1st Conference on Information-Theoretic Cryptography

ITC 2020, June 17–19, 2020, Boston, MA, USA

Edited by

Yael Tauman Kalai
Adam D. Smith
Daniel Wichs

Editors

Yael Tauman Kalai
Microsoft Research New England, Cambridge, MA, USA
yael@microsoft.com

Adam D. Smith
Boston University, MA, USA
ads22@bu.edu

Daniel Wichs
Northeastern University, Boston, MA, USA
NTT Research, Boston, MA, USA
wichs@ccs.neu.edu

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

ISBN 978-3-95977-151-1

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/978-3-95977-151-1.

Publication date
June, 2020

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.

Digital Object Identifier: 10.4230/LIPIcs.ITC.2020.0

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.

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Regular Papers

1st Conference on Information-Theoretic Cryptography (ITC 2020).
Editors: Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

The first Conference on Information-Theoretic Cryptography (ITC 2020) was originally planned to take place at Boston University, Boston, MA USA on June 17-19, 2020. However, due to the COVID-19 pandemic, the conference is being held entirely virtually. The general chairs are Yael Tauman Kalai and Adam D. Smith, and the program chair is Daniel Wichs. The conference is held in cooperation with the IACR.

The conference received 39 submissions, of which the Program Committee (PC) accepted 16. Since this was the first iteration of the conference, we felt the need to set a very high bar for quality to cement its standing as a top venue in the field. Each submission was reviewed by at least three PC members, often more. The 19 PC members, were helped by many additional external reviewers. The proceedings consist of the revised version of the 16 accepted papers. The revisions were not reviewed, and the authors bear full responsibility for the content.

We used Shai Halevi's excellent web-review software, and are extremely indebted to him for having written it, for setting it up for us, and for patiently providing much technical help whenever we had any questions. In addition to asking reviewers for an overall score, we asked reviewers to also give numeric scores for "Importance of End Results", "Conceptual Novelty", and "Technical Novelty". Although we did not rely on these scores in the final decision making process, we felt that they helped frame the discussion and clarified what aspects reviewers liked and disliked about the paper.

In addition to the accepted papers, the conference includes six invited talks. These were selected by the PC and represent the "greatest hits" results in the area of information theoretic cryptography from the last few years.

We would like to thank the many people who made ITC a success. First of all, a big thanks to all the authors who submitted papers to the conference. There were many excellent submissions, and in our goal to keep the quality high, it is all but certain that we rejected some papers whose importance we failed to grasp. We would like to thank all the PC members for their efforts in providing detailed reviews, verifying correctness, and discussing the merits and limitations in depth. We are also thankful to the external reviewers for providing valuable expert opinions. We are grateful to the ITC Steering Committee, and especially Benny Applebaum, for guidance and advice. And last, but not least, we thank all the invited speakers, presenting authors, and participants for committing their time to making the virtual conference a success, despite all the difficulties of the ongoing COVID-19 pandemic.

1st Conference on Information-Theoretic Cryptography (ITC 2020).
Editors: Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs
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 (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)

1st Conference on Information-Theoretic Cryptography (ITC 2020).
Editors: Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Organization

General Chairs

- Yael Tauman Kalai (MSR and MIT)
- Adam Smith (BU)

Program Chair

- Daniel Wichs (Northeastern and NTT Research)

Program Committee

- Shweta Agrawal (IIT Madras)
- Amos Beimel (Ben Gurion University)
- Anne Broadbent (University of Ottawa)
- Mahdi Cheraghchi (University of Michigan Ann Arbor)
- Kai-Min Chung (Academia Sinica)
- Stefan Dziembowski (University of Warsaw)
- Serge Fehr (CWI Amsterdam and Leiden University)
- Siyao Guo (New York University Shanghai)
- Iftach Haitner (Tel Aviv University)
- Mohammad Hajiabadi (UC Berkeley)
- Ilan Komargodski (NTT Research)
- Hemanta Maji (Purdue University)
- Moni Naor (Weizmann Institute of Science)
- Jesper Buus Nielsen (Aarhus University)
- Christian Schaffner (QuSoft and University of Amsterdam)
- Stefano Tessaro (University of Washington)
- Jonathan Ullman (Northeastern University)
- Mary Wootters (Stanford University)
- Mark Zhandry (Princeton University and NTT Research)

External Reviewers

Mark Abspoel, Alexander Barg, James Bartusek, Jeremiah Blocki, Christian Badertscher, Clement Canonne, Andre Chailloux, Suvradip Chakraborty, Eshan Chattopadhyay, Albert Cheu, Ran Cohen, Wei Dai, Pratik Dand, Akshay Degwekar, Itai Dinur, Jack Doerner, Jelle Don, Yfke Dulek, Bill Fefferman, Venkatesan Guruswami, Brett Hemenway, Viet Tung Hoang, Joseph Jaeger, Matthew Joseph, Raza Ali Kazmi, Xiao Liang, Qipeng Liu, Tianren Liu, Yanyi Liu, Sébastien Lord, Fermi Ma, Mohammad Mahmoody, Nikolaos Makriyannis, Aleksandar (Sasho) Nikolov, Maciej Obremski, Naty Peter, Supartha Podder, Antigoni Polychroniadou, Divya Ravi, Joao Ribeiro, Lior Rotem, Sruthi Sekar, Or Sheffet, Pratik Soni, Jad Silbak, Mark Simkin,Florian Speelman, Akshayaram Srinivasan, Benjamin Terner, Ameya Velingker, Vanessa Vitse, Mingyuan Wang, Tianhao Wang, Chris Williamson, Jiapeng Zhang, Vincent Zucca

1st Conference on Information-Theoretic Cryptography (ITC 2020).
Editors: Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

List of Authors

Radune Artiom (12)
Ariel University, Ariél, Israel; The Open University, Raanana, Israel

Gilad Asharov (11)
Bar-Ilan University, Ramat-Gan, Israel

Victor Balcer (1)
School of Engineering & Applied Sciences, Harvard University, Cambridge, MA, United States

Amos Beimel (14)
Dept. of Computer Science, Ben-Gurion University, Beer-Sheva, Israel

Jeremiah Blocki (16)
Purdue University, West Lafayette, IN, USA

Mahdi Cheraghchi (7)
EECS Department, University of Michigan, Ann Arbor, MI, USA

Albert Cheu (1)
Khoury College of Computer Sciences, Northeastern University, Boston, MA, United States

Kai-Min Chung (8)
Academia Sinica, Taipei, Taiwan

Claude Crépeau (4)
School of Computer Science, McGill University, Montréal, Québec, Canada

Hendrik Eerikson (5)
Cybernetica AS, Tartu, Estonia

Reo Eriguchi (2)
Graduate School of Information Science and Technology, The University of Tokyo, Japan

Badih Ghazi (15)
Google Research, Mountain View, CA, USA

Noah Golowich (15)
Google Research, Mountain View, CA, USA; MIT, Cambridge, MA, USA

Venkatesan Guruswami (7)
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Martin Hirt (3)
ETH Zurich, Switzerland

Marcel Keller (5)
CSIRO's Data61, Eveleigh, Australia

Ilan Komargodski (11)
NTT Research, East Palo Alto, CA, USA

Aleksandra Korolova (14)
Dept. of Computer Science, University of Southern California, Los Angeles, CA, USA

Shubhang Kulkarni (16)
Purdue University, West Lafayette, IN, USA

Ravi Kumar (15)
Google Research, Mountain View, CA, USA

Noboru Kunihiro (2)
Department of Computer Science, University of Tsukuba, Japan

Tai-Ning Liao (8)
National Taiwan University, Taipei, Taiwan

Fuchun Lin (7)
Department of Electrical and Electronic Engineering, Imperial College London, UK

Wei-Kai Lin (11)
Cornell University, Ithaca, NY, USA

Pasin Manurangsi (15)
Google Research, Mountain View, CA, USA

Arnaud Y. Massenet (4)
École Normale Supérieure Paris-Saclay, Gif-sur-Yvette, France

Marta Mularczyk (3)
ETH Zurich, Switzerland

Moni Naor (9)
Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel

Kobbi Nissim (14)
Dept. of Computer Science, Georgetown University, Washington, DC, USA

Claudio Orlandi (5)
Department of Computer Science, DIGIT, Aarhus University, Denmark

Rasmus Pagh (15)
Google Research, Mountain View, CA, USA; IT University of Copenhagen, Denmark; Basic Algorithms Research Copenhagen, Denmark

Anat Paskin-Cherniavsky (12)
Ariel University, Ariél, Israel

1st Conference on Information-Theoretic Cryptography (ITC 2020).
Editors: Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Enoch Peserico (11)
Università degli Studi di Padova, Italy

Naty Peter (13)
Ben-Gurion University of the Negev, Beer-Sheva, Israel

Pille Pullonen (5)
Cybernetica AS, Tartu, Estonia

Joonas Puura (5)
Institute of Computer Science, University of Tartu, Estonia

Luowen Qian (8)
Boston University, MA, USA

Peter Michael Reichstein Rasmussen (6)
Basic Algorithms Research Copenhagen, University of Copenhagen, Denmark

Lior Rotem (9)
School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem 91904, Israel

Reihaneh Safavi-Naini (7)
Department of Computer Science, University of Calgary, CA

Amit Sahai (6)
UCLA, Los Angeles, CA, USA

Louis Salvail (4)
Département d'Informatique et de R.O., Université de Montréal, Montréal, Québec, Canada

Gil Segev (9, 10)
School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem 91904, Israel

Ido Shahaf (10)
School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem, Israel

Or Sheffet (14)
Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel

Elaine Shi (11)
Cornell University, Ithaca, NY, USA

Mark Simkin (5)
Department of Computer Science, DIGIT, Aarhus University, Denmark

Uri Stemmer (14)
Dept. of Computer Science, Ben-Gurion University, Beer-Sheva, Israel; Google Research

Lucas Shigeru Stinchcombe (4)
Bloomberg L.P., Tokyo, Japan

Rotem Tsabary (13)
Weizmann Institute of Science, Rehovot, Israel

Ameya Velingker (15)
Google Research, Mountain View, CA, USA

Huaxiong Wang (7)
Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore, SG

Hoeteck Wee (13)
CNRS, ENS, PSL, Paris, France

Nan Yang (4)
Canadian Centre for Cyber Security, Ottawa, Ontario, Canada; Concordia University, Montréal, Québec, Canada

Samson Zhou (16)
Carnegie Mellon University, Pittsburgh, PA, USA

Separating Local & Shuffled Differential Privacy via Histograms

Victor Balcer
School of Engineering & Applied Sciences, Harvard University, Cambridge, MA, United States

Albert Cheu
Khoury College of Computer Sciences, Northeastern University, Boston, MA, United States

Abstract

Recent work in differential privacy has highlighted the *shuffled model* as a promising avenue to compute accurate statistics while keeping raw data in users' hands. We present a protocol in this model that estimates histograms with error *independent of the domain size*. This implies an arbitrarily large gap in sample complexity between the shuffled and local models. On the other hand, we show that the models are equivalent when we impose the constraints of pure differential privacy and single-message randomizers.

