updates for accumulators. Also, the accumulators that we study are transparent. Namely, there is no secret state hold by the accumulator.

In the remainder of this section, we first give the formal definition of transparent additive accumulators. Then, we give a general construction of such accumulators from collision resistant hash functions and online mergers. Finally, by applying our results on online merging, we will study the relation between the length of the digest and the number of times that the witnesses of membership need to be updated in such accumulators.

▶ **Definition 40** (Accumulators). *An accumulator scheme consists of four algorithms.*

- ***Key Generation.*** $\mathsf{Gen}(1^\kappa) \to \mathsf{k}$: *The randomized algorithm* Gen *takes as input the security parameter* 1^κ *and outputs an accumulator key* k.
- ***Addition.*** $\mathsf{Add}^{[\mathsf{aux}]}(\mathsf{k}, \mathsf{pp}, \mathsf{v}) \to \mathsf{pp}'$: *The deterministic algorithm* Add *takes as input an accumulator key* k, *a digest* pp *and a value* v, *has* read *and* write *access to the auxiliary information* aux, *and outputs a new digest* pp'.
- ***Witness update.*** $\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{v}) \to \mathsf{w}$: *The deterministic algorithm* Upd *takes as input a digest* pp, *a value* v, *has* read-only *access to* aux *and outputs a witness* w.
- ***Verification.*** $\mathsf{Ver}(\mathsf{k}, \mathsf{pp}, \mathsf{v}, \mathsf{w}) \to \mathsf{b}$: *The deterministic algorithm* Ver *takes as input an accumulator key* k, *a digest* pp, *a value* v, *a witness* w *and outputs a bit* $\mathsf{b} \in \{0,1\}$.

▶ **Definition 41** (Completeness of accumulators). *For any interactive computationally unbounded adversary $\mathcal{A}$ that still has a limited* $\mathsf{poly}(\kappa)$ *round complexity, consider the following game* $\mathsf{Comp}_{\mathcal{A}}(\kappa)$ *between $\mathcal{A}$ and a challenger $\mathcal{C}$.*

1. ***Initialization.*** *The challenger $\mathcal{C}$ sets* $\mathsf{pp} = \bot$, $\mathsf{aux} = \bot$, $\mathsf{t} = \bot$, $\mathcal{V} = \emptyset$, *generates* $\mathsf{k} \leftarrow \mathsf{Gen}(1^\kappa)$ *and sends* k *to the adversary $\mathcal{A}$.*
2. *Till $\mathcal{A}$ continues (which is at most* $\mathsf{poly}(\kappa)$ *steps), $\mathcal{A}$ chooses one of the following operations to perform.*

a. ***Addition.*** *$\mathcal{A}$ sends a value $\mathsf{v} \notin \mathcal{V}$ to $\mathcal{C}$ and $\mathcal{C}$ sets $\mathsf{pp} \leftarrow \mathsf{Add}^{[\mathsf{aux}]}(\mathsf{k}, \mathsf{pp}, \mathsf{v})$ and adds v to $\mathcal{V}$.*
b. ***Set target.*** *$\mathcal{A}$ sends a value $\mathsf{v} \in \mathcal{V}$ to $\mathcal{C}$. If $\mathsf{t} \neq \perp$, $\mathcal{C}$ does nothing. Otherwise, $\mathcal{C}$ sets $\mathsf{t} \leftarrow \mathsf{v}$.*
c. ***Witness generation.*** *If $\mathsf{t} = \perp$, $\mathcal{C}$ does nothing. Otherwise, $\mathcal{C}$ generates $\mathsf{w} \leftarrow \mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{t})$.*

The adversary $\mathcal{A}$ wins the game if $\mathsf{Ver}(\mathsf{k}, \mathsf{pp}, \mathsf{t}, \mathsf{w}) = 0$ for at least once in Step 2c. We call an accumulator scheme complete if $\Pr[\mathcal{A}$ wins in $\mathsf{Comp}_{\mathcal{A}}(\kappa)] = \mathsf{negl}(\kappa)$ for any $\mathcal{A}$.

Let $n = |\mathcal{V}|$ be the number of values added till a specific moment during the game $\mathsf{Comp}_{\mathcal{A}}(\kappa)$. We define the number of updates of an accumulator system at time n to be the number of all possible different witnesses generated for value t in Step 2c.

Security requires that no PPT adversary can find a witness for values that are not added.

▶ **Definition 42** (Security of accumulators). *For any interactive PPT adversary $\mathcal{A}$, consider the following game $\mathsf{Sec}_{\mathcal{A}}(\kappa)$ between $\mathcal{A}$ and a challenger $\mathcal{C}$.*

1. ***Initialization.*** *The challenger $\mathcal{C}$ sets $\mathsf{pp} = \perp$, $\mathsf{aux} = \perp$, $\mathcal{V} = \emptyset$, generates $\mathsf{k} \leftarrow \mathsf{Gen}(1^\kappa)$ and sends k to the adversary $\mathcal{A}$.*
2. ***Addition.*** *Till $\mathcal{A}$ continues (which is at most $\mathsf{poly}(\kappa)$ steps), $\mathcal{A}$ sends a value $\mathsf{v} \notin \mathcal{V}$ to $\mathcal{C}$ and $\mathcal{C}$ sets $\mathsf{pp} \leftarrow \mathsf{Add}^{[\mathsf{aux}]}(\mathsf{k}, \mathsf{pp}, \mathsf{v})$ and adds v to $\mathcal{V}$.*
3. *The adversary $\mathcal{A}$ outputs $\mathsf{v} \notin \mathcal{V}$ and a witness w and wins the game if $\mathsf{Ver}(\mathsf{k}, \mathsf{pp}, \mathsf{v}, \mathsf{w}) = 1$.*

We call an accumulator scheme secure if $\Pr[\mathcal{A}$ wins in $\mathsf{Sec}_{\mathcal{A}}(\kappa)] = \mathsf{negl}(\kappa)$ for any PPT $\mathcal{A}$.

To construct accumulators, we first construct a special kind of merger called *hash tree merger* that has a hash function H as subroutine. The difference is that now the nodes in the trees also have a string as its value. Let x be a node. For simplicity, we assume that every node x is uniquely identified. We use $\mathsf{Val}(x)$ to denote its value. When merging a list of trees whose roots are $(\mathsf{rt}_1, \ldots, \mathsf{rt}_k)$, the root of the new tree has value $H(\mathsf{Val}(\mathsf{rt}_1), \ldots, \mathsf{Val}(\mathsf{rt}_k))$.

▶ **Definition 43** (Hash tree merger). *Let M be an online merger and H be a hash function. At every round, the* hash tree merger $\mathsf{HMer}(\mathsf{M}, H)$ *first uses M to merge trees. When a set of trees whose roots are $(\mathsf{rt}_1, \ldots, \mathsf{rt}_k)$ are being merged, they will first be ordered according to their unique identifiers. Without loss of generality, we can assume that $\mathsf{rt}_1 < \cdots < \mathsf{rt}_k$. Then, $\mathsf{HMer}(\mathsf{M}, H)$ assigns the root of the new tree the value $H(\mathsf{Val}(\mathsf{rt}_1), \ldots, \mathsf{Val}(\mathsf{rt}_k))$.*

Now, we give the construction of accumulators from collision resistant hash functions, where an online merger M is given as a subroutine. Looking ahead, the auxiliary information aux will be a list of hash trees where the accumulated values are stored in the leaves of the trees. The digest pp is simply the values of all the roots.

▶ **Construction 44.** *Let M be an online merger and $(\mathsf{Gen}, \mathsf{Hash})$ be a collision resistant hash function. Using them, we implement an accumulator according to Definition 40 as follows.*

- $\mathsf{Gen}(1^\kappa) \to \mathsf{k}$: *Sample and output a key $\mathsf{k} \leftarrow \mathsf{Gen}(1^\kappa)$ for the function $(\mathsf{Gen}, \mathsf{Hash})$. Initialize $\mathsf{HMer}(\mathsf{M}, H)$ such that $H(\mathsf{Val}(\mathsf{x}_1), \ldots, \mathsf{Val}(\mathsf{x}_t)) = 1||\mathsf{Hash}(\mathsf{k}, (\mathsf{Val}(\mathsf{x}_1), \ldots, \mathsf{Val}(\mathsf{x}_t)))$ for arbitrary $t \in \mathbb{N}$.*
- $\mathsf{Add}^{[\mathsf{aux}]}(\mathsf{k}, \mathsf{pp}, \mathsf{v}) \to \mathsf{pp}'$: *Parse aux as a list of trees. Add a new tree with a single node whose value is $0||\mathsf{v}$. Let HMer handle the merging. Output the values of the roots of the trees as pp'.*
- $\mathsf{Upd}^{\mathsf{aux}}(\mathsf{pp}, \mathsf{v}) \to \mathsf{w}$: *First parse aux as a list of hash trees and locate one tree containing a leaf of value $0||v$. If no such tree exists, halt. Otherwise, let d be the depth of the tree and $(p_d, \ldots, p_0)$ be the path from p_d, which has value $0||\mathsf{v}$, to the root p_0. Let ℓ_i (resp. r_i) be*

the tuple of values of left (resp. right) siblings of p_i for $i \in [d]$. Note that when computing hashes, we first order the roots according to their identifiers. Thus, it is legitimate to talk about tuples of left siblings and right siblings. Output $\mathsf{w} = ((\ell_d, r_d), \ldots, (\ell_1, r_1))$.

- $\mathsf{Ver}(\mathsf{k}, \mathsf{pp}, \mathsf{v}, \mathsf{w}) \to \mathsf{b}$: *First parse* $\mathsf{pp} = \{\mathsf{rt}_1, \ldots, \mathsf{rt}_k\}$ *and* $\mathsf{w} = ((\ell_d, r_d), \ldots, (\ell_1, r_1))$. *Then, compute* $\mathsf{tmp}_{i-1} = 1||\mathsf{Hash}(\mathsf{k}, (\ell_i, \mathsf{tmp}_i, r_i))$ *for* $i = d, \ldots, 1$, *where* $\mathsf{tmp}_d = 0||\mathsf{v}$. *Output 1 if* $\mathsf{tmp}_0 \in \mathsf{pp}$ *and 0 otherwise.*

Completeness of Construction 44 is straightforward. We now bound the length of digest and number of updates.

▶ **Proposition 45** (Length of digest and number of updates of Construction 44)**.** *In Construction 44, after adding n inputs, the length of digest is bounded by $(\kappa+1) \cdot w_{[n]}$ and the number of update is bounded by* $\mathsf{dep}_{[n]}$.

Proof. The length of digest equals the number of trees times the length of the root, which is bounded by $(\kappa + 1) \cdot w_{[n]}$. An update for an accumulated value is required only when the tree it belongs get merged. Therefore, the number of update is bounded by the depth of the merger which is $\mathsf{dep}_{[n]}$. ◀

We now prove that Construction 44 satisfies the security requirement of Definition 42.

▶ **Proposition 46** (Security of Construction 44)**.** *Construction 44 is a secure accumulator according to Definition 42.*

Proof. It suffices to show that when $\mathcal{A}$ wins $\mathsf{Sec}_{\mathcal{A}}(\kappa)$, a collision for the underlying hash function is found. Let v be the value outputted by $\mathcal{A}$. Note that v can not be one of the trees which has only a single node. Otherwise, it means $\mathcal{A}$ has already added v. Therefore, $\mathcal{A}$ must have also outputted a witness $\mathsf{w} = ((\ell_{d'}, r_{d'}), \ldots, (\ell_1, r_1))$. Let $\mathsf{tmp}_{i-1} = 1||\mathsf{Hash}(\mathsf{k}, (\ell_i, \mathsf{tmp}_i, r_i))$ for $i = d', \ldots, 1$ where $\mathsf{tmp}_{d'} = 0||\mathsf{v}$. Since v is accepted, we know there must exist a tree $\mathcal{T}$ whose root $rt = \mathsf{tmp}_0$. Let d be the depth of $\mathcal{T}$. Let $\{\mathsf{tmp}_k, \ldots, \mathsf{tmp}_0\}$ be the longest sub-path of $\{\mathsf{tmp}_{d'}, \ldots, \mathsf{tmp}_0\}$ that exist in $\mathcal{T}$.

1. Suppose $k = d$ and $k = d'$. This contradicts the assumption that $\mathcal{A}$ wins the game as v must be one of the leaves.
2. Suppose $k = d$ and $k < d'$. Note that tmp_k begins with 1 while the values of all nodes of depth k in $\mathcal{T}$ begins with 0, which means we have found a collision.
3. Suppose $k < d$ and $k = d'$. Note that tmp_k begins with 0 while the values of all nodes of depth k in $\mathcal{T}$ begins with 1, which means we have found a collision.
4. Suppose $k < d$ and $k < d'$. In this case we have also found a collision because $\mathsf{tmp}_{k+1} \neq \mathsf{tmp}'_{k+1}$. ◀

Lower Bounds for Secret-Sharing Schemes for k-Hypergraphs

Amos Beimel
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

Abstract

A secret-sharing scheme enables a dealer, holding a secret string, to distribute shares to parties such that only pre-defined authorized subsets of parties can reconstruct the secret. The collection of authorized sets is called an access structure. There is a huge gap between the best known upper bounds on the share size of a secret-sharing scheme realizing an arbitrary access structure and the best known lower bounds on the size of these shares. For an arbitrary n-party access structure, the best known upper bound on the share size is $2^{O(n)}$. On the other hand, the best known lower bound on the total share size is much smaller, i.e., $\Omega(n^2/\log(n))$ [Csirmaz, *Studia Sci. Math. Hungar.*]. This lower bound was proved more than 25 years ago and no major progress has been made since.

In this paper, we study secret-sharing schemes for k-hypergraphs, i.e., for access structures where all minimal authorized sets are of size exactly k (however, unauthorized sets can be larger). We consider the case where k is small, i.e., constant or at most $\log(n)$. The trivial upper bound for these access structures is $O(n \cdot \binom{n-1}{k-1})$ and this can be slightly improved. If there were efficient secret-sharing schemes for such k-hypergraphs (e.g., 2-hypergraphs or 3-hypergraphs), then we would be able to construct secret-sharing schemes for arbitrary access structures that are better than the best known schemes. Thus, understanding the share size required for k-hypergraphs is important. Prior to our work, the best known lower bound for these access structures was $\Omega(n\log(n))$, which holds already for graphs (i.e., 2-hypergraphs).

We improve this lower bound, proving a lower bound of $\Omega(n^{2-1/(k-1)}/k)$ on the total share size for some explicit k-hypergraphs, where $3 \leq k \leq \log(n)$. For example, for 3-hypergraphs we prove a lower bound of $\Omega(n^{3/2})$. For $\log(n)$-hypergraphs, we prove a lower bound of $\Omega(n^2/\log(n))$, i.e., we show that the lower bound of Csirmaz holds already when all minimal authorized sets are of size $\log(n)$. Our proof is simple and shows that the lower bound of Csirmaz holds for a simple variant of the access structure considered by Csirmaz. Using our results, we prove a near quadratic separation between the required share size for realizing an explicit access structure and the monotone circuit size describing the access structure, i.e., the share size in $\Omega(n^2/\log(n))$ and the monotone circuit size is $O(n\log(n))$ (where the circuit has depth 3).

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Cryptographic primitives

Keywords and phrases Secret Sharing, Share Size, Lower Bounds, Monotone Circuits

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.16

Related Version *Full Version*: https://eprint.iacr.org/2023/289

Funding Research supported by ERC grant 742754 (project NTSC) and Israel Science Foundation grant no. 391/21.

1 Introduction

Secret-sharing schemes are a tool used in many cryptographic protocols. A secret-sharing scheme involves a dealer who has a secret, a set of n parties, and an access structure Γ – a collection of (authorized) subsets of the parties. A secret-sharing scheme for Γ is a method by which the dealer distributes strings (called shares) to the parties such that: (1) any subset in Γ can reconstruct the secret from its shares, and (2) any subset not in Γ cannot reveal any


© Amos Beimel;
licensed under Creative Commons License CC-BY 4.0

partial information on the secret. The share size of a scheme is the maximum share size in the scheme (i.e., the maximum length of the strings representing the shares). Originally motivated by the problem of secure information storage, secret-sharing schemes have found numerous other applications in cryptography, distributed computing, and complexity, e.g., Byzantine agreement [54], secure multiparty computations [13, 24, 26], threshold cryptography [36], access control [51], attribute-based encryption [42, 62], generalized oblivious transfer [56, 61], and proving NP-hardness of the partial minimum circuit size problem [43].

Secret-sharing schemes were introduced by Blakley [16] and Shamir [55] for the threshold case. Secret-sharing schemes for general access structures were introduced and constructed by Ito, Saito, and Nishizeki [44]. More efficient construction for specific families of access structure were given in [14, 57, 20, 45, 15, 17, 37]. For general n-party access structures, the share size in the schemes of [44] is 2^n; for 30 years no schemes with share size better than $2^{n-o(n)}$ were known. Liu and Vaikuntanathan [47], in a breakthrough paper, constructed for every access structure a secret-sharing scheme with share size $2^{0.994n}$. This was improved in a sequence of works [3, 5, 7], where the currently best known scheme has share size $(3/2)^{(1+o(1))n} < 2^{0.585n}$ [7]. The best known lower bound was proved by Csirmaz [28, 29], stating that, for every n, there is an n-party access structure such that sharing ℓ-bit secrets requires that the total share size (i.e., the sum of sizes of the n shares) is $\Omega((n^2/\log(n)) \cdot \ell)$. The question if there exist more efficient schemes, or if there exist access structures that do not have (space) efficient schemes remains open.

In this paper, we consider a natural class of access structure – k-hypergraph access structures, in which the size of the minimal authorized sets is exactly k. Two-hypergraph access structures are called graph access structures and they have been studied extensively, e.g., [21, 22, 23, 38, 18, 30, 35, 31, 32, 10, 40, 33]. k-hypergraph access structures for $k > 2$ have also been studied previously (although not as much as graph secret sharing), e.g., [58, 60, 52, 27, 34, 10, 9]. The naive way to construct a secret-sharing scheme for a k-hypergraph is to share the secret independently for each minimal authorized set; this results in a scheme with total share size $k \cdot \binom{n}{k}$. A result of Erdös and Pyber [39] implies that every n-vertex graph can be realized by a secret-sharing scheme with ℓ-bit secrets and total share size $O((n^2/\log(n))\ell)$ (for secrets of size $\ell \geq \log(n)$). Using this result and Stinson's decomposition technique [59], every n-party k-hypergraph can be realized by a secret-sharing scheme with ℓ-bit secrets and total share size $O((\binom{n}{k}/\log(n)) \cdot \ell)$ (for secrets of size $\ell \geq k^4 \log(n)$ and for $k \leq n/2$) (see Remark 8). In contrast, the best known lower bound on the total share size in secret-sharing realizing a graph with an ℓ-bit secret is $\Omega(n \log(n)\ell)$ [38, 30]. Prior to our work, this was the best known lower bound for k-hypergraphs. Blundo et al. [18] showed a lower bound of $\Omega(n/\log(n) \cdot \ell)$ on the max share size for an access structure in which the size of the minimal authorized sets is *at most* $\log n$. In Table 1, we summarize the known upper bounds and lower bounds on the share size in secret-sharing schemes.

One reason for studying secret-sharing schemes for k-hypergraphs is that they can be used to construct secret-sharing schemes for arbitrary access structures. Every n-party access structure is a union of k-hypergraph access structures, thus, to construct more efficient secret-sharing schemes for arbitrary access structures, it suffices to construct efficient secret-sharing schemes for k-hypergraphs. Moreover, even if we have efficient secret-sharing schemes for k-hypergraphs for a small k, then, as described in the next lemma, for every access structure there is a secret-sharing scheme that is better than the best known secret-sharing schemes (the proof of the lemma for $k = 2$ appears in [53]; for completeness the proof of this lemma appears in Appendix A).

Table 1 Summary of known results on upper and lower bounds on the total share size for secret-sharing schemes.

	Upper bound	Lower bounds
Arbitrary access structures	$O(2^{0.585n})$ [7]	$\Omega(n^2/\log(n))$ [29]
Graph access structures	$O(n^2/\log(n))$ [39]	$\Omega(n\log(n))$ [38, 30]
k-hypergraph access structures for $k \leq \log(n)$	$\binom{n}{k}\frac{k^2}{\log(n)}$	$\Omega(n^{2-1/(k-1)}/k)$ **[This paper]**

▶ **Lemma 1.** *Assume that there exists constants k, c such that every N-party k-hypergraph access structure can be realized by a secret-sharing scheme with total share size $O(N^c)$. Then every n-party access structure can be realized by a secret-sharing scheme with total share size $\tilde{O}(2^{cn/k})$.*

Another reason for studying secret-sharing schemes for k-hypergraphs is that there were no improvements in their share size for more than two decades and the share size in the best known schemes for them is almost as big as the naive scheme for them. This should be compared with the new secret-sharing schemes for arbitrary access structures [47, 3, 5, 7] and the new CDS protocols and secret-sharing schemes for uniform access structures [11, 41, 48, 2, 49, 12, 1, 3]. Furthermore, k-hypergraph access structures resemble k-uniform access structures, in which all sets of size smaller than k are unauthorized, all sets of size larger than k are authorized, and some sets of size k are authorized and some are not. The best known share size for k-uniform access structure is $2^{\tilde{O}(\sqrt{k\log(n)})}$ [49, 3], i.e., it is much smaller than the best known share size for k-hypergraphs. It is interesting to understand if the difference in the share size is inherent.

1.1 Our Results

Our main result is a new lower bound on the share size in secret-sharing schemes for k-hypergraphs.

▶ **Theorem 2** (Informal). *For every n, every $3 \leq k \leq \log(n)$, there is an explicit n-party k-hypergraph access structure such that for every secret length ℓ in every secret-sharing scheme realizing the access structure the total share size is at least $\Omega\left(\frac{n^{2-1/(k-1)}}{k} \cdot \ell\right)$.*

Our lower bound applies to k-partite hypergraph access structures, i.e., access structures in which the parties are partitioned to k parts and each minimal authorized set contains exactly one party from each part. k-partite hypergraph access structures are very useful, i.e., they are used in the proof of Lemma 1. The uniform access structure that are equivalent to CDS protocols are also k-partite.

For $k = 3$, we get a 3-partite hypergraph access structure that requires total share size $\Omega(n^{3/2} \cdot \ell)$. This implies that the applying Lemma 1 with $k = 3$ cannot result in a secret-sharing scheme with share size smaller than $2^{n/2}$. For $k = \log(n)$, we get a $\log(n)$-partite hypergraph access structure that requires total share size $\Omega((n^2/\log(n)) \cdot \ell)$, i.e., the best known lower bound on the share size.

For the interesting case of graph secret-sharing schemes, i.e., $k = 2$, our lower bound is $\Omega(n \cdot \ell)$; this is a trivial lower bound as Karnin et al. [46] proved that the size of the share of any non-redundant party is $\Omega(\ell)$. Improving the lower bound of $\Omega(n \log(n) \cdot \ell)$ for graph secret sharing, or constructing better schemes for graphs, is left as an open question.

We observe that the $\log(n)$-partite access structure for which we prove a lower bound of $O(n^2/\log(n))$ on the total share size can be described by a monotone *circuit* of size $O(n \log(n))$ and depth 3 (where we count the number of wires in the circuit). That is, we prove a near quadratic separation between the required share size and the monotone circuit size. In contrast, the size of the monotone *formula* describing an access structure is an upper bound on the share size required to realize the access structure [14]. Monotone *circuits* describing an access structure imply a computational secret-sharing scheme for the access structure [63];[1] our result raises the question if monotone circuits can be used to construct secret-sharing schemes with information-theoretic security.

To prove Theorem 2, we take the access structure used by Csirmaz in [28, 29] and transform it to a k-hypergraph access structure. The access structures that we construct to prove the lower bounds are quite simple. For example, for $k = 3$, we take two parts D_1, D_2 of size $\sqrt{n}$ and a third part D_3 of size $n - 2\sqrt{n}$; for every $a_1 \in D_1, a_2 \in D_2$ we take a distinct party $c_3 \in D_3$ and add the minimal authorized set $\{a_1, a_2, c_3\}$. See Figure 1 for an illustration of this construction.

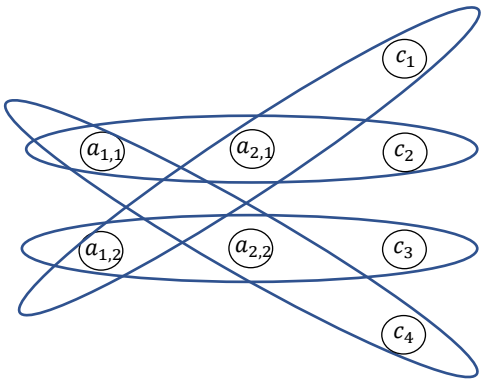


Figure 1 Illustration of the 3-partite hypergraph access structure 3-CSI^8. The parties are $\{a_{1,1}, a_{1,2}, a_{2,1}, a_{2,2}, c_1, c_2, c_3, c_4\}$ and the 4 minimal authorized sets are described by blue circles.

As explained above, the lower bound of Csirmaz [28, 29] of $\Omega((n^2/\log(n)) \cdot \ell)$ on the total size of shares for a some access structure is the best known lower bound on the share size in secret-sharing schemes. This lower bound was used to derive a separation between the size of monotone real formulas and the size of shares in secret-sharing schemes [6] and a separation between the size of shares in information theoretic secret-sharing schemes and the size of shares in computational secret-sharing schemes [4]. Recently, in a work that inspired this work, it was used to prove exponential lower bounds on the size of the shares in *evolving* secret-sharing schemes [50]. The result we use to derive our lower bound (Theorem 10) was generalized by Blundo et al. [18] with the so-called independent sequence method. They constructed, using this method, an access structure in which the size of the minimal authorized sets is *at most* $\log n$ and the maximum share size is $\Omega((n/\log(n)) \cdot \ell)$.

[1] The size of the public information in this scheme is the number of wires in the monotone circuit and the size of each share is the security parameter.

2 Preliminaries

In this section, we define secret-sharing schemes realizing general access structures. We start by defining a secret-sharing scheme, which is a randomized mapping whose input is a string, called the secret, and output is the n strings, called shares.

▶ **Definition 3** (Secret-Sharing Schemes). *Let $\{p_1, \ldots, p_n\}$ be a set of parties. A* secret-sharing scheme Π *with domain of secrets S is a* randomized *mapping from S to a set of n-tuples $S_1 \times S_2 \times \cdots \times S_n$, where S_j is called the* domain of shares *of p_j, that is, given a secret $s \in S$, the secret-sharing scheme outputs the shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$. For a set $A \subseteq \{p_1, \ldots, p_n\}$, we denote $\Pi_A(s)$ as the restriction of $\Pi(s)$ to its A-entries, i.e. $\langle \mathsf{sh}_i \rangle_{p_i \in A}$.*

Informally, in a secret-sharing scheme, we consider a dealer that distributes a secret $s \in S$ according to Π by first sampling a vector of *shares* $\langle \mathsf{sh}_1, \ldots, \mathsf{sh}_n \rangle \leftarrow \Pi(s)$, and privately communicating each share sh_j to party p_j.

▶ **Definition 4** (Access Structures). *Let $\{p_1, \ldots, p_n\}$ be a set of parties. A collection $\Gamma \subseteq 2^{\{p_1,\ldots,p_n\}}$ is* monotone *if $B \in \Gamma$ and $B \subseteq C$ imply that $C \in \Gamma$. An* access structure *is a monotone collection $\Gamma \subseteq 2^{\{p_1,\ldots,p_n\}}$ of non-empty subsets of $\{p_1, \ldots, p_n\}$. Sets in Γ are called* authorized*, and sets not in Γ are called* unauthorized*.*

We next define the correctness and perfect security of a secret-sharing scheme realizing a general access structure; we require that such scheme is secure against an unbounded adversary, i.e., its security is information-theoretic. The definition is based on [25, 8] and does not assume any probability distribution on the secrets.

▶ **Definition 5** (Secret-Sharing Schemes Realizing an Access Structure). *Let S be a finite set of secrets, where $|S| \geq 2$. A secret-sharing scheme Π with domain of secrets S realizes an access structure Γ if the following two requirements hold:*

Perfect Correctness. *The secret s can be reconstructed by any authorized set of parties. That is, for any set $B \in \Gamma$ (where $B = \{p_{i_1}, \ldots, p_{i_{|B|}}\}$), there exists a* reconstruction function $\mathsf{Recon}_B : S_{i_1} \times \cdots \times S_{i_{|B|}} \to S$ *such that for every $s \in S$,*

$$\Pr[\, \mathsf{Recon}_B(\Pi_B(s)) = s \,] = 1. \tag{1}$$

Perfect Security. *Every unauthorized set cannot learn anything about the secret (in the information theoretic sense) from their shares. Formally, for any set $T \notin \Gamma$, for every two secrets $s_1, s_2 \in S$, and for every possible vector of shares $\langle \mathsf{sh}_j \rangle_{p_j \in T}$:*

$$\Pr[\, \Pi_T(s_1) = \langle \mathsf{sh}_j \rangle_{p_j \in T} \,] = \Pr[\, \Pi_T(s_2) = \langle \mathsf{sh}_j \rangle_{p_j \in T} \,], \tag{2}$$

where the probabilities are over the randomness of Π.