2012 ACM Subject Classification Security and privacy → Privacy-preserving protocols

Keywords and phrases Differential Privacy, Distributed Protocols, Histograms

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.1

Related Version A full version of the paper is available at `https://arxiv.org/abs/1911.06879`.

Funding *Victor Balcer*: Supported by NSF grant CNS-1565387.
Albert Cheu: Supported by NSF grants CCF-1718088, CCF-1750640, and CNS-1816028.

Acknowledgements We are grateful to Daniel Alabi and Maxim Zhilyaev for discussions that shaped the early stages of this work. We are also indebted to Matthew Joseph and Jieming Mao for directing us to the pointer-chasing and multi-party pointer-jumping problems. Finally, we thank Salil Vadhan for editorial comments and providing a simpler construction for Claim 19.

1 Introduction

The *local model* of private computation has minimal trust assumptions: each user executes a randomized algorithm on their data and sends the output *message* to an analyzer [17, 19, 11]. While this model has appeal to the users – their data is never shared in the clear – the noise in every message poses a roadblock to accurate statistics. For example, a locally private d-bin histogram has, on some bin, error scaling with $\sqrt{\log d}$. But when users trust the analyzer with their raw data (the *central model*), there is an algorithm that achieves error independent of d on every bin.

Because the local and central models lie at the extremes of trust, recent work has focused on the intermediate *shuffled model* [6, 8]. In this model, users execute ranomization like in the local model but now a trusted *shuffler* applies a uniformly random permutation to all user messages before the analyzer can view them. The anonymity provided by the shuffler allows users to introduce less noise than in the local model while achieving the same level of privacy. This prompts the following questions:

In terms of accuracy, how well separated is the shuffled model from the local model?

How close is the shuffled model to the central model?

1.1 Our Results

In Section 3, we provide a new protocol for histograms in the shuffled model. To quantify accuracy, we bound the *simultaneous error*: the maximum difference over all bins between each bin's estimated frequency and its true frequency in the input dataset.

▶ **Theorem 1** (Informal). *For any $\varepsilon < 1$ and $\delta = o(1/n)$, there exists a shuffled protocol that satisfies (ε, δ)-differential privacy and reports a histogram with simultaneous error $O(\log(1/\delta)/(\varepsilon^2 n))$ with constant probability.*

For comparison, [8] give a protocol with error $O(\sqrt{\log d \cdot \log 1/\delta}/(\varepsilon n))$. Our protocol has smaller error when $\log(1/\delta) = o(\log d)$. In the natural regime where $\delta = \Theta(poly(1/n))$, that condition is satisfied when $\log n = o(\log d)$. An example for this setting would be a dataset holding the browser home page of each user. The data universe could consist of all strings up to a certain length which would far exceed the number of users.

In Section 3.3, we show that the histogram protocol has strong implications for the *distributional* setting. Here, the rows of the dataset are independently drawn from a probability distribution. We focus on the *sample complexity*, which is the number of samples needed to identify or estimate some feature of the distribution. We prove that the separation in sample complexity between the local and shuffled models can be made arbitrarily large:

▶ **Theorem 2** (Informal). *There is a distributional problem where the sample complexity in the local model scales with a parameter of the problem, but the sample complexity in the shuffled model is independent of that parameter.*

We also show that there is a distributional problem which requires polynomially more samples in the *sequentially interactive* local model than in the shuffled model. This is done by reusing the techniques to prove Theorem 2.

A natural conjecture is that there are progressively weaker versions of Theorem 2 for progressively constrained versions of the model. In Section 4, we prove that the shuffled model collapses to the local model when constraints are too strong:

▶ **Theorem 3** (Informal). *For every single-message shuffled protocol that satisfies pure differential privacy, there is a local protocol with exactly the same privacy and sample complexity guarantees.*

1.2 Related Work

Table 1 presents our histogram result alongside existing results for the problem – all previous bounds on simultaneous error in the shuffled model depend on d. Although we focus on simultaneous error, error metrics focusing on per-bin error are also used in the literature such as mean squared error (MSE) and high probability confidence intervals on each bin. When considering these alternative metrics or when d is not large, other histogram protocols may outperform ours (see e.g. [18]).

Quantitative separations between the local and shuffled models exist in the literature [8, 1, 13, 12]. As a concrete example, [8] implies that the sample complexity of Bernoulli mean estimation in the shuffled model is $O(1/\alpha^2 + \log(1/\delta)/(\alpha\varepsilon))$. In contrast, [5] gives a lower bound of $\Omega(1/\alpha^2\varepsilon^2)$ in the local model.

Prior work have shown limits of the shuffled model, albeit under communication constraints. The first set of results follow from a lemma in [8]: a single-message shuffled protocol implies a local protocol with a weaker differential privacy guarantee. Specifically, if the shuffled protocol obeys (ε, δ)-differential privacy, then the local protocol obeys $(\varepsilon + \ln n, \delta)$-differential privacy. Lower bounds for the local model can then be invoked, as done in [8, 13].

Table 1 Comparison of results for the histogram problem. To simplify the presentation, we assume constant success probability, $\varepsilon < 1$, $\delta < 1/\log d$ for results from [13], and $e^{-O(n\varepsilon^2)} \leq \delta < 1/n$ for our result.

Model	Simultaneous Error	No. Messages per User	Source
Local	$\Theta\left(\frac{1}{\varepsilon\sqrt{n}} \cdot \sqrt{\log d}\right)$	1	[3]
Shuffled	$O\left(\frac{1}{\varepsilon n} \cdot \sqrt{\log d \cdot \log \frac{1}{\delta}}\right)$	$O(d)$	[8]
	$O\left(\frac{1}{\varepsilon n}\sqrt{\log^3 d \cdot \log(\frac{1}{\delta}\log d)}\right)$	$O\left(\frac{1}{\varepsilon^2}\log^3 d \cdot \log(\frac{1}{\delta}\log d)\right)$ w.h.p.	[13]
	$O\left(\frac{\log d}{n} + \frac{1}{\varepsilon n} \cdot \sqrt{\log d \cdot \log \frac{1}{\varepsilon\delta}}\right)$	$O\left(\frac{1}{\varepsilon^2}\log \frac{1}{\varepsilon\delta}\right)$	[13]
	$O\left(\frac{1}{\varepsilon^2 n}\log \frac{1}{\delta}\right)$	$O(d)$	[Theorem 12]
Central	$\Theta\left(\frac{1}{\varepsilon n}\min\left(\log d, \log \frac{1}{\delta}\right)\right)$	N/A	[9, 7, 4, 14]

Another class of lower bound comes from a lemma in [12]: when a shuffled protocol obeys ε-differential privacy and bounded communication complexity, the set of messages output by each user is insensitive to their personal data. Specifically, changing their data causes the set's distribution to change by $\leq 1 - 2^{-O_\varepsilon(m^2\ell)}$ in statistical distance, where m denotes number of messages and ℓ the length of each message. This is strong enough to obtain a lower bound on binary sums.

The amplification-by-shuffling lemmas in [2, 10] show that uniformly permuting the messages generated by a local protocol improves privacy guarantees: an ε-private local protocol becomes an (ε', δ)-private shuffled protocol where $\varepsilon' \ll \varepsilon$ and $\delta > 0$. One might conjecture weaker versions of these lemmas where $\delta = 0$ but Theorem 3 eliminates that possibility.

2 Preliminaries

We define a *dataset* $\vec{x} \in \mathcal{X}^n$ to be an ordered tuple of n rows where each row is drawn from a *data universe* $\mathcal{X}$ and corresponds to the data of one user. Two datasets $\vec{x}, \vec{x}\,' \in \mathcal{X}^n$ are considered *neighbors* (denoted as $\vec{x} \sim \vec{x}\,'$) if they differ in exactly one row.

▶ **Definition 4** (Differential Privacy [9]). *An algorithm $\mathcal{M} : \mathcal{X}^n \to \mathcal{Z}$ satisfies* (ε, δ)-differential privacy *if*

$$\forall \vec{x} \sim \vec{x}\,' \quad \forall T \subseteq \mathcal{Z} \qquad \Pr[\mathcal{M}(\vec{x}) \in T] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(\vec{x}\,') \in T] + \delta.$$

We say an (ε, δ)-differentially private algorithm satisfies pure differential privacy *when $\delta = 0$ and* approximate differential privacy *when $\delta > 0$. For pure differential privacy, we may omit the δ parameter from the notation.*

▶ **Definition 5** (Local Model [17]). *A protocol $\mathcal{P}$ in the* (non-interactive[1]) local model *consists of two randomized algorithms:*

- *A* randomizer $\mathcal{R} : \mathcal{X} \to \mathcal{Y}$ *that takes as input a single user's data and outputs a message.*

[1] The literature also includes interactive variants; see [15] for a definition of *sequential* and *full* interactivity.

- *An* analyzer $\mathcal{A} : \mathcal{Y}^* \to \mathcal{Z}$ *that takes as input all user messages and computes the output of the protocol.*

We denote the protocol $\mathcal{P} = (\mathcal{R}, \mathcal{A})$*. We assume that the number of users* n *is public and available to both* $\mathcal{R}$ *and* $\mathcal{A}$*. Let* $\vec{x} \in \mathcal{X}^n$*. The evaluation of the protocol* $\mathcal{P}$ *on input* $\vec{x}$ *is The evaluation of the protocol* $\mathcal{P}$ *on input* $\vec{x}$ *is*

$$\mathcal{P}(\vec{x}) = (\mathcal{A} \circ \mathcal{R})(\vec{x}) = \mathcal{A}(\mathcal{R}(x_1), \ldots, \mathcal{R}(x_n)).$$

▶ **Definition 6** (Differential Privacy for Local Protocols)**.** *A local protocol* $\mathcal{P} = (\mathcal{R}, \mathcal{A})$ *satisfies* (ε, δ)-differential privacy *for* n *users if its randomizer* $\mathcal{R} : \mathcal{X} \to \mathcal{Y}$ *is* (ε, δ)*-differentially private (for datasets of size one).*

▶ **Definition 7** (Shuffled Model [6, 8])**.** *A protocol* $\mathcal{P}$ *in the* shuffled model *consists of three randomized algorithms:*

- *A* randomizer $\mathcal{R} : \mathcal{X} \to \mathcal{Y}^*$ *that takes as input a single user's data and outputs a vector of messages whose length may be randomized. If, on all inputs, the probability of sending a single message is 1, then the protocol is said to be* single-message*. Otherwise, the protocol is said to be* multi-message*.*
- *A* shuffler $\mathcal{S} : \mathcal{Y}^* \to \mathcal{Y}^*$ *that concatenates all message vectors and then applies a uniformly random permutation to (the order of) the concatenated vector. For example, when there are three users each sending two messages, there are* 6! *permutations and all are equally likely to be the output of the shuffler.*
- *An* analyzer $\mathcal{A} : \mathcal{Y}^* \to \mathcal{Z}$ *that takes a permutation of messages to generate the output of the protocol.*

As in the local model, we denote the protocol $\mathcal{P} = (\mathcal{R}, \mathcal{A})$ *and assume that the number of users* n *is accessible to both* $\mathcal{R}$ *and* $\mathcal{A}$*. The evaluation of the protocol* $\mathcal{P}$ *on input* $\vec{x}$ *is*

$$\mathcal{P}(\vec{x}) = (\mathcal{A} \circ \mathcal{S} \circ \mathcal{R})(\vec{x}) = \mathcal{A}(\mathcal{S}(\mathcal{R}(x_1), \ldots, \mathcal{R}(x_n))).$$

▶ **Definition 8** (Differential Privacy for Shuffled Protocols [8])**.** *A shuffled protocol* $\mathcal{P} = (\mathcal{R}, \mathcal{A})$ *satisfies* (ε, δ)-differential privacy *for* n *users if the algorithm* $(\mathcal{S} \circ \mathcal{R}) : \mathcal{X}^n \to \mathcal{Y}^*$ *is* (ε, δ)*-differentially private.*

We note a difference in robustness between the local and shuffled models. A user in a local protocol only has to trust that their own execution of $\mathcal{R}$ is correct to ensure differential privacy. In contrast, a user in a shuffled protocol may not have the same degree of privacy when other users deviate from the protocol.

For any $d \in \mathbb{N}$, let $[d]$ denote the set $\{1, \ldots, d\}$. For any $j \in [d]$, we define the function $c_j : [d]^n \to \mathbb{R}$ as the normalized count of j in the input:

$$c_j(\vec{x}) = (1/n) \cdot |\{i \in [n] \,:\, x_i = j\}|.$$

We use *histogram* to refer to the vector of normalized counts $(c_1(\vec{x}), \ldots, c_d(\vec{x}))$. For measuring the accuracy of a histogram protocol $\mathcal{P} : [d]^n \to \mathbb{R}^d$, we use the following metrics:

▶ **Definition 9.** *A histogram protocol* $\mathcal{P} : [d]^n \to \mathbb{R}^d$ *has* (α, β)-per-query accuracy *if*

$$\forall \vec{x} \in [d]^n \quad \forall j \in [d] \quad \Pr[|\mathcal{P}(\vec{x})_j - c_j(\vec{x})| \leq \alpha] \geq 1 - \beta.$$

▶ **Definition 10.** *A histogram protocol* $\mathcal{P} : [d]^n \to \mathbb{R}^d$ *has* (α, β)-simultaneous accuracy *if*

$$\forall \vec{x} \in [d]^n \quad \Pr[\forall j \in [d] \quad |\mathcal{P}(\vec{x})_j - c_j(\vec{x})| \leq \alpha] \geq 1 - \beta.$$

3 The Power of Multiple Messages for Histograms

In this section, we present an (ε, δ)-differentially private histogram protocol in the shuffled model whose simultaneous error does not depend on the universe size. We start by presenting a private protocol for releasing a binary sum that always outputs 0 if the true count is 0 and otherwise outputs a noisy estimate. The histogram protocol uses this counting protocol to estimate the frequency of every domain element. Its simultaneous error is the maximum noise introduced to the nonzero counts. There are at most n such counts.

For comparison, a protocol in [8] adds independent noise to all counts without the zero-error guarantee. The simultaneous error is therefore the maximum noise over *all* d counts, which introduces a $\log d$ instead of a $\log n$ factor.

3.1 A Two-Message Protocol for Binary Sums

In the protocol $\mathcal{P}^{\text{zsum}}_{\varepsilon,\delta}$ (Figure 1), each user reports a vector whose length is the sum of their data and a Bernoulli random variable. The contents of each vector will be copies of 1. Because the shuffler only reports a uniformly random permutation, the observable information is equivalent to the sum of user data, plus noise. The noise is distributed as $\text{Bin}(n, p)$, where p is chosen so that there is sufficient variance to ensure (ε, δ)-differential privacy. We take advantage of the fact that the binomial distribution is bounded: if the sum of the data is zero, the noisy sum is *never* more than n. Hence, the analyzer will perform truncation when the noisy sum is small. We complete our proof by arguing that it is unlikely for large values to be truncated.

To streamline the presentation and analysis, we assume that $\sqrt{(100/n) \cdot \ln(2/\delta)} \leq \varepsilon \leq 1$ so that $p \in (1/2, 1)$. We can achieve (ε, δ) privacy for a broader parameter regime by setting p to a different function; we refer the interested reader to Theorem 4.11 in [8].

Randomizer $\mathcal{R}^{\text{zsum}}_{\varepsilon,\delta}(x \in \{0,1\})$ for $\varepsilon, \delta \in [0,1]$**:**
1. Let $p \leftarrow 1 - \frac{50}{\varepsilon^2 n} \ln(2/\delta)$.
2. Sample $z \sim \text{Ber}(p)$.
3. Output $(\underbrace{1, \ldots, 1}_{x+z \text{ copies}})$.

Analyzer $\mathcal{A}^{\text{zsum}}_{\varepsilon,\delta}(\vec{y} \in \{1\}^*)$ for $\varepsilon, \delta \in [0,1]$**:**
1. Let $p \leftarrow 1 - \frac{50}{\varepsilon^2 n} \ln(2/\delta)$.
2. Let $c^* = \frac{1}{n} \cdot |\vec{y}|$, where $|\vec{y}|$ is the length of $\vec{y}$.
3. Output $\begin{cases} c^* - p & \text{if } c^* > 1 \\ 0 & \text{otherwise} \end{cases}$.

Figure 1 The pseudocode for $\mathcal{P}^{\text{zsum}}_{\varepsilon,\delta}$, a private shuffled protocol for normalized binary sums.

▶ **Theorem 11.** *For any $\varepsilon, \delta \in [0,1]$ and any $n \in \mathbb{N}$ such that $n \geq (100/\varepsilon^2) \cdot \ln(2/\delta)$, the protocol $\mathcal{P}^{\text{zsum}}_{\varepsilon,\delta} = (\mathcal{R}^{\text{zsum}}_{\varepsilon,\delta}, \mathcal{A}^{\text{zsum}}_{\varepsilon,\delta})$ has the following properties:*

(i) *$\mathcal{P}^{\text{zsum}}_{\varepsilon,\delta}$ is (ε, δ)-differentially private in the shuffled model.*

(ii) *For every* $\beta \geq \delta^{25}$ *, the error is* $|\mathcal{P}^{\text{zsum}}_{\varepsilon,\delta}(\vec{x}) - \frac{1}{n}\sum x_i| \leq \alpha$ *with probability* $\geq 1-\beta$ *where*

$$\begin{aligned}\alpha &= \frac{50}{\varepsilon^2 n}\log\frac{2}{\delta} + \frac{1}{\varepsilon n}\cdot\sqrt{200\log\frac{2}{\delta}\cdot\log\frac{2}{\beta}} \\ &= O\left(\frac{1}{\varepsilon^2 n}\log\frac{1}{\delta}\right)\end{aligned}$$

(iii) $\mathcal{P}^{\text{zsum}}_{\varepsilon,\delta}(\underbrace{(0,\ldots,0)}_{n \text{ copies}}) = 0.$

(iv) *Each user sends at most two one-bit messages.*

Proof of Part (i). If we let z_i be the random bit generated by the i-th user, the total number of messages is $|\vec{y}| = \sum_{i=1}^n x_i + z_i$. Observe that learning $|\vec{y}|$ is sufficient to represent the output of shuffler since all messages have the same value. Thus, the privacy of this protocol is equivalent to the privacy of

$$\mathcal{M}(\vec{x}) = \sum_{i=1}^n x_i + \text{Bin}(n,p) \sim -\left(-\sum_{i=1}^n x_i + \text{Bin}(n,1-p)\right) + n.$$

By post-processing, it suffices to show the privacy of $\mathcal{M}_{\text{neg}}(\vec{x}) = -\sum_{i=1}^n x_i + \text{Bin}(n,1-p)$ where $1-p = \frac{50}{\varepsilon^2 n}\ln\frac{2}{\delta}$. Because privacy follows almost immediately from technical claims in [13], we defer the proof to Appendix A. ◀

Proof of Part (ii). Fix any $\vec{x} \in \{0,1\}^n$. For shorthand, we define $\alpha' = 2\cdot\sqrt{\frac{p(1-p)}{n}\cdot\ln(2/\beta)}$ so that $\alpha = (1-p) + \alpha'$. A Chernoff bound implies that for $\beta \geq 2e^{-np(1-p)}$, the following event occurs with probability $\geq 1-\beta$:

$$\left|\frac{1}{n}\cdot\sum_{i=1}^n z_i - p\right| \leq \alpha' \tag{1}$$

The inequality $\beta \geq 2e^{-np(1-p)}$ follows from our bounds on ε, β, and n.

The remainder of the proof will condition on (1). If $c^* > 1$, then the analyzer outputs $c^* - p$. We show that the error of $c^* - p$ is at most α':

$$\begin{aligned}\left|(c^*-p) - \frac{1}{n}\cdot\sum_{i=1}^n x_i\right| &= \left|\frac{1}{n}\cdot\sum_{i=1}^n (x_i+z_i) - p - \frac{1}{n}\cdot\sum_{i=1}^n x_i\right| && \text{(By construction)} \\ &= \left|\frac{1}{n}\cdot\sum_{i=1}^n z_i - p\right| \\ &\leq \alpha' && \text{(By (1))}\end{aligned}$$

If $c^* \leq 1$, then the analyzer will output 0. In this case, the error is exactly $\frac{1}{n}\sum x_i$. We argue that $c^* \leq 1$ implies $\frac{1}{n}\sum x_i \leq \alpha$.

$$\begin{aligned}1 &\geq c^* \\ &= \frac{1}{n}\cdot\sum_{i=1}^n (x_i+z_i) && \text{(By construction)} \\ &\geq \frac{1}{n}\cdot\sum_{i=1}^n x_i + p - \alpha' && \text{(By (1))}\end{aligned}$$

Rearranging terms yields

$$\frac{1}{n} \cdot \sum_{i=1}^{n} x_i \leq (1-p) + \alpha' = \alpha$$

which concludes the proof. ◀

Proof of Part (iii). If $\vec{x} = (0, \ldots, 0)$, then $|\vec{y}|$ is drawn from $0 + \mathrm{Bin}(n, p)$, which implies $c^* \leq 1$ with probability 1. Hence, $\mathcal{P}^{\mathrm{zsum}}_{\varepsilon,\delta}(\vec{x}) = 0$. ◀

3.2 A Multi-Message Protocol for Histograms

In the protocol $\mathcal{P}^{\mathrm{hist}}_{\varepsilon,\delta}$ (Figure 2), users encode their data $x_i \in [d]$ as a one-hot vector $\vec{b} \in \{0,1\}^d$. Then protocol $\mathcal{P}^{\mathrm{zsum}}_{\varepsilon,\delta}$ is executed on each coordinate j of $\vec{b}$. The executions are done in one round of shuffling. To remove ambiguity between executions, each message in execution j has value j.