The most important complexity measure that we study in secret-sharing schemes in the share size.

▶ **Definition 6** (Share Size). *The size of the secret in a secret-sharing scheme Π with domain of secrets S and domains of shares $S_1, \cdots, S_n$ is $\log(|S|)$, the share size of party p_i is $\log(|S_i|)$, the* max *share size is $\max_{1 \leq j \leq n} \log(|S_j|)$, and the* total *share size is $\sum_{1 \leq j \leq n} \log(|S_j|)$.*

▶ **Definition 7.** *An access structure Γ is a k-hypergraph access structure (also called k-homogeneous access structure) if the size of every minimal authorized set in $A \in \Gamma$ is* exactly *k. An access structure Γ is a k-partite hypergraph access structure if there exists a partition of the set of parties to k sets $D_1, \ldots, D_k$ such that every minimal authorized set $A \in \Gamma$ contains exactly one party from each D_i, that is $|A \cap D_i| = 1$ for every $1 \leq i \leq k$.*

Note that that the size of unauthorized sets in a k-hypergraph access structure can be much larger that k. For example, in a graph access structure (i.e., in a 2-hypergraph access strucure), the minimal authorized sets are the edges of the graph and the unauthorized sets are independent sets.

▶ Remark 8. Erdös and Pyber [39] have proved that every graph can be partitioned into complete bipartite graphs such that each vertex is contained in at most $O(n/\log(n))$ complete bipartite graphs. Blundo et al. [19] observed that this implies that for every n-vertex graph there is a secret-sharing realizing the graph with 1-bit secret, max share size $O(n/\log(n))$, and, in particular, total share size $O(n^2/\log(n))$.

This secret-sharing scheme can be used to construct a secret-sharing scheme for k-hypergraphs as follows. Given a k-hypergraph Γ with a set of parties P, define for every set of parties A of size exactly $k-2$ an access structure $\Gamma_A = \{B \subseteq P \setminus A : A \cup B \in \Gamma\}$. Notice that Γ_A is a graph access structure. We independently share the secret s for each access structure Γ_A, that is, we independently choose $k-2$ random bits $r_1, \ldots, r_{k-2}$, give each bit to a party in A, and share $s \oplus r_1 \oplus \cdots \oplus r_{k-2}$ using a graph secret-sharing for the graph access structure Γ_A. The total share in this scheme is $O\left(\binom{n}{k-2}\frac{n^2}{\log(n)}\right)$; this expression is equal to $O\left(\binom{n}{k}\frac{k^2}{\log(n)}\right)$ for $k \leq n/2$. Observe that each minimal authorized set (of size k) is an authorized set in $\binom{k}{2}$ access structures Γ_A. Thus, we can use Stinson's decomposition [59] to construct a secret-sharing scheme realizing Γ with ℓ-bit secrets, where $\ell > k^4\log(n)$, and total share size $O\left(\binom{n}{k}\frac{1}{\log(n)} \cdot \ell\right)$.

3 Lower Bounds on the Size of the Shares in k-Partite Hypergraph Access Structures

Lower bounds for secret-sharing schemes have been proved in, e.g., [46, 23, 19, 38, 28, 29, 18]. The best lower bound was proved by Csirmaz [28, 29], who proved that for every n there exists an explicit n-party access structure such that every secret-sharing scheme realizing it with an ℓ-bit secret requires total share size $\Omega(n^2/\log(n) \cdot \ell)$. We use this lower bound to prove lower bounds for k-partite hypergraphs. We do this in two stages, we first define in Definition 11 a k-partite access structure in which the max share size is $\Omega\left(n^{1-1/(k-1)}\ell/k\right)$ and then define in Definition 15 a k-partite access structure in which the total share size is $\Omega\left(n^{2-1/(k-1)}\ell/k\right)$.

3.1 A Lower Bound on the Max Share Size

We first define a family of access structures $\mathtt{CSI}$; access structures from this family were used by Csirmaz [28] to prove his lower bound. Each access structure in the family is defined by a given sequence of subsets satisfying the following condition: we say that a sequence of subsets $A_1, A_1, \ldots, A_m$ is valid if $A_i \not\subseteq A_j$ for every $i < j$ (e.g., $|A_i| \geq |A_{i+1}|$ for $1 \leq i \leq m$).

▶ **Definition 9** ([28]). *Let A be a set and $A_1, A_2, \ldots, A_m$ be a valid sequence of subsets of A. Furthermore, let $B = \{b_1, \ldots, b_m\}$ and define $B_i = \{b_1, \ldots, b_i\}$ for $1 \leq i \leq m$. We assume that $A \cap B = \emptyset$. Define the access structure $\mathtt{CSI}^{A_1,\ldots,A_m}$, whose parties are $A \cup B$ and the minimal authorized sets of $\mathtt{CSI}^{A_1,\ldots,A_m}$ are $A_1 \cup B_1, A_2 \cup B_2, \ldots, A_m \cup B_m$.*

▶ **Theorem 10** ([28]). *For every valid sequence of subsets $A_1, A_2, \ldots, A_m$ and every integer $\ell \in \mathbb{N}$, in every secret-sharing scheme realizing $\mathtt{CSI}^{A_1,A_1,\ldots,A_m}$ with domain of secrets $\{0,1\}^\ell$, the total share size of the parties A (i.e., $\sum_{p\in A}|\mathsf{sh}_p|$) is at least $(m-1)\cdot\ell$.*

Csirmaz considered the case in which $A = \{p_1, \ldots, p_k\}$, $m = 2^k$, and $A_1 \ldots, A_m$ is some valid ordering of all subsets of A. In this case the number of parties in the access structure is $n = O(2^k)$ and Theorem 10 implies that there is at least one party whose share size is $\Omega(2^k/|A|) = \Omega(n/\log(n))$. However, Csirmaz's proof applies to any access structure defined for a valid sequence. We will use larger sets A.

The main contribution of this paper is a construction of a k-hypergraph access structure k-$\mathtt{CSI}^n$ from $\mathtt{CSI}$; this access structure requires long shares. An illustration of the 3-$\mathtt{CSI}^8$ access structure appears in Figure 1.

▶ **Definition 11** (The Access Structure k-$\mathtt{CSI}^n$)**.** *Fix k, n and let t be the maximal number such that $(k-1)\cdot t + t^{k-1} \le n$. Let $D_i = \{a_{i,1}, \ldots, a_{i,t}\}$ for $1 \le i \le k-1$, $A = \cup_{i=1}^{k-1} D_i$, $m = t^{k-1}$, and $A_1, \ldots, A_m$ be any ordering of the subsets of A of size $k-1$ that contain exactly one element from each D_i (that is, $|A_j \cap D_i| = 1$ for every $1 \le j \le m, 1 \le i \le k-1$). Finally, let $C = \{c_1, \ldots, c_{n-(k-1)\cdot t}\}$. Define the access structure k-$\mathtt{CSI}^n$, whose parties are $A \cup C$ and the minimal sets of k-$\mathtt{CSI}^n$ are $A_1 \cup \{c_1\}, A_2 \cup \{c_2\}, \ldots, A_m \cup \{c_m\}$.*

Every minimal authorized set in k-$\mathtt{CSI}^n$ contains exactly one party from each part $D_1, \ldots, D_{k-1}, C$, i.e., there is a minimal authorized set $\{a_{1,j_1}, a_{2,j_2}, \ldots, a_{k-1,j_{k-1}}, c_j\}$ for every sequence $(j_1, j_2, \ldots, j_{k-1}) \in [t]^{k-1}$ and the appropriate j.

▶ Remark 12. To define an access structure with exactly n parties, we added the redundant parties $c_{m+1}, \ldots, c_{n-(k-1)\cdot t}$. These parties do not belong to any minimal authorized set and they can be ignored.

▶ **Theorem 13.** *For every n, every $k \le \log(n)$, every $\ell \in \mathbb{N}$, in every secret-sharing scheme realizing the n-party k-hypergraph access structure k-$\mathtt{CSI}^n$ with domain of secrets $\{0,1\}^\ell$, the total share size of the parties in A is $\Omega(n \cdot \ell)$, in particular, there is at least one party with share size $\Omega\left(\frac{n^{1-1/(k-1)}}{k} \cdot \ell\right)$.*

Proof. Consider any secret-sharing scheme Π realizing k-$\mathtt{CSI}^n$ with domain of secrets $\{0,1\}^\ell$. We construct from it a secret-sharing Π' realizing $\mathtt{CSI}^{A_1,\ldots,A_m}$ (where $A_1, \ldots, A_m$ are all the subsets of A that contain exactly one party from each D_i) such that the share of each $a_{i,j}$ is the same in both schemes.

The construction of Π' is as follows:

- Share the secret using the scheme Π. Let $\mathsf{sh}^a_{i,j}$ be the share of $a_{i,j}$ for $1 \le i \le k-1, 1 \le j \le t$ and sh^c_i be the share of c_i for $1 \le i \le m$.
- For $1 \le i \le m$, share sh^c_i using an i-out-of-i secret-sharing scheme. Denote the shares by $\mathsf{sh}^c_{i,j}$ for $1 \le j \le i$.
- The share of $a_{i,j}$ is $\mathsf{sh}^a_{i,j}$ and the share of b_j is $\mathsf{sh}^c_{i,j}$ for $j \le i \le m$.

Any authorized set $A_i \cup B_i \in \mathtt{CSI}^{A_1,\ldots,A_m}$ holds in Π' the shares of A_i and can reconstruct the share sh^c_i, hence the parties in $A_i \cup B_i$ can reconstruct the secret. Next we argue that the scheme Π' is secure. Consider an unauthorized set $T' \notin \mathtt{CSI}^{A_1,\ldots,A_m}$ and let $j \le m$ be the minimal index such that $b_j \notin T'$; if such index does not exist set $j = m+1$. Define $T = (T' \cap A) \cup \{c_1, \ldots, c_{j-1}\}$. Since $T' \notin \mathtt{CSI}^{A_1,\ldots,A_m}$ and $\{b_1, \ldots, b_{j-1}\} \subseteq T'$, it must be that $A_i \not\subseteq T'$ for every $i \le j-1$. This implies that $T \notin k$-$\mathtt{CSI}^n$. By the properties of the i-out-of-i secret-sharing scheme, the parties in T' have no information on sh^c_i for $i \ge j$. I.e., the parties in T' only have the shares of the unauthorized set T in Π and get no information on the secret.

We next analyze the lower bound on the share size that we get. By the choice of the parameters in k-$\texttt{CSI}^n$, we get that $t^{k-1} = m = \Theta(n)$ and $|A| = (k-1)\cdot t = (k-1)\Omega(n^{1/(k-1)})$. By Theorem 10, the total share size of the parties in A in Π', hence also in Π, is $\Omega(m \cdot \ell) = \Omega(n \cdot \ell)$, in particular, there exists a party $p \in A$ whose share size in Π is

$$\Omega\left(\frac{n}{|A|} \cdot \ell\right) = \Omega\left(\frac{n}{(k-1)n^{1/(k-1)}} \cdot \ell\right). \qquad \blacktriangleleft$$

▶ Remark 14. We proved the lower bound by using Theorem 10 as a black-box. An alternative proof for Theorem 13 can directly apply the information inequalities as in the proof of [28].

3.2 A Lower Bound on the Total Share Size

We next construct a k-partite hypergraph access structure k-$\texttt{TotCSI}^n$ that requires large *total* share size. The construction is similar to the construction of Csirmaz [29], who showed how to construct an access structure requiring total share size $\Omega(n^2/\log(n) \cdot \ell)$; to show a small monotone circuit for this access structure, we use a variant of [4] of this construction. Recall that in the access structure k-$\texttt{CSI}^n$ there is a small set A, whose total share size is large. To construct k-$\texttt{TotCSI}^n$, we will take many copies of the access structure k-$\texttt{CSI}^n$ using the same set C, i.e., we only use many copies of the set A. Since the set A is small, the number of parties in k-$\texttt{TotCSI}^n$ will be small. On the other hand, we have many copies of the set A, each copy requires large share size, hence the *total* share size is large. Specifically, we take $\alpha = O(n/(k \cdot t))$ copies of each party in A and for each minimal authorized set $A_j \cup \{c_j\}$ in k-$\texttt{CSI}^n$ we take α^{k-1} minimal authorized sets in k-$\texttt{TotCSI}^n$ by replacing each party a^1_{i,j_i} in A_j by each a^h_{i,j_i}.

▶ **Definition 15** (The Access Structure k-$\texttt{TotCSI}^n$)**.** *Fix k, n, take t as the maximal integer such that $t^{k-1} \leq n/2$, and let $m = t^{k-1}$ and $\alpha = \lfloor n/(2(k-1)\cdot t) \rfloor$. For every $1 \leq h \leq \alpha$, let $D^h_i = \left\{a^h_{i,1}, \ldots, a^h_{i,t}\right\}$ for $1 \leq i \leq k-1$, $A^h = \cup_{i=1}^{k-1} D^h_i$, and $A_1, \ldots, A_m$ be any ordering of the subsets of A^1 of size $k-1$ that contain exactly one element from each D^1_i (that is, $|A_j \cap D^1_i| = 1$ for every $1 \leq j \leq m, 1 \leq i \leq k-1$). Furthermore, let $A = \cup_{1 \leq h \leq \alpha} A^h$ and let $C = \left\{c_1, \ldots, c_{n-|A|}\right\}$. Define the access structure k-$\texttt{TotCSI}^n$, whose parties are $A \cup C$ and for every $1 \leq j \leq m$ we have the following α^{k-1} minimal authorized sets in k-$\texttt{TotCSI}^n$: Let $A_j = \left\{a^1_{1,j_1}, \ldots, a^1_{k-1,j_{k-1}}\right\}$ for a sequence $(j_1, \ldots, j_{k-1}) \in [t]^{k-1}$; for every sequence $h_1, \ldots, h_{k-1} \in [\alpha]^{k-1}$ the set $\left\{a^{h_1}_{1,j_1}, \ldots, a^{h_{k-1}}_{k-1,j_{k-1}}, c_j\right\}$ is a minimal authorized set in k-$\texttt{TotCSI}^n$.*

Note that k-$\texttt{TotCSI}^n$ is a k-partite hypergraph access structure, where the parts are $\cup_{h=1}^{\alpha} D^h_1, \ldots, \cup_{h=1}^{\alpha} D^h_{k-1}, C$. The access structure has $(t \cdot \alpha)^{k-1}$ minimal authorized sets.

▶ **Theorem 16.** *For every n, every $k \leq \log(n)$, every integer $\ell \in \mathbb{N}$, in every secret-sharing scheme realizing the n-party k-hypergraph access structure k-$\texttt{TotCSI}^n$ with domain of secrets $\{0,1\}^\ell$, the total share size is at least $\Omega\left(\frac{n^{2-1/(k-1)}}{k} \cdot \ell\right)$.*

Proof. For every $1 \leq h \leq \alpha$, the access structure k-$\texttt{TotCSI}^n$ restricted to the parties in $A^h \cup \{c_1, \ldots, c_{t^{k-1}}\}$ is isomorphic to k-$\texttt{CSI}^{n'}$, where $n' = (k-1)t + t^{k-1} = \Theta(n)$. Thus, by Theorem 13, in any secret-sharing scheme realizing k-$\texttt{TotCSI}^n$ the total share size of the parties in A^h is $\Omega(n \cdot \ell)$, hence the total share size of the parties in A is

$$\Omega(\alpha n \cdot \ell) = \Omega\left(\frac{n^2}{k \cdot t} \cdot \ell\right) = \Omega\left(\frac{n^2}{k \cdot n^{1/(k-1)}} \cdot \ell\right). \qquad \blacktriangleleft$$

3.3 Secret Sharing vs. Monotone Circuits

We next observe that the access structure $\log(n)$-$\texttt{TotCSI}^n$ can be described by a shallow monotone circuit of size $O(n\log(n))$; that is, we derive an almost quadratic separation between the total share size required in any secret-sharing scheme realizing k-$\texttt{TotCSI}^n$ and the size of the monotone circuit for it.

▶ **Theorem 17.** *The access structure* $\log(n)$-$\texttt{TotCSI}^n$ *can be described by a monotone circuit of size* $O(n\log(n))$ *and depth 3.*

Proof. The access structure $\log(n)$-$\texttt{CSI}^{n'}$, where $t=2$ and $n'=(\log(n)-1)2+2^{\log(n)-1}=\Theta(n)$ has $2^{\log(n)-1}=n/2$ minimal authorized sets of size $\log(n)$, thus it can be described by a monotone CNF formula F of size $O(n\log(n))$;[2] denote the variables of this formula by $\{a_{i,j_i}\}_{i\in[\log(n)-1],j_i\in[t]}\cup\{c_1,\ldots,c_{n/2}\}$. For every $i\in[\log(n)-1], j_i\in[t]$, we compute $\wedge_{h\in[\alpha]}a^h_{i,j_i}$ and connect this AND gate to each leaf in F labeled by a_{i,j_i}. The resulting monotone circuit describes the access structure $\log(n)$-$\texttt{TotCSI}^n$. The size of the circuit is $O(n\log(n)+\alpha(\log(n)-1)2)=O(n\log(n))$ and its depth is 3. ◀

References

1 Benny Applebaum and Barak Arkis. On the power of amortization in secret sharing: d-uniform secret sharing and CDS with constant information rate. In *TCC 2018*, volume 11239 of *LNCS*, pages 317–344, 2018.

2 Benny Applebaum, Barak Arkis, Pavel Raykov, and Prashant Nalini Vasudevan. Conditional disclosure of secrets: Amplification, closure, amortization, lower-bounds, and separations. In *CRYPTO 2017*, volume 10401 of *LNCS*, pages 727–757, 2017.

3 Benny Applebaum, Amos Beimel, Oriol Farràs, Oded Nir, and Naty Peter. Secret-sharing schemes for general and uniform access structures. In *EUROCRYPT 2019*, volume 11478 of *LNCS*, pages 441–471, 2019.

4 Benny Applebaum, Amos Beimel, Yuval Ishai, Eyal Kushilevitz, Tianren Liu, and Vinod Vaikuntanathan. Succinct computational secret sharing. In *55th STOC*, pages 1553–1566, 2023.

5 Benny Applebaum, Amos Beimel, Oded Nir, and Naty Peter. Better secret sharing via robust conditional disclosure of secrets. In *52nd STOC*, pages 280–293, 2020.

6 Benny Applebaum, Amos Beimel, Oded Nir, Naty Peter, and Toniann Pitassi. Secret sharing, slice formulas, and monotone real circuits. In *ITCS 2022*, volume 215 of *LIPIcs*, pages 8:1–8:23, 2022.

7 Benny Applebaum and Oded Nir. Upslices, downslices, and secret-sharing with complexity of 1.5^n. In *CRYPTO 2021*, volume 12827 of *LNCS*, pages 627–655, 2021.

8 Amos Beimel and Benny Chor. Universally ideal secret-sharing schemes. *IEEE Trans. on Information Theory*, 40(3):786–794, 1994.

9 Amos Beimel and Oriol Farràs. The share size of secret-sharing schemes for almost all access structures and graphs. *IACR Cryptol. ePrint Arch.*, 2020:664, 2020. Conference version in TCC 2020, volume 12552 of LNCS, pages 499–529, 2020. URL: `https://eprint.iacr.org/2020/664`.

10 Amos Beimel, Oriol Farràs, and Yuval Mintz. Secret-sharing schemes for very dense graphs. *J. of Cryptology*, 29(2):336–362, 2016.

11 Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In *TCC 2014*, volume 8349 of *LNCS*, pages 317–342, 2014.

[2] It can also be described by a monotone formula of size $O(n)$ and depth $O(\log(n))$.

12 Amos Beimel and Naty Peter. Optimal linear multiparty conditional disclosure of secrets protocols. In *ASIACRYPT 2018*, volume 11274 of *LNCS*, pages 332–362, 2018.

13 Michael Ben-Or, Shaffi Goldwasser, and Avi Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *20th STOC*, pages 1–10, 1988.

14 Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *CRYPTO '88*, volume 403 of *LNCS*, pages 27–35, 1988.

15 Michael Bertilsson and Ingemar Ingemarsson. A construction of practical secret sharing schemes using linear block codes. In *AUSCRYPT '92*, volume 718 of *LNCS*, pages 67–79, 1992.

16 George Robert Blakley. Safeguarding cryptographic keys. In *Proc. of the 1979 AFIPS National Computer Conference*, volume 48, pages 313–317, 1979.

17 George Robert Blakley and Grigory A. Kabatianskii. Linear algebra approach to secret sharing schemes. In *Error Control, Cryptology, and Speech Compression*, volume 829 of *LNCS*, pages 33–40. Springer, 1994.

18 Carlo Blundo, Alfredo De Santis, Roberto De Simone, and Ugo Vaccaro. Tight bounds on the information rate of secret sharing schemes. *Des. Codes Cryptography*, 11(2):107–122, 1997.

19 Carlo Blundo, Alfredo De Santis, Luisa Gargano, and Ugo Vaccaro. On the information rate of secret sharing schemes. *Theoretical Computer Science*, 154(2):283–306, 1996.

20 Ernest F. Brickell. Some ideal secret sharing schemes. *Journal of Combin. Math. and Combin. Comput.*, 6:105–113, 1989.

21 Ernest F. Brickell and Daniel M. Davenport. On the classification of ideal secret sharing schemes. *J. of Cryptology*, 4(73):123–134, 1991.

22 Ernest F. Brickell and Douglas R. Stinson. Some improved bounds on the information rate of perfect secret sharing schemes. *J. of Cryptology*, 5(3):153–166, 1992.

23 Renato M. Capocelli, Alfredo De Santis, Luisa Gargano, and Ugo Vaccaro. On the size of shares for secret sharing schemes. *J. of Cryptology*, 6(3):157–168, 1993.

24 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *20th STOC*, pages 11–19, 1988.

25 Benny Chor and Eyal Kushilevitz. Secret sharing over infinite domains. *J. of Cryptology*, 6(2):87–96, 1993.

26 Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 316–334, 2000.

27 Giovanni Di Crescenzo and Clemente Galdi. Hypergraph decomposition and secret sharing. In *14th ISAAC*, volume 2906 of *LNCS*, pages 645–654, 2003.

28 László Csirmaz. The size of a share must be large. In *EUROCRYPT '94*, volume 950 of *LNCS*, pages 13–22, 1994. doi:10.1007/BFb0053420.

29 László Csirmaz. The dealer's random bits in perfect secret sharing schemes. *Studia Sci. Math. Hungar.*, 32(3–4):429–437, 1996.

30 László Csirmaz. Secret sharing schemes on graphs. Technical Report 2005/059, Cryptology ePrint Archive, 2005.

31 László Csirmaz. An impossibility result on graph secret sharing. *Des. Codes Cryptography*, 53(3):195–209, 2009. doi:10.1007/s10623-009-9304-0.

32 László Csirmaz. Secret sharing on the d-dimensional cube. *Des. Codes Cryptography*, 74(3):719–729, 2015.

33 László Csirmaz and Péter Ligeti. Secret sharing on large girth graphs. *Cryptogr. Commun.*, 11(3):399–410, 2019. doi:10.1007/s12095-018-0338-x.

34 László Csirmaz, Péter Ligeti, and Gábor Tardos. Erdös-pyber theorem for hypergraphs and secret sharing. *Graphs and Combinatorics*, 31(5):1335–1346, 2014.

35 László Csirmaz and Gábor Tardos. Optimal information rate of secret sharing schemes on trees. *IEEE Trans. Inf. Theory*, 59(4):2527–2530, 2013. doi:10.1109/TIT.2012.2236958.

36 Yvo Desmedt and Yair Frankel. Shared generation of authenticators and signatures. In *CRYPTO '91*, volume 576 of *LNCS*, pages 457–469, 1991.

37 Marten van Dijk. A linear construction of perfect secret sharing schemes. In *EUROCRYPT '94*, volume 950 of *LNCS*, pages 23–34, 1995.

38 Marten van Dijk. On the information rate of perfect secret sharing schemes. *Des. Codes Cryptography*, 6(2):143–169, 1995.

39 Paul Erdös and László Pyber. Covering a graph by complete bipartite graphs. *Discrete Mathematics*, 170(1–3):249–251, 1997.

40 Oriol Farràs, Tarik Kaced, Sebastià Martín, and Carles Padró. Improving the linear programming technique in the search for lower bounds in secret sharing. In *EUROCRYPT 2018*, LNCS, pages 597–621, 2018.

41 Romain Gay, Iordanis Kerenidis, and Hoeteck Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption. In *CRYPTO 2015*, volume 9216 of *LNCS*, pages 485–502, 2015.

42 Viput Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *13th CCS*, pages 89–98, 2006.

43 Shuichi Hirahara. NP-hardness of learning programs and partial MCSP. In *63rd FOCS*, pages 968–979, 2022.

44 Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing schemes realizing general access structure. In *Globecom 87*, pages 99–102, 1987. Journal version: Multiple assignment scheme for sharing secret. *J. of Cryptology*, 6(1), 15-20, 1993.

45 Mauricio Karchmer and Avi Wigderson. On span programs. In *8th Structure in Complexity Theory*, pages 102–111, 1993.

46 Ehud D. Karnin, Jonathan W. Greene, and Martin E. Hellman. On secret sharing systems. *IEEE Trans. on Information Theory*, 29(1):35–41, 1983.

47 Tianren Liu and Vinod Vaikuntanathan. Breaking the circuit-size barrier in secret sharing. In *50th STOC*, pages 699–708, 2018.

48 Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Conditional disclosure of secrets via non-linear reconstruction. In *CRYPTO 2017*, volume 10401 of *LNCS*, pages 758–790, 2017.

49 Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Towards breaking the exponential barrier for general secret sharing. In *EUROCRYPT 2018*, volume 10820 of *LNCS*, pages 567–596, 2018.

50 Noam Mazor. A lower bound on the share size in evolving secret sharing. *Electron. Colloquium Comput. Complex.*, TR23-013, 2023. URL: `https://eccc.weizmann.ac.il/report/2023/013`.

51 Moni Naor and Avishai Wool. Access control and signatures via quorum secret sharing. *IEEE Transactions on Parallel and Distributed Systems*, 9(1):909–922, 1998.

52 Carles Padró and Germán Sáez. Lower bounds on the information rate of secret sharing schemes with homogeneous access structure. *Inform. Process. Lett.*, 83(6):345–351, 2002.

53 Naty Peter. *Secret-Sharing Schemes and Conditional Disclosure of Secrets Protocols*. PhD thesis, Ben-Gurion University of the Negev, 2020. URL: `https://primo.bgu.ac.il/permalink/972BGU_INST/23v028/alma9926575584104361`.

54 Michael O. Rabin. Randomized Byzantine generals. In *24th FOCS*, pages 403–409, 1983.

55 Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

56 Bhavani Shankar, Kannan Srinathan, and C. Pandu Rangan. Alternative protocols for generalized oblivious transfer. In *9th ICDCN*, volume 4904 of *LNCS*, pages 304–309, 2008. `doi:10.1007/978-3-540-77444-0_31`.

57 Gustavus J. Simmons, Wen-Ai Jackson, and Keith M. Martin. The geometry of shared secret schemes. *Bulletin of the ICA*, 1:71–88, 1991.

58 Douglas R. Stinson. New general lower bounds on the information rate of secret sharing schemes. In *CRYPTO '92*, volume 740 of *LNCS*, pages 168–182, 1993.

59 Douglas R. Stinson. Decomposition construction for secret sharing schemes. *IEEE Trans. on Information Theory*, 40(1):118–125, 1994.

60 Hung-Min Sun and Shiuh-Pyng Shieh. Constructing perfect secret sharing schemes for general and uniform access structures. *J. Inf. Sci. Eng.*, 15(5):679–689, 1999.

61 Tamir Tassa. Generalized oblivious transfer by secret sharing. *Des. Codes Cryptography*, 58(1):11–21, 2011.

62 Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC 2011*, volume 6571 of *LNCS*, pages 53–70, 2011.

63 Andrew Chi-Chih Yao. Unpublished manuscript, 1989. Presented at Oberwolfach and DIMACS workshops.

A A Secret-Sharing Scheme for an Arbitrary Access Structure from a Secret-Sharing Scheme for k-Hypergraph

We next describe a simple reduction from realizing an arbitrary access structure to realizing k-hypergraphs. Given an access structure Γ with parties $p_1, \ldots, p_n$ we define the following k-hypergraph access structure Γ_k with $k \cdot N$ vertices, where $N = 2^{n/k}$ (for simplicity assume that n/k is an integer):

- Let $P_i = \left\{p_{(i-1)\cdot n/k+1}, \ldots, p_{i\cdot n/k}\right\}$ for $1 \leq i \leq k$, $D_i = 2^{P_i}$, and $D = \cup_{i=1}^{k} D_i$. The parties in Γ_k are D, i.e., each party in Γ_k is a subset of the parties in Γ.[3]
- For every minimal authorized set A in Γ, the set

  $$\{A \cap P_1, \ldots, A \cap P_k\}$$

  is a minimal authorized set in Γ_k.