Randomizer $\mathcal{R}^{\mathrm{hist}}_{\varepsilon,\delta}(x \in [d])$ for $\varepsilon, \delta \in [0,1]$**:**
1. For each $j \in [d]$, let $b_j \leftarrow \mathbb{1}[x = j]$ and compute scalar product $\vec{m}_j \leftarrow j \cdot \mathcal{R}^{\mathrm{zsum}}_{\varepsilon,\delta}(b_j)$.
2. Output the concatenation of all $\vec{m}_j$.

Analyzer $\mathcal{A}^{\mathrm{hist}}_{\varepsilon,\delta}(\vec{y} \in [d]^*)$ for $\varepsilon, \delta \in [0,1]$**:**
1. For each $j \in [d]$, let $\vec{y}_{(j)} \leftarrow$ all messages of value j, then compute $\tilde{c}_j \leftarrow \mathcal{A}^{\mathrm{zsum}}_{\varepsilon,\delta}(\vec{y}_{(j)})$.
2. Output $(\tilde{c}_1, \ldots, \tilde{c}_d)$.

Figure 2 The pseudocode for $\mathcal{P}^{\mathrm{hist}}_{\varepsilon,\delta}$, a private shuffled protocol for histograms.

▶ **Theorem 12.** *For any $\varepsilon, \delta \in [0,1]$ and any $n \in \mathbb{N}$ such that $n \geq (100/\varepsilon^2) \cdot \ln(2/\delta)$, the protocol $\mathcal{P}^{\mathrm{hist}}_{\varepsilon,\delta} = (\mathcal{R}^{\mathrm{hist}}_{\varepsilon,\delta}, \mathcal{A}^{\mathrm{hist}}_{\varepsilon,\delta})$ has the following properties:*

(i) *$\mathcal{P}^{\mathrm{hist}}_{\varepsilon,\delta}$ is $(2\varepsilon, 2\delta)$-differentially private in the shuffled model.*

(ii) *For every $\beta \geq \delta^{25}$, $\mathcal{P}^{\mathrm{hist}}_{\varepsilon,\delta}$ has (α, β)-per-query accuracy for*

$$\alpha = O\left(\frac{1}{\varepsilon^2 n} \log \frac{1}{\delta}\right)$$

(iii) *For every $\beta \geq n \cdot \delta^{25}$, $\mathcal{P}^{\mathrm{hist}}_{\varepsilon,\delta}$ has (α, β)-simultaneous accuracy for*

$$\alpha = O\left(\frac{1}{\varepsilon^2 n} \log \frac{1}{\delta}\right)$$

(iv) *Each user sends at most $1 + d$ messages each of length $O(\log d)$.*

The accuracy guaranteed by this protocol is close to what is possible in the central model: there is a stability-based algorithm with simultaneous error $O((1/(\varepsilon n)) \cdot \ln(1/\delta))$ [7]. However, in $\mathcal{P}^{\mathrm{hist}}_{\varepsilon,\delta}$, each user communicates $O(d)$ messages of $O(\log d)$ bits. It remains an open question as to whether or not this can be improved while maintaining similar accuracy.

Because the simultaneous error of a *single-message* histogram protocol is at least $\Omega((1/(\varepsilon n)) \cdot \mathrm{poly}(\log d))$ [8], this protocol is also proof that the single-message model is

a strict subclass of the multi-message model. This separation was previously shown by [2, 1] for the summation problem.[2]

Proof of Part (i). Fix any neighboring pair of datasets $\vec{x} \sim \vec{x}\,'$. Let $\vec{y} \leftarrow (\mathcal{S} \circ \mathcal{R}^{\text{hist}}_{\varepsilon,\delta})(\vec{x})$ and $\vec{y}\,' \leftarrow (\mathcal{S} \circ \mathcal{R}^{\text{hist}}_{\varepsilon,\delta})(\vec{x}\,')$. For any $j \neq j'$, the count of j in output of the shuffler is independent of the count of j' in the output because each execution of $\mathcal{R}^{\text{zsum}}_{\varepsilon,\delta}$ is independent. As in Step (1) of $\mathcal{A}^{\text{hist}}_{\varepsilon,\delta}$, for $j \in [d]$, let $\vec{y}_{(j)}$ ($\vec{y}\,'_{(j)}$ resp.) be the vector of all messages in $\vec{y}$ ($\vec{y}\,'$ resp.) that have value j.

For any $j \in [d]$ where $c_j(\vec{x}) = c_j(\vec{x}\,')$, $\vec{y}_{(j)}$ is identically distributed to $\vec{y}\,'_{(j)}$. For each of the two $j \in [d]$ where $c_j(\vec{x}) \neq c_j(\vec{x}\,')$, we will show that the distribution of $\vec{y}_{(j)}$ is close to that of $\vec{y}\,'_{(j)}$. Let $\vec{r}, \vec{r}\,' \in \{0,1\}^n$ where $r_i = \mathbb{1}[x_i = j]$ and $r'_i = \mathbb{1}[x'_i = j]$. Now,

$$\vec{y}_{(j)} \sim j \cdot (\mathcal{S} \circ \mathcal{R}^{\text{zsum}}_{\varepsilon,\delta})(\vec{r}) \quad \text{and} \quad \vec{y}\,'_{(j)} \sim j \cdot (\mathcal{S} \circ \mathcal{R}^{\text{zsum}}_{\varepsilon,\delta})(\vec{r}\,').$$

So by Theorem 11 Part (i), for any $T \subseteq \{j\}^*$,

$$\Pr[\vec{y}_{(j)} \in T] \leq e^{\varepsilon} \cdot \Pr[\vec{y}\,'_{(j)} \in T] + \delta.$$

$(2\varepsilon, 2\delta)$-differential privacy follows by composition. ◀

Proof of Part (ii)-(iii). Notice that the j-th element in the output $\tilde{c}_j$ is identically distributed with an execution of the counting protocol on the bits $b_{i,j}$ indicating if $x_i = j$. Formally, $\tilde{c}_j \sim \mathcal{P}^{\text{zsum}}_{\varepsilon,\delta}(\{b_{i,j}\}_{i\in[n]})$ for all $j \in [d]$. Per-query accuracy immediately follows from Theorem 11 Part (ii).

To bound simultaneous error, we leverage the property that when $c_j(\vec{x}) = 0$, the counting protocol will report a nonzero value with probability 0. Let $Q = \{j \in [d] : c_j(\vec{x}) > 0\}$ and let α be the error bound defined in Theorem 11 Part (ii) for the failure probability β/n.

$$\begin{aligned}
&\Pr\left(\exists j \in [d] \text{ s.t. } |\tilde{c}_j - c_j(\vec{x})| > \alpha\right) \\
\leq &\Pr\left(\exists j \in Q \text{ s.t. } |\tilde{c}_j - c_j(\vec{x})| > \alpha\right) + \Pr\left(\exists j \notin Q \text{ s.t. } |\tilde{c}_j - c_j(\vec{x})| > \alpha\right) \\
= &\Pr\left(\exists j \in Q \text{ s.t. } |\tilde{c}_j - c_j(\vec{x})| > \alpha\right) && \text{(Theorem 11 Part (iii))} \\
\leq &\sum_{j \in Q} \Pr\left(|\tilde{c}_j - c_j(\vec{x})| > \alpha\right) \\
\leq &\sum_{j \in Q} \beta/n && \text{(Theorem 11 Part (ii))} \\
\leq &\beta && (|Q| \leq n)
\end{aligned}$$

This concludes the proof. ◀

3.3 Applications

In this section, we use our histogram protocol to solve two distributional problems; one of these results implies a very strong separation in sample complexity between the non-interactive local model and the shuffled model. Both distributional problems reduce to what we call *support identification*:

[2] In particular, a private unbiased estimator for $\sum_i x_i$ with real-valued $x_i \in [0,1]$ in the single-message shuffled model must have error $\Omega(n^{1/6})$ [2] while there exists a multi-message shuffled model protocol for estimating summation with error $O(1/\varepsilon)$ [1].

▶ **Definition 13** (Support Identification Problem). *The* support identification *problem has positive integer parameters $h \leq d$. Let D be a set of size d and let $\mathbf{U}_H$ be the uniform distribution over any $H \subseteq D$. The set of problem instances is $\{\mathbf{U}_H : H \subseteq D \text{ and } |H| = h\}$. A protocol solves the* (h, d)-support identification problem with sample complexity n *if, given n users with data independently sampled from any problem instance $\mathbf{U}_H$, it identifies H with probability at least 99/100.*

We now show how to solve this problem in the shuffled model.

▷ Claim 14. Fix any $\varepsilon \in (0, 1]$ and $\delta < (1/200h)^{1/25}$. Under (ε, δ)-differential privacy, the sample complexity of the (h, d)-support identification problem is $O(h \log h \cdot (1/\varepsilon^2) \cdot \log(1/\delta))$ in the shuffled model.

Proof. For the purposes of this proof, we assume there is some bijection f between D and $[d]$ so that any reference to $j \in [d]$ corresponds directly to some $f(j) \in D$ and vice versa. Consider the following protocol: execute $\mathcal{P}^{\text{hist}}_{\varepsilon,\delta}$ on n samples from $\mathbf{U}_H$ and then choose the items whose estimated frequencies are at least $(t + 1)/n$ (the magnitude of t will be determined later). We will prove that the items returned by the protocol are precisely those of H, with probability at least 99/100.

Let E_{samp} be the event that every element in support H has frequency at least $(2t+1)/n$ in the sample. Let E_{priv} be the event that the histogram protocol estimates the frequency of every element in D with error at most t/n. If both events occur, every element in H has estimated frequency at least $(t + 1)/n$ and every element outside H has estimated frequency at most t/n. Hence, it suffices to show that E_{samp} and E_{priv} each occur with probability $\geq 199/200$.

We lower bound the probability of E_{samp} via a coupon collector's argument. That is, if we have $n = O(kh \log h)$ samples from $\mathbf{U}_H$ then each element of H appears at least k times with probability at least 199/200. Hence we set $k = (2t + 1)$.

To lower bound the probability of E_{priv}, we simply invoke Theorem 12: given that $\varepsilon \in (0, 1]$ and $\delta > (1/200h)^{1/25}$, the frequency of every item in D is estimated up to error t/n for some $t = O((1/\varepsilon^2) \cdot \log(1/\delta))$ with probability $\geq 199/200$.[3] ◁

In the above analysis, if we had used a protocol with simultaneous error that depends on the domain size d, then t would in turn depend on d. For example, using the histogram protocol in [8] would give $t = \Omega((1/\varepsilon) \cdot \sqrt{\log d \cdot \log(1/\delta)})$. This results in a protocol whose sample complexity grows with d in addition to h.

So having shown how to solve the support identification problem with few samples, we now describe two different problems and explain how to reduce these to support identification. This will imply low sample complexity in the shuffled model.

▶ **Definition 15** (Pointer-Chasing Problem [16]). *The* pointer chasing problem *is denoted* $\text{PC}(k, \ell)$ *where k, ℓ are positive integer parameters. A problem instance is $\mathbf{U}_{\{(1,a),(2,b)\}}$ where a, b are permutations of $[\ell]$. A protocol* solves $\text{PC}(k, \ell)$ with sample complexity n *if, given n independent samples from any $\mathbf{U}_{\{(1,a),(2,b)\}}$, it outputs the k-th integer in the sequence $a_1, b_{a_1}, a_{b_{a_1}} \ldots$ with probability at least* 99/100.

To solve $\text{PC}(k, \ell)$, note that it suffices to identify $\{(1, a), (2, b)\}$ and directly perform the pointer chasing. Because the support has size $h = 2$, $\mathcal{P}^{\text{hist}}_{\varepsilon,\delta}$ can be used to solve the problem with just $O((1/\varepsilon^2) \cdot \log(1/\delta))$ samples, independent of k and ℓ. But in the case

[3] The bound on δ in Theorem 12 is a function of n. This is derived from a pessimistic bound on the number of unique values in the input. But in this reduction, we know that data takes one of h values.

where $k = 2$, [16] gives a lower bound of $\Omega(\ell/e^{\varepsilon})$ for non-interactive local protocols. So there is an arbitrarily large separation between the non-interactive shuffled and non-interactive local models (Theorem 2).

▶ **Definition 16** (Multi-Party Pointer Jumping Problem [15]). *The* multi-party pointer jumping problem *is denoted* $\mathrm{MPJ}(s,h)$ *where* s, h *are positive integer parameters. A problem instance is* $\mathbf{U}_{\{Z_1,\ldots,Z_h\}}$ *where each* Z_i *is a labeling of the nodes at level* i *in a complete* s*-ary tree. Each label* $Z_{i,j}$ *is an integer in* $\{0,\ldots,s-1\}$*. The labeling implies a root-leaf path: if the* i*-th node in the path has label* $Z_{i,j}$*, then the* $(i+1)$*-st node in the path is the* $(Z_{i,j})$*-th child of the* i*-th node. A protocol* solves $\mathrm{MPJ}(s,h)$ with sample complexity n *if, given* n *samples from any* $\mathbf{U}_{\{Z_1,\ldots,Z_h\}}$*, it identifies the root-leaf path with probability at least* $99/100$*.*

As with pointer-chasing, we can solve $\mathrm{MPJ}(s,h)$ when the support is identified. This takes $O(h\log h\cdot(1/\varepsilon^2)\cdot\log(1/\delta))$ samples in the shuffled model. But [15] gives a lower bound of $\Omega(h^3/(\varepsilon^2\log h))$ in the local model when $s = h^4$, even allowing for sequential interactivity. However, we do not claim a polynomial separation between the shuffled model and sequentially interactive local model. This would require a proof that every sequentially interactive local protocol has a counterpart in the shuffled model.

Note that the reductions we employ can also be applied in the central model. That is, instead of executing $\mathcal{P}^{\mathrm{hist}}_{\varepsilon,\delta}$ in the reduction (Claim 14), execute the central model algorithm, from [7], with simultaneous error $O((1/(\varepsilon n))\cdot\log(1/\delta))$. This improves the bounds by $1/\varepsilon$.

Table 2 The sample complexity of private pointer-chasing (PC) and multi-party pointer jumping (MPJ). Shuffled and central results follow from a reduction to histograms.

Model	$\mathrm{PC}(k,\ell)$	$\mathrm{MPJ}(s,h)$
Local	$\Omega(\ell/e^{\varepsilon})$ [16] for $k=2$	$\Omega(h^3/(\varepsilon^2\log h))$ [15] for $s=h^4$, seq. interactive
Shuffled	$O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$	$O\left(h\log h\cdot\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$ for $\delta<(1/200h)^{1/25}$
Central	$O\left(\frac{1}{\varepsilon}\log\frac{1}{\delta}\right)$	$O\left(h\log h\cdot\frac{1}{\varepsilon}\log\frac{1}{\delta}\right)$

4 Pure Differential Privacy in the Shuffled Model

In this section, we prove that any single-message shuffled protocol that satisfies ε-differential privacy can be simulated by a local protocol under the same privacy constraint.

▶ **Theorem 17** (Formalization of Thm. 3). *For any single-message shuffled protocol* $\mathcal{P} = (\mathcal{R},\mathcal{A})$ *that satisfies* ε*-differential privacy, there exists a local protocol* $\mathcal{P}_L = (\mathcal{R}_L,\mathcal{A}_L)$ *that satisfies* ε*-differential privacy and* $\mathcal{P}_L(\vec{x})$ *is identically distributed to* $\mathcal{P}(\vec{x})$ *for every input* $\vec{x}\in\mathcal{X}^n$*.*

We start with the following claim, which strengthens a theorem in [8] for the special case of pure differential privacy in the shuffled model:

▷ Claim 18. Let $\mathcal{P} = (\mathcal{R},\mathcal{A})$ be any single-message shuffled protocol that satisfies ε-differential privacy. Then $\mathcal{R}$ is an ε-differentially private algorithm.

Proof. Assume for contradiction that $\mathcal{R}$ is not ε-differentially private. So there are values $x, x' \in \mathcal{X}$ and a set $Y \subseteq \mathcal{Y}$ such that

$$\Pr[\mathcal{R}(x) \in Y] > e^{\varepsilon} \cdot \Pr[\mathcal{R}(x') \in Y].$$

Let $\vec{x} = (\underbrace{x, \ldots, x}_{n \text{ copies}})$ and $\vec{x}\,' = (x', \underbrace{x, \ldots, x}_{n-1 \text{ copies}})$. Now consider Y^n, the set of message vectors where each message belongs to Y.

$$\begin{aligned}
\Pr[(\mathcal{S} \circ \mathcal{R})(\vec{x}) \in Y^n] &= \Pr[\mathcal{R}(\vec{x}) \in Y^n] \\
&= \Pr[\mathcal{R}(x) \in Y]^n \\
&> e^{\varepsilon} \cdot \Pr[\mathcal{R}(x') \in Y] \cdot \Pr[\mathcal{R}(x) \in Y]^{n-1} \\
&= e^{\varepsilon} \cdot \Pr[(\mathcal{S} \circ \mathcal{R})(\vec{x}\,') \in Y^n]
\end{aligned}$$

which contradicts the fact that $\mathcal{S} \circ \mathcal{R}$ is ε-differentially private. ◁

Now we are ready to prove Theorem 17.

Proof of Theorem 17. Consider the aggregator $\mathcal{A}_L$ that applies a uniformly random permutation to its input and then executes $\mathcal{A}$. Then $\mathcal{P}_L = (\mathcal{R}, \mathcal{A}_L)$ is a local protocol that simulates $\mathcal{P}$, in the sense that $\mathcal{P}_L(\vec{x})$ is identically distributed to $\mathcal{P}(\vec{x})$ for every $\vec{x} \in \mathcal{X}^n$. And by Claim 18, the randomizer is ε-differentially private. ◀

4.1 Roadblocks to Generalizing Theorem 17

One might conjecture Claim 18 also holds for multi-message protocols and thus immediately generalize Theorem 17. However, this is not the case:

▷ Claim 19. There exists a multi-message shuffled protocol that is ε-differentially private for all $\varepsilon \geq 0$ but its randomizer is not ε-differentially private for *any* finite ε.

Proof. Consider the randomizer $\mathcal{R}^{\infty}$ that on input $x \in \{0, 1\}$ outputs two messages x and $1 - x$. The output of the shuffler $S \circ \mathcal{R}^{\infty}$ is 0-differentially private since for all inputs the output is a random permutation of exactly n 0s and n 1s. However, $\mathcal{R}^{\infty}$ is not ε-differentially private for any finite ε as the first message of $\mathcal{R}^{\infty}(x)$ is that user's bit x. ◁

We note that it is without loss of accuracy or privacy to suppose that a randomizer shuffles its messages prior to sending them to the shuffler. We call these *pre-shuffle* randomizers. Observe that the pre-shuffle version of $\mathcal{R}^{\infty}$ (i.e. $\mathcal{S} \circ \mathcal{R}^{\infty}$ for 1 user) satisfies 0-differential privacy. So one might conjecture Claim 18 holds for pre-shuffle randomizers and thus generalize Theorem 17. But this too is not the case:

▷ Claim 20. There exists a multi-message shuffled protocol that is ε-differentially private for some finite ε but its pre-shuffle randomizer is not ε-differentially private for *any* finite ε.

Proof. Consider any randomizer $\mathcal{R}^{\text{gap}}$ that takes binary input and outputs four binary messages with the following constraint: the messages can take any value when the input is 0 but on input 1, there cannot be exactly two 1s. Formally, the supports are $\text{supp}(\mathcal{R}^{\text{gap}}(0)) = \{0, 1\}^4$ and $\text{supp}(\mathcal{R}^{\text{gap}}(1)) = \{\vec{y} \in \{0, 1\}^4 : \sum_i y_i \neq 2\}$.

The pre-shuffle randomizer $\mathcal{S} \circ \mathcal{R}^{\text{gap}}$ cannot satisfy pure differential privacy because $(0, 0, 1, 1) \in \text{supp}(\mathcal{R}^{\text{gap}}(0))$ but $(0, 0, 1, 1) \notin \text{supp}(\mathcal{R}^{\text{gap}}(1))$. On the other hand, for all $n \geq 2$ and $\vec{x} \in \{0, 1\}^n$,

$$\text{supp}(\mathcal{S} \circ \mathcal{R}^{\text{gap}}(\vec{x})) = \{0, 1\}^{4n}$$

This follows from the fact that every number in $\{0, \ldots, 4n\}$ – the number of 1s sent to the shuffler – can be expressed as the sum of n numbers from $\{0, 1, 3, 4\}$. Thus, there is some finite ε for which the protocol with randomizer $\mathcal{R}^{\text{gap}}$ is ε-differentially private. ◁

References

1 Borja Balle, James Bell, Adria Gascon, and Kobbi Nissim. Differentially private summation with multi-message shuffling. *arXiv preprint arXiv:1906.09116*, 2019.

2 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. 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 638–667. Springer, 2019. doi:10.1007/978-3-030-26951-7_22.

3 Raef Bassily and Adam D. Smith. Local, private, efficient protocols for succinct histograms. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 127–135. ACM, 2015. doi:10.1145/2746539.2746632.

4 Amos Beimel, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*, pages 437–454. Springer, 2010. doi:10.1007/978-3-642-11799-2_26.

5 Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 451–468. Springer, 2008. doi:10.1007/978-3-540-85174-5_25.

6 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 *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 441–459. ACM, 2017. doi:10.1145/3132747.3132769.

7 Mark Bun, Kobbi Nissim, and Uri Stemmer. Simultaneous private learning of multiple concepts. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 369–380. ACM, 2016. doi:10.1145/2840728.2840747.

8 Albert Cheu, Adam D. Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 375–403. Springer, 2019. doi:10.1007/978-3-030-17653-2_13.