An illustration of a construction of such access structure for $k = 2$ appears in Figure 2.

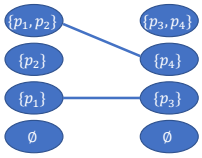


Figure 2 The access structure Γ_2 constructed from the access structure with two minimal authorized sets $\{p_1, p_3\}$ and $\{p_1, p_2, p_4\}$. The minimal authorized sets of Γ_2 are described by an edge.

The secret-sharing Π for Γ is as follows:

- Construct the above hypergraph access structure Γ_k from the access structure Γ.
- Share the secret s using any secret-sharing scheme Π_k for Γ_k. Let sh_C be the share in this scheme of the vertex $C \in D$.
- For every non-empty set $C \in D$, independently share sh_C using a $|C|$-out-of-$|C|$ secret-sharing scheme among the parties of C. In addition, give the shares of $\emptyset \in D_i$ for each $1 \leq i \leq k$ to all parties in Γ.

We next argue the correctness and security of the scheme. First, let $A = A_1 \cup \cdots \cup A_k$ be a minimal authorized set in Γ, where $A_i \subseteq P_i$ for every $1 \leq i \leq k$. By the construction of Γ_k, the set $\{A_1, \ldots, A_k\}$ is an authorized set in Γ_k, thus $\mathsf{sh}_{A_1}, \ldots, \mathsf{sh}_{A_k}$ determine the secret. Furthermore, the parties in A can reconstruct $\mathsf{sh}_{A_1}, \ldots, \mathsf{sh}_{A_k}$, hence, can reconstruct the secret.

[3] There is party for the empty set in each D_i. These are k distinct parties.

For the security of the scheme, consider an unauthorized set $T = T_1 \cup \cdots \cup T_k \notin \Gamma$, where $T_i \subseteq P_i$ for every $1 \leq i \leq k$. Clearly, the parties in T can reconstruct sh_{T_i} for every $1 \leq i \leq k$; however they can also reconstruct the shares of every subset of T_i. On the other hand, for any other set $B \in D$, the parties in T miss at least one party in B. Hence, the parties in T have no information of the shares of these sets.

Since T is unauthorized, every subset of T is unauthorized and for every $T'_1 \subseteq T_1, \ldots, T'_k \subseteq T_k$, the set $\{T'_1, \ldots, T'_k\}$ is unauthorized in Γ_k. Thus, the shares in Π_k that the parties in T can reconstruct are shares of an unauthorized set in Γ_k and these shares do not reveal any information on the secret s.

The total share size of the scheme Π is at most n times the total share size of the scheme Π', i.e., n times the share size required to realize a $(k \cdot 2^{n/k})$-party k-hypergraph.

Differentially Private Aggregation via Imperfect Shuffling

Badih Ghazi
Google Research, Mountain View, CA, USA

Ravi Kumar
Google Research, Mountain View, CA, USA

Pasin Manurangsi
Google Research, Mountain View, CA, USA

Jelani Nelson
University of California at Berkeley, CA, USA
Google Research, Mountain View, CA, USA

Samson Zhou
University of California at Berkeley, CA, USA
Rice University, Houston, TX, USA

Abstract

In this paper, we introduce the imperfect shuffle differential privacy model, where messages sent from users are shuffled in an *almost* uniform manner before being observed by a curator for private aggregation. We then consider the private summation problem. We show that the standard split-and-mix protocol by Ishai et. al. [FOCS 2006] can be adapted to achieve near-optimal utility bounds in the imperfect shuffle model. Specifically, we show that surprisingly, there is no additional error overhead necessary in the imperfect shuffle model.

2012 ACM Subject Classification Security and privacy → Human and societal aspects of security and privacy

Keywords and phrases Differential privacy, private summation, shuffle model

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.17

1 Introduction

Differential privacy (DP) [18] has emerged as a popular concept that mathematically quantifies the privacy of statistics-releasing mechanisms. Consequently, DP mechanisms have been recently deployed in industry [29, 21, 35, 16], as well as by government agencies such as the US Census Bureau [2]. DP is parameterized by ε and δ, where ε is a privacy loss parameter that is generally a small positive constant such as 1 and δ is an approximation parameter or "failure" probability that is typically (smaller than) inverse-polynomial in n:

▶ **Definition 1** (Differential privacy; [18, 17]). *Given $\varepsilon > 0$ and $\delta \in (0,1)$, a randomized algorithm $\mathcal{A} : X \to Y$ is* (ε, δ)-differentially private *if, for every neighboring datasets x and x', and for all $S \subseteq Y$, $\mathbf{Pr}[\mathcal{A}(x) \in S] \leq e^{\varepsilon} \cdot \mathbf{Pr}[\mathcal{A}(x') \in S] + \delta$.*

In this paper, we study the real summation problem, where each of n parties holds a number $x_i \in [0,1]$ for all $i \in [n]$ and the goal is to privately (approximately) compute $\sum_{i=1}^{n} x_i$. Due to its fundamental nature, the private real summation problem has a wide range of applications, such as private distributed mean estimation [40, 10], e.g., in federated learning [33, 27, 31], private stochastic gradient descent [37, 8, 1, 3, 13], databases and information systems [34, 43], and clustering [39, 38].


© Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Jelani Nelson, and Samson Zhou;
licensed under Creative Commons License CC-BY 4.0

In the central model of DP, where a curator is given full access to the raw data in order to release the private statistic or data structure, the Laplace mechanism [18] can achieve, for real summation, additive error $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$, which is known to be nearly optimal for $\varepsilon \leq 1$ [26].

However, the ability for the curator to observe the full data is undesirable in many commercial settings, where the users do not want their raw data to be sent to a central curator. To address this shortcoming, the local model of DP [32, 42] (LDP) demands that all messages sent from an individual user to the curator is private. Unfortunately, although the local model enjoys near-minimal trust assumptions, numerous basic tasks provably must suffer from significantly larger estimation errors compared to their counterparts in the central model. For the real summation problem, [9] achieves additive error $\mathcal{O}_\varepsilon(\sqrt{n})$ and it is known that smaller error bounds cannot be achieved [12].

Consequently, the shuffle model [11, 20, 14] of DP was introduced as an intermediary between the generous central model and the strict local model. In the shuffle model, the messages sent from the users are randomly permuted before being observed by the curator, in an encode-shuffle-analyze architecture. Surprisingly, when users are allowed to send multiple messages, there exist protocols in the shuffle model of DP that achieve additive error $\mathcal{O}_\varepsilon(1)$ for the private real summation problem [25, 6, 24]. Unfortunately, practical applications can lack the ideal settings that provide the full assumptions required by the shuffle model of DP.

1.1 Model and Motivation

We first define a natural generalization of the uniform shuffler that tolerates imperfections. Let Π be the set of permutations on $[n]$. For $\pi, \pi' \in \Pi$, we define $\mathrm{Swap}(\pi, \pi')$ to be the minimum number of coordinate *swaps*[1] that can be applied to π to obtain π'.

▶ **Definition 2** (γ-Imperfect Shuffler). *For a distortion parameter $\gamma > 0$, we say that $\mathcal{S}$ is a γ-imperfect shuffler if, for all $\pi, \pi' \in \Pi$,*

$$\mathbf{Pr}\left[\mathcal{S} = \pi\right] \leq e^{\gamma \cdot \mathrm{Swap}(\pi, \pi')} \mathbf{Pr}\left[\mathcal{S} = \pi'\right].$$

We call an output from such a shuffler a *γ-imperfect shuffle* or a *γ-I-shuffle*, for short. Here, γ represents an upper bound on the multiplicative distortion of the output probabilities of the distributions of the shuffler, i.e., how the distribution deviates from a perfectly symmetric shuffler. For example, $\gamma = 0$ corresponds to a perfectly symmetric shuffler while $\gamma \to \infty$ represents almost no guarantee from the shuffler.

To understand the motivation behind this definition, consider a setting where a number of user devices collect statistics to be sent to an intermediate buffer, which is ultimately sent to a central curator for processing. The devices may choose to perform this collection over different periods of time, so that immediately sending their statistics over to the curator could reveal information about their identity, through the timestamp.

For example, consider a setting where sensors are monitoring traffic in US cities during peak afternoon hours. Then reports that are received earlier in the day by the curator are more likely to correspond to cities that are in the east, while reports that are received later in the day by the curator are more likely to correspond to cities in the west. To mitigate this, the sensors instead could choose a universally fixed hour during the day to broadcast their reports from the previous day, at some random time during the hour.

[1] We say that π' results from an application of a coordinate *swap* on π iff $\pi(i) = \pi'(i)$ on all except two $i \in [n]$.

Specifically, each user $i \in [n]$ could choose a time t_i, say normalized without loss of generality to $t_i \in [0,1]$, and send their messages at time t_i. If the t_i are chosen uniformly at random and this protocol was executed perfectly, it would result in a uniform shuffle of the messages for a buffer that strips both the source information and the exact time of arrival, e.g., [41].[2] However, issues may arise such as different clock skews, where users may not perfectly synchronize the fixed hour during which the messages should be sent, or communication delays, either because an intermediate link has failed or simply because the latency varies across different networks. It is unclear how to model the imperfect shuffle resulting from these issues using the standard shuffle model.

For a better handle on modeling the imperfection, we can assume that each t_i is adversarially chosen in $[0,1]$. Moreover, each message transmission time can now be altered by a random offset from the intended release time, where the offset is drawn, e.g., from a Laplacian distribution. Specifically, each user $i \in [n]$ draws an offset τ_i from the (centered) Laplacian distribution with scale $\frac{2}{\gamma}$, and sends their message at time $t_i + \tau_i$ instead of at time t_i.

In other words, each user $i \in [n]$ sends their message at time $t_i + \tau_i$, which is determined by the two following quantities:

(1) t_i is an arbitrary and possibly adversarially chosen offset due to nature or some other external source, e.g., clock skews, transmission failure, communication delay.

(2) τ_i is an internal source of noise that the protocol can sample from a predetermined distribution to mitigate the negative privacy effects of t_i.

Note that whereas two permutations π, π' on $[n]$ with swap distance one were equally likely to be output by the shuffler, this may now no longer be the case. On the other hand, for fixed $i, j \in [n]$ and conditioning on the values of $\{t_1, \tau_1, \ldots, t_n, \tau_n\} \setminus \{t_i, \tau_i, t_j, \tau_j\}$, we can see that for $a, b \in [n]$, the probability that $t_a + \tau_a \le t_i + \tau_i \le t_{a+1} + \tau_{a+1}$ and $t_b + \tau_b \le t_j + \tau_j \le t_{b+1} + \tau_{b+1}$ is within an e^γ factor of the probability that $t_a + \tau_a \le t_j + \tau_j \le t_{a+1} + \tau_{a+1}$ and $t_b + \tau_b \le t_i + \tau_i \le t_{b+1} + \tau_{b+1}$.

Specifically, let $\mathcal{E}_1$ be the event that $\tau_i \in [t_a + \tau_a - t_i, t_{a+1} + \tau_{a+1} - t_i]$, where τ_i is a (centered) Laplace random variable and scale $\frac{2}{\gamma}$. Similarly, let $\mathcal{E}_2$ be the event that $\tau_j \in [t_b + \tau_b - t_j, t_{b+1} + \tau_{b+1} - t_j]$ where τ_j is a (centered) Laplace random variable and scale $\frac{2}{\gamma}$. Furthermore, let $\mathcal{E}_3$ be the event that $\tau_j \in [t_a + \tau_a - t_j, t_{a+1} + \tau_{a+1} - t_j]$ and $\mathcal{E}_4$ be the event that $\tau_i \in [t_b + \tau_b - t_i, t_{b+1} + \tau_{b+1} - t_i]$. Then by the properties of the Laplace distribution and the assumption that $t_i, t_j \in [0,1]$, we have $\mathbf{Pr}[\mathcal{E}_1 \wedge \mathcal{E}_2] = \mathbf{Pr}[\mathcal{E}_1]\mathbf{Pr}[\mathcal{E}_2] \le (e^{\gamma/2} \cdot \mathbf{Pr}[\mathcal{E}_3])(e^{\gamma/2} \cdot \mathbf{Pr}[\mathcal{E}_4]) = e^\gamma \cdot \mathbf{Pr}[\mathcal{E}_3 \wedge \mathcal{E}_4]$. Thus, the resulting distribution over permutations is captured by the γ-I-shuffle model.

We can naturally generalize this setting to the model where each user sends m messages, e.g., m buffers collect messages from n users, which results in times $\{t_{i,j}\}_{i\in[n],j\in[m]}$ and offsets $\{\tau_{i,j}\}_{i\in[n],j\in[m]}$. Formally, for m rounds of messages for the n users, $\{m_{i,j}\}_{i\in[n],j\in[m]}$, a separate permutation π_j drawn from a γ-imperfect shuffler is used to shuffle the messages $\{m_{i,j}\}_{i\in[n]}$, for each $j \in [m]$. For example, $\{m_{i,1}\}_{i\in[n]}$ is shuffled according to a permutation π_1 drawn from a γ-imperfect shuffler, $\{m_{i,2}\}_{i\in[n]}$ is shuffled according to an independent permutation π_2 drawn from the same γ-imperfect shuffler, and so on and so forth.

[2] We assume in this example that the buffer can queue the messages, and then forward them to the analyst at some point of time, but that it cannot further shuffle them. The (imperfect) shuffling we consider stems solely from the randomization of the transmission time of the messages by the users.

We remark that the above model is sometimes referred to as the *m-parallel shuffling* model; another model that has been considered in literature is one where all the mn messages are shuffled together using a single shuffler. We only focus on the former in this paper. It remains an interesting open question whether our results can be extended to the latter model.

1.2 Our Contributions

Surprisingly, we present a protocol for the real summation problem that matches the utility bounds of the best protocols in the shuffle model. Thus, we show that there is no additional error overhead necessary in the γ-I-shuffle model, i.e., there is no utility loss due to the imperfect shuffler.

▶ **Theorem 3.** *Let $n \geq 19$ and $\gamma \leq \frac{\log\log n}{80}$ be a distortion parameter. Then there exists an (ε, δ)-DP protocol for summation in the γ-I-shuffle model with expected absolute error $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ and $m = \mathcal{O}\left(e^{4\gamma} + \frac{e^{4\gamma}(\log\frac{1}{\delta} + \log n)}{\log n}\right)$ messages per party. Each message uses $\mathcal{O}(\log q)$ bits, for $q = \lceil 2n^{3/2} \rceil$.*

Observe that when δ is inverse-polynomial in n and the distortion parameter γ is a constant $\mathcal{O}(1)$, then the number of messages m sent by each player in Theorem 3 is a constant. Moreover, under these settings, Theorem 3 recovers the guarantees in the standard shuffle model from [6, 25], though we remark that more communication efficient protocols [24] are known in the standard shuffle model across more general settings. Regardless, we again emphasize that the privacy and utility guarantees of the protocol are independent of the distortion parameter γ.

1.3 Overview of our Techniques

In this section, we describe both our protocol for private real summation in the γ-I-shuffle model and the corresponding analysis for correctness and privacy.

A natural starting point is the recent framework by [44, 45], which achieves amplification of privacy using *differentially oblivious* (DO) shufflers that nearly match amplification of privacy results using fully anonymous shufflers [20, 4, 14, 22]. Unfortunately, the framework crucially uses LDP protocols, which are known to not give optimal bounds even with fully anonymous shufflers. For instance, [6, 14, 5] showed that any single-message shuffled protocol for summation based on LDP protocols must exhibit mean squared error $\Omega(n^{1/3})$ or absolute error $\Omega(n^{1/6})$.

Another natural approach is to adapt recent works for private real summation in the shuffle model, e.g., [25, 24]. One challenge in generalizing these proofs is that they often leverage the fully anonymous shuffler by analyzing a random sample from an alternate view of the output of the local randomizers, which often have some algebraic or combinatorial interpretation that facilitates the proof of specific desirable properties. However, these properties often seem substantially more difficult to prove once the symmetry of the fully anonymous shuffler is lost. In fact, we do not even know the mass that the γ-imperfect shuffler places on each permutation.

From private real summation to statistical security of summation on fixed fields. We first use an observation from [6] that reduces the problem of private real summation to the problem of private summation on a fixed field of size q, so that each user has an input $x_i \in \mathbb{F}_q$ for all $i \in [n]$. We then consider the well-known split-and-mix protocol [30], where

each user i outputs a set of m messages $x_{i,1}, \ldots, x_{i,m} \in \mathbb{F}_q$ uniformly at random conditioned on $x_{i,1} + \ldots + x_{i,m} = x_i$. For the private summation on a fixed field problem, we adapt a well-known reduction [6] for the split-and-mix protocol in the shuffle DP model to the notion of statistical security in the γ-I-shuffle model. Statistical security demands small total variation between the output of a protocol on input x and input x', if $\sum_{i=1}^n x_i = \sum_{i=1}^n x'_i$. In other words, it suffices to show that the output distribution looks "similar" for two inputs with the same sum. See Definition 5 for a formal definition of statistical security.

To show statistical security, we first upper-bound the total variation distance in terms of the probability that two independent instances of the same protocol with the *same* input give the same output. Balle et al. [6] use a similar approach, but then utilizes the symmetry of the fully anonymous shuffler to further upper-bound this quantity in terms of the probability that $\vec{\mathcal{R}}(\vec{\mathsf{X}}) = \mathcal{S} \circ \vec{\mathcal{R}}'(\vec{\mathsf{X}})$, where $\vec{\mathsf{X}} = (x_1, \ldots, x_n)$ is the input vector, $\vec{\mathcal{R}}$ and $\vec{\mathcal{R}}'$ are independent instances of the local randomizer, and $\mathcal{S}$ is an instance of the uniform shuffler. We do not have access to such symmetries in the γ-I-shuffle model or even explicit probabilities that the γ-imperfect shuffler places on each permutation.

Connected components of a communication graph. Instead, we first upper-bound the total variation distance by $\vec{\mathcal{R}}(\vec{\mathsf{X}}) = \mathcal{S}^{-1} \circ \mathcal{S}' \circ \vec{\mathcal{R}}'(\vec{\mathsf{X}})$, where $\mathcal{S}^{-1}$ is the inverse of an instance of a γ-imperfect shuffle and $\mathcal{S}'$ is an independent instance of the same γ-imperfect shuffle. Intuitively, $\vec{\mathcal{R}}(\vec{\mathsf{X}})$ and $\mathcal{S}^{-1} \circ \mathcal{S}' \circ \vec{\mathcal{R}}'(\vec{\mathsf{X}})$ can look very different if there exists a large number of users whose messages are not shuffled with those of other users. Formally, this can be captured by looking at the number of connected components in the communication graph of $\mathcal{S}^{-1} \circ \mathcal{S}' \circ \vec{\mathcal{R}}'(\vec{\mathsf{X}})$, so that there exists an edge connecting users i and j if the protocol swaps one of their messages. Hence, evaluating the number of connected components in the communication graph is closely related to analyzing the probability that there is no edge between S and $[n] \setminus S$, for a given set $S \subseteq [n]$.

Although this quantity would be somewhat straightforward to evaluate for a uniform shuffler [6], it seems more challenging to evaluate for γ-imperfect shufflers, since we do not have explicit probabilities for each permutation. Therefore, we develop a novel coupling argument to relate the probability that there is no edge between S and $[n] \setminus S$ in the γ-I-shuffle model to the probability of this event in the shuffle model. In particular, a specific technical challenge that our argument handles is when both S and $[n] \setminus S$ has large cardinality, because then there can be a permutation π that swaps many coordinates while still leaving S and $[n] \setminus S$ disconnected. However, if we simply relate the probability of Π in the shuffle and the γ-I-shuffle model, we incur a gap of $e^{t \cdot \gamma}$, where γ is the distortion parameter and t is the number of swaps by Π, which can have size $\Omega(n)$. Thus without additional care, this gap can overwhelm the probability achieved from the coupling argument. We circumvent this issue by considering a subset of S with size k and coupling the "good" permutations in the shuffle and the γ-I-shuffle model, which results in a smaller gap of $e^{k \cdot \gamma}$. For more details, see Lemma 26.

Putting things together. At this point, we are almost done. Unfortunately, our coupling only addresses the case where a single imperfect shuffle is performed on a local randomizer, but we require the bound for the composition $\mathcal{S}^{-1} \circ \mathcal{S}' \circ \vec{\mathcal{R}}'(\vec{\mathsf{X}})$, which seems significantly more challenging because communication between users i and j under $\mathcal{S}'$ may be "erased" by $\mathcal{S}^{-1}$. Instead, we show a simple observation for γ-imperfect shuffling, which states that if $\mathcal{S}, \mathcal{S}'$ are two shufflers such that $\mathcal{S}$ is a γ-imperfect shuffler, then $\mathcal{S}' \circ \mathcal{S}$ is a γ-imperfect shuffler. This statement, presented in Lemma 23, can be considered as a post-processing preservation

property of γ-imperfect shuffling. In light of this statement, we can now view $\mathcal{S}^{-1} \circ \mathcal{S}' \circ \vec{\mathcal{R}}'(\vec{\mathsf{X}})$ as a single γ-imperfect shuffler applied to the local randomizer, and use our new results upper-bounding the number of connected components in the resulting communication graph to ultimately show σ-security.

1.4 Preliminaries

For an integer $n > 0$, we define $[n] := \{1, \ldots, n\}$.

▶ **Definition 4** (Total variation distance). *Given probability measures μ, ν on a domain Ω, their* total variation distance *is defined by* $\mathrm{TVD}(\mu, \nu) = \frac{1}{2}\|\mu - \nu\|_1 = \frac{1}{2}\sum_{x \in \Omega} |\mu(x) - \nu(x)|$.

▶ **Definition 5** (σ-security). *Given a security parameter $\sigma > 0$, a protocol $\mathcal{P}$ is* σ-secure *for computing a function $f : \mathcal{X}^n \to Z$ if, for any $x, x' \in \mathcal{X}^n$ such that $f(x) = f(x')$, we have* $\mathrm{TVD}(\mathcal{P}(x), \mathcal{P}(x')) \leq 2^{-\sigma}$.

Recall the following two well-known properties of DP:

▶ **Theorem 6** (Basic Composition, e.g., [19]). *Let $\varepsilon, \delta \geq 0$. Any mechanism that permits k adaptive interactions with mechanisms that preserve (ε, δ)-DP is $(k\varepsilon, k\delta)$-DP.*

▶ **Theorem 7** (Post-processing [19]). *Let $\mathcal{M} : \mathcal{U}^* \to X$ be an (ε, δ)-DP algorithm. Then, for any arbitrary random mapping $g : X \to X'$, we have that $g(\mathcal{M}(x))$ is (ε, δ)-DP.*

We use $\mathsf{Ber}(p)$ to denote the Bernoulli distribution with parameter p and use $\mathsf{DLap}(\alpha)$ to denote the discrete Laplace distribution, so that $Z \sim \mathsf{DLap}(\alpha)$ has the probability mass function $\mathbf{Pr}[Z = k] \propto \alpha^{|k|}$ for $k \in \mathbb{Z}$. We use $\mathsf{Polya}(r, p)$ to denote the Polya distribution with parameter $r > 0, p \in (0, 1)$, which induces the probability density function $k \mapsto \binom{k+r-1}{k} p^k (1-p)^r$ for $k \in \mathbb{Z}_{\geq 0}$. We require the following equivalence between a discrete Laplacian random variable and the sum of a differences of Polya random variables.

▶ **Fact 8.** *Let $x_1, \ldots, x_n, y_1, \ldots, y_n \sim \mathsf{Polya}\left(\frac{1}{n}, \alpha\right)$. Then $z = \sum_{i=1}^n (x_i - y_i) \sim \mathsf{DLap}(\alpha)$.*

We also require the following property about randomized rounding.

▶ **Lemma 9** ([4]). *Given a precision $p \geq 1$, let $x_1, \ldots, x_n \in \mathbb{R}$ and $y_i = \lfloor x_i p \rfloor + \mathsf{Ber}(x_i p - \lfloor x_i p \rfloor)$ for each $i \in [n]$. Then $\mathbb{E}\left[\left(\sum_{i=1}^n \left(x_i - \frac{y_i}{p}\right)\right)^2\right] \leq \frac{n}{4p^2}$.*

1.5 Related Work

To amplify the privacy, the trusted shuffler is the key component of the shuffle model, which in some sense only shifts the point of vulnerability from the curator to the shuffler, particularly in the case where the shuffler may be colluding with the curator. Hence among the various relaxations for distributed DP protocols, e.g., [7, 15], the DO shuffle model has been recently proposed [36, 28] to permit some privacy leakage in the shuffling stage, called a DO shuffle. In fact, [36, 28] showed that DO shuffling can be more efficient to achieve than a fully anonymous shuffle while [44, 45] showed that locally private protocols can be used in conjunction with a DO shuffler to achieve almost the same privacy amplification bounds as with a fully anonymous shuffler, up to a small additive loss resulting from the DO shuffle. However, the best known results in the shuffle model of DP do not utilize LDP protocols, and thus cannot directly be applied in the framework of [44, 45].

2 A Simple Reduction

In this section, we briefly describe a simple reduction for showing amplification of privacy for imperfect shuffling. The result can be viewed as in the same spirit as similar privacy amplification statements, e.g., [22, 44, 23], but for imperfect shuffling. In particular, the following well-known result achieves privacy amplification for local randomizers in the shuffle model:

▶ **Theorem 10** ([22]). *For any domain $\mathcal{D}$ and $i \in [n]$, let $\mathcal{R}^{(i)} : \mathcal{S}^{(1)} \times \ldots \times \mathcal{S}^{(i-1)} \times \mathcal{D} \to \mathcal{S}^{(i)}$, where $\mathcal{S}^{(i)}$ is the range space of $\mathcal{R}^{(i)}$, such that $\mathcal{R}^{(i)}(z_{1:i-1}, \cdot)$ is an ε_0-DP local randomizer for all values of auxiliary inputs $z_{1:i-1} \in \mathcal{S}^{(1)} \times \ldots \times \mathcal{S}^{(i-1)}$. Let $\mathcal{A}_s : \mathcal{D}^n \to \mathcal{S}^{(1)} \times \ldots \times \mathcal{S}^{(n)}$ be the algorithm that given a dataset $x_{1:n} \in \mathcal{D}^n$, samples a uniform random permutation π over $[n]$ and sequentially computes $z_i = \mathcal{R}^{(i)}(z_{1:i-1}, x_{\pi(i)})$ for $i \in [n]$ and outputs $z_{1:n}$. Then for any $\delta \in [0,1]$ such that $\varepsilon_0 \leq \log\left(\frac{n}{16\log(2/\delta)}\right)$, $\mathcal{A}_s$ is (ε, δ)-DP for $\varepsilon \leq \log\left(1 + \frac{e^{\varepsilon_0}-1}{e^{\varepsilon_0}+1}\left(\frac{8\sqrt{e^{\varepsilon_0}\log(4/\delta)}}{\sqrt{n}}\right)\right)$.*

We would like to show privacy amplification statements for the imperfect shuffle model that are qualitatively similar to Theorem 10. To that end, we first recall the following definition of DO shufflers.

▶ **Definition 11** (Differentially Oblivious Shuffle). *A shuffle protocol is (ε, δ)-*differentially oblivious *if for all adversaries $\mathcal{V}$, all $\pi, \pi' \in \Pi$, and all subsets S of the view space, $\mathbf{Pr}\left[\mathsf{View}^{\mathcal{V}}(\pi) \in S\right] \leq e^{\varepsilon \cdot \mathrm{Swap}(\pi,\pi')}\mathbf{Pr}\left[\mathsf{View}^{\mathcal{V}}(\pi') \in S\right] + \delta$.*

Zhou and Shi [44] showed that DO shufflers also amplify privacy.

▶ **Theorem 12** (Theorem 1 in [44]). *For any domain $\mathcal{D}$ and range space $\mathcal{S}$, $i \in [n]$, let $\mathcal{R}^{(1)}, \ldots, \mathcal{R}^{(n)} : \mathcal{D} \to \mathcal{S}$ be ε_0-DP local randomizers and let $\mathcal{A}_s$ be a $(\varepsilon_1, \delta_1)$-DO shuffler. Then the composed protocol $\mathcal{A}_s(\mathcal{R}^{(1)}, \ldots, \mathcal{R}^{(n)})$ is $(\varepsilon + \varepsilon_1, \delta + \delta_1)$-DP for $\varepsilon = \mathcal{O}\left(\frac{(1-e^{\varepsilon_0})e^{\varepsilon_0/2}\sqrt{\log(1/\delta)}}{\sqrt{n}}\right)$.*

It turns out that imperfect shufflers can be parametrized by DO shufflers, i.e., imperfect shufflers are a specific form of DO shufflers. Therefore, we can immediately apply the previous statement to obtain the following statement for privacy amplification for imperfect shufflers.

▶ **Theorem 13.** *For any domain $\mathcal{D}$ and range space $\mathcal{S}$, $i \in [n]$, let $\mathcal{R}^{(1)}, \ldots, \mathcal{R}^{(n)} : \mathcal{D} \to \mathcal{S}$ be ε_0-DP local randomizers and let $\mathcal{A}_s$ be a γ-imperfect shuffler. Then the composed protocol $\mathcal{A}_s(\mathcal{R}^{(1)}, \ldots, \mathcal{R}^{(n)})$ is $(\varepsilon + \gamma, \delta)$-DP for $\varepsilon = \mathcal{O}\left(\frac{(1-e^{\varepsilon_0})e^{\varepsilon_0/2}\sqrt{\log(1/\delta)}}{\sqrt{n}}\right)$.*

Proof. By the definition of γ-imperfect shuffle, we have that for all $\pi, \pi' \in \Pi$, $\mathbf{Pr}[\mathcal{S} = \pi] \leq e^{\gamma \cdot \mathrm{Swap}(\pi,\pi')}\mathbf{Pr}[\mathcal{S} = \pi']$. Since no additional information is leaked by the shuffler, then for all adversaries $\mathcal{V}$ and all subsets S of the view space,

$$\mathbf{Pr}\left[\mathsf{View}^{\mathcal{V}}(\pi) \in S\right] \leq e^{\gamma \cdot \mathrm{Swap}(\pi,\pi')}\mathbf{Pr}\left[\mathsf{View}^{\mathcal{V}}(\pi') \in S\right].$$

In other words, the γ-imperfect shuffler is a $(\gamma, 0)$-DO shuffler. Thus by Theorem 12, the composed protocol $\mathcal{A}_s(\mathcal{R}^{(1)}, \ldots, \mathcal{R}^{(n)})$ is $(\varepsilon+\gamma, \delta)$-DP for $\varepsilon = \mathcal{O}\left(\frac{(1-e^{\varepsilon_0})e^{\varepsilon_0/2}\sqrt{\log(1/\delta)}}{\sqrt{n}}\right)$. ◀

3 Differentially Private Summation

In this section, we first introduce the structural statements necessary to argue privacy for the standard split-and-mix protocol [30]. We then assume correctness of these statements, deferring their proofs to subsequent sections, and we prove the guarantees of Theorem 3. We also give an application to private vector aggregation as a simple corollary of Theorem 3.

We first relate DP protocols for summation under a γ-imperfect shuffler to σ-secure protocols. Lemma 4.1 in [4] showed this relationship for uniform shufflers. It turns out their proof extends to γ-imperfect shufflers as well.

▶ **Lemma 14** (Lemma 4.1 in [4]). *Given a σ-secure protocol in the γ-I-shuffle model for n-party private summation on $\mathbb{Z}_q$ such that each player sends $f(n, q, \sigma)$ bits of messages, there exists an $(\varepsilon, (1 + e^{\varepsilon})2^{-\sigma-1})$-DP protocol for any $\varepsilon \leq O(1)$ in the γ-I-shuffle model for n-party private summation on real numbers with expected absolute error $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ such that each player sends $f(n, O(n^{3/2}), \sigma)$ bits of messages.*

In Section 4, we prove the following guarantees about the split-and-mix protocol from [30].

▶ **Theorem 15.** *Let $n \geq 19$ and $\gamma \leq \frac{\log\log n}{80}$ be a distortion parameter. For worst-case statistical security with parameter σ, it suffices to use $m = \mathcal{O}\left(e^{4\gamma} + \frac{e^{4\gamma}(\sigma + \log n)}{\log n}\right)$ messages, where each message has $\mathcal{O}(\log q)$ bits, for $q = \lceil 2n^{3/2} \rceil$.*

By Lemma 14 and Theorem 15, we have our main statement:

▶ **Theorem 3.** *Let $n \geq 19$ and $\gamma \leq \frac{\log\log n}{80}$ be a distortion parameter. Then there exists an (ε, δ)-DP protocol for summation in the γ-I-shuffle model with expected absolute error $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ and $m = \mathcal{O}\left(e^{4\gamma} + \frac{e^{4\gamma}(\log\frac{1}{\delta} + \log n)}{\log n}\right)$ messages per party. Each message uses $\mathcal{O}(\log q)$ bits, for $q = \lceil 2n^{3/2} \rceil$.*

Applications to private vector summation. An immediate application of our results is to the problem of private vector aggregation, where n parties have vectors $\vec{x_1}, \ldots, \vec{x_n} \in [0, 1]^d$ and the goal is to privately compute $\vec{X} = \sum_{i=1}^{n} \vec{x_i} \in \mathbb{R}^d$. Given a protocol $\mathcal{P}$ for private summation where n players each send m messages, the n players can perform a protocol $\mathcal{P}'$ for vector aggregation by performing $\mathcal{P}$ on each of their d coordinates. In particular, the n players can first perform $\mathcal{P}$ on the first coordinate of their vectors, then perform $\mathcal{P}$ on the second coordinate of their vectors, and so on and so forth, by sending md messages in total. Equivalently, the n players can perform $\mathcal{P}$ on a field of size q^d rather than size q and just send m messages in total. However, the total communication size is still the same, because each message increases by a factor of d due to the larger field size. Thus we consider the approach where the n players perform $\mathcal{P}$ on each of the d coordinates.

To argue privacy, we observe that the n players run d iterations of the protocol $\mathcal{P}$, once for each of the coordinates. By composition of DP, i.e., Theorem 6, to guarantee ε-privacy for the overall protocol, it suffices to run each of the d iterations with privacy $\varepsilon' = \frac{\varepsilon}{d}$ and failure probability $\delta' = \frac{\delta}{d}$. By post-processing of DP, i.e., Theorem 7, the resulting vector where each coordinate is computed using the corresponding protocol is (ε, δ)-DP.

Then as a corollary to Theorem 3 with privacy parameter $\varepsilon' = \frac{\varepsilon}{d}$ and failure probability $\delta' = \frac{\delta}{d}$, we obtain:

▶ **Theorem 16.** *Let $n \geq 19$, $d \geq 1$, $\varepsilon > 0$ be a (constant) privacy parameter, and $\gamma \leq \frac{\log\log n}{80}$ be a distortion parameter. Then there exists an (ε, δ)-DP protocol for vector summation in the γ-I-shuffle model with expected absolute error $\mathcal{O}\left(\frac{d}{\varepsilon}\right)$ per coordinate and $m = \mathcal{O}\left(d\left(e^{4\gamma} + \frac{e^{4\gamma}(\log\frac{d}{\delta} + \log n)}{\log n}\right)\right)$ messages per party. Each message uses $\mathcal{O}(\log q)$ bits, for $q = \lceil 2n^{3/2} \rceil$.*

4 Security of Split-and-Mix Protocol

In this section, we prove the σ-security of the split-and-mix protocol. The proof largely attempts to follow the outline of the split-and-mix protocol analysis for private aggregation by [4], which first reduces from worst-case input to average-case input and then analyzes the connectivity of the resulting communication graph induced by a uniform shuffle.

We similarly first reduce from worst-case input to average-case input and then analyze the connectivity of the resulting communication graph induced by a uniform shuffle. The former appears in Section 4.1 and the latter appears in Section 4.2.

However, the main challenge is that the symmetric properties of the uniform shuffler is often crucially utilized in various steps of the approach. Unfortunately, these properties do not often seem to translate to γ-imperfect shufflers, where we might not even know the mass that is placed on each permutation. Thus we need to handle a number of technical challenges to recover qualitatively similar structural properties to the uniform shuffling model. Along the way, we show that the composition of two shufflers, where the inner shuffler is a γ-imperfect shuffler, is also a γ-imperfect shuffler with the same parameter, which can be interpreted as a post-processing statement for γ-imperfect shuffling.

We first formally define the split-and-mix protocol:

▶ **Definition 17** (Split-and-Mix Protocol, e.g., [30]). *Given an integer parameter $m \geq 1$, the m-message n-player* split-and-mix protocol $\mathcal{P}_{m,n}$ *is defined as follows. Each player i outputs a set of m messages $x_{i,1}, \ldots, x_{i,m}$ uniformly at random conditioned on $x_{i,1} + \ldots + x_{i,m} = x_i$. For each $j \in [m]$, the set of messages $x_{1,j}, \ldots, x_{n,j}$ are then swapped according to a γ-imperfect shuffler $\mathcal{S}^{(j)}$.*

4.1 Worst-case to Average-case Reduction

In this section, we show a reduction from worst-case input to average-case input. In other words, rather than analyze the split-and-mix protocol over the worst-case input, we show it suffices to analyze the expected performance of the split-and-mix protocol across all possible inputs. The approach is nearly identical to that of [6], but they can further simplify their final expression due to the symmetric properties of the uniform shuffler, which do not hold for the γ-imperfect shuffler.

Let $\mathcal{P}_{m,n}$ denote the m-message n-player split-and-mix protocol and let $\tilde{\mathcal{P}}_{m,n}$ be defined as follows. Each player i outputs a set of $m+1$ messages $x_{i,1}, \ldots, x_{i,m+1}$ uniformly at random conditioned on $x_{i,1} + \ldots + x_{i,m+1} = x_i$. For each $i \in [n]$, we use the notation $\mathcal{R}_m(x_i) = (x_{i,1}, \ldots, x_{i,m})$ to denote the choice of the m messages for player i. Let $\mathbb{G} = \mathbb{F}_q$ and for $j \in [m]$, let $\mathcal{S}^{(j)} : \mathbb{G}^n \to \mathbb{G}^n$ be independent shufflers. Then the output of $\tilde{\mathcal{P}}_{m,n}$ is $\mathcal{S}^{(j)}$ applied to the first m messages of each player, concatenated with the unshuffled final message of each player, i.e., $\tilde{\mathcal{P}}_{m,n}(x_1, \ldots, x_n) = \mathcal{S}^{(1)}(x_{1,1}, \ldots, x_{n,1}) \circ \ldots \circ \mathcal{S}^{(m)}(x_{1,m}, \ldots, x_{n,m}) \circ x_{1,m+1}, \ldots, x_{n,m+1}$.

We first reduce the problem to average-case statistical security using the approach of Lemma 6.1 in [6]. Formally, we say that a protocol $\mathcal{P}_{m,n}$ provides average-case statistical security with parameter σ if $\mathbb{E}_{\vec{\mathsf{X}},\vec{\mathsf{X}}'}[\mathrm{TVD}_{|\vec{\mathsf{X}},\vec{\mathsf{X}}'}(\mathcal{P}_{m,n}(\vec{\mathsf{X}}), \mathcal{P}_{m,n}(\vec{\mathsf{X}}'))] \leq 2^{-\sigma}$, where $\vec{\mathsf{X}}$ and $\vec{\mathsf{X}}'$ are

each drawn uniformly at random from all pairs of vectors in $\mathbb{G}^n$ with the same sum. Here we use the notation $\mathrm{TVD}_{|\vec{\mathsf{X}},\vec{\mathsf{X}}'}$ to denote the total variation distance between two distributions conditioned on fixings of $\vec{\mathsf{X}}$ and $\vec{\mathsf{X}}'$.

▶ **Lemma 18.** *Suppose $\mathcal{P}_{m,n}$ provides average-case statistical security with parameter σ, then $\mathcal{P}_{m+1,n}$ and $\tilde{\mathcal{P}}_{m,n}$ provide worst-case statistical security with parameter σ.*

Proof. Let $\vec{x}$ and $\vec{x'}$ be a pair of vectors in $\mathbb{G}^n$ with the same sum. Given an output of $\tilde{\mathcal{P}}_{m,n}(\vec{x})$, the protocol $\mathcal{P}_{m+1,n}(\vec{x})$ can be simulated by using an additional application of $\mathcal{R}_{m+1}$ to randomly permute the last message of each of the players according to the distribution of the γ-imperfect shuffle. Hence, $\mathrm{TVD}(\mathcal{P}_{m+1,n}(\vec{x}), \mathcal{P}_{m+1,n}(\vec{x'})) \leq \mathrm{TVD}(\tilde{\mathcal{P}}_{m,n}(\vec{x}), \tilde{\mathcal{P}}_{m,n}(\vec{x'}))$. It thus suffices to upper bound the worst-case statistical security of $\tilde{\mathcal{P}}_{m,n}$ by σ.

The worst-case security of $\tilde{\mathcal{P}}_{m,n}$ is reduced to the average-case security of $\tilde{\mathcal{P}}_{m,n}$ by noting that the addition of the $(m+1)$st message to each player can effectively be viewed as adding a random value to each player's input and thus transforming each input value x_i into a uniformly random value in $\mathbb{G}$. More formally, consider the definition $\mathcal{R}_{m+1}(x) = (\mathcal{R}_m(x - \mathsf{U}), \mathsf{U})$, for $x \in \mathbb{G}$, where U is a uniformly random element of $\mathbb{G}$.

Since $\vec{x} - \vec{\mathsf{U}}$ is a uniformly random vector in $\mathbb{G}^n$, then we can couple the randomness observed from two instances $\vec{\mathsf{U}}, \vec{\mathsf{U}}'$ resulting from two independent executions of $\mathcal{P}_{m,n}$ with two inputs having the same sum. Therefore,

$$\begin{aligned}\mathrm{TVD}(\tilde{\mathcal{P}}_{m+1,n}(\vec{x}), \tilde{\mathcal{P}}_{m+1,n}(\vec{x'})) &= \mathrm{TVD}((\mathcal{P}_{m,n}(\vec{x} - \vec{\mathsf{U}}), \vec{\mathsf{U}}), (\mathcal{P}_{m,n}(\vec{x'} - \vec{\mathsf{U}}'), \vec{\mathsf{U}}')) \\ &= \mathbb{E}_{\vec{\mathsf{U}},\vec{\mathsf{U}}'}[\mathrm{TVD}(\mathcal{P}_{m,n}(\vec{x} - \vec{\mathsf{U}}), \mathcal{P}_{m,n}(\vec{x'} - \vec{\mathsf{U}}'))]] \\ &= \mathbb{E}_{\vec{\mathsf{X}},\vec{\mathsf{X}}'}[\mathrm{TVD}(\mathcal{P}_{m,n}(\vec{\mathsf{X}}), \mathcal{P}_{m,n}(\vec{\mathsf{X}}'))],\end{aligned}$$

where $\vec{\mathsf{X}}, \vec{\mathsf{X}}'$ are chosen uniformly at random conditioned on $\vec{\mathsf{X}} = \vec{\mathsf{X}}' + \vec{x} - \vec{x'}$. ◀

We now upper bound the expected total variation distance between the two independent executions of the γ-imperfect shuffle, using an approach similar to Lemma C.1 in [6].

▶ **Lemma 19.** *Let $\vec{\mathsf{X}}$ and $\vec{\mathsf{X}}'$ be drawn uniformly at random from all pairs of vectors in $\mathbb{G}^n$ with the same sum, noting that $\vec{\mathsf{X}}$ and $\vec{\mathsf{X}}'$ are not independent. For two independent executions $\mathcal{P}_{m,n}$ and $\mathcal{P}'_{m,n}$ of the γ-imperfect shuffle, $\mathbb{E}_{\vec{\mathsf{X}},\vec{\mathsf{X}}'}[\mathrm{TVD}_{|\vec{\mathsf{X}},\vec{\mathsf{X}}'}(\mathcal{P}_{m,n}(\vec{\mathsf{X}}), \mathcal{P}_{m,n}(\vec{\mathsf{X}}'))] \leq \sqrt{q^{mn-1}\mathbf{Pr}\left[\mathcal{P}_{m,n}(\vec{\mathsf{X}}) = \mathcal{P}'_{m,n}(\vec{\mathsf{X}})\right] - 1}$.*

We note that the probability that two independent executions of the protocol can be decomposed into the split protocol and the mix protocol as follows. By comparison, Lemma C.2 in [6] was able to prove a simpler relationship by leveraging properties of their symmetric shuffler, which we do not have for an imperfect shuffler.

▶ **Lemma 20.** *Let $\mathcal{R}_{m,n}$ and $\mathcal{R}'_{m,n}$ denote two independent executions of the split protocol in $\mathcal{P}_{m,n}$ so that $\mathcal{P}_{m,n} = \mathcal{S}_{m,n} \circ \mathcal{R}_{m,n}$. Then*

$$\mathbf{Pr}\left[\mathcal{P}_{m,n}(\vec{\mathsf{X}}) = \mathcal{P}'_{m,n}(\vec{\mathsf{X}})\right] = \mathbf{Pr}\left[\mathcal{R}_{m,n}(\vec{\mathsf{X}}) = \mathcal{S}^{-1}_{m,n} \circ \mathcal{S}'_{m,n} \circ \mathcal{R}'_{m,n}(\vec{\mathsf{X}})\right].$$

Proof. Note that

$$\begin{aligned}\mathbf{Pr}\left[\mathcal{P}_{m,n}(\vec{\mathsf{X}}) = \mathcal{P}'_{m,n}(\vec{\mathsf{X}})\right] &= \mathbf{Pr}\left[\mathcal{S}_{m,n} \circ \mathcal{R}_{m,n}(\vec{\mathsf{X}}) = \mathcal{S}'_{m,n} \circ \mathcal{R}'_{m,n}(\vec{\mathsf{X}})\right] \\ &= \mathbf{Pr}\left[\mathcal{R}_{m,n}(\vec{\mathsf{X}}) = \mathcal{S}^{-1}_{m,n} \circ \mathcal{S}'_{m,n} \circ \mathcal{R}'_{m,n}(\vec{\mathsf{X}})\right].\end{aligned}$$

◀

From Lemma 19 and Lemma 20, we have

▶ **Lemma 21.** *For two independent executions $\mathcal{P}_{m,n}$ and $\mathcal{P}'_{m,n}$ of the split-and-mix protocol with a γ-imperfect shuffler,*

$$\mathbb{E}_{\vec{\mathsf{X}},\vec{\mathsf{X}}'}[\mathrm{TVD}(\mathcal{P}_{m,n}(\vec{\mathsf{X}}),\mathcal{P}_{m,n}(\vec{\mathsf{X}}'))] \le \sqrt{q^{mn-1}\mathbf{Pr}\left[\mathcal{R}_{m,n}(\vec{\mathsf{X}}) = \mathcal{S}_{m,n}^{-1}\circ\mathcal{S}'_{m,n}\circ\mathcal{R}'_{m,n}(\vec{\mathsf{X}})\right]-1}.$$

4.2 Reduction to Connected Components

In this section, we prove the following general statement upper bounding the probability that the shuffler $\mathcal{S}_{m,n}^{-1}\circ\mathcal{S}'_{m,n}(\cdot)$ on the output of a randomizer achieves the same output as an independent instance of the randomizer by the expectation of a quantity relating to the number of connected components in the communication graph of the shuffler $\mathcal{S}_{m,n}^{-1}\circ\mathcal{S}'_{m,n}(\cdot)$. Specifically, we can view a protocol $\mathcal{P}_{m,n}$ that is an ordered tuple $\pi_1,\ldots,\pi_m$, where π_j is a permutation on $[n]$ for each $j\in[m]$, so that in each round $j\in[m]$, user $i\in[n]$ sends a message to user $\pi_j(i)$.

Then we can define the *communication graph* for the multi-message shuffle protocol $\mathcal{P}_{m,n}$ as follows. The graph G consists of n vertices, which we associate with $[n]$, corresponding to the players $[n]$ participating in the protocol $\mathcal{P}_{m,n}$. We add an edge between vertices i and j if player i passes one of their m messages to player j.

The following proof is the same as Lemma C.4 in [6].

▶ **Lemma 22.** *Let G be the graph on n vertices formed the communication graph of the shuffle $\mathcal{S}^{-1}\circ\mathcal{S}'$. Let $C(G)$ be the number of connected components of G. Then* $\mathbf{Pr}\left[\vec{\mathcal{R}}(\vec{\mathsf{X}}) = \mathcal{S}^{-1}\circ\mathcal{S}'\circ\vec{\mathcal{R}}'(\vec{\mathsf{X}})\right] \le \mathbb{E}\left[q^{C(G)-mn}\right]$.

Proof. By the law of total expectation,

$$\mathbf{Pr}\left[\vec{\mathcal{R}}(\vec{\mathsf{X}}) = \mathcal{S}^{-1}\circ\mathcal{S}'\circ\vec{\mathcal{R}}'(\vec{\mathsf{X}})\right] = \mathbb{E}\left[\mathbf{Pr}\left[\vec{\mathcal{R}}(\vec{\mathsf{X}}) = \mathcal{S}^{-1}\circ\mathcal{S}'\circ\vec{\mathcal{R}}'(\vec{\mathsf{X}}) \mid \mathcal{S},\mathcal{S}'\right]\right].$$

Thus for the graph G conditioned on $\mathcal{S}$ and $\mathcal{S}'$, it suffices to show that

$$\mathbf{Pr}\left[\vec{\mathcal{R}}(\vec{\mathsf{X}}) = \mathcal{S}^{-1}\circ\mathcal{S}'\circ\vec{\mathcal{R}}'(\vec{\mathsf{X}}) \mid \mathcal{S},\mathcal{S}'\right] = q^{C(G)-mn}.$$

Note that $C(G)$ depends on the choices of $\mathcal{S}$ and $\mathcal{S}'$ but we omit these dependencies in the notation for the sake of presentation. Recall that $\mathcal{P}_{m,n}(\vec{\mathsf{X}}) = \mathcal{S}_{m,n}\circ\vec{\mathcal{R}}_{m,n}(\vec{\mathsf{X}})$ is currently indexed so that the first message of each player after the shuffle protocol completes are the first n indices, followed by the second message of each of the n players and so forth. We thus define a re-indexing permutation $\psi:[mn]\to[mn]$ to that the m messages of the first player will be the first m indices, followed by the m messages of the second player and so forth. That is, $\psi(j) = \left\lfloor\frac{j-1}{m}\right\rfloor + n(j-1 \bmod n)+1$. Let $\mathsf{W},\mathsf{W}'\in\mathbb{G}^{mn}$ be defined so that $\mathsf{W}_j = \psi(\vec{\mathcal{R}}(\vec{\mathsf{X}}))_j$ and $\mathsf{W}'_j = \psi(\mathcal{S}^{-1}\circ\mathcal{S}'\circ\vec{\mathcal{R}}'(\vec{\mathsf{X}}))_j$. The task then becomes to show that $\mathbf{Pr}\left[\mathsf{W}=\mathsf{W}' \mid \mathcal{S},\mathcal{S}'\right] = q^{C(G)-mn}$. Toward that end, for each $j\in[mn]$, we define $\mathcal{E}_j$ to be the event that $\mathsf{W}_j=\mathsf{W}'_j$ and $p_j = \mathbf{Pr}\left[\mathcal{E}_j \mid \mathcal{E}_1,\ldots,\mathcal{E}_{j-1}\right]$, so that $\mathbf{Pr}\left[\mathsf{W}=\mathsf{W}' \mid \mathcal{S},\mathcal{S}'\right] = \prod_{j=1}^{m} np_j$.

Firstly, consider the messages that are not the last message by a particular player, i.e., consider the values of $j\in[mn]$ that are not divisible by m. Observe that conditioning on fixed values of $\vec{\mathsf{X}}$ and $\vec{\mathcal{R}}'$, as well as the events $\mathcal{E}_1,\ldots,\mathcal{E}_{j-1}$, the value of W_j remains uniformly distributed and has probability q^{-1} of being equal to to W'_j. Hence, we have $p_j = q^{-1}$.

For the cases where j is divisible by m, we further consider two subcases. In particular, we consider the case where j is the largest index in C_j and the case where j is not the largest index in C_j, where C_j is the set of vertices in the same connected component as j in G.

In the first subcase, the multisets of W' and $\vec{\mathcal{R}}'(\mathsf{X}')$ restricted to C_i are the same and thus the multisets of the summands are the same, so that $\sum_{i|C_i=C_j} \mathsf{W}'_i = \sum_{i|C_i=C_j} \psi(\vec{\mathcal{R}}'(\mathsf{X}'))_i$. Moreover, since the indices corresponding to all messages of a fixed player are in the same connected component, then $\sum_{i|C_i=C_j} \psi(\vec{\mathcal{R}}'(\mathsf{X}'))_i = \sum_{i|C_i=C_j} \mathsf{W}_i$. Finally, we have that conditioning on $\mathcal{E}_1, \ldots, \mathcal{E}_{j-1}$ and the fact that j is the largest index in C_j, $\sum_{i|C_i=C_j, i\neq j} \mathsf{W}'_i = \sum_{i|C_i=C_j, i\neq j} \mathsf{W}_i$. Therefore, we have $p_j = 1$.

For the second subcase, we shall show that $p_j = q^{-1}$. Let $\mathcal{T}$ be the subset of $(\mathsf{W}, \mathsf{W}') \in \mathbb{G}^{2mn}$ that are consistent with $\mathcal{E}_1, \ldots, \mathcal{E}_{j-1}$ and a fixed value of $\vec{\mathsf{X}}$. We show there exists a homomorphism $\phi : \mathbb{G} \to \mathbb{G}^{2mn}$ that maps from $g \in \mathbb{G}$ to a $u_g \in \mathbb{G}^{2mn}$ with a specific property to be defined. We then consider the action of $\mathbb{G}^{2mn}$ on itself by addition of u_g. Then the property of ϕ that we show is that u_g fixes $\mathcal{T}$ and W_j but adds g to W'_j. Consider the partitioning of $\mathcal{T}$ into equivalence classes where two elements of $\mathcal{T}$ are equivalent if they are equal under addition by u_g for some g. Then the homomorphism induces a partitioning of $\mathcal{T}$ into subsets of size q such that each subset contains exactly one element for which $\mathcal{E}_j$ holds. Since each value of $\mathcal{T}$ is equally probable, it then follows that $p_j = q^{-1}$ as desired.

We now define the homomorphism ϕ as follows. Since there exists a path in G from the vertex with the jth message to a higher index vertex, then there exists some path parameter ℓ and a corresponding path $(a_1, b_1, \ldots, a_\ell, b_\ell, a_{\ell+1})$ such that the following hold. Firstly, each of the terms a_i, b_i are elements of $[mn]$ that will ultimately map to indices of elements in $\mathbb{G}^{mn}$. Secondly, for all $i \in [\ell]$, we have $\pi(b_i) = a_i$ for the permutation π induced by the m message n player protocol and moreover, b_i and a_{i+1} correspond to the same vertex. Finally, it holds that $a_1 = j$, $b_\ell > j$, $a_i \neq a_{i'}$ for any $i \neq i'$, and $b_i < j$ for all $i < \ell$. Then we implicitly define the homomorphism ϕ by defining u_g to be the element of $\mathbb{G}^{2mn}$ with the value g in the entries $a_2, \ldots, a_{\ell+1}, b_1 + mn, \ldots, b_\ell + mn$ and the identity 0 in all other coordinates, where we recall that the elements a_i and b_i correspond to indices of elements in $\mathbb{G}^{mn}$.

We observe that the group action of addition by u_g does not affect the realization of $\mathcal{E}_1, \ldots, \mathcal{E}_{j-1}$ since W_{a_i} and $\mathsf{W}'_{a_i} = \vec{\mathcal{R}}'(\vec{\mathsf{X}})_{b_i}$ are increased by exactly the same amount by u_g, except for the case when $i = 1$ or $i = \ell + 1$. However, note that $a_i \geq j$ for both of the cases where $i = 1$ and $i = \ell + 1$, which does not affect the realization of $\mathcal{E}_1, \ldots, \mathcal{E}_{j-1}$. Hence, u_g has the desired properties and so it follows that $p_j = q^{-1}$.

Therefore, conditioned on any fixed realization of $\mathcal{S}$, we have that $\prod_{j=1}^{mn} p_j = q^{C(G)-mn}$, so that in summary $\mathbf{Pr}\left[\vec{\mathcal{R}} = \mathcal{S}^{-1} \circ \mathcal{S}' \circ \vec{\mathcal{R}}'(\vec{\mathsf{X}})\right] \leq \mathbb{E}[q^{C(G)-mn}]$. ◀

We remark that the statement of Lemma 22 holds even for a general shuffler $\mathcal{S}$ with the corresponding communication graph, rather than the specific shuffler $\mathcal{S}^{-1}_{m,n} \circ \mathcal{S}'_{m,n}(\cdot)$.

We now show that the composition of two shufflers, where the inner shuffler is a γ-imperfect shuffler, is also a γ-imperfect shuffler with the same parameter.