9 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006. doi:10.1007/11681878_14.

10 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 Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2468–2479. SIAM, 2019. doi:10.1137/1.9781611975482.151.

11 Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In Frank Neven, Catriel Beeri, and Tova Milo, editors, *PODS*, pages 211–222. ACM, 2003. doi:10.1145/773153.773174.

12 Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Pure differentially private summation from anonymous messages. *CoRR*, abs/2002.01919, 2020.

13 Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the power of multiple anonymous messages. *IACR Cryptology ePrint Archive*, 2019:1382, 2019.

14 Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 705–714. ACM, 2010. doi:10.1145/1806689.1806786.

15 Matthew Joseph, Jieming Mao, Seth Neel, and Aaron Roth. The role of interactivity in local differential privacy. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 94–105. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00015.

16 Matthew Joseph, Jieming Mao, and Aaron Roth. Exponential separations in local differential privacy. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 515–527. SIAM, 2020. doi:10.1137/1.9781611975994.31.

17 Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What can we learn privately? In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 531–540. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.27.

18 Tianhao Wang, Min Xu, Bolin Ding, Jingren Zhou, Ninghui Li, and Somesh Jha. Practical and robust privacy amplification with multi-party differential privacy. *arXiv preprint arXiv:1908.11515*, 2019.

19 Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

A Privacy via Smooth Distributions

Ghazi, Golowich, Kumar, Pagh and Velingker [13] identify a class of distributions and argue that, if η is sampled from such a distribution, adding η to a 1-sensitive sum ensures differential privacy of that sum.

▶ **Definition 21** (Smooth Distributions, [13]). *A distribution* $\mathbf{D}$ *over* $\mathbb{Z}$ *is* (ε, δ, k)*-smooth if for all* $k' \in [-k, k]$*,*

$$\Pr_{Y \sim \mathbf{D}}\left[\frac{\Pr_{Y' \sim \mathbf{D}}[Y' = Y]}{\Pr_{Y' \sim \mathbf{D}}[Y' = Y + k']} \geq e^{|k'|\varepsilon}\right] \leq \delta.$$

▶ **Lemma 22** (Smoothness for Privacy, [13]). *Let* $f : \mathbb{Z}^n \to \mathbb{Z}$ *be a function such that* $|f(\vec{x}) - f(\vec{x}')| \leq 1$ *for all* $\vec{x} \sim \vec{x}'$*. Let* $\mathbf{D}$ *be an* $(\varepsilon, \delta, 1)$*-smooth distribution. The algorithm that takes as input* $\vec{x} \in \mathbb{Z}^n$*, then samples* $\eta \sim \mathbf{D}$ *and reports* $f(\vec{x}) + \eta$ *satisfies* (ε, δ)*-differential privacy.*

▶ **Lemma 23** (Binomial Distribution is Smooth, [13]). *For any positive integer* n*,* $\gamma \in [0, 1/2]$*,* $\alpha \in [0, 1]$*, and any* $k \leq \alpha\gamma n/2$*, the distribution* $\mathrm{Bin}(n, \gamma)$ *is* (ε, δ, k)*-smooth with*

$$\varepsilon = \ln \frac{1+\alpha}{1-\alpha} \qquad \text{and} \qquad \delta = \exp\left(-\frac{\alpha^2\gamma n}{8}\right) + \exp\left(-\frac{\alpha^2\gamma n}{8+2\alpha}\right).$$

▶ **Corollary 24.** *Fix any $\varepsilon, \delta \in [0,1]$. Let $n \geq (100/\varepsilon^2) \cdot \ln(2/\delta)$. The algorithm $\mathcal{M}_{neg}$ that takes as input $\vec{x} \in \{0,-1\}^n$ then samples*

$$\eta \sim \mathrm{Bin}\left(n,\, 50 \cdot \frac{\ln(2/\delta)}{n\varepsilon^2}\right)$$

and reports $\eta + \sum x_i$ satisfies (ε, δ)-differential privacy.

Proof. When $\alpha = (e^\varepsilon - 1)/(e^\varepsilon + 1)$ observe that $\alpha \in [\varepsilon/\sqrt{5}, 1)$ and Lemma 23 implies that η is sampled from an $(\varepsilon, \delta, 1)$-smooth distribution:

$$\ln \frac{1+\alpha}{1-\alpha} = \ln \frac{(e^\varepsilon + 1) + (e^\varepsilon - 1)}{(e^\varepsilon + 1) - (e^\varepsilon - 1)} = \varepsilon$$

and

$$\begin{aligned} \exp\left(-\frac{\alpha^2\gamma n}{8}\right) + \exp\left(-\frac{\alpha^2\gamma n}{8+2\alpha}\right) &\leq 2\exp\left(-\frac{\alpha^2\gamma n}{10}\right) && (\alpha < 1) \\ &\leq 2\exp\left(-\frac{\gamma\varepsilon^2 n}{50}\right) \\ &= \delta. \end{aligned}$$

So by Lemma 22, we have $\mathcal{M}_{\mathrm{neg}}$ is (ε, δ)-differentially private. ◀

d-Multiplicative Secret Sharing for Multipartite Adversary Structures

Reo Eriguchi
Graduate School of Information Science and Technology, The University of Tokyo, Japan
reo-eriguchi@g.ecc.u-tokyo.ac.jp

Noboru Kunihiro
Department of Computer Science, University of Tsukuba, Japan
kunihiro@cs.tsukuba.ac.jp

Abstract

Secret sharing schemes are said to be d-multiplicative if the i-th shares of any d secrets $s^{(j)}$, $j \in [d]$ can be converted into an additive share of the product $\prod_{j\in[d]} s^{(j)}$. d-Multiplicative secret sharing is a central building block of multiparty computation protocols with minimum number of rounds which are unconditionally secure against possibly non-threshold adversaries. It is known that d-multiplicative secret sharing is possible if and only if no d forbidden subsets covers the set of all the n players or, equivalently, it is private with respect to an adversary structure of type Q_d. However, the only known method to achieve d-multiplicativity for any adversary structure of type Q_d is based on CNF secret sharing schemes, which are not efficient in general in that the information ratios are exponential in n.

In this paper, we explicitly construct a d-multiplicative secret sharing scheme for any ℓ-partite adversary structure of type Q_d whose information ratio is $O(n^{\ell+1})$. Our schemes are applicable to the class of all the ℓ-partite adversary structures, which is much wider than that of the threshold ones. Furthermore, our schemes achieve information ratios which are polynomial in n if ℓ is constant and hence are more efficient than CNF schemes. In addition, based on the standard embedding of ℓ-partite adversary structures into $\mathbb{R}^\ell$, we introduce a class of ℓ-partite adversary structures of type Q_d with good geometric properties and show that there exist more efficient d-multiplicative secret sharing schemes for adversary structures in that family than the above general construction. The family of adversary structures is a natural generalization of that of the threshold ones and includes some adversary structures which arise in real-world scenarios.

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

Keywords and phrases Secret sharing scheme, multiplicative secret sharing scheme, multipartite adversary structure

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.2

Funding This research was partially supported by JST CREST Grant Number JPMJCR14D6, Japan and JSPS KAKENHI Grant Number JP19K22838.

1 Introduction

Secret sharing is a cryptographic technique introduced in [4, 20] to protect a secret from leakage by dividing it into several shares and distributing them to n players. Let P be the set of the n players. A subset of players is called *forbidden* if it reveals no information on the secret and *authorized* if it determines the secret. In this paper, we only consider *perfect* secret sharing, in which each subset is either forbidden or authorized. For a family of subsets of players $\Delta \subseteq 2^P$, we say that a secret sharing scheme is Δ-*private* if any subset in Δ is forbidden. The efficiency of a secret sharing scheme is measured by the (total) *information ratio*, which is defined as the ratio of the total size of shares to that of a secret.

In addition to its direct application to distributed storage of secret data, secret sharing is a central buiding block of unconditionally secure multiparty computation protocols. Secure *multiparty computation* (MPC) is an important problem in cryptography, in which several players jointly compute an agreed function over their inputs without revealing no information on them to an adversary. More precisely, we assume that there are m clients holding their secret inputs $x_j \in \mathbb{F}$, $j \in [m] := \{1, 2, \ldots, m\}$ represented as elements in some finite field $\mathbb{F}$ and n servers who help the computation of the function. Furthermore, a (passive) adversary corrupts a subset of servers and learns the entire internal information of the corrupted servers. We identify the set of servers with P and let $\Delta \subseteq 2^P$ denote the family of all subsets of servers which the adversary can corrupt, which we call the *adversary structure*.

Barkol et al. [1] show that it is possible to construct a protocol with two rounds of interaction, which is the minimum number of rounds, to securely evaluate a multivariate polynomial of total degree at most d if there exists a Δ-private secret sharing scheme satisfying an additional property called *d-multiplicativity*. A secret sharing scheme is said to be *d-multiplicative* if the i-th shares of any d secrets $s^{(j)}$, $j \in [d]$ can be converted into an element c_i such that the sum $\sum_{i\in[n]} c_i$ is equal to the product $\prod_{j\in[d]} s^{(j)}$. Note that the dominant term of the communication complexity in the protocol is $m\sigma \log |\mathbb{F}|$, where m is the number of clients and σ is the information ratio of the underlying secret sharing scheme.

Barkol et al. [1] also characterize the existence of d-multiplicative secret sharing schemes: Δ-private d-multiplicative secret sharing is possible if and only if Δ is of *type* Q_d. An adversary structure Δ is called a *Q_d-adversary structure* if $A_1 \cup \cdots \cup A_d \neq P$ for any $A_1, \ldots, A_d \in \Delta$.

In particular, if we focus on the adversary who can corrupt any subset of k servers, i.e., $\Delta = \mathcal{T}_k^n := \{A \subseteq P : |A| \leq k\}$, then $\mathcal{T}_k^n$-private d-multiplicative secret sharing is possible if and only if $n > dk$. The "if" part follows from the (k, n)-*Shamir secret sharing scheme* [20], which is based on Lagrange interpolation for some polynomial over any finite field $\mathbb{F}$ with $|\mathbb{F}| > n$. It is known that Shamir's scheme achieves the optimal information ratio [16]. On the other hand, since it costs a lot to control a single server, the number of servers n should be as small as possible. The above characterization implies that we must consider non-threshold Q_d-adversary structures to design d-multiplicative secret sharing among $n \leq dk$ players.

For any (possibly non-threshold) adversary structure Δ of type Q_d, the *CNF secret sharing scheme* [15] for Δ is known to be d-multiplicative. However, the CNF scheme is inefficient in general since its information ratio is $\sum_{i\in P} |\Delta_i^+|$, which is exponential in n in the worst case. Here, Δ_i^+ denotes the set of all the maximal subsets in Δ not containing the player $i \in P$. This large information ratio of the CNF scheme leads to a large amount of communication when the scheme is used in the protocol. Therefore, it is important to devise a method to construct efficient d-multiplicative secret sharing schemes for non-threshold Q_d-adversary structures.