▶ **Lemma 23.** *Let $\mathcal{S}, \mathcal{S}'$ be two shufflers such that $\mathcal{S}$ is a γ-imperfect shuffler. Then, $\mathcal{S}' \circ \mathcal{S}$ is a γ-imperfect shuffler.*

Proof. Let $\mathcal{S}'$ be an arbitrary shuffler and $\mathcal{S}$ be a γ-imperfect shuffler. Then, for any $\pi, \pi' \in \Pi$,

$$\begin{aligned}\mathbf{Pr}\left[\mathcal{S}' \circ \mathcal{S} = \pi\right] &= \mathbf{Pr}\left[\mathcal{S} = (\mathcal{S}')^{-1} \circ \pi\right] \leq e^{\gamma \cdot \mathrm{Swap}((\mathcal{S}')^{-1}\circ\pi, (\mathcal{S}')^{-1}\circ\pi')} \mathbf{Pr}\left[\mathcal{S} = (\mathcal{S}')^{-1} \circ \pi'\right] \\ &= e^{\gamma \cdot \mathrm{Swap}(\pi,\pi')} \mathbf{Pr}\left[\mathcal{S}' \circ \mathcal{S} = \pi'\right].\end{aligned}$$ ◀

We now show a few structural statements that upper bound the probability that there exists no edge from a set $S \subset [n]$ to $[n] \setminus S$ for a communication graph induced by a γ-imperfect shuffler.

▶ **Lemma 24.** *Let G be the communication graph of a γ-imperfect shuffler (on an n-player m-message protocol). For a fixed set S of size s, the probability that there exists no edge from S to $[n] \setminus S$ in G is at most $e^{2sm\gamma}\binom{n}{s}^{-m}$ for $s \leq \frac{n}{2}$ and at most $e^{2(n-s)m\gamma}\binom{n}{s}^{-m}$ for $s \geq \frac{n}{2}$.*

Proof. Without loss of generality, let $S = [s]$, i.e., S is the first s integers of $[n]$. Then for a permutation to not induce an edge between S and $[n] \setminus S$, the permutation can be decomposed into a permutation of the first s integers and a permutation of the remaining $n - s$ integers. Hence, there are $s!(n-s)!$ permutations of $[n]$ such that S is preserved. Let Π_S be the set of permutations that preserves S so that $|\Pi_S| = s!(n-s)!$.

For each permutation $\pi \in \Pi_S$, we define a subset C_π of permutations so that (1) $C_{\pi'} \cap C_\pi = \emptyset$ for all $\pi, \pi' \in \Pi_S$ with $\pi \neq \pi'$, (2) π is the only permutation of C_π that preserves S, (3) $|C_\pi| = \binom{n}{s}$, and (4) π and π' have swap distance at most $2s$ for any $\pi' \in C_\pi$, hence implying that $\mathbf{Pr}[\mathcal{S} = \pi] \leq e^{2s\gamma} \cdot \mathbf{Pr}[\mathcal{S} = \pi']$. Recall that since $\pi \in \Pi_S$, then π can be decomposed into permutations π_1 of the first s integers and permutations π_2 of the remaining $n - s$ integers.

Let A be any set of s indices of $[n]$, sorted in increasing order. Consider the following transformation T_A on a permutation π to produce a permutation ψ. Place the elements of π in positions $[s]$ in order into the s indices of A, so that $\pi'(A_i) = \pi(i)$. For the supplanted indices that have not been assigned to indices in A, place them in order into the remaining positions of $[s]$. Formally, let $X = [s] \setminus A$ and $Y = A \setminus [s]$. Then we set $\pi'(X_i) = \pi(Y_i)$ for all $i \in [|X|]$, noting that $|X| = |Y|$. We then define C_π to be the set of permutations that can be obtained from this procedure, i.e., $C_\pi = \{\pi' : \exists A \text{ with } \pi = T_A(\pi)\}$. See Figure 1 for an example of the application of such an example T_A.

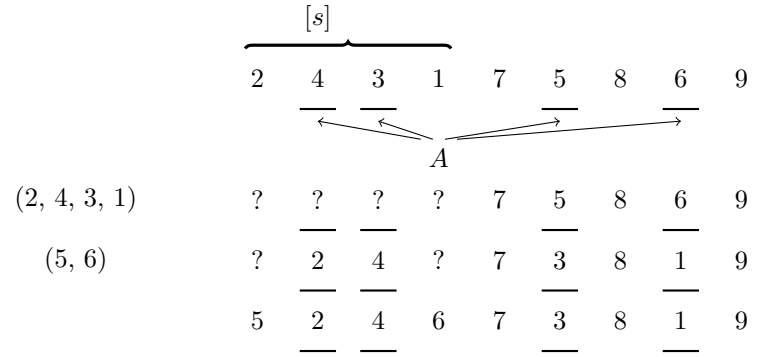


Figure 1 An example of the transformation T_A for the permutation $\pi = (8, 4, 6, 2, 1, 3, 7, 5, 9)$, with $n = 9$, $s = 4$, and $A = (2, 3, 6, 8)$. Note that the order $(8, 4, 6, 2)$ is preserved within the indices of A in the resulting permutation $\pi' = T_A(\pi)$ and the order $(3, 5)$ is preserved within the indices $[s] \setminus A$.

We first claim that $C_{\pi'} \cap C_\pi = \emptyset$ for all $\pi, \pi' \in \Pi_S$ with $\pi \neq \pi'$. Suppose by way of contradiction, there exists $\psi \in C_\pi \cap C_{\pi'}$, so that there exist sets A and A' with $\psi = T_A(\pi) = T_{A'}(\pi')$. Recall that since $\pi, \pi' \in \Pi_S$, then π, π' can be decomposed into permutations π_1, π_1' of the first s integers and permutations π_2, π_2' of the remaining $n - s$ integers. After applying T_A to π, then the first s integers are in the indices of A', in some order. Similarly, after applying $T_{A'}$ to π', then the first s integers are in the indices of A, in some order. Hence for $\psi = T_A(\pi) = T_{A'}(\pi')$, it follows that $A = A'$, so it suffices to show that T_A is injective for a fixed A. To that end, note that T_A preserves the order of $[s]$ within A and thus if $\pi = \pi_1 \circ \pi_2$, then π_1 is the restriction of $T_A(\pi)$ to A. Similarly, note that T_A does not touch the indices outside of $A \cup [s]$ and so π_2 is preserved by $T_A(\pi)$ in the restriction of $[n] \setminus (A \cup [s])$. Finally,

T_A preserves the relative order of π_2 inside the indices of $[s] \setminus A$. Therefore, given A and $T_A(\pi)$, we can completely recover π_1 and π_2 and thus π. In other words, T_A is injective, so that $T_A(\pi) = T_A(\pi')$ implies $\pi = \pi'$, which is a contradiction.

To see that π is the only permutation of C_π that preserves S, note that if any of the s positions are picked outside $[s]$, then there exists a value of $[s]$ outside of the first s positions and so the resulting permutation does not preserve S. However, there is only a single way to pick s indices from $[n]$ that are all inside $[s]$, which corresponds to π. Hence, π is the only permutation of C_π that preserves S.

To see the third property, note that A is formed by choosing s indices of $[n]$. Hence, $|A| = \binom{n}{s}$. Since A is exactly the set of positions for which π_1 is mapped to, then each element of A corresponds to a unique element in C_π. Thus, $|C_\pi| = \binom{n}{s}$.

To see the fourth property, note that the only swaps are indices in A with indices in $[s]$, meaning that at most $2s$ indices are changed. Thus, π and π' have swap distance at most $2s$ for any $\pi' \in C_\pi$. Then by the γ-imperfect shuffle property, $\mathbf{Pr}[\mathcal{S} = \pi] \le e^{2s\gamma} \cdot \mathbf{Pr}[\mathcal{S} = \pi']$.

Since we have associated each $\pi \in \Pi_S$ with a set C_π of size $\binom{n}{s}$ such that $\pi' \notin C_\pi$ for $\pi' \in \Pi_S$ with $\pi' \neq \pi$ and $\mathbf{Pr}[\mathcal{S} = \pi] \le e^{s\gamma} \cdot \mathbf{Pr}[\mathcal{S} = \pi']$, then it follows from a coupling argument that the probability that there exists no edge from S to $[n] \setminus S$ after one iteration of the γ-imperfect shuffle is at most $e^{2s\gamma}\binom{n}{s}^{-1}$. By independence, the probability that there exists no edge from S to $[n] \setminus S$ in G after m iterations is at most $e^{2sm\gamma}\binom{n}{s}^{-m}$.

By symmetry for sets S with size s and $n - s$, we have the probability is at most $\min\left(e^{2sm\gamma}\binom{n}{s}^{-m}, e^{2(n-s)m\gamma}\binom{n}{s}^{-m}\right)$ across all ranges of s. ◀

▶ **Lemma 25.** *Let G be the communication graph of a γ-imperfect shuffler (on an n-player m-message protocol). For a fixed set S with size s, the probability that there exists no edge from S to $[n] \setminus S$ in G is at most $e^{km\gamma}\binom{n/2}{k}^{-m}$ for any integer k with $0 \le k \le \min(s, n-s)$.*

By Lemma 24 and Lemma 25, we have:

▶ **Lemma 26.** *Let G be the communication graph of a γ-imperfect shuffler (on an n-player m-message protocol). For a fixed set S with size s, the probability that there exists no edge from S to $[n] \setminus S$ in G is at most $e^{2sm\gamma}\binom{n}{s}^{-m}$ for $s \le \frac{n}{2}$, at most $e^{2(n-s)m\gamma}\binom{n}{s}^{-m}$ for $s \ge \frac{n}{2}$, and at most $e^{km\gamma}\binom{n/2}{k}^{-m}$ for any integer k with $0 \le k \le \min(s, n-s)$.*

Lemma 23 and Lemma 26 are the two main structural properties of imperfect shufflers that we use to overcome the challenge of adapting the analysis of [6] to shufflers without symmetry.

We now upper bound the probability that the number of connected components of G is c, where G is the underlying communication graph for the split-and-mix-protocol under a γ-imperfect shuffle.

▶ **Lemma 27.** *Let $n \ge 19$ and $m \ge 8e^{4\gamma}$. Let G be the communication graph of a γ-imperfect shuffler (on an n-player m-message protocol). Let $p(n, c)$ denote the probability that the number of connected components of G is c. Then $p(n, c) \le \frac{2^{c-1}}{c!}\left(\frac{e}{n}\right)^{\frac{(m-1)(c-1)}{32e^{4\gamma}}} \cdot e^{2\gamma(m-1)(c-1)}$.*

We now upper bound the expected value of $\mathbb{E}\left[q^{C(G)}\right]$ for the purposes of upper bounding the right hand side of Lemma 22.

▶ **Lemma 28.** *Let $n \ge 19$, $m \ge 8e^{4\gamma}$, and $q \le \left(\frac{n}{e}\right)^{\frac{(m-1)}{32e^{4\gamma}}} e^{2\gamma(1-m)}$. Let G be the graph on n vertices formed a random instantiation of the split-and-mix protocol $\mathcal{P}_{m,n}$ with m messages for each of n players, using a γ-imperfect shuffler $\mathcal{S}$. That is, let G have an edge between i and j if and only if player i passes one of their m messages to player j. Then $\mathbb{E}\left[q^{C(G)}\right] \le q + 3q^2 e^{2\gamma(m-1)}\left(\frac{e}{n}\right)^{\frac{m-1}{32e^{4\gamma}}}$.*

Proof. By Lemma 27, we have $p(n,c) \le \frac{2^{c-1}}{c!}\left(\frac{e}{n}\right)^{\frac{(m-1)(c-1)}{32e^{4\gamma}}} \cdot e^{2\gamma(m-1)(c-1)}$. Taking the expectation, we have $\mathbb{E}\left[q^{C(G)}\right] \le \sum_{c=1}^{n} q^c \frac{2^{c-1}}{c!}\left(\frac{e}{n}\right)^{\frac{(m-1)(c-1)}{32e^{4\gamma}}} \cdot e^{2\gamma(m-1)(c-1)}$. Since term in the summand after the second term is at most $\frac{2q}{3}\left(\frac{e}{n}\right)^{\frac{(m-1)}{32e^{4\gamma}}} e^{2\gamma(m-1)}$ times the previous term in the summand, then $\mathbb{E}\left[q^{C(G)}\right] \le q + q^2 e^{2\gamma(m-1)}\left(\frac{e}{n}\right)^{\frac{m-1}{32e^{4\gamma}}} \sum_{i=0}^{\infty}\left(\frac{2q}{3}\left(\frac{e}{n}\right)^{\frac{(m-1)}{32e^{4\gamma}}} e^{2\gamma(m-1)}\right)^i$. Since $q \le \left(\frac{n}{e}\right)^{\frac{(m-1)}{32e^{4\gamma}}} e^{2\gamma(1-m)}$ by assumption, then $\mathbb{E}\left[q^{C(G)}\right] \le q + 3q^2 e^{2\gamma(m-1)}\left(\frac{e}{n}\right)^{\frac{m-1}{32e^{4\gamma}}}$. ◀

We now analyze the statistical security of the split-and-mix protocol.

▶ **Lemma 29.** *Let $n \ge 19$, $m \ge 8e^{4\gamma}$, and $q \le \left(\frac{n}{e}\right)^{\frac{(m-1)}{32e^{4\gamma}}} e^{2\gamma(1-m)}$. Then we have worst-case statistical security with parameter $\sigma \le (m-1)\left(\frac{\log n - \log e}{64e^{4\gamma}} - 2\gamma\log e\right) - 3\log(3q)$,*

Proof. By Lemma 21 and Lemma 22, we have

$$\mathbb{E}_{\vec{\mathsf{X}},\vec{\mathsf{X}}'}[\mathrm{TVD}(\mathcal{P}_{m,n}(\vec{\mathsf{X}}), \mathcal{P}_{m,n}(\vec{\mathsf{X}}'))] \le \sqrt{q^{mn-1}\mathbb{E}\left[q^{C(G)-mn}\right] - 1},$$

where $C(G)$ is the communication graph for the shuffle $\mathcal{S}^{-1} \circ \mathcal{S}'$. By Lemma 23 and the fact that $\mathcal{S}'$ is a γ-imperfect shuffler, we have that $\mathcal{S}^{-1} \circ \mathcal{S}'$ is also a γ-imperfect shuffler and thus it suffices to upper bound $\mathbb{E}\left[q^{C(G)-mn}\right]$ where $C(G)$ is the communication graph for an arbitrary γ-imperfect shuffler $\mathcal{S}$. Therefore by Lemma 28, we have average case statistical security less than or equal to $2^{-\sigma} \ge \sqrt{3q^3 e^{2\gamma(m-1)}\left(\frac{e}{n}\right)^{\frac{m-1}{32e^{4\gamma}}}}$, which holds for $\sigma \le (m-1)\left(\frac{\log n - \log e}{64e^{4\gamma}} - 2\gamma\log e\right) - 3\log(3q)$. The claim then follows by the reduction of worst-case input to average-case input by Lemma 18. ◀

Now it can be verified that by restricting $\gamma \le \frac{\log\log n}{80}$, then we have both $728e^{4\gamma} \le \log n$ and $\lceil 2n^{3/2}\rceil \le \left(\frac{n}{e}\right)^{\frac{(m-1)}{32e^{4\gamma}}} e^{2\gamma(1-m)}$. These conditions imply that 1) $\left(\frac{\log n - \log e}{64e^{4\gamma}} - 2\gamma\log e\right) = \mathcal{O}\left(\frac{\log n}{e^{4y}}\right)$, so that the parameter σ has a non-empty range in the statement of Lemma 29, and 2) $q = \lceil 2n^{3/2}\rceil$ satisfies $q \le \left(\frac{n}{e}\right)^{\frac{(m-1)}{32e^{4\gamma}}} e^{2\gamma(1-m)}$ in the statement of Lemma 29. As a corollary, we obtain the following guarantees for worst-case statistical security:

▶ **Theorem 15.** *Let $n \ge 19$ and $\gamma \le \frac{\log\log n}{80}$ be a distortion parameter. For worst-case statistical security with parameter σ, it suffices to use $m = \mathcal{O}\left(e^{4\gamma} + \frac{e^{4\gamma}(\sigma + \log n)}{\log n}\right)$ messages, where each message has $\mathcal{O}(\log q)$ bits, for $q = \lceil 2n^{3/2}\rceil$.*

5 Conclusion and Discussion

In this work, we introduce the imperfect shuffle DP model, as a means of abstracting out real-world scenarios that prevent perfect shuffling. We also give a real summation protocol with nearly optimal error and small communication complexity. The protocol, which is based on the split-and-mix protocol [30], is similar to that of the (perfect) shuffle model [6, 25], while the main challenge comes in the analysis. Although we overcome this hurdle for this particular protocol, our techniques are quite specific. Therefore, an interesting open question is whether there is a general theorem that transfer the privacy guarantee in the perfect shuffle model to that in the imperfect shuffle model, possibly with some loss in the privacy parameters.

References

1 Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS*, pages 308–318, 2016.

2 John M Abowd. The US census bureau adopts differential privacy. In *KDD*, pages 2867–2867, 2018.

3 Naman Agarwal, Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and Brendan McMahan. cpSGD: communication-efficient and differentially-private distributed SGD. In *NeurIPS*, pages 7575–7586, 2018.

4 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. Differentially private summation with multi-message shuffling. *CoRR*, abs/1906.09116, 2019. `arXiv:1906.09116`.

5 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *CRYPTO*, pages 638–667, 2019.

6 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. Private summation in the multi-message shuffle model. In *CCS*, pages 657–676, 2020.

7 Borja Balle, Peter Kairouz, Brendan McMahan, Om Dipakbhai Thakkar, and Abhradeep Thakurta. Privacy amplification via random check-ins. In *NeurIPS*, 2020.

8 Raef Bassily, Adam D. Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*, pages 464–473, 2014.

9 Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *CRYPTO*, pages 451–468, 2008.

10 Sourav Biswas, Yihe Dong, Gautam Kamath, and Jonathan R. Ullman. CoinPress: Practical private mean and covariance estimation. In *NeurIPS*, 2020.

11 Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnés, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *SOSP*, pages 441–459, 2017.

12 T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Optimal lower bound for differentially private multi-party aggregation. In *ESA*, pages 277–288, 2012.

13 Xiangyi Chen, Zhiwei Steven Wu, and Mingyi Hong. Understanding gradient clipping in private SGD: A geometric perspective. In *NeurIPS*, 2020.

14 Albert Cheu, Adam D. Smith, Jonathan R. Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *EUROCRYPT*, pages 375–403, 2019.

15 Albert Cheu and Chao Yan. Necessary conditions in multi-server differential privacy. In *ITCS*, pages 36:1–36:21, 2023.

16 Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *NIPS*, pages 3571–3580, 2017.

17 Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

18 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

19 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

20 Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *SODA*, pages 2468–2479, 2019.

21 Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067, 2014.

22 Vitaly Feldman, Audra McMillan, and Kunal Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *FOCS*, pages 954–964, 2021.

23 Vitaly Feldman, Audra McMillan, and Kunal Talwar. Stronger privacy amplification by shuffling for renyi and approximate differential privacy. In *SODA*, pages 4966–4981, 2023.

24 Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Amer Sinha. Differentially private aggregation in the shuffle model: Almost central accuracy in almost a single message. In *ICML*, pages 3692–3701, 2021.

25 Badih Ghazi, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Private aggregation from fewer anonymous messages. In *EUROCRYPT*, pages 798–827, 2020.

26 Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM J. Comput.*, 41(6):1673–1693, 2012.

27 Antonious M. Girgis, Deepesh Data, Suhas N. Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of federated learning: Privacy, accuracy and communication trade-offs. *IEEE J. Sel. Areas Inf. Theory*, 2(1):464–478, 2021.

28 S. Dov Gordon, Jonathan Katz, Mingyu Liang, and Jiayu Xu. Spreading the privacy blanket: Differentially oblivious shuffling for differential privacy. In *ACNS*, pages 501–520, 2022.

29 Andy Greenberg. Apple's "differential privacy" is about collecting your data – but not your data, June 2016.

30 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *FOCS*, pages 239–248, 2006.

31 Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021.

32 Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011.

33 Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *CoRR*, abs/1610.05492, 2016. `arXiv:1610.05492`.

34 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.

35 Stephen Shankland. How google tricks itself to protect chrome user privacy. *CNET, October*, 2014.

36 Elaine Shi and Ke Wu. Non-interactive anonymous router. In *EUROCRYPT*, pages 489–520, 2021.

37 Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. Stochastic gradient descent with differentially private updates. In *GlobalSIP*, pages 245–248, 2013.

38 Uri Stemmer. Locally private k-means clustering. *J. Mach. Learn. Res.*, 22:176:1–176:30, 2021.

39 Uri Stemmer and Haim Kaplan. Differentially private k-means with constant multiplicative error. In *NeurIPS*, pages 5436–5446, 2018.

40 Ananda Theertha Suresh, Felix X. Yu, Sanjiv Kumar, and H. Brendan McMahan. Distributed mean estimation with limited communication. In *ICML*, pages 3329–3337, 2017.

41 Martin Thomson and Christopher A. Wood. Oblivious http, 2023. URL: `https://datatracker.ietf.org/doc/draft-ietf-ohai-ohttp/`.

42 Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *JASA*, 60(309):63–69, 1965.

43 Royce J. Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. Differentially private SQL with bounded user contribution. *PoPETS*, 2020(2):230–250, 2020.

44 Mingxun Zhou and Elaine Shi. The power of the differentially oblivious shuffle in distributed privacy mechanisms. *IACR Cryptol. ePrint Arch.*, page 177, 2022.

45 Mingxun Zhou, Elaine Shi, T.-H. Hubert Chan, and Shir Maimon. A theory of composition for differential obliviousness. In *EUROCRYPT*, 2023.

A Missing Proofs

Proof of Lemma 19. We write $\mathcal{P}$ and $\mathcal{P}'$ as shorthand for $\mathcal{P}_{m,n}$ and $\mathcal{P}'_{m,n}$, respectively. Let $\vec{\mathsf{V}}$ be a uniformly random vector drawn from $\mathbb{G}^{mn}$, conditioned on $\vec{\mathsf{V}}$ having the same sum as $\vec{\mathsf{X}}$ and $\vec{\mathsf{X}}'$. Then by the triangle inequality,

$$\begin{aligned}\mathbb{E}_{\vec{\mathsf{X}},\vec{\mathsf{X}}'}[\underset{|\vec{\mathsf{X}},\vec{\mathsf{X}}'}{\mathrm{TVD}}(\mathcal{P}(\vec{\mathsf{X}}),\mathcal{P}(\vec{\mathsf{X}}'))] &\le \mathbb{E}_{\vec{\mathsf{X}},\vec{\mathsf{X}}'}[\underset{|\vec{\mathsf{X}},\vec{\mathsf{X}}'}{\mathrm{TVD}}(\mathcal{P}(\vec{\mathsf{X}}),\vec{\mathsf{V}}) + \underset{|\vec{\mathsf{X}},\vec{\mathsf{X}}'}{\mathrm{TVD}}(\vec{\mathsf{V}},\mathcal{P}(\vec{\mathsf{X}}'))] \\ &= \mathbb{E}_{\vec{\mathsf{X}}}[\underset{|\vec{\mathsf{X}}}{\mathrm{TVD}}(\mathcal{P}(\vec{\mathsf{X}}),\vec{\mathsf{V}})] + \mathbb{E}_{\vec{\mathsf{X}}'}[\underset{|\vec{\mathsf{X}}'}{\mathrm{TVD}}(\vec{\mathsf{V}},\mathcal{P}(\vec{\mathsf{X}}'))] \\ &= 2\mathbb{E}_{\vec{\mathsf{X}}}[\underset{|\vec{\mathsf{X}}}{\mathrm{TVD}}(\mathcal{P}(\vec{\mathsf{X}}),\vec{\mathsf{V}})].\end{aligned}$$

Moreover, considering the distribution over $\vec{\mathsf{V}}$,

$$\begin{aligned}2\,\underset{|\vec{\mathsf{X}}}{\mathrm{TVD}}(\mathcal{P}(\vec{\mathsf{X}}),\vec{\mathsf{V}}) &= \sum_{\vec{v}\in\mathbb{G}^{mn}} \left|\mathbf{Pr}\left[\mathcal{P}(\vec{\mathsf{X}})=\vec{v}\right] - \mathbf{Pr}\left[\vec{\mathsf{V}}=\vec{v}\right]\right| \\ &= \sum_{\vec{v}\in\mathbb{G}^{mn},\sum\vec{v}=\sum\vec{\mathsf{X}}} |\mathbf{Pr}\left[\mathcal{P}(\vec{\mathsf{X}})=\vec{v}\right] - q^{1-mn}| \\ &= q^{mn-1}\mathbb{E}_{\vec{\mathsf{V}}}\left[\left|\mathbf{Pr}\left[\mathcal{P}(\vec{\mathsf{X}})=\vec{\mathsf{V}}\right] - q^{1-mn}\right|\right].\end{aligned}$$

Since $\vec{\mathsf{V}}$ is a uniformly random vector from $\mathbb{G}^{mn}$ with its sum being equal to that of $\vec{\mathsf{X}}$, then for the random variable $\mathcal{Z} := \mathcal{Z}(\mathsf{X},\mathsf{V}) := \mathbf{Pr}\left[\mathcal{P}(\vec{\mathsf{X}})=\vec{\mathsf{V}}\right]$, we have $\mathbb{E}[\mathcal{Z}] = q^{1-mn}$. Therefore, $2\,\mathrm{TVD}_{|\vec{\mathsf{X}}}(\mathcal{P}(\vec{\mathsf{X}}),\vec{\mathsf{V}}) \le q^{mn-1}\mathbb{E}[|\mathcal{Z}-\mathbb{E}[\mathcal{Z}]|]$. By convexity, $\mathbb{E}[|\mathcal{Z}-\mathbb{E}[\mathcal{Z}]|] \le \sqrt{\mathbb{E}[\mathcal{Z}^2]}$. Since we have

$$\mathbb{E}_{\vec{\mathsf{V}}}[\mathcal{Z}^2] = q^{1-mn}\sum_{\vec{v}\in\mathbb{G}^{mn},\sum\vec{v}=\sum\vec{\mathsf{X}}} \mathbf{Pr}\left[\mathcal{P}(\vec{\mathsf{X}})=\vec{v}\right]^2 = q^{1-mn}\mathbf{Pr}\left[\mathcal{P}(\vec{\mathsf{X}})=\mathcal{P}'(\vec{\mathsf{X}})\right],$$

we thus have

$$\begin{aligned}\mathbb{E}_{\vec{\mathsf{X}},\vec{\mathsf{X}}'}[\underset{|\vec{\mathsf{X}},\vec{\mathsf{X}}'}{\mathrm{TVD}}(\mathcal{P}(\vec{\mathsf{X}}),\mathcal{P}(\vec{\mathsf{X}}')] &\le 2\,\underset{|\vec{\mathsf{X}}}{\mathrm{TVD}}(\mathcal{P}(\vec{\mathsf{X}}),\vec{\mathsf{V}}) \\ &\le q^{mn-1}\mathbb{E}_{\mathcal{V}(\vec{\mathsf{X}}')}[|\mathbf{Pr}\left[\mathcal{P}(\vec{\mathsf{X}})=\vec{\mathsf{V}}\right] - q^{1-mn}|] \\ &\le \sqrt{q^{mn-1}\mathbf{Pr}\left[\mathcal{P}_{m,n}(\vec{\mathsf{X}})=\mathcal{P}'_{m,n}(\vec{\mathsf{X}})\right] - 1}.\end{aligned}$$

◀

Proof of Lemma 25. We can similarly show that the probability that there exists no edge from S to $[n]\setminus S$ in G after the m iterations is at most $e^{km\gamma}\binom{n/2}{k}^{-m}$ for any integer k with $0 \le k \le \min(s, n-s)$ by the following modifications to the coupling argument. We again let $S = [s]$ without loss of generality and let $k \le \min(s, n-s)$ be a fixed non-negative integer.

Recall that there are $s!(n-s)!$ permutations of $[n]$ such that S is preserved. We define Π_S to be the set of permutations that preserves S so that $|\Pi_S| = s!(n-s)!$ and we define a transformation $T_A(\pi)$ for a permutation $\pi \in \Pi_S$ as follows.

If $s \leq \frac{n}{2}$, we let A be a set of k positions in $\{s+1, \ldots, n\}$, sorted in increasing order. We then initialize $\psi = \pi$ and iteratively perform the following procedure k times. For each $i \in [k]$, we swap the value in the ith index of ψ with the value in the A_ith index of A. We then output set $T_A(\pi)$ to be the result of ψ after applying these k swaps. Note that since $[s]$ and A are disjoint, we can also explicitly define the resulting $\psi = T_A(\pi)$ by

$$\psi(i) = \begin{cases} \pi(i), & i \notin (A \cup [k]) \\ \pi(A_i), & i \in [k] \\ \pi(j), & j = A_i, i \in [k] \end{cases}.$$

Similarly, if $s \geq \frac{n}{2}$, we let A be a set of k positions in $[n-s]$, sorted in increasing order, and initialize $\psi = \pi$. Then for each $i \in [k]$, we swap the value in the $(n-i+1)$st index of ψ with the value in the ith index of A. Alternatively, we can also explicitly define the resulting $\psi = T_A(\pi)$ by

$$\psi(i) = \begin{cases} \pi(i), & i \notin (A \cup \{n-k+1, \ldots, n\}) \\ \pi(A_i), & i \in \{n-k+1, \ldots, n\} \\ \pi(j), & j = A_i, i \in [k] \end{cases}.$$

We again define C_π to be the set of permutations that can be obtained from this procedure, i.e., $C_\pi = \{\pi' : \exists A \text{ with } \pi = T_A(\pi)\}$. By the same argument as in Lemma 25, we have (1) $C_{\pi'} \cap C_\pi = \emptyset$ for all $\pi, \pi' \in \Pi_S$ with $\pi \neq \pi'$, (2) π is the only permutation of C_π that preserves S, (3) $|C_\pi| = \binom{n}{k}$. By the construction of T_A performing k swaps on π, we also have that π and π' have swap distance at most k for any $\pi' \in C_\pi$, so that $\mathbf{Pr}[\mathcal{S} = \pi] \leq e^{k\gamma} \cdot \mathbf{Pr}[\mathcal{S} = \pi']$.