Besides Shamir's scheme and CNF schemes, several multiplicative secret sharing schemes have been proposed. The notion of an *arithmetic codex* is introduced in [7]. Arithmetic codices are defined as linear codes with some multiplicative property and can be used as d-multiplicative secret sharing schemes. In particular, arithmetic codices based on algebraic geometric codes [8] are important. Let d and k be any positive integers and assume that C is an algebraic curve of genus $g(C)$ defined over $\mathbb{F}$. If $dk + 2dg(C) < n < |C(\mathbb{F})|$, then there exists a $\mathcal{T}_k^n$-private d-multiplicative secret sharing scheme over $\mathbb{F}$. Here, $C(\mathbb{F})$ is the set of all $\mathbb{F}$-rational places on C. Although we need additional $2dg(C)$ players, the condition $|C(\mathbb{F})| > n$ is weaker than $|\mathbb{F}| > n$.

2-Multiplicative secret sharing schemes have received a lot of attention. For general adversary structures, one of the most significant results is that any linear secret sharing scheme for a Q_2-adversary structure can be converted into a 2-multiplicative scheme for the same adversary structure [9]. Since the information ratio of the resulting scheme is twice that of the initial scheme, it achieves $2\lambda_{\mathbb{F}}(\Delta)$, where $\lambda_{\mathbb{F}}(\Delta)$ is the minimum information ratio of Δ-private linear secret sharing schemes. In [17, 18], more efficient 2-multiplicative schemes are proposed for specific classes of Q_2-adversary structures.

1.1 Our Results

In this paper, we focus on multipartite adversary structures. The class of *multipartite adversary structures* has been well studied because they correspond to many realistic situations and can be described in a compact way. Please refer to [12] for a comprehensive survey of multipartite adversary structures. For a partition $\Pi = (P_1, \ldots, P_\ell)$ of P, Π-partite adversary structures are defined as the ones in which each player is classified into some part P_j and all players in the same part play an equivalent role. There is a useful geometric representation [11]: any Π-partite adversary structure can be embedded in $\mathbb{R}^\ell$ via the map $\Phi^\Pi : 2^P \to \mathbb{R}^\ell$, $\Phi^\Pi(X) = (|X \cap P_1|, \ldots, |X \cap P_\ell|)$. In particular, a Π-partite adversary structure Δ is uniquely determined by $\max \Phi^\Pi(\Delta)$, where $\max \Phi^\Pi(\Delta)$ is the set of all maximal elements in $\Phi^\Pi(\Delta)$ with respect to the coordinatewise order on $\mathbb{R}^\ell$.

The main contribution of this paper is twofold. First, for any ℓ-partite adversary structure Δ of type Q_d, we explicitly construct a Δ-private d-multiplicative secret sharing scheme whose information ratio is $n|\max \Phi^\Pi(\Delta)| = O(n^{\ell+1})$ (Theorem 4). The scheme can be defined over any finite field $\mathbb{F}$ with $|\mathbb{F}| > n$. It is obtained by a simple application of well-known decomposition techniques [21]. However, to our best knowledge, this is the first time to prove that the scheme satisfies d-multiplicativity if Δ is of type Q_d. As a result, we obtain d-multiplicative schemes for the class of all the ℓ-partite adversary structures of type Q_d, which is much wider than that of the threshold ones. Furthermore, our schemes achieve information ratios which are polynomial in n if ℓ is constant and hence are more efficient than CNF schemes.

Second, we show that there exists a more efficient Δ-private d-multiplicative secret sharing scheme than the scheme from the above general construction if $\Phi^\Pi(\Delta)$ has some good geometric property (Theorem 9). Specifically, let $C = \mathrm{Conv}(\Phi^\Pi(\Delta))$ be the convex hull of $\Phi^\Pi(\Delta)$ in $\mathbb{R}^\ell$ and set $\boldsymbol{p} = (1/d)\Phi^\Pi(P)$. Assume that $\boldsymbol{p} \notin C$. If $\mathrm{dist}(\boldsymbol{p}, C)$, the distance between $\boldsymbol{p}$ and C, is at least $\epsilon > 0$, then Δ is of type Q_d and it is possible to construct a Δ-private d-multiplicative scheme whose information ratio is at most $O(\ell n^2/\epsilon)$ over a finite field $\mathbb{F}$ with $|\mathbb{F}| = \Omega(\ell n^2/\epsilon)$. For example, if ϵ is a constant independent of n, then the information ratio is smaller than that of the above general construction.

The information ratio of our scheme depends only on the distance between the point $\boldsymbol{p}$ and the convex hull of $\Phi^\Pi(\Delta)$. In other words, our scheme provides the same upper bound on the minimum information ratio of Δ-private secret sharing schemes regardless of how the adversary structure Δ is represented. We demonstrate an example of adversary structures (Example 11) for which an upper bound obtained by a naive approach based on weighted threshold secret sharing would grow infinitely depending on the description of Δ.

Our construction for such Δ is a natural generalization of Shamir's scheme. Indeed, when $\Delta = \mathcal{T}_k^n$, the condition $\boldsymbol{p} \notin C$ holds if and only if $n > dk$ and then our scheme is the same as the (k, n)-Shamir secret sharing scheme.

Table 1 Comparison amongst existing d-multiplicative secret sharing schemes. The symbol "$d = *$" denotes any value of d.

Scheme	d	Adversary structure	Information ratio	Assumption on $\mathbb{F}$
Shamir [20]	$*$	$\mathcal{T}_k^n$ with $n > dk$	n	$\lvert\mathbb{F}\rvert > n$
Chen and Cramer [8]	$*$	$\mathcal{T}_k^n$ with $n > dk + 2dg(C)$	n	$\lvert C(\mathbb{F})\rvert > n$
CNF [15]	$*$	Δ of type Q_d	$\sum_{i\in P} \lvert\Delta_i^+\rvert$	–
Cramer et al. [9]	2	Δ of type Q_2	$2\lambda_{\mathbb{F}}(\Delta)$	–
Theorem 4	$*$	ℓ-partite Δ of type Q_d	$n\lvert\max \Phi^{\Pi}(\Delta)\rvert$	$\lvert\mathbb{F}\rvert > n$
Theorem 9	$*$	ℓ-partite Δ such that $\operatorname{dist}(\boldsymbol{p}, \operatorname{Conv}(\Phi^{\Pi}(\Delta))) \geq \epsilon$	$O(\ell n^2/\epsilon)$	$\lvert\mathbb{F}\rvert = \Omega(\ell n^2/\epsilon)$

1.2 Our Techniques

Let $\Pi = (P_1, \ldots, P_\ell)$ be a partition of P and Δ be any Π-partite adversary structure of type Q_d. First, we explain a high-level idea of our proposed construction of a Δ-private d-multiplicative secret sharing scheme using decomposition techniques [21].

Write $\max \Phi^{\Pi}(\Delta) = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N\}$ and $\boldsymbol{a}_j = (\boldsymbol{a}_j(1), \ldots, \boldsymbol{a}_j(\ell))$. For each $j \in [N]$, define Δ_j as the Π-partite adversary structure such that $\Phi^{\Pi}(\Delta_j) = \{\boldsymbol{x} \in \Phi^{\Pi}(2^P) : \boldsymbol{x} \preceq \boldsymbol{a}_j\}$, where $\preceq$ is the coordinatewise order on $\mathbb{R}^\ell$. Then we can decompose Δ into N adversary structures $\Delta_1, \ldots, \Delta_N$ as $\Delta = \bigcup_{j\in[N]} \Delta_j$.

We construct a Δ-private scheme Σ as follows. Let $\mathbb{F}$ be a finite field with $\lvert\mathbb{F}\rvert > n$ and fix n distinct nonzero elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$. Let $s \in \mathbb{F}$ be a secret to be shared. We randomly split s into $s_1, \ldots, s_N$, i.e., $s = s_1 + \cdots + s_N$. For each $j \in [N]$ and $i \in [\ell]$, we run the $(\boldsymbol{a}_j(i), \lvert P_i\rvert)$-Shamir secret sharing scheme with secret s_j in parallel. In other words, we assign to each player $k \in P_i$ the evaluation of some polynomial f_{ij} with $f_{ij}(0) = s_j$ of degree at most $\boldsymbol{a}_j(i)$ at the point α_k. Then the player $k \in P_i$ receives N field elements $\{s_{i,j,k} := f_{ij}(\alpha_k) : j \in [N]\}$. As for privacy, if $A \in \Delta$, then $\Phi^{\Pi}(A) \preceq \boldsymbol{a}_j$ for some $j \in [N]$ and the players in A cannot obtain any information about s_j and hence the secret s. Since each player receives N field elements for a secret $s \in \mathbb{F}$, the information ratio of Σ is nN.

To prove the scheme Σ is d-multiplicative, let $s^{(1)}, \ldots, s^{(d)}$ be d secrets. Since we split $s^{(m)}$ into $s_1^{(m)}, \ldots, s_N^{(m)}$, the product $s^{(1)} \cdots s^{(d)}$ can be represented as the sum of N^d monomials $\{s_{j_1}^{(1)} \cdots s_{j_d}^{(d)} : j_1 \in [N], \ldots, j_d \in [N]\}$. It follows from Δ being of type Q_d that, for any tuple $(j_1, \ldots, j_d)$, there exists an index $i \in [\ell]$ such that $\boldsymbol{a}_{j_1}(i) + \cdots + \boldsymbol{a}_{j_d}(i) < \lvert P_i\rvert$. Let $(j_1, \ldots, j_d)$ be any tuple and i be such an index. From the definition of the $(\boldsymbol{a}_{j_m}(i), \lvert P_i\rvert)$-Shamir secret sharing scheme, it can be observed that $s_{j_m}^{(m)}$ is the evaluation of some polynomial of degree at most $\boldsymbol{a}_{j_m}(i)$ at the point 0. Then the monomial $s_{j_1}^{(1)} \cdots s_{j_d}^{(d)}$ is the evaluation of some polynomial f of degree at most $\boldsymbol{a}_{j_1}(i) + \cdots + \boldsymbol{a}_{j_d}(i)$ at the point 0. Each player k in P_i can obtain the evaluation of f at the point α_k by multiplying their shares $s_{i,j_1,k}^{(1)}, \ldots, s_{i,j_d,k}^{(d)}$. Since there are more than $\boldsymbol{a}_{j_m}(i) + \cdots + \boldsymbol{a}_{j_d}(i)$ players in P_i, the monomial $s_{j_1}^{(1)} \cdots s_{j_d}^{(d)}$ can be obtained from their shares by Lagrange interpolation. Finally, the product $s^{(1)} \cdots s^{(d)}$ can be obtained by doing this process for all the N^d tuple $(j_1, \ldots, j_d)$.