Also by construction, we have $|C_\pi| \geq \binom{n/2}{k}$ and so by adapting the above coupling argument, we have that the probability that there exists no edge from S to $[n] \setminus S$ in G after the m iterations is at most $e^{km\gamma}\binom{n/2}{k}^{-m}$. ◀

Proof of Lemma 27. For a fixed set S, let $\mathbb{P}_S$ denote the probability that there is no edge from S to $[n] \setminus S$. Let $p(n, c)$ denote the probability that the number of connected components of G is c. Then

$$p(n,c) \leq \frac{1}{c} \sum_{S \subseteq [n]} \mathbb{P}_S \cdot p(n - |S|, c-1) \leq \frac{1}{c} \sum_{s=1}^{n-c+1} \binom{n}{s} \mathbb{P}_S \cdot p(n-|S|, c-1).$$

We decompose this sum and apply Lemma 26.

By Lemma 26, we have $\mathbb{P}_S \leq \min(e^{2(n-s)m\gamma}\binom{n}{s}^{-m}, e^{2sm\gamma}\binom{n}{s}^{-m})$. By Lemma 26, we also have $\mathbb{P}_S \leq e^{km\gamma}\binom{n/2}{k}^{-m}$ for any $k \leq \min(s, n-s)$. Observe that for $k \geq n - s \geq \frac{n}{2}$, we have $e^{2(n-s)m\gamma}\binom{n}{s}^{-m} \leq e^{2km\gamma}\binom{n}{k}^{-m} \leq e^{2km\gamma}\binom{n/2}{k}^{-m}$. Thus for $k = \frac{n}{4e^{4\gamma}}$,

$$\begin{aligned} p(n,c) \leq{}& \frac{1}{c} \sum_{s=1}^{k} \binom{n}{s}\binom{n}{s}^{-m} e^{2sm\gamma} \cdot p(n-|S|, c-1) \\ &+ \frac{1}{c} \sum_{s=k+1}^{n-c+1} \binom{n}{s}\binom{n/2}{k}^{-m} e^{2km\gamma} \cdot p(n-|S|, c-1). \end{aligned}$$

Observe that $k = \frac{n}{4e^{4\gamma}}$ implies that

$$\begin{aligned} e^{2\gamma} &\leq \left(\frac{n}{2k}\right)^{1/2} \\ e^{2km\gamma} &\leq \left(\frac{n}{2k}\right)^{km/2} \leq \binom{n/2}{k}^{m/2} \\ \binom{n/2}{k}^{-m} e^{2km\gamma} &\leq \binom{n/2}{k}^{-m/2} \leq \binom{n}{k}^{-m/2}. \end{aligned}$$

Thus we have

$$\begin{aligned} p(n,c) \leq \frac{1}{c}\sum_{s=1}^{k}\binom{n}{s}\binom{n}{s}^{-m} e^{2sm\gamma}\cdot p(n-|S|,c-1) \\ + \frac{1}{c}\sum_{s=k+1}^{n-c+1}\binom{n}{s}\binom{n}{k}^{-m/2}\cdot p(n-|S|,c-1). \end{aligned}$$

Since $k = \frac{n}{4e^{4\gamma}}$, then

$$\binom{n}{k}^{-m/2} \leq (4e^{4\gamma})^{-\frac{nm}{8e^{4\gamma}}} \leq (2e)^{-\frac{nm}{8e^{4\gamma}}} \leq \binom{n}{n/2}^{-\frac{m}{4e^{4\gamma}}} \leq \binom{n}{s}^{-\frac{m}{4e^{4\gamma}}}.$$

Hence,

$$\begin{aligned} p(n,c) \leq \frac{1}{c}\sum_{s=1}^{k}\binom{n}{s}\binom{n}{s}^{-m} e^{2sm\gamma}\cdot p(n-|S|,c-1) \\ + \frac{1}{c}\sum_{s=k+1}^{n-c+1}\binom{n}{s}^{1-\frac{m}{4e^{4\gamma}}}\cdot p(n-|S|,c-1). \end{aligned}$$

For $m \geq 8e^{4\gamma}$, we have $1 \leq \frac{m}{8e^{4\gamma}}$ and thus

$$\begin{aligned} p(n,c) \leq \frac{1}{c}\sum_{s=1}^{k}\binom{n}{s}\binom{n}{s}^{-m} e^{2sm\gamma}\cdot p(n-|S|,c-1) \\ + \frac{1}{c}\sum_{s=k+1}^{n-c+1}\binom{n}{s}^{-\frac{m}{8e^{4\gamma}}}\cdot p(n-|S|,c-1). \end{aligned}$$

We first apply the induction hypothesis that $p(n,c) \leq \frac{2^{c-1}}{c!}\left(\frac{e}{n}\right)^{\frac{(m-1)(c-1)}{32e^{4\gamma}}}\cdot e^{\gamma(m-1)(c-1)}$:

$$\begin{aligned} p(n,c) \leq \frac{2^{c-1}}{c!}\left(\frac{e}{n}\right)^{\frac{(m-1)(c-1)}{32e^{4\gamma}}}\cdot e^{2\gamma(m-1)(c-1)}\cdot\frac{1}{2}\cdot e^{\frac{(1-m)}{32e^{4\gamma}}}\cdot e^{2\gamma(1-m)} \\ \cdot\left(\sum_{s=1}^{k}\binom{n}{s}^{1-m} e^{2sm\gamma}\left(\frac{n^{c-1}}{(n-s)^{c-2}}\right)^{\frac{m-1}{32e^{4\gamma}}} + \sum_{s=k+1}^{n-c+1}\left(\frac{(n-s)!s!n^{c-1}}{n!(n-s)^{c-2}}\right)^{\frac{m-1}{32e^{4\gamma}}}\right). \end{aligned}$$

We upper bound $p(n,c)$ by upper bounding the summation across the first k terms, i.e., the head of the summation, then upper bounding the tail terms of the summation, i.e., the terms with $s \geq \frac{3n}{4}$, and finally upper bounding the remaining terms of the summation, i.e., $s \in \left[k, \frac{3n}{4}\right]$.

Upper bounding the head terms in the summation. We now upper bound the summation across all $s \le k$. Let $a_s = \binom{n}{s}^{1-m} e^{2sm\gamma} \left(\frac{n^{c-1}}{(n-s)^{c-2}}\right)^{\frac{m-1}{32e^{4\gamma}}}$. For $s \le k = \frac{n}{4e^{4\gamma}}$ and $m \ge 8e^{4\gamma}$,

$$\begin{aligned}
\frac{a_s}{a_{s-1}} &= \left(\frac{s}{n-s+1}\right)^{m-1} e^{2m\gamma} \left(\frac{n-s+1}{n-s}\right)^{\frac{(m-1)(c-2)}{32e^{4\gamma}}} \\
&\le \left(\frac{1}{8e^{4\gamma}}\right)^{m-1} e^{2m\gamma} e^{\frac{(m-1)(c-2)}{n-s}} \\
&\le \left(\frac{1}{8e^{4\gamma}}\right)^{m-1} (e^{4\gamma})^{m-1} e^{\frac{4(m-1)}{3}} \\
&\le \left(\frac{e^{4/3}}{8}\right)^{m-1} \le \left(\frac{1}{2}\right)^{m-1} \le \frac{1}{25}.
\end{aligned}$$

Then through a geometric series, we bound the summation

$$\begin{aligned}
\sum_{s=1}^{k} a_s &\le \sum_{s=1}^{\infty} \frac{a_1}{25^{s-1}} \le \frac{26a_1}{25} \\
&\le \frac{26}{25} n^{1-m} e^{m\gamma} \left(\frac{n^{c-1}}{(n-1)^{c-2}}\right)^{\frac{m-1}{32e^{4\gamma}}} \\
&\le \frac{26}{25} e^{m\gamma} e^{\frac{m-1}{32e^{4\gamma}}}
\end{aligned}$$

Upper bounding the tail terms in the summation. We now upper bound the summation across all $s \ge \lceil \frac{3n}{4} \rceil$. Let $b_s = \left(\frac{(n-s)!s!n^{c-1}}{n!(n-s)^{c-2}}\right)^{\frac{m-1}{32e^{4\gamma}}}$. Then for $s \ge \frac{3n}{4}$,

$$\begin{aligned}
\frac{b_s}{b_{s-1}} &= \left(\frac{s}{n-s+1}\left(\frac{n-s+1}{n-s}\right)^{c-2}\right)^{\frac{m-1}{32e^{4\gamma}}} \\
&\ge \left(\frac{s}{n-s}\right)^{\frac{m-1}{32e^{4\gamma}}} \ge 9.
\end{aligned}$$

We again bound another subset of the sum through a geometric series:

$$\begin{aligned}
\sum_{s=\lceil 3n/4 \rceil}^{n-c+1} b_s &\le \sum_{s=\lceil 3n/4 \rceil}^{n-c+1} \frac{b_{n-c+1}}{9^{n-c+1-s}} \\
&\le \sum_{s=-\infty}^{n-c+1} \frac{b_{n-c+1}}{9^{n-c+1-s}} \\
&= \frac{9b_{n-c+1}}{8} \\
&= \frac{9}{8}\left(\frac{(c-1)!(n-c+1)!n^{c-1}}{n!(c-1)^{c-2}}\right)^{\frac{m-1}{32e^{4\gamma}}}.
\end{aligned}$$

Similar to [6], we bound the last expression using Sterling's bound, $\sqrt{2\pi}n^{n+\frac{1}{2}}e^{-n} \le n! \le en^{n+\frac{1}{2}}e^{-n}$, so that $\frac{9}{8}\left(\frac{(c-1)!(n-c+1)!n^{c-1}}{n!(c-1)^{c-2}}\right)^{\frac{m-1}{32e^{4\gamma}}} \le \frac{9}{8}\left(\frac{e}{\sqrt{2\pi}}(c-1)^{1.5}\left(1-\frac{(c-1)}{n}\right)^{n-c+1.5}\right)^{\frac{m-1}{32e^{4\gamma}}}$, which is maximized at $c = 3$ for $n \ge 19$, $m \ge 8e^{4\gamma}$, and $c \le \frac{n}{4}$. Thus,

$$\frac{9}{8}\left(\frac{e}{\sqrt{2\pi}}(c-1)^{1.5}\left(1-\frac{(c-1)}{n}\right)^{n-c+1.5}\right)^{\frac{m-1}{32e^{4\gamma}}} \leq \frac{9}{8}\left(\frac{2e}{\sqrt{\pi}}\left(1-\frac{2}{n}\right)^{n-1.5}\right)^{\frac{m-1}{32e^{4\gamma}}} \leq \frac{9}{8}(1.27)^{\frac{m-1}{32e^{4\gamma}}}.$$

Upper bounding the middle terms in the summation. It remains to upper bound the summation across $s \in \left[\frac{n}{4e^{4\gamma}}, \frac{3n}{4}\right]$. We have for $\alpha = \frac{s}{n}$, $b_s = \left(\frac{((1-\alpha)n)!(\alpha n)!}{(n-1)!(1-\alpha)^{c-2}}\right)^{\frac{m-1}{32e^{4\gamma}}}$. By Sterling's bound, we have $b_s \leq \left(\frac{e^2}{\sqrt{2\pi}}\sqrt{n}(1-\alpha)^{2.5-c+(1-\alpha)n}\alpha^{\alpha n+\frac{1}{2}}\right)^{\frac{m-1}{32e^{4\gamma}}} \leq \left(\frac{e^2\sqrt{n}}{\sqrt{2\pi}}\alpha^{\alpha n}\right)^{\frac{m-1}{32e^{4\gamma}}}$. Since there are at most n such terms b_s, then $\sum_{s=k+1}^{\lceil 3n/4\rceil-1} b_s \leq n\left(\frac{e^2\sqrt{n}}{\sqrt{2\pi}}\left(\frac{3}{4}\right)^{\frac{3n}{4}}\right)^{\frac{m-1}{32e^{4\gamma}}} \leq 2\left(en\left(\frac{3}{4}\right)^{\frac{3n}{4}}\right)^{\frac{m-1}{32e^{4\gamma}}} \leq 2$.

Putting things together. Combining the upper bounds across the three summations, we have

$$\begin{aligned}\sum_{s=1}^{k}\binom{n}{s}^{1-m}e^{2sm\gamma}\left(\frac{n^{c-1}}{(n-s)^{c-2}}\right)^{\frac{m-1}{32e^{4\gamma}}} + \sum_{s=k+1}^{n-c+1}\left(\frac{(n-s)!s!n^{c-1}}{n!(n-s)^{c-2}}\right)^{\frac{m-1}{32e^{4\gamma}}} &\leq \frac{26}{25}e^{m\gamma}e^{\frac{m-1}{32e^{4\gamma}}} + 2 + \frac{9}{8}(1.27)^{\frac{m-1}{32e^{4\gamma}}} \\ &\leq 2e^{\frac{m-1}{32e^{4\gamma}}}\cdot e^{m\gamma} \leq 2e^{\frac{m-1}{32e^{4\gamma}}}\cdot e^{2\gamma(m-1)}.\end{aligned}$$

Therefore, we have $p(n,c) \leq \frac{2^{c-1}}{c!}\left(\frac{e}{n}\right)^{\frac{(m-1)(c-1)}{32e^{4\gamma}}}\cdot e^{2\gamma(m-1)(c-1)}$, as desired. ◀

Exponential Correlated Randomness Is Necessary in Communication-Optimal Perfectly Secure Two-Party Computation

Keitaro Hiwatashi
The University of Tokyo, Japan
National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan

Koji Nuida
Kyushu University, Japan
National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan

Abstract

Secure two-party computation is a cryptographic technique that enables two parties to compute a function jointly while keeping each input secret. It is known that most functions cannot be realized by information-theoretically secure two-party computation, but any function can be realized in the correlated randomness (CR) model, where a trusted dealer distributes input-independent CR to the parties beforehand. In the CR model, three kinds of complexities are mainly considered; the size of CR, the number of rounds, and the communication complexity.

Ishai et al. (TCC 2013) showed that any function can be securely computed with optimal online communication cost, i.e., the number of rounds is one round and the communication complexity is the same as the input length, at the price of exponentially large CR. In this paper, we prove that exponentially large CR is necessary to achieve perfect security and online optimality for a general function and that the protocol by Ishai et al. is asymptotically optimal in terms of the size of CR. Furthermore, we also prove that exponentially large CR is still necessary even when we allow multiple rounds while keeping the optimality of communication complexity.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases Secure Computation, Correlated Randomness, Lower Bound

Digital Object Identifier 10.4230/LIPIcs.ITC.2023.18

Funding Supported by JST CREST Grant Number JPMJCR2113.
Keitaro Hiwatashi: This research was in part conducted under a contract of "Research and development on new generation cryptography for secure wireless communication services" among "Research and Development for Expansion of Radio Wave Resources (JPJ000254)," which was supported by the Ministry of Internal Affairs and Communications, Japan. The author was also partially supported by JSPS KAKENHI Grant Number JP21J20186 and JP22KJ0546.
Koji Nuida: Supported by JSPS KAKENHI Grant Number JP19H01109, JP22K11906, and JST AIP Acceleration Research JPMJCR22U5.

1 Introduction

Secure multi-party computation (MPC) is a cryptographic technique that enables some parties to compute a function jointly while keeping each input secret. Secure multi-party computation has been extensively studied since Yao advocated it in the 1980s [25]. From a theoretical point of view, Kushilevitz [18] gave a complete characterization of a function class that can be realized by a two-party protocol perfectly secure against a semi-honest adversary, and Chor and Kushilevitz [7] gave a complete characterization of a boolean function that can be realized by a perfectly secure multi-party protocol in the dishonest-majority and


© Keitaro Hiwatashi and Koji Nuida;
licensed under Creative Commons License CC-BY 4.0

semi-honest adversary setting. From their result, most functions (including even simple functions such as the AND function) cannot be securely realized by a multi-party protocol in the dishonest-majority setting. However, if we are allowed to use *correlated randomness* (CR, in short), any function can be securely realized. Indeed, by using additive secret sharing and Beaver Triple [2], we can securely compute any boolean circuits. For simplicity, we focus on the two-party case and consider a semi-honest adversary in this paper.

The CR model, also known as a preprocessing model, is a model where we are allowed to use CR, which is a randomness independent of an input of a function. In the CR model, a protocol is divided into two phases: the offline phase and the online phase. In the offline phase, CR is generated and distributed to the parties. In the online phase, the parties securely compute a function with their input and the CR distributed in the offline phase. In most cases, the online phase consists of lightweight computation and is fast enough, and therefore many recent works (e.g., [1, 5, 6, 9, 17]) adopted the CR model. Some papers (e.g., [1, 5, 6]) assume that a trusted dealer generates and distributes CR in the offline phase. In the CR model, three kinds of complexities are mainly considered: the size of CR, the number of rounds of the online phase, and the communication complexity of the online phase. By using Beaver Triple, we can construct a secure protocol for a boolean circuit C with $O(s)$-bit CR, $O(\mathrm{depth}(C))$ rounds, and $O(s)$-bit communication complexity, where s is the size of C and $\mathrm{depth}(C)$ is the depth of C. It is a major open problem whether it is possible to make the communication complexity sublinear in the circuit size.

Ishai et al. [16] partially solved the above open problem by developing a one-time truth table. In their protocol, the communication complexity is independent of the circuit size and is linear in the input length. Furthermore, their protocol using a one-time truth table achieves online optimality, i.e., the number of rounds is one round and the communication complexity is the same as the input length. However, the size of CR of their protocol is exponential in the input length.[1] Indeed, their protocol needs $O(N^2)$-bit CR where N is the cardinality of their input domain (which is exponential in the input length). Beimel et al. [4], the full version of [3], reduced the size of CR to $O(N^{1/2})$ bits, at the price of increasing the number of rounds to two rounds and the communication complexity to $O(N^{1/2})$ bits. Although there might have to be some trade-off among the three kinds of complexities mentioned above, the quantitative property of such a trade-off has not been well investigated in the literature. For example, focusing on Ishai et al.'s protocol, it is even not known whether the size of CR can be reduced to $o(N^2)$ bits while keeping the single round and optimal communication complexity.

1.1 Our Contributions

In this paper, we show some trade-offs by giving lower bounds of the size of CR in *online-restricted* settings where online communication cost (i.e., the number of rounds and the communication complexity) is restricted. Here we assume "shared-output" setting that the outputs (y_0, y_1) by two parties satisfy that the function value is reconstructed by $y_0 + y_1$ where '+' is some additive group operation. We discuss this setting in Section 1.2.

More concretely, we give (exponential) lower bounds of the size of CR in two types of online-restricted settings: online-optimal setting and communication-optimal setting. As described above, "online-optimal" means that the number of rounds is one round and the communication

[1] It is still an open problem whether it is possible to make the communication complexity sublinear in the circuit size in the setting that the time complexity (and therefore the size of CR) is polynomial in the input length.

complexity is the same as the input length. On the other hand, "communication-optimal" means that we allow multiple rounds but the total size of messages sent by a party to the other party during the multiple rounds is still equal to the size of the input of the party.

Our results are summarized as follows, where N is the cardinality of the domain $\mathcal{X}$:

1. We prove that there exists a function $f\colon \mathcal{X} \times \mathcal{X} \to \{0,1\}^2$ such that any *online-optimal* perfectly secure two-party protocol for f needs CR of at least $(N-1)^2 = \Omega(N^2)$ bits.
2. We prove that there exists a function $f\colon \mathcal{X} \times \mathcal{X} \to \{0,1\}^2$ such that any *communication-optimal* perfectly secure two-party protocol for f needs CR of at least $N - 1 = \Omega(N)$ bits.

The first result implies that the one-time truth table [16], which is an online-optimal secure two-party protocol with $O(N^2)$-bit CR, is asymptotically optimal among online-optimal ones for general functions in terms of the size of CR.

1.2 Shared-output vs. Plain-output

Some papers, including Ishai et al. [16] and Beimel et al. [4], considered the "plain-output" setting, in which both parties output the function value itself, not the share of the function value. The shared-output setting is more general in the sense that shared-output protocols can be converted into plain-output protocols by adding a reconstruction step. The protocols in both Ishai et al. [16] and Beimel et al. [4] contained shared-output protocols in the sense that their protocols can be divided into two steps: Both protocols (Fig.1 in [16] and Theorem D.1 in [4]) compute the share of the function value first, and then reconstruct the function value by exchanging the shares.

Furthermore, the shared-output setting is suitable for the situation where the protocol is used as a subprotocol in another MPC protocol since the shared output does not leak any information about the function value itself. Due to this composability, many recent MPC protocols (e.g., [1, 5, 6, 24]) adopt the shared-output setting.

1.3 Related Work

The prior work most relevant to our setting is a combination of [6] and [14]. In [6], Boyle et al. showed that distributed point functions (DPF) can be constructed from an online-optimal (shared-output) equality protocol. In more detail, they showed that given an online-optimal shared-output equality protocol with r-bit CR, a DPF scheme with $O(r)$-bit key size can be constructed[2]. On the other hand, as Gilboa et al. [14] mentioned, the key size of an information-theoretic DPF scheme is at least $2^{\Omega(\log N)} = N^{\Omega(1)}$ bits, where N corresponds to the cardinality of the domain of point functions. Combining it with the reduction by Boyle et al., it can be proved that the size of CR of an online-optimal protocol for the equality function $EQ\colon [N] \times [N] \to \{0,1\}$ is at least $N^{\Omega(1)}$ bits. To the best of our knowledge, this is the only prior result showing an exponential lower bound of the size of CR.

There are several results on the randomness complexity not on the size of CR (e.g., [13, 15, 19, 20, 21, 22, 23]). We note that they consider MPC in the plain model (not in the CR model) and with more than two parties. There are also several results on the communication complexity in multi-party computation (e.g., [8, 10, 11, 12]).

[2] Though they only considered the computational security, their reduction can be applied also in the information-theoretic setting.

1.4 Organization

We provide the notations used in this paper and the definitions of online- or communication-optimal secure two-party protocols in Section 2. We provide a technical overview in Section 3. In Section 4, we prove the lower bound in the online-optimal setting. We prove the lower bound in the communication-optimal setting in Section 5.

2 Preliminaries

2.1 Notations

For an integer N, let $[N]$ denote the set $\{0, 1, \ldots, N-1\}$. Let P_0 and P_1 be the parties participating in a two-party protocol. We use $\Delta^{k\times\ell}(i,j)$ to denote the $k \times \ell$ matrix for which the (i,j)-th element is 1 and the other elements are 0. We let $\mathbb{G}$ denote an Abelian group, $+$ denote the operation on $\mathbb{G}$, and 0 denote the identity element of $\mathbb{G}$. For a boolean value $b \in \{0,1\}$, let $\bar{b}$ be the negation of b. For a matrix M, $M[x,y]$ denotes the (x,y)-th element of M.

2.2 Online-Optimal Protocols

▶ **Definition 1.** *An online-optimal secure two-party protocol for $f\colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathbb{G}$ with correlated randomness $\mathcal{CR} \subseteq \mathcal{R}_0 \times \mathcal{R}_1$ consists of three algorithms* $(\mathsf{Gen}, \mathsf{Msg}, \mathsf{Eval})$ *with following syntax:*

- Gen*:* Gen *outputs a correlated randomness $(r_0, r_1) \in \mathcal{CR} \subseteq \mathcal{R}_0 \times \mathcal{R}_1$ without taking any input.*
- Msg*: Taking party id $b \in \{0,1\}$, input $x \in \mathcal{X}_b$, and randomness $r \in \mathcal{R}_b$,* Msg *(deterministically) outputs a message $m \in \mathcal{M}_b$.*
- Eval*: Taking party id $b \in \{0,1\}$, input $x \in \mathcal{X}_b$, randomness $r \in \mathcal{R}_b$, and message $m \in \mathcal{M}_{\bar{b}}$,* Eval *(deterministically) outputs $g \in \mathbb{G}$.*

satisfying the following three requirements:

- *Optimality: For $b \in \{0,1\}$, the size of $\mathcal{M}_b$ is equal to the size of $\mathcal{X}_b$.*
- *Correctness: For all $(x_0, x_1) \in \mathcal{X}_0 \times \mathcal{X}_1$,*

$$\Pr\left[\, g_0 + g_1 = f(x_0,x_1) \;\middle|\; \begin{array}{c} (r_0,r_1) \leftarrow \mathsf{Gen}, \\ m_0 \leftarrow \mathsf{Msg}(0,x_0,r_0), \\ m_1 \leftarrow \mathsf{Msg}(1,x_1,r_1), \\ g_0 \leftarrow \mathsf{Eval}(0,x_0,r_0,m_1), \\ g_1 \leftarrow \mathsf{Eval}(1,x_1,r_1,m_0) \end{array} \right] = 1.$$

- *Security: For $b \in \{0,1\}$, the distribution of $\{(r_b, \mathsf{Msg}(\bar{b}, x, r_{\bar{b}}))\}_{(r_0,r_1)\leftarrow\mathsf{Gen}}$ is independent of $x \in \mathcal{X}_{\bar{b}}$.*

Without loss of generality, we assume that the randomness space is not redundant. That is, we assume that the probability $\Pr[(r_0,r_1) \leftarrow \mathsf{Gen}]$ is positive for all $(r_0,r_1) \in \mathcal{CR}$ and that for all $r_0 \in \mathcal{R}_0$ ($r_1 \in \mathcal{R}_1$, resp.), there exists $r_1 \in \mathcal{R}_1$ ($r_0 \in \mathcal{R}_0$, resp.) such that $(r_0,r_1) \in \mathcal{CR}$.

2.3 Communication-Optimal Protocols

▶ **Definition 2.** *A (T-round) communication-optimal secure two-party protocol for $f\colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathbb{G}$ with correlated randomness $\mathcal{CR} \subseteq \mathcal{R}_0 \times \mathcal{R}_1$ consists of algorithms* $(\mathsf{Gen}, \mathsf{Msg}, \mathsf{Eval})$ *with following syntax:*

- Gen*:* Gen *outputs a correlated randomness* $(r_0, r_1) \in \mathcal{CR} \subseteq \mathcal{R}_0 \times \mathcal{R}_1$ *without taking any input.*
- Msg*: Taking* $(x_0, r_0) \in \mathcal{X}_0 \times \mathcal{R}_0$ *and* $(x_1, r_1) \in \mathcal{X}_1 \times \mathcal{R}_1$*,* Msg *(deterministically) outputs messages* $(\mathrm{mes}_1, \ldots, \mathrm{mes}_T) \in M^1 \times \cdots \times M^T$*. Here* mes_i *is a message sent to* $P_{i \bmod 2}$ *from* $P_{i+1 \bmod 2}$ *which is determined by* $P_{i+1 \bmod 2}$*'s input* $x_{i+1 \bmod 2}$*,* $P_{i+1 \bmod 2}$*'s randomness* $r_{i+1 \bmod 2}$ *and the messages* $(\mathrm{mes}_1, \ldots, \mathrm{mes}_{i-1})$ *exchanged so far.*
- Eval*: Taking party id* $b \in \{0, 1\}$*, input* $x \in \mathcal{X}_b$*, randomness* $r \in \mathcal{R}_b$*, and messages* $(\mathrm{mes}_1, \ldots, \mathrm{mes}_T)$ *exchanged so far,* Eval *(deterministically) outputs* $g \in \mathbb{G}$*.*

satisfying the following three requirements:

- *Optimality: For* $b \in \{0, 1\}$*, the size of the message space* $\mathcal{M}_b$ *is equal to that of the input space* $\mathcal{X}_b$*, where* $\mathcal{M}_0 = M^1 \times M^3 \times M^5 \times \cdots$ *and* $\mathcal{M}_1 = M^2 \times M^4 \times M^6 \times \cdots$*.*
- *Correctness: For all* $(x_0, x_1) \in \mathcal{X}_0 \times \mathcal{X}_1$*,*

$$\Pr\left[g_0 + g_1 = f(x_0, x_1) \;\middle|\; \begin{array}{c} (r_0, r_1) \leftarrow \mathsf{Gen}, \\ (\mathrm{mes}_1, \ldots, \mathrm{mes}_T) \leftarrow \mathsf{Msg}(x_0, r_0, x_1, r_1), \\ g_b \leftarrow \mathsf{Eval}(b, x_b, r_b, (\mathrm{mes}_1, \ldots, \mathrm{mes}_T)) \end{array} \right] = 1.$$

- *Security: For* $b \in \{0, 1\}$*, the distribution of* $\{(r_b, \mathsf{Msg}(x_0, r_0, x_1, r_1))\}_{(r_0, r_1) \leftarrow \mathsf{Gen}}$ *is independent of* $x_{\bar{b}} \in \mathcal{X}_{\bar{b}}$*.*

As an online-optimal secure two-party protocol, we assume that the randomness space is not redundant. In our definition, P_0 is the first party who sends a message. Note that a two-party protocol where P_1 is the first party who sends a message can be reduced to a protocol where P_0 is the first party who sends a message by letting the first message space M_1 be a singleton.

2.4 Non-Redundant Functions

Throughout this paper, we consider non-redundant functions in the following sense:

▶ **Definition 3.** *We say that a function* $f\colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathbb{G}$ *is non-redundant for* P_0 *if* $f(x_0, \cdot) - f(x_0', \cdot)\colon \mathcal{X}_1 \to \mathbb{G}$ *is not constant for all* $x_0 \neq x_0' \in \mathcal{X}_0$*; a function* $f\colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathbb{G}$ *is non-redundant for* P_1 *if* $f(\cdot, x_1) - f(\cdot, x_1')\colon \mathcal{X}_0 \to \mathbb{G}$ *is not constant for all* $x_1 \neq x_1' \in \mathcal{X}_1$*; and a function* $f\colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathbb{G}$ *is non-redundant if* f *is non-redundant for* P_0 *and* P_1*.*

A two-party protocol for a redundant (i.e., not non-redundant) function f is reducible to a two-party protocol for a non-redundant function f' without any overhead. See Appendix A for more details.

3 Technical Overview

Similar research (e.g., [11, 12]) on the lower bounds for communication or randomness complexity mainly focuses on information entropy, e.g. the Shannon entropy. However, in such arguments, it is difficult to effectively handle the correctness requirement as a restraint condition, and thus the obtained bounds may not be tight in general. We consider this may be the main reason why strict bounds for our setting have not been obtained so far, and to overcome this issue, in this work, we instead directly utilize the algebraic aspect of the correctness requirement. This may require a more complicated argument than the entropy-based approach, but it would allow the correctness requirement to be fully utilized in the derivation of the tight lower bound. We provide a more detailed explanation of our approach in the following subsections.

3.1 The Case of Online-Optimal Setting

We give the $\Omega(N^2)$-bit lower bound for some function $f\colon [N] \times [N] \to \{0,1\}^2$ in two steps:

1. By the security, correctness and optimality requirements, we show that the following equation (hereinafter referred to as the *correctness equation*) holds: For all $(r_0, r_1) \in \mathcal{CR}$,

$$A_{0,r_0} + A_{1,r_1} = P_{0,r_0}^{\mathrm{T}} F P_{1,r_1}, \tag{1}$$

   where A_{b,r_b} is an $N \times N$ matrix determined by r_b, P_{b,r_b} is an $N \times N$ permutation matrix determined by r_b, and F is an $N \times N$ matrix whose (i,j)-th element is equal to $f(i,j)$.
2. For some $r_0 \in \mathcal{R}_0$, we prove that the size of the set $\{A_{0,r_0} - A_{0,r_0'}\}_{r_0' \in \mathcal{R}_0}$ is at least $2^{(N-1)^2}$. This implies that $\log |\mathcal{R}_0| \geq (N-1)^2$ and proves the $\Omega(N^2)$-bit lower bound.

Roughly speaking, each element of A_{b,r_b} corresponds to an output of Eval and P_{b,r_b} is a permutation matrix corresponding to Msg.

3.1.1 The First Step

The correctness equation Eq.(1) is deduced from the correctness requirement and the fact that $\mathsf{Msg}(b, \cdot, r_b)\colon \mathcal{X}_b \to \mathcal{M}_b$ is bijective for all $b \in \{0,1\}$ and $r_b \in \mathcal{R}_b$ (Lemma 6). First, we give an informal proof of the fact that Msg is bijective. Since the size of the domain $\mathcal{X}_b$ and the range $\mathcal{M}_b$ are the same, it is enough to prove that the function is injective. Without loss of generality, we set $b = 0$. Suppose on the contrary that there exist $x_0 \neq x_0' \in \mathcal{X}_0$ such that $\mathsf{Msg}(0, x_0, r_0) = \mathsf{Msg}(0, x_0', r_0) = m_0$. This assumption implies that for any input x_1 of P_1, P_1's output of Eval for the case of P_0's input being x_0 is the same as that for the case of x_0'. Therefore, from the correctness requirement, $f(x_0, x_1) - f(x_0', x_1)$ is equal to $\mathsf{Eval}(0, x_0, r_0, m_1) - \mathsf{Eval}(0, x_0', r_0, m_1)$ where m_1 is P_1's message with some $r_1 \in \mathcal{R}_1$ satisfying $(r_0, r_1) \in \mathcal{CR}$. The former depends on x_1 by the non-redundancy of f, while the latter can be computed solely by P_0, contradicting the security requirement that P_0 should not obtain any information on x_1.

Since $\mathsf{Msg}(b, \cdot, r_b)$ is bijective, the input of P_b can be determined by the correlated randomness r_b and the message m_b sent from P_b. Therefore, P_b's output of Eval can be computed from the messages (m_0, m_1) and the randomness r_b. Let A_{b,r_b} be an $N \times N$ matrix whose (m_0, m_1)-th element is equal to P_b's output of Eval when the messages and the randomness are (m_0, m_1) and r_b, respectively. From the correctness requirement, for all (m_0, m_1), $A_{0,r_0}[m_0, m_1] + A_{1,r_1}[m_0, m_1]$ is equal to the entry of the matrix F at the $\pi_{0,r_0}^{-1}(m_0)$-th row and the $\pi_{1,r_1}^{-1}(m_1)$-th column, where π_{b,r_b} denotes the bijection $\mathsf{Msg}(b, \cdot, r_b)$. This implies the correctness equation Eq.(1), where P_{0,r_0} and P_{1,r_1} are permutation matrices corresponding to π_{0,r_0}^{-1} and π_{1,r_1}^{-1}, respectively.

3.1.2 The Second Step

For simplicity, let F be $\Delta^{N \times N}(0,0)$, $\mathbb{G}$ be $\{0,1\}$ and the operation on $\mathbb{G}$ be XOR. Then, the right term of the correctness equation Eq.(1) is equal to $\Delta^{N \times N}(\mathsf{Msg}(0,0,r_0), \mathsf{Msg}(1,0,r_1))$. The notable point is that, given $r_0 \in \mathcal{R}_0$, we can choose the value of $\mathsf{Msg}(1,0,r_1)$ arbitrarily (Lemma 7). That is, for all $m_1 \in \mathcal{M}_1$, there exists $r_1 \in \mathcal{R}_1$ such that $(r_0, r_1) \in \mathcal{CR}$ and $\mathsf{Msg}(1,0,r_1) = m_1$ (otherwise, the fact that P_0 receives P_1's message m_1 would tell P_0 that P_1's input is not 0, contradicting the security).

Let $r_0 \in \mathcal{R}_0$ satisfy $\mathsf{Msg}(0,0,r_0) = 0$. From the property described above, for all $m_0 \in \mathcal{M}_0 \backslash \{0\}$ and $m_1 \in \mathcal{M}_1 \backslash \{0\}$, there exist $r'_0, r''_0 \in \mathcal{R}_0$ and $r_1, r'_1 \in \mathcal{R}_1$ such that $(r_0, r_1), (r'_0, r_1), (r'_0, r'_1), (r''_0, r'_1) \in \mathcal{CR}$, $\mathsf{Msg}(1,0,r'_1) = m_1$, $\mathsf{Msg}(0,0,r'_0) = m_0$, $\mathsf{Msg}(1,0,r'_1) = 0$, and $\mathsf{Msg}(0,0,r''_0) = 0$. Taking the sum of both sides of the four correctness equations, we have

$$A_{0,r_0} + A_{0,r''_0} = \Delta^{N\times N}(0,m_1) + \Delta^{N\times N}(m_0,m_1) + \Delta^{N\times N}(m_0,0) + \Delta^{N\times N}(0,0).$$

Note that $A + A = 0$ since the operation is XOR. This implies that the bottom right corner of $A_{0,r_0} + A_{0,r''_0}$ is equal to $\Delta^{(N-1)\times(N-1)}(m_0 - 1, m_1 - 1)$ (Theorem 9). Taking the sum of these equations with various values of m_0 and m_1, it follows that for all $M \in \{0,1\}^{(N-1)\times(N-1)}$, there exists $r'_0 \in \mathcal{R}_0$ such that $\mathsf{Msg}(0,0,r'_0) = 0$ and the bottom right corner of $A_{0,r_0} + A_{0,r'_0}$ is equal to M (Corollary 10). This implies that the size of $\{A_{0,r_0} + A_{0,r'_0}\}_{r'_0 \in \mathcal{R}_0}$ is at least the size of $\{0,1\}^{(N-1)\times(N-1)}$, i.e., $2^{(N-1)^2}$ and this proves the $\Omega(N^2)$-bit lower bound.

3.2 The Case of Communication-Optimal Setting

The basic approach to proving the $\Omega(N)$-bit lower bound is the same as in the online-optimal setting. The main difference is that the message sent from one party at the second or later round may depend on the other party's input or randomness. Nevertheless, the situation is still similar due to the following facts:

1. The map $T_{r_0,r_1} \colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathcal{M}_0 \times \mathcal{M}_1$, which maps the pair of inputs to the transcript when the correlated randomness is (r_0, r_1), is bijective. (Lemma 13)
2. The input of P_b can be determined by the correlated randomness r_b and the transcript (m_0, m_1), and therefore, P_b's output of Eval can be computed from the transcript (m_0, m_1) and the randomness r_b. (Lemma 15)

From these facts, even in the present setting, the correctness equation (with a slight modification on the right side) holds: $A_{0,r_0} + A_{1,r_1} = T_{r_0,r_1} \circ F$, where $T_{r_0,r_1} \circ F$ is an $\mathcal{M}_0 \times \mathcal{M}_1$ matrix whose (m_0, m_1)-th element is equal to the $T^{-1}_{r_0,r_1}(m_0, m_1)$-th element of F.

Let F be $\Delta^{N\times N}(0,0)$. Unlike the online-optimal setting, now the right side of the correctness equation is equal to $\Delta^{N\times N}(T_{r_0,r_1}(0,0))$. A notable point is that each of the row and column entries of $T_{r_0,r_1}(0,0)$ depends on both r_0 and r_1, in contrast to the online-optimal setting where the row entry $\mathsf{Msg}(0,0,r_0)$ (resp. the column entry $\mathsf{Msg}(1,0,r_1)$) depends only on r_0 (resp. r_1). However, we can still somehow control the value of $T_{r_0,r_1}(0,0)$ (Lemma 12 and Lemma 14), and we can set the $(N-1) \times 1$ submatrix at the bottom left corner of $A_{0,r_0} + A_{0,r'_0}$ arbitrarily by varying r'_0. This proves the $\Omega(N)$-bit lower bound.

4 The Case of Online-Optimal Setting

In this section, we prove the optimality of Ishai et al.'s protocol [16] among online-optimal two-party protocols in terms of the size of CR for the "worst" function. In Section 4.1, we give a matrix representation of the three requirements for an online-optimal secure two-party protocol given in Section 2 (Theorem 8). In Section 4.2, we give a function whose domain is $[N] \times [N]$ such that any online-optimal two-party protocol for f needs $\Omega(N^2)$-bit CR.

4.1 Matrix Representation

Throughout this subsection, we let $(\mathsf{Gen}, \mathsf{Msg}, \mathsf{Eval})$ denote an online-optimal secure two-party protocol for $f \colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathbb{G}$ with correlated randomness $\mathcal{CR} \subseteq \mathcal{R}_0 \times \mathcal{R}_1$. For $b \in \{0,1\}$, $x \in \mathcal{X}_b$, and $m \in \mathcal{M}_b$, let $\mathcal{R}_{b,x,m} = \{r \in \mathcal{R}_b \mid \mathsf{Msg}(b,x,r) = m\}$.

First, we give four lemmas for the matrix representation:

▶ **Lemma 4.** *For all $b \in \{0,1\}$ and $x \in \mathcal{X}_b$, the following hold:*
- $\mathcal{R}_{b,x,m} \cap \mathcal{R}_{b,x,m'} = \varnothing$ *for all* $m \neq m' \in \mathcal{M}_b$.
- $\cup_{m \in \mathcal{M}_b} \mathcal{R}_{b,x,m} = \mathcal{R}_b$.

Proof. These two statements are deduced from the fact that r is in $\mathcal{R}_{b,x,m}$ if and only if m is equal to $\mathsf{Msg}(b,x,r)$. ◀

▶ **Lemma 5.** *For all $b \in \{0,1\}$ and $m \in \mathcal{M}_b$, the following hold:*
- $\mathcal{R}_{b,x,m} \cap \mathcal{R}_{b,x',m} = \varnothing$ *for all* $x \neq x' \in \mathcal{X}_b$.
- $\cup_{x \in \mathcal{X}_b} \mathcal{R}_{b,x,m} = \mathcal{R}_b$.

Proof. We fix $b = 0$ in this proof. In the case of $b = 1$, the statement can be proved similarly.

First, we prove the first statement. Suppose on the contrary that an $r \in \mathcal{R}_{0,x,m} \cap \mathcal{R}_{0,x',m}$ exists. By the non-redundancy of the randomness space, there is an $r' \in \mathcal{R}_1$ such that $(r,r') \in \mathcal{CR}$. Now it suffices to show that there exist $y, y' \in \mathcal{X}_1$ such that $y \neq y'$ and $\mathsf{Msg}(1,y',r'') \neq \mathsf{Msg}(1,y,r')$ for any $r'' \in \mathcal{R}_1$ with $(r,r'') \in \mathcal{CR}$; indeed, this implies that $(r, \mathsf{Msg}(1,y,r'))$ belongs to $\{(r^*, \mathsf{Msg}(1,y,r^{**}))\}_{(r^*,r^{**}) \in \mathcal{CR}}$ but not to $\{(r^*, \mathsf{Msg}(1,y',r^{**}))\}_{(r^*,r^{**}) \in \mathcal{CR}}$, which contradicts the security requirement. To show the claim, suppose on the contrary that for any $y, y' \in \mathcal{X}_1$ with $y \neq y'$, there is an $r'' \in \mathcal{R}_1$ such that $(r,r'') \in \mathcal{CR}$ and $\mathsf{Msg}(1,y',r'') = \mathsf{Msg}(1,y,r')$. From the correctness requirement, we have

$$\mathsf{Eval}(0,x,r,\mathsf{Msg}(1,y,r')) + \mathsf{Eval}(1,y,r',\mathsf{Msg}(0,x,r)) = f(x,y),$$
$$\mathsf{Eval}(0,x',r,\mathsf{Msg}(1,y,r')) + \mathsf{Eval}(1,y,r',\mathsf{Msg}(0,x,r)) = f(x',y)$$

(note that now $\mathsf{Msg}(0,x',r) = m = \mathsf{Msg}(0,x,r)$ by the choice of r), and therefore

$$\mathsf{Eval}(0,x,r,\mathsf{Msg}(1,y,r')) - \mathsf{Eval}(0,x',r,\mathsf{Msg}(1,y,r')) = f(x,y) - f(x',y). \tag{2}$$

By the same argument for (y',r'') instead of (y,r'), we also have

$$\mathsf{Eval}(0,x,r,\mathsf{Msg}(1,y',r'')) - \mathsf{Eval}(0,x',r,\mathsf{Msg}(1,y',r'')) = f(x,y') - f(x',y'). \tag{3}$$

By the choice of r'', the left-hand sides of Equations (2) and (3) are equal, therefore we have $f(x,y) - f(x',y) = f(x,y') - f(x',y')$. Since $y \neq y'$ were arbitrary, this implies that the function $f(x,\cdot) - f(x',\cdot)$ on $\mathcal{X}_1$ is constant, contradicting the non-redundancy of f. Hence, we have $\mathcal{R}_{0,x,m} \cap \mathcal{R}_{0,x',m} = \varnothing$.

Next, we prove the second statement. Suppose that $\cup_{x \in \mathcal{X}_0} \mathcal{R}_{0,x,m} \subsetneq \mathcal{R}_0$. Then, we have

$$\begin{aligned}\sum_{m \in \mathcal{M}_0} \sum_{x \in \mathcal{X}_0} |\mathcal{R}_{0,x,m}| &= \sum_{m \in \mathcal{M}_0} |\cup_{x \in \mathcal{X}_0} \mathcal{R}_{0,x,m}| \ (\because \text{the first statement}) \\ &< \sum_{m \in \mathcal{M}_0} |\mathcal{R}_0| = |\mathcal{M}_0| \cdot |\mathcal{R}_0|.\end{aligned}$$

From Lemma 4, we have

$$\sum_{x \in \mathcal{X}_0} \sum_{m \in \mathcal{M}_0} |\mathcal{R}_{0,x,m}| = \sum_{x \in \mathcal{X}_0} |\mathcal{R}_0| = |\mathcal{X}_0| \cdot |\mathcal{R}_0|.$$

This means that $|\mathcal{X}_0| \cdot |\mathcal{R}_0| < |\mathcal{M}_0| \cdot |\mathcal{R}_0|$ and contradicts the optimality requirement. Hence, we have $\cup_{x \in \mathcal{X}_0} \mathcal{R}_{0,x,m} = \mathcal{R}_0$. ◀

▶ **Lemma 6.** *For all $b \in \{0,1\}$ and $r \in \mathcal{R}_b$, $\mathsf{Msg}(b,\cdot,r) \colon \mathcal{X}_b \to \mathcal{M}_b$ is a bijection.*

Proof. From Lemma 5, $\mathsf{Msg}(b, \cdot, r)$ is an injection. Since $|\mathcal{X}_b| = |\mathcal{M}_b|$ from the optimality requirement, the injective function $\mathsf{Msg}(b, \cdot, r)$ is a bijection. ◀

▶ **Lemma 7.** *For all $b \in \{0,1\}$, $r_{\bar{b}} \in \mathcal{R}_{\bar{b}}$, and $(x_b, m_b) \in \mathcal{X}_b \times \mathcal{M}_b$, there exists $r_b \in \mathcal{R}_b$ such that $(r_0, r_1) \in \mathcal{CR}$ and $\mathsf{Msg}(b, x_b, r_b) = m_b$.*

Proof. We fix $b = 0$ in this proof. In the case of $b = 1$, the statement can be proved similarly.

Let $r_1 \in \mathcal{R}_1$ and $(x_0, m_0) \in \mathcal{X}_0 \times \mathcal{M}_0$. By the non-redundancy of the randomness space, there is an $r^* \in \mathcal{R}_0$ with $(r^*, r_1) \in \mathcal{CR}$. By Lemma 6, there exists an $x' \in \mathcal{X}_0$ such that $\mathsf{Msg}(0, x', r^*) = m_0$ and therefore $(r_1, m_0) \in \{(r', \mathsf{Msg}(0, x', r))\}_{(r,r') \in \mathcal{CR}}$. Since this set is independent of x' from the security requirement, we also have $(r_1, m_0) \in \{(r', \mathsf{Msg}(0, x_0, r))\}_{(r,r') \in \mathcal{CR}}$, therefore there is an $r_0 \in \mathcal{R}_0$ such that $(r_0, r_1) \in \mathcal{CR}$ and $\mathsf{Msg}(0, x_0, r_0) = m_0$. This proves the statement. ◀

Then, we give a matrix representation of the three requirements for an online-optimal secure two-party protocol:

▶ **Theorem 8.** *Given an online-optimal secure two-party protocol $(\mathsf{Gen}, \mathsf{Msg}, \mathsf{Eval})$ for $f \colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathbb{G}$ with correlated randomness $\mathcal{CR} \subseteq \mathcal{R}_0 \times \mathcal{R}_1$, let F be an $\mathcal{X}_0 \times \mathcal{X}_1$ matrix whose (x_0, x_1)-th element is $f(x_0, x_1)$. Then, for any $b \in \{0,1\}$ and $r \in \mathcal{R}_b$, there exist an $\mathcal{M}_0 \times \mathcal{M}_1$ matrix $A_{b,r}$ and an $\mathcal{X}_b \times \mathcal{M}_b$ permutation matrix $P_{b,r}$ such that*

- *for all $(r_0, r_1) \in \mathcal{CR}$, $A_{0,r_0} + A_{1,r_1} = P_{0,r_0}^{\mathrm{T}} F P_{1,r_1}$ holds;*
- *for all $r_{\bar{b}} \in \mathcal{R}_{\bar{b}}$ and $(x_b, m_b) \in \mathcal{X}_b \times \mathcal{M}_b$, there exists $r_b \in \mathcal{R}_b$ such that $(r_0, r_1) \in \mathcal{CR}$ and $P_{b,r_b}[x_b, m_b] = 1$.*

Note that, roughly speaking, the optimality requirement corresponds to $P_{b,r}$ being a permutation matrix, and the correctness and security requirements correspond to the first and the second conditions of the theorem, respectively.

Proof. For $b \in \{0,1\}$, let E_{b,r_b} for $r_b \in \mathcal{R}_{r_b}$ be an $\mathcal{X}_b \times \mathcal{M}_{\bar{b}}$ matrix whose $(x_b, m_{\bar{b}})$-th element is equal to $\mathsf{Eval}(b, x_b, r_b, m_{\bar{b}})$. For $b \in \{0,1\}$, let P_{b,r_b} for $r_b \in \mathcal{R}_{r_b}$ be an $\mathcal{X}_b \times \mathcal{M}_b$ matrix whose $(x_b, \mathsf{Msg}(b, x_b, r_b))$-th element is 1 and other elements are 0. Since the $(x_b, x_{\bar{b}})$-th element of $E_{b,r_b} P_{\bar{b},r_{\bar{b}}}^{\mathrm{T}}$ is equal to $\mathsf{Eval}(b, x_b, r_b, \mathsf{Msg}(\bar{b}, x_{\bar{b}}, r_{\bar{b}}))$, the correctness requirement can be expressed by the following:

$$E_{0,r_0} P_{1,r_1}^{\mathrm{T}} + \left(E_{1,r_1} P_{0,r_0}^{\mathrm{T}}\right)^{\mathrm{T}} = F \tag{4}$$

for all $(r_0, r_1) \in \mathcal{CR}$. From Lemma 6, P_{b,r_b} is a permutation matrix and therefore P_{b,r_b}^{T} is its inverse. By multiplying P_{0,r_0}^{T} from the left (P_{1,r_1} from the right, resp.) to both sides of Equation (4), we have

$$P_{0,r_0}^{\mathrm{T}} E_{0,r_0} + E_{1,r_1}^{\mathrm{T}} P_{1,r_1} = P_{0,r_0}^{\mathrm{T}} F P_{1,r_1}. \tag{5}$$

Therefore, $(A_{0,r_0}, A_{1,r_1}) = (P_{0,r_0}^{\mathrm{T}} E_{0,r_0}, E_{1,r_1}^{\mathrm{T}} P_{1,r_1})$ satisfies the first condition of the statement.

The second condition of the statement is deduced from Lemma 7. ◀

4.2 Lower Bound

We prove the $\Omega(N^2)$-bit lower bound for the function $f \colon [N] \times [N] \to \{0,1\}^2$ defined as follows:

$$f(x_0, x_1) = \begin{cases} 11 & (x_0 = x_1 = 0) \\ 01 & (x_0 = x_1 \neq 0) \\ 00 & (\text{otherwise}). \end{cases}$$

That is, we prove that any online-optimal secure two-party protocol for f needs $\Omega(N^2)$-bit CR. Note that the operation '+' on $\{0,1\}^2$ is bitwise XOR here and that f is non-redundant.

In the rest of this section, we write $[N]$ instead of $\mathcal{X}_b$ and $\mathcal{M}_b$. Without loss of generality, we consider the lower bound for the size of P_0's CR (i.e., $\log|\mathcal{R}_0|$).

We use the notation A_{b,r_b} and P_{b,r_b} for representing $N \times N$ matrices whose existence is guaranteed by Theorem 8. Since the operation on $\{0,1\}^2$ is bitwise XOR, Equation (5) holds even if we focus on the first bit of each element of A_{b,r_b} and F. Therefore, we focus on the first bit and use the same notation A_{b,r_b} and F. Then we have $F = \Delta^{N\times N}(0,0)$ in the current setting.

First, we prove the following theorem:

▶ **Theorem 9.** *Suppose that $r_0 \in \mathcal{R}_0$ satisfies $\mathsf{Msg}(0,0,r_0) = 0$. Then, for all $i,j \in [N-1]$, there exists an $r'_0 \in \mathcal{R}_0$ such that*

- $\mathsf{Msg}(0,0,r'_0) = 0$,
- *the $(N-1)\times(N-1)$ submatrix in the bottom right corner of $A_{0,r_0} + A_{0,r'_0}$ is equal to $\Delta^{(N-1)\times(N-1)}(i,j)$.*

Proof. From the definition of P_{b,r_b} (see the proof of Theorem 8), the right term of Equation (5) is equal to $\Delta^{N\times N}(\mathsf{Msg}(0,0,r_0), \mathsf{Msg}(1,0,r_1))$. From Theorem 8, there exists an $r_1 \in \mathcal{R}_1$ such that $(r_0,r_1) \in \mathcal{CR}$ and $\mathsf{Msg}(1,0,r_1) = j+1$, and there exists $r''_0 \in \mathcal{R}_0$ such that $(r''_0,r_1) \in \mathcal{CR}$ and $\mathsf{Msg}(0,0,r''_0) = i+1$. From the property mentioned at the beginning, we have

$$A_{0,r_0} + A_{1,r_1} = \Delta^{N\times N}(0,j+1) \text{ and } A_{0,r''_0} + A_{1,r_1} = \Delta^{N\times N}(i+1,j+1),$$

and therefore

$$\begin{aligned} A_{0,r_0} + A_{0,r''_0} &= (A_{0,r_0} + A_{1,r_1}) + (A_{0,r''_0} + A_{1,r_1}) \\ &= \Delta^{N\times N}(0,j+1) + \Delta^{N\times N}(i+1,j+1). \end{aligned}$$

Similarly, there exist an $r'_1 \in \mathcal{R}_1$ and an $r'_0 \in R_0$ such that $(r''_0,r'_1) \in \mathcal{CR}$, $\mathsf{Msg}(1,0,r'_1) = 0$, $(r'_0,r'_1) \in \mathcal{CR}$, and $\mathsf{Msg}(0,0,r'_0) = 0$. Then, we have

$$A_{0,r''_0} + A_{1,r'_1} = \Delta^{N\times N}(i+1,0) \text{ and } A_{0,r'_0} + A_{1,r'_1} = \Delta^{N\times N}(0,0),$$

and

$$A_{0,r''_0} + A_{0,r'_0} = (A_{0,r''_0} + A_{1,r'_1}) + (A_{0,r'_0} + A_{1,r'_1}) = \Delta^{N\times N}(i+1,0) + \Delta^{N\times N}(0,0).$$

Hence, we have

$$A_{0,r_0} + A_{0,r'_0} = \Delta^{N\times N}(0,j+1) + \Delta^{N\times N}(i+1,j+1) + \Delta^{N\times N}(i+1,0) + \Delta^{N\times N}(0,0).$$

Especially, the $(N-1)\times(N-1)$ submatrix in the bottom right corner of $A_{0,r_0} + A_{0,r'_0}$ is equal to $\Delta^{(N-1)\times(N-1)}(i,j)$. Therefore, r'_0 satisfies the condition of the statement. ◀

Using Theorem 9 sequentially, we have the following corollary:

▶ **Corollary 10.** *Suppose that $r_0 \in \mathcal{R}_0$ satisfies $\mathsf{Msg}(0,0,r_0) = 0$. Then, for all $M \in \{0,1\}^{(N-1)\times(N-1)}$, there exists an $r'_0 \in \mathcal{R}_0$ such that*

- $\mathsf{Msg}(0,0,r'_0) = 0$,
- *the $(N-1)\times(N-1)$ submatrix in the bottom right corner of $A_{0,r_0} + A_{0,r'_0}$ is equal to M.*

Proof. For $r \in \mathcal{R}_0$, we use the notation $A'_{0,r}$ for the $(N-1) \times (N-1)$ submatrix in the bottom right corner of $A_{0,r}$. Let I be the set of indices where the element of M is equal to 1, i.e., $I = \{(i,j) \in [N-1] \times [N-1] \mid M[i,j] = 1\}$. Let $M_k = \Delta^{(N-1)\times(N-1)}(i_k, j_k)$ for $k \geq 0$, where (i_k, j_k) is the k-th element of I (in some ordering). We define the sequence $r_{0,0}, r_{0,1}, \ldots, r_{0,|I|}$ as follows:

- $r_{0,0} = r_0$.
- For $k \geq 1$, $r_{0,k}$ is an element of $\mathcal{R}_0$ such that $A'_{0,r_{0,k-1}} + A'_{0,r_{0,k}}$ is equal to M_{k-1} and $\mathsf{Msg}(0,0,r_{0,k})$ is equal to 0. The existence of such $r_{0,k}$ is guaranteed by Theorem 9.

We have

$$A'_{0,r_{0,0}} + A'_{0,r_{0,|I|}} = \sum_{k=1}^{|I|} (A'_{0,r_{0,k-1}} + A'_{0,r_{0,k}}) = \sum_{k=1}^{|I|} M_{k-1} = M,$$

and therefore $r'_0 = r_{0,|I|}$ satisfies the condition of the statement. ◀

The lower bound of the size of P_0's CR is derived from Corollary 10:

▶ **Corollary 11.** *The size of CR delivered to P_0 is $\Omega(N^2)$ bits. More concretely, it is greater than or equal to $(N-1)^2$ bits.*

Proof. Let $r_0 \in \mathcal{R}_0$ satisfy $\mathsf{Msg}(0,0,r_0) = 0$. (The existence of such r_0 is guaranteed by Lemma 7.) From Corollary 10, the following inequality holds:

$$\left|\{A_{0,r_0} + A_{0,r'_0}\}_{r'_0 \in \mathcal{R}_0}\right| \geq \left|\{0,1\}^{(N-1)\times(N-1)}\right|.$$

Since the left term of the above inequality is upper-bounded by $|\mathcal{R}_0|$, we have

$$|\mathcal{R}_0| \geq \left|\{0,1\}^{(N-1)\times(N-1)}\right| = 2^{(N-1)^2}.$$

Therefore, the size of P_0's CR is greater than or equal to $(N-1)^2$ bits. ◀

5 The Case of Communication-Optimal Setting

In this section, we prove the $\Omega(N)$-bit lower bound for the size of CR of a communication-optimal two-party protocol for the concrete function f given in Section 4.2.

We give a matrix representation of the three requirements for a communication-optimal secure two-party protocol in Section 5.1 (Theorem 16). In Section 5.2, we prove that any communication-optimal two-party protocol for f given in Section 4.2 needs $\Omega(N)$-bit CR.

5.1 Matrix Representation

Throughout this subsection, we let $(\mathsf{Gen}, \mathsf{Msg}, \mathsf{Eval})$ denote a communication-optimal secure two-party protocol for $f\colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathbb{G}$ with correlated randomness $\mathcal{CR} \subseteq \mathcal{R}_0 \times \mathcal{R}_1$. For $(r_0, r_1) \in \mathcal{R}_0 \times \mathcal{R}_1$, let $T_{r_0,r_1}\colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathcal{M}_0 \times \mathcal{M}_1$ be a function such that $T_{r_0,r_1}(x_0, x_1) = (m_0, m_1)$, where m_b is a message which P_b sends to $P_{\bar{b}}$ in the online phase whose input (CR, resp.) is (x_0, x_1) ((r_0, r_1), resp.). That is, m_0 is equal to $(\mathrm{mes}_1, \mathrm{mes}_3, \ldots)$ and m_1 is equal to $(\mathrm{mes}_2, \mathrm{mes}_4, \ldots)$, where $(\mathrm{mes}_1, \mathrm{mes}_2, \ldots) = \mathsf{Msg}(x_0, r_0, x_1, r_1)$. Note that the message m_b which P_b sends to $P_{\bar{b}}$ is uniquely determined by $(x_b, r_b, m_{\bar{b}})$, where x_b is P_b's input, r_b is P_b's CR, and $m_{\bar{b}}$ is a message sent to P_b by $P_{\bar{b}}$. We define a function $g^b_{x_b,r_b}\colon \mathcal{M}_{\bar{b}} \to \mathcal{M}_b$ based on the above correspondence. Let $S^0_{x_0,r_0}$ be the set $\{(g^0_{x_0,r_0}(m_1), m_1)\}_{m_1 \in \mathcal{M}_1} \subseteq \mathcal{M}_0 \times \mathcal{M}_1$ and let $S^1_{x_1,r_1}$ be the set $\{(m_0, g^1_{x_1,r_1}(m_0))\}_{m_0 \in \mathcal{M}_0} \subseteq \mathcal{M}_0 \times \mathcal{M}_1$.

First, we give four lemmas for the matrix representation:

▶ **Lemma 12.** *$S^0_{x_0,r_0} \cap S^1_{x_1,r_1} = \{T_{r_0,r_1}(x_0,x_1)\}$ holds for all $(r_0,r_1) \in \mathcal{CR}$ and $(x_0,x_1) \in \mathcal{X}_0 \times \mathcal{X}_1$.*

Proof. Let $(m_0,m_1) := T_{r_0,r_1}(x_0,x_1)$. By the definition, $g^0_{x_0,r_0}(m_1) = m_0$ and $g^1_{x_1,r_1}(m_0) = m_1$, and therefore $(m_0,m_1) \in S^0_{x_0,r_0} \cap S^1_{x_1,r_1}$. Suppose on the contrary that there exists $(m'_0,m'_1) \in S^0_{x_0,r_0} \cap S^1_{x_1,r_1}$ such that $(m'_0,m'_1) \neq (m_0,m_1)$. Let t be the first round where the two transcripts determined by (m_0,m_1) and (m'_0,m'_1) differ and let P_b be the party who sends a message at t-th round. Since $g^b_{x_b,r_b}(m_{\bar{b}}) = m_b$ and $g^b_{x_b,r_b}(m'_{\bar{b}}) = m'_b$, the t-th messages msg_t and msg'_t in the transcripts (m_0,m_1) and (m'_0,m'_1) are determined by (x_b,r_b) and the $(t-1)$-th or earlier messages in the transcripts (m_0,m_1) and (m'_0,m'_1), respectively. Since the latter messages are equal by the minimality of t, we have $\mathrm{msg}_t = \mathrm{msg}'_t$, contradicting the choice of t. Therefore, there is no $(m'_0,m'_1) \in S^0_{x_0,r_0} \cap S^1_{x_1,r_1}$ such that $(m'_0,m'_1) \neq (m_0,m_1)$, and the statement holds. ◀

▶ **Lemma 13.** *For all $(r_0,r_1) \in \mathcal{CR}$, T_{r_0,r_1} is bijective.*

Proof. Since $|\mathcal{X}_0 \times \mathcal{X}_1| = |\mathcal{M}_0 \times \mathcal{M}_1|$ from the optimality requirement, it is enough to prove that T_{r_0,r_1} is injective. Suppose on the contrary that there exist $(x_0,x_1) \neq (x'_0,x'_1) \in \mathcal{X}_0 \times \mathcal{X}_1$ such that $T_{r_0,r_1}(x_0,x_1) = T_{r_0,r_1}(x'_0,x'_1) =: (m_0,m_1)$. We assume that $x_0 \neq x'_0$ in the proof; the other case $x_1 \neq x'_1$ is similar.

For any $x''_1 \in \mathcal{X}_1$, there exists an $r''_1 \in \mathcal{R}_1$ such that $(r_0,r''_1) \in \mathcal{CR}$ and $T_{r_0,r''_1}(x_0,x''_1) = (m_0,m_1)$, since we have $\{(r^*,T_{r^*,r^{**}}(x_0,x_1))\}_{(r^*,r^{**})\in\mathcal{CR}} = \{(r^*,T_{r^*,r^{**}}(x_0,x''_1))\}_{(r^*,r^{**})\in\mathcal{CR}}$ by the security requirement and the left-hand side contains $(r_0,(m_0,m_1))$. By Lemma 12, (m_0,m_1) belongs to all of $S^0_{x_0,r_0}$, $S^0_{x'_0,r_0}$, and $S^1_{x''_1,r''_1}$, therefore $T_{r_0,r''_1}(x_0,x''_1) = T_{r_0,r''_1}(x'_0,x''_1) = (m_0,m_1)$ by Lemma 12 again. Then, from the correctness requirement, we have

$$\mathsf{Eval}(0,x_0,r_0,(m_0,m_1)) + \mathsf{Eval}(1,x''_1,r''_1,(m_0,m_1)) = f(x_0,x''_1),$$
$$\mathsf{Eval}(0,x'_0,r_0,(m_0,m_1)) + \mathsf{Eval}(1,x''_1,r''_1,(m_0,m_1)) = f(x'_0,x''_1),$$

and therefore

$$\mathsf{Eval}(0,x_0,r_0,(m_0,m_1)) - \mathsf{Eval}(0,x'_0,r_0,(m_0,m_1)) = f(x_0,x''_1) - f(x'_0,x''_1). \tag{6}$$

Since x''_1 was arbitrary, it follows that the function $f(x_0,\cdot) - f(x'_0,\cdot)$ is constant on $\mathcal{X}_1$, contradicting the non-redundancy of f. Hence the statement holds. ◀

▶ **Lemma 14.** *For all $b \in \{0,1\}$, $r_b \in \mathcal{R}_b$, $x_{\bar{b}} \in X_{\bar{b}}$, and $(m_0,m_1) \in \mathcal{M}_0 \times \mathcal{M}_1$, there exists $r_{\bar{b}} \in \mathcal{R}_{\bar{b}}$ such that $(r_0,r_1) \in \mathcal{CR}$ and $(m_0,m_1) \in S^b_{x_{\bar{b}},r_{\bar{b}}}$.*

Proof. We prove the statement for the case of $b = 0$; the other case $b = 1$ is similar. By the non-redundancy of the randomness space, there is an $r'_1 \in \mathcal{R}_1$ such that $(r_0,r'_1) \in \mathcal{CR}$. By Lemma 13, there is $(x'_0,x'_1) \in \mathcal{X}_0 \times \mathcal{X}_1$ such that $(m_0,m_1) = T_{r_0,r'_1}(x'_0,x'_1)$. Therefore we have $(r_0,(m_0,m_1)) \in \{(r^*,T_{r^*,r^{**}}(x'_0,x'_1))\}_{(r^*,r^{**})\in\mathcal{CR}}$, while this set is equal to $\{(r^*,T_{r^*,r^{**}}(x'_0,x_1))\}_{(r^*,r^{**})\in\mathcal{CR}}$ by the security requirement. This implies that there is an $r_1 \in \mathcal{R}_1$ such that $(r_0,r_1) \in \mathcal{CR}$ and $T_{r_0,r_1}(x'_0,x_1) = (m_0,m_1)$, therefore $(m_0,m_1) \in S^1_{x_1,r_1}$ by Lemma 12. Hence the statement holds. ◀

▶ **Lemma 15.** *For all $(m_0,m_1) \in \mathcal{M}_0 \times \mathcal{M}_1$, $b \in \{0,1\}$, and $r_b \in \mathcal{R}_b$, there exists a unique $x_b \in \mathcal{X}_b$ such that $g^b_{x_b,r_b}(m_{\bar{b}}) = m_b$.*

Proof. We prove the statement for the case of $b = 0$; the other case $b = 1$ is similar. First, we prove the existence. By definition, $g^0_{x_0,r_0}(m_1) = m_0$ holds if and only if $(m_0, m_1) \in S^0_{x_0,r_0}$. Let $r_1 \in \mathcal{R}_1$ satisfy $(r_0, r_1) \in \mathcal{CR}$. From Lemma 13, there exists $(x_0, x_1) \in \mathcal{X}_0 \times \mathcal{X}_1$ such that $T_{r_0,r_1}(x_0, x_1) = (m_0, m_1)$. From Lemma 12, $S^0_{x_0,r_0}$ contains $T_{r_0,r_1}(x_0, x_1)$ and this proves the existence.

Then, we prove the uniqueness. Suppose on the contrary that there exist $x_0 \neq x'_0 \in \mathcal{X}_0$ such that $g^0_{x_0,r_0}(m_1) = g^0_{x'_0,r_0}(m_1) = m_0$ and therefore $S^0_{x_0,r_0}$ and $S^0_{x'_0,r_0}$ contain (m_0, m_1). From Lemma 14, for all $x_1 \in \mathcal{X}_1$, there exists $r_1 \in \mathcal{R}_1$ such that $(r_0, r_1) \in \mathcal{CR}$ and $(m_0, m_1) \in S^1_{x_1,r_1}$. Since $(m_0, m_1) \in S^0_{x_0,r_0} \cap S^1_{x_1,r_1}$ and $(m_0, m_1) \in S^0_{x'_0,r_0} \cap S^1_{x_1,r_1}$, we have $T_{r_0,r_1}(x_0, x_1) = T_{r_0,r_1}(x'_0, x_1) = (m_0, m_1)$ from Lemma 12. This contradicts the fact that T_{r_0,r_1} is bijective (Lemma 13). This proves the uniqueness. ◀

Then, we give a matrix representation of the three requirements for a communication-optimal secure two-party protocol:

▶ **Theorem 16.** *Given a communication-optimal secure two-party protocol* $(\mathsf{Gen}, \mathsf{Msg}, \mathsf{Eval})$ *for* $f\colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathbb{G}$ *with correlated randomness* $CR \subseteq \mathcal{R}_0 \times \mathcal{R}_1$*, let* F *be an* $\mathcal{X}_0 \times \mathcal{X}_1$ *matrix whose* (x_0, x_1)*-th element is* $f(x_0, x_1)$*. Then, for* $b \in \{0, 1\}$ *and* $r_b \in \mathcal{R}_b$*, there exist an* $\mathcal{M}_0 \times \mathcal{M}_1$ *matrix* A_{b,r_b} *and a bijection* $T_{r_0,r_1}\colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathcal{M}_0 \times \mathcal{M}_1$ *such that*

- *for all* $(r_0, r_1) \in \mathcal{CR}$*,* $A_{0,r_0} + A_{1,r_1} = T_{r_0,r_1} \circ F$ *holds;*
- *for all* $b \in \{0, 1\}$*,* $r_b \in \mathcal{R}_b$*,* $x_{\bar{b}} \in \mathcal{X}_{\bar{b}}$ *and* $(m_0, m_1) \in \mathcal{M}_0 \times \mathcal{M}_1$*, there exists* $r_{\bar{b}} \in \mathcal{R}_{\bar{b}}$ *such that* $(r_0, r_1) \in \mathcal{CR}$ *and* $(m_0, m_1) \in S^{\bar{b}}_{x_{\bar{b}},r_{\bar{b}}}$*.*

Here, $T_{r_0,r_1} \circ F$ *is an* $M_0 \times \mathcal{M}_1$ *matrix whose* (m_0, m_1)*-th element is equal to* $T^{-1}_{r_0,r_1}(m_0, m_1)$*-th element of* F*.*

Note that, roughly speaking, the optimality requirement corresponds to T_{r_0,r_1} being a bijection,the correctness requirement corresponds to the first condition of the theorem, and the security requirement corresponds to the second condition of the theorem.

Proof. $\mathsf{Eval}(b, x_b, r_b, (m_0, m_1))$ is determined by (m_0, m_1, r_b) since x_b is uniquely determined by (m_0, m_1, r_b) from Lemma 15. Let A_{b,r_b} be an $\mathcal{M}_0 \times \mathcal{M}_1$ matrix whose (m_0, m_1)-th element is equal to $\mathsf{Eval}(b, x_b, r_b, (m_0, m_1))$. Note that T_{r_0,r_1} is bijective from Lemma 13.

The first condition is deduced from the correctness requirement and the definition of A_{b,r_b} and T_{r_0,r_1}. The second condition is the same as Lemma 14. ◀

5.2 Lower Bound

We prove the $\Omega(N)$-bit lower bound for the function $f\colon [N] \times [N] \to \{0, 1\}^2$ defined in Section 4.2. That is, we prove that any communication-optimal secure two-party protocol for f needs $\Omega(N)$-bit CR.

In the rest of this section, we write $[N]$ instead of $\mathcal{X}_b$ and $\mathcal{M}_b$. We consider the lower bound for the size of P_0's CR (i.e., $\log|\mathcal{R}_0|$); the lower bound for the size of P_1's CR is similar. We use the notation A_{b,r_b} for representing $N \times N$ matrices whose existence is guaranteed by Theorem 16, and as in Section 4.2, we focus on the first bit and use the same notation A_{b,r_b} and F. Then we have $F = \Delta^{N\times N}(0, 0)$ in the current setting.

First, we prove the following theorem:

▶ **Theorem 17.** *Suppose that* $r_0 \in \mathcal{R}_0$ *satisfies* $(0, 0) \in S^0_{0,r_0}$*. Then, for all* $i \in [N-1]$*, there exists* $r'_0 \in \mathcal{R}_0$ *such that*

- $(0, 0) \in S^0_{0,r'_0}$*.*
- *The* $(N-1)\times 1$ *submatrix at the bottom left corner of* $A_{0,r_0} + A_{0,r'_0}$ *is equal to* $\Delta^{(N-1)\times 1}(i, 0)$*.*

Proof. Let B_{b,r_b} be the $(N-1)\times 1$ submatrix at the bottom left corner of A_{b,r_b}. From the definition of the operation $\circ$, $T_{r_0,r_1}\circ F$ is equal to $\Delta^{N\times N}(T_{r_0,r_1}(0,0))$. From Theorem 16, there exists $r_1\in\mathcal{R}_1$ such that $(r_0,r_1)\in\mathcal{CR}$ and $(i+1,0)\in S^1_{0,r_1}$, and there exists $r''_0\in\mathcal{R}_0$ such that $(r''_0,r_1)\in\mathcal{CR}$ and $(i+1,0)\in S^0_{0,r''_0}$. Let $M:=([N]\times[N])\backslash\{(m,0)\mid m=1,\dots,N-1\}$.

From the definition of S^0_{0,r_0} and the assumption that $(0,0)$ belongs to S^0_{0,r_0}, $g^0_{0,r_0}(0)=0$ and S^0_{0,r_0} is equal to $\{(0,0)\}\cup\{(g^0_{0,r_0}(m),m)\}_{m=1,\dots,N-1}\subseteq M$. Therefore, from Lemma 12, we have $T_{r_0,r_1}(0,0)\in S^0_{0,r_0}\cap S^1_{1,r_1}\subseteq M$ and $B_{0,r_0}+B_{1,r_1}$ is the zero matrix. Also, $T_{r''_0,r_1}(0,0)=(i+1,0)$ since $(i+1,0)\in S^0_{0,r''_0}\cap S^1_{0,r_1}$. Therefore, $B_{0,r''_0}+B_{1,r_1}$ is equal to $\Delta^{(N-1)\times 1}(i,0)$ and we have

$$B_{0,r_0}+B_{0,r''_0}=(B_{0,r_0}+B_{1,r_1})+(B_{0,r''_0}+B_{1,r_1})=\Delta^{(N-1)\times 1}(i,0).$$

Let $(m'_0,m'_1)\in S^0_{0,r''_0}\backslash\{(i,0)\}$. Note that $(m'_0,m'_1)\in M$ since $S^0_{0,r''_0}\backslash\{(i,0)\}$ is equal to $\{(g^0_{0,r''_0}(m),m)\}_{m=1,\dots,N-1}\subseteq M$. From Theorem 16, there exists $r'_1\in\mathcal{R}_1$ such that $(r''_0,r'_1)\in\mathcal{CR}$ and $(m'_0,m'_1)\in S^1_{0,r'_1}$, and there exists $r'_0\in\mathcal{R}_0$ such that $(r'_0,r'_1)\in\mathcal{CR}$ and $(0,0)\in S^0_{0,r'_0}$. From Lemma 12 and the fact that $(m'_0,m'_1)\in S^0_{0,r''_0}\cap S^1_{0,r'_1}$, we have $T_{r''_0,r'_1}(0,0)=(m'_0,m'_1)\in M$ and $B_{0,r''_0}+B_{1,r'_1}$ is the zero matrix. Also, from the definition of S^0_{0,r'_0} and the fact that $(0,0)\in S^0_{0,r'_0}$, S^0_{0,r'_0} is equal to $\{(0,0)\}\cup\{(g^0_{0,r'_0}(m),m)\}_{m=1,\dots,N-1}\subseteq M$. From Lemma 12, we have $T_{r'_0,r'_1}(0,0)\in S^0_{0,r'_0}\cap S^1_{0,r'_1}\subseteq M$ and $B_{0,r'_0}+B_{1,r'_1}$ is the zero matrix. Therefore, $B_{0,r''_0}+B_{0,r'_0}=(B_{0,r''_0}+B_{1,r'_1})+(B_{0,r'_0}+B_{1,r'_1})$ is the zero matrix.

Hence, $B_{0,r_0}+B_{0,r'_0}=(B_{0,r_0}+B_{0,r''_0})+(B_{0,r''_0}+B_{0,r'_0})$ is equal to $\Delta^{(N-1)\times 1}(i,0)$, and therefore r'_0 satisfies the conditions of the statement. ◀

Using Theorem 17 sequentially, we have the following corollary:

▶ **Corollary 18.** *Suppose that $r_0\in\mathcal{R}_0$ satisfies $(0,0)\in S^0_{0,r_0}$. Then, for all $M\in\{0,1\}^{[N-1]\times[1]}$, there exists $r'_0\in\mathcal{R}_0$ such that*

- $(0,0)\in S^0_{0,r'_0}$.
- *The $(N-1)\times 1$ submatrix at the bottom left corner of $A_{0,r_0}+A_{0,r'_0}$ is equal to M.*

Proof. We can prove this corollary similarly to Corollary 10. ◀

The lower bound of the size of P_0's CR is derived from Corollary 18:

▶ **Corollary 19.** *The size of CR delivered to P_0 is $\Omega(N)$ bits. More concretely, it is greater than or equal to $N-1$ bits.*

Proof. We can prove this corollary similarly to Corollary 11. ◀

References

1 Nuttapong Attrapadung, Goichiro Hanaoka, Takahiro Matsuda, Hiraku Morita, Kazuma Ohara, Jacob C. N. Schuldt, Tadanori Teruya, and Kazunari Tozawa. Oblivious linear group actions and applications. In *CCS'21*, pages 630–650. ACM, 2021. doi:10.1145/3460120.3484584.

2 Donald Beaver. Efficient multiparty protocols using circuit randomization. In *11th CRYPTO*, volume 576 of *LNCS*, pages 420–432. Springer, 1991. doi:10.1007/3-540-46766-1_34.

3 Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In *11th TCC*, volume 8349 of *LNCS*, pages 317–342. Springer, 2014. doi:10.1007/978-3-642-54242-8_14.

4 Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions (full version of [3]). https://people.csail.mit.edu/ranjit/papers/BIKK.pdf, 2014.

5 Elette Boyle, Nishanth Chandran, Niv Gilboa, Divya Gupta, Yuval Ishai, Nishant Kumar, and Mayank Rathee. Function secret sharing for mixed-mode and fixed-point secure computation. In *40th EUROCRYPT*, volume 12697 of *LNCS*, pages 871–900. Springer, 2021. doi:10.1007/978-3-030-77886-6_30.

6 Elette Boyle, Niv Gilboa, and Yuval Ishai. Secure computation with preprocessing via function secret sharing. In *17th TCC*, volume 11891 of *LNCS*, pages 341–371. Springer, 2019. doi:10.1007/978-3-030-36030-6_14.

7 Benny Chor and Eyal Kushilevitz. A zero-one law for boolean privacy. In *21st STOC*, pages 62–72. ACM, 1989. doi:10.1145/73007.73013.

8 Geoffroy Couteau. A note on the communication complexity of multiparty computation in the correlated randomness model. In *38th EUROCRYPT*, volume 11477 of *LNCS*, pages 473–503. Springer, 2019. doi:10.1007/978-3-030-17656-3_17.

9 Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The tinytable protocol for 2-party secure computation, or: Gate-scrambling revisited. In *37th CRYPTO*, volume 10401 of *LNCS*, pages 167–187. Springer, 2017. doi:10.1007/978-3-319-63688-7_6.

10 Ivan Damgård, Jesper Buus Nielsen, Antigoni Polychroniadou, and Michael Raskin. On the communication required for unconditionally secure multiplication. In *36th CRYPTO*, volume 9815 of *LNCS*, pages 459–488. Springer, 2016. doi:10.1007/978-3-662-53008-5_16.

11 Ivan Bjerre Damgård, Boyang Li, and Nikolaj Ignatieff Schwartzbach. More communication lower bounds for information-theoretic mpc. In *2nd ITC*, volume 199 of *LIPIcs*, pages 2:1–2:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITC.2021.2.

12 Deepesh Data, Manoj M. Prabhakaran, and Vinod M. Prabhakaran. On the communication complexity of secure computation. In *34th CRYPTO*, volume 8617 of *LNCS*, pages 199–216. Springer, 2014. doi:10.1007/978-3-662-44381-1_12.

13 Anna Gál and Adi Rosén. Lower bounds on the amount of randomness in private computation. In *35th STOC*, pages 659–666. ACM, 2003. doi:10.1145/780542.780638.

14 Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In *33rd EUROCRYPT*, volume 8441 of *LNCS*, pages 640–658. Springer, 2014. doi:10.1007/978-3-642-55220-5_35.

15 Vipul Goyal, Yuval Ishai, and Yifan Song. Tight bounds on the randomness complexity of secure multiparty computation. In *42nd CRYPTO*, volume 13510 of *LNCS*, pages 483–513. Springer, 2022. doi:10.1007/978-3-031-15985-5_17.

16 Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *10th TCC*, volume 7785 of *LNCS*, pages 600–620. Springer, 2013. doi:10.1007/978-3-642-36594-2_34.

17 Marcel Keller, Emmanuela Orsini, and Peter Scholl. Mascot: Faster malicious arithmetic secure computation with oblivious transfer. In *CCS'16*, pages 830–842. ACM, 2016. doi:10.1145/2976749.2978357.

18 Eyal Kushilevitz. Privacy and communication complexity. In *30th FOCS*, pages 416–421. IEEE Computer Society, 1989. doi:10.1109/sfcs.1989.63512.

19 Eyal Kushilevitz and Yishay Mansour. Randomness in private computations. In *15th PODC*, pages 181–190. ACM Press, 1996. doi:10.1145/248052.248089.

20 Eyal Kushilevitz, Rafail Ostrovsky, Emmanuel Prouff, Adi Rosén, Adrian Thillard, and Damien Vergnaud. Lower and upper bounds on the randomness complexity of private computations of and. In *17th TCC*, volume 11892 of *LNCS*, pages 386–406. Springer, 2019. doi:10.1007/978-3-030-36033-7_15.

21 Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Characterizing linear size circuits in terms of privacy. In *28th STOC*. ACM, 1996. doi:10.1145/237814.238002.

22 Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Amortizing randomness in private multiparty computations. In *17th PODC*, pages 81–90. ACM, 1998. doi:10.1145/277697.277710.

23 Eyal Kushilevitz and Adi Rosén. A randomness-rounds tradeoff in private computation. In *14th CRYPTO*, volume 839 of *LNCS*, pages 397–410. Springer, 1994. doi:10.1007/3-540-48658-5_36.

24 Arpita Patra, Thomas Schneider, Ajith Suresh, and Hossein Yalame. ABY2.0: improved mixed-protocol secure two-party computation. In *30th USENIX Security Symposium*, pages 2165–2182. USENIX Association, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/patra.

25 Andrew C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society, 1986. doi:10.1109/SFCS.1986.25.

A Reduction to Protocol for Non-Redundant Function

In this section, we reduce a protocol for $f\colon \mathcal{X}_0 \times \mathcal{X}_1 \to \mathbb{G}$ to a protocol for a non-redundant function f'. We define the binary relations '$\sim$' on $\mathcal{X}_0$ as follows: $x_0 \sim x_0'$ if and only if $f(x_0, \cdot) - f(x_0', \cdot)\colon \mathcal{X}_1 \to \mathbb{G}$ is constant. Note that is an equivalence relation. Let $\mathcal{X}_0' \subseteq \mathcal{X}_0$ be a complete system of representatives, and let ϕ_0 be the natural sujection $\mathcal{X}_0 \to \mathcal{X}_0'$. By the definition, $f(x, \cdot) - f(\phi(x), \cdot)\colon \mathcal{X}_1 \to \mathbb{G}$ is constant and we denote $h_0(x)$ as the constant. Similarly, we define $\mathcal{X}_1'$, ϕ_1, and $h_1(x)$.

Let $f'\colon \mathcal{X}_0' \times \mathcal{X}_1' \to \mathbb{G}$ be a restriction of f. Note that f' is non-redundant. Then, we can construct a two-party protocol Π for f from a two-party protocol Π' for f' with the same CR size, the number of rounds, and the communication complexity: $\Pi(x_0, x_1)$ computes $(g_0, g_1) \leftarrow \Pi'(\phi_0(x_0), \phi(x_1))$ and outputs $(g_0 + h_0(x_0), g_1 + h_1(x_1))$. CR size, the number of rounds, and the communication complexity of Π is the same as Π' and the security, and Π is secure when Π' is secure.