Next, let Δ be a Π-partite adversary structure with $\operatorname{dist}(\boldsymbol{p}, C) \geq \epsilon > 0$, where $\boldsymbol{p} = (1/d)\Phi^{\Pi}(P)$ and $C = \operatorname{Conv}(\Phi^{\Pi}(\Delta))$. We explain how to prove the existence of a Δ-private d-multiplicative secret sharing scheme whose information ratio is $O(\ell n^2/\epsilon)$. Roughly speaking, we show that Δ is contained by some weighted threshold adversary structure. For a vector

of non-negative integers $\boldsymbol{w}$ and a non-negative integer t, the *weighted threshold adversary structure* $\mathcal{W}^{\Pi}_{\boldsymbol{w},t}$ is defined as the Π-partite adversary structure such that $\Phi^{\Pi}(\mathcal{W}^{\Pi}_{\boldsymbol{w},t}) = \{\boldsymbol{x} \in \Phi^{\Pi}(2^P) : \boldsymbol{w} \cdot \boldsymbol{x} \leq t\}$, where $\boldsymbol{w} \cdot \boldsymbol{x}$ is the standard inner product in $\mathbb{R}^{\ell}$. If $\boldsymbol{w} \cdot \Phi^{\Pi}(P) > dt$, it is possible to construct a $\mathcal{W}^{\Pi}_{\boldsymbol{w},t}$-private d-multiplicative scheme whose information ratio is $\boldsymbol{w} \cdot \Phi^{\Pi}(P)$ by assigning multiple shares of Shamir's scheme to each player [20].

More precisely, let $\boldsymbol{c}^*$ be the closest point in C to $\boldsymbol{p}$ and $\boldsymbol{h}$ be the unit vector which is parallel to $\boldsymbol{p} - \boldsymbol{c}^*$. Then it can be verified that $\boldsymbol{h} \cdot (\boldsymbol{x} - \boldsymbol{p}) \leq -\epsilon$ for any $\boldsymbol{x} \in \Phi^{\Pi}(\Delta)$. Although $\boldsymbol{h}$ is not necessarily a non-negative integer vector, we can approximate it by a vector $\boldsymbol{h}'$ such that all the entries are rational numbers with common denominator $q = O(\ell n/\epsilon)$ and $\boldsymbol{h}' \cdot (\boldsymbol{x} - \boldsymbol{p}) < 0$ for any $\boldsymbol{x} \in \Phi^{\Pi}(\Delta)$. By setting $\boldsymbol{w} = q\boldsymbol{h}'$ and $t = \max_{\boldsymbol{x} \in \Phi^{\Pi}(\Delta)}\{\boldsymbol{w} \cdot \boldsymbol{x}\}$, we have that $\Delta \subseteq \mathcal{W}^{\Pi}_{\boldsymbol{w},t}$. Since $dt < \boldsymbol{w} \cdot \Phi^{\Pi}(P)$ and each entry of $\boldsymbol{w}$ can be upper bounded by $q = O(\ell n/\epsilon)$, we obtain a Δ-private d-multiplicative scheme Σ with information ratio $O(\ell n^2/\epsilon)$.

1.3 Related Work

There is another kind of MPC protocols based on secret sharing, in which the function to be computed is represented as an arithmetic circuit and servers interactively evaluate it gate by gate. In the threshold setting, the protocol in [3] is classically known and a more efficient protocol is proposed in [10]. For a non-threshold adversary, Cramer et al. [9] construct an MPC protocol based on 2-multiplicative secret sharing schemes and Maurer [19] and Hirt and Tschudi [14] construct protocols based on CNF secret sharing schemes. Their protocols are secure against an adversary whose adversary structure is of type Q_2 (or Q_3 in the setting with perfect active security) and hence they are more flexible than that of [1]. However, the servers need to interact with each other whenever they evaluate a multiplication gate.

d-Multiplicative secret sharing can also be defined in the context of *homomorphic secret sharing*. Recently, several homomorphic secret sharing schemes have been proposed in the literature (e.g. [5, 6]). However, the security of the schemes relies on some computational assumptions.

1.4 Notations

Let $\mathbb{Z}_+$ and $\mathbb{R}_+$ denote the set of all non-negative integers and the set of all non-negative real numbers, respectively. Define $[\ell] = \{1, \ldots, \ell\}$ for $\ell \in \mathbb{N}$. Let $P = \{p_1, \ldots, p_n\}$ be the set of n players. The power set of a set X is denoted by 2^X and X^m is the Cartesian product of m copies of X. Let $\mathbb{F}$ be any field. The vector $\mathbf{1} \in \mathbb{F}^m$ is the vector whose entries are all one and $\boldsymbol{e}_i \in \mathbb{F}^m$ is the i-th unit vector, i.e., the vector such that the i-th entry is one and the other entries are all zero. The i-th component of $\boldsymbol{v}$ is denoted by $\boldsymbol{v}(i)$. For two real vectors $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^m$, we write $\boldsymbol{v} \preceq \boldsymbol{w}$ if $\boldsymbol{v}(i) \leq \boldsymbol{w}(i)$ for any $i \in [m]$ and $\boldsymbol{v} \prec \boldsymbol{w}$ if $\boldsymbol{v} \preceq \boldsymbol{w}$ and $\boldsymbol{v} \neq \boldsymbol{w}$. The standard inner product of $\boldsymbol{v}$ and $\boldsymbol{w}$ is $\boldsymbol{v} \cdot \boldsymbol{w} = \boldsymbol{v}(1)\boldsymbol{w}(1) + \cdots + \boldsymbol{v}(m)\boldsymbol{w}(m)$. The length of $\boldsymbol{v} \in \mathbb{R}^m$ is measured by the Euclidean norm: $\|\boldsymbol{v}\| = \sqrt{\boldsymbol{v} \cdot \boldsymbol{v}}$. For two closed subsets C_1 and C_2 in $\mathbb{R}^m$, the distance between C_1 and C_2 is defined by $\mathrm{dist}(C_1, C_2) = \min\{\|\boldsymbol{c}_1 - \boldsymbol{c}_2\| : \boldsymbol{c}_1 \in C_1, \boldsymbol{c}_2 \in C_2\}$.

2 Preliminaries

2.1 Adversary Structures

A family Δ of subsets of P is *monotone decreasing* if $A \in \Delta$ and $A \supseteq B$ implies $B \in \Delta$ for any $A, B \subseteq P$. We call a monotone decreasing family of subsets of P an *adversary structure* on P.

Let Δ be an adversary structure on P and $d \geq 2$. We say that Δ is of *type* Q_d if $A_1 \cup \cdots \cup A_d \neq P$ for any $A_1, \ldots, A_d \in \Delta$.

The (k, n)-*threshold adversary structure* $\mathcal{T}_k^n$ is the most well-known adversary structure, which is defined by $\mathcal{T}_k^n = \{A \subseteq P : |A| \leq k\}$. It can be seen that $\mathcal{T}_k^n$ is of type Q_d if and only if $n > dk$.

Let $\Pi = (P_1, \ldots, P_\ell)$ be a partition of P, i.e., $P_i \cap P_j = \emptyset$ for $i \neq j$ and $P = \bigcup_{j \in [\ell]} P_j$. A permutation τ on P is called a Π-*permutation* if $\tau(P_j) = P_j$ for any $j \in [\ell]$. An adversary structure Δ is called Π-*partite* if $\tau(B) \in \Delta$ for any $B \in \Delta$ and any Π-permutation τ. There is a useful geometric representation of Π-partite adversary structures. Let $\Phi^\Pi : 2^P \to \mathbb{R}^\ell$ be a map defined by $\Phi^\Pi(X) = (|X \cap P_j|)_{j \in [\ell]}$. The image of Φ^Π is the set of all integer points in the hyperrectangle determined by $\mathbf{0}$ and $\Phi^\Pi(P)$, that is, $\Phi^\Pi(2^P) = \{\boldsymbol{x} \in \mathbb{Z}^\ell : \mathbf{0} \preceq \boldsymbol{x} \preceq \Phi^\Pi(P)\}$. It follows that a Π-partite adversary structure Δ is uniquely determined by $\Phi^\Pi(\Delta)$. Note that, if $\boldsymbol{a} \in \Phi^\Pi(\Delta)$ and $\boldsymbol{a} \succeq \boldsymbol{b}$, then $\boldsymbol{b} \in \Phi^\Pi(\Delta)$ for any $\boldsymbol{a}, \boldsymbol{b} \in \Phi^\Pi(2^P)$. Thus, any Π-partite adversary structure Δ is uniquely determined only by specifying $\max \Phi^\Pi(\Delta)$, where

$$\max \Phi^\Pi(\Delta) := \{\boldsymbol{a} \in \Phi^\Pi(\Delta) : \boldsymbol{a} \prec \boldsymbol{b} \preceq \Phi^\Pi(P) \Rightarrow \boldsymbol{b} \notin \Phi^\Pi(\Delta)\}.$$

2.2 Secure Polynomial Evaluation Based on d-Multiplicative Secret Sharing

We provide the definition of d-multiplicative secret sharing and an MPC protocol proposed in [1] to securely evaluate a multivariate polynomial.

A *secret sharing scheme* [2] is a tuple $\Sigma = (\mathcal{K}, \mathcal{R}, \mathcal{S}, \varphi)$, where $\mathcal{K}$ is a domain of secrets, $\mathcal{R}$ is a set of random strings, $\mathcal{S}$ is a domain of shares, and $\varphi : \mathcal{K} \times \mathcal{R} \to \mathcal{S}^n$ is a map. For $A \subseteq P$, $\varphi(s, r)_A$ denotes the restriction of $\varphi(s, r)$ to the entries indexed by A. For each $i \in P$, we define $\mathcal{S}_i \subseteq \mathcal{S}$ as $\mathcal{S}_i = \{\varphi(s, r)_{\{i\}} : s \in \mathcal{K}, r \in \mathcal{R}\}$. The (total) *information ratio* $\sigma(\Sigma)$ is defined as $\sigma(\Sigma) = \sum_{i \in P} \log |\mathcal{S}_i| / \log |\mathcal{K}|$.

We say that a secret sharing scheme $\Sigma = (\mathcal{K}, \mathcal{R}, \mathcal{S}, \varphi)$ is *private with respect to* an adversary structure Δ on P or Δ-private for short if, for any $A \in \Delta$, any two secrets $s, t \in \mathcal{K}$, and any possible tuple of shares $(x_i)_{i \in A}$, it holds that

$$\Pr[\varphi(s, r)_A = (x_i)_{i \in A}] = \Pr[\varphi(t, r)_A = (x_i)_{i \in A}],$$

where the probabilities are taken over the random choice of $r \in \mathcal{R}$. In other words, the players in $A \in \Delta$ cannot obtain any information about a secret. Clearly, if Σ is Δ_1-private and $\Delta_1 \supseteq \Delta_2$, then Σ is also Δ_2-private.

Suppose that $\mathcal{K}$ is a finite field $\mathbb{F}$. For $d \geq 2$, a secret sharing scheme Σ is said to be d-*multiplicative* if there exists a map $\mathrm{MULT} : P \times \mathcal{S}^d \to \mathbb{F}$ such that

$$\prod_{j \in [d]} s^{(j)} = \sum_{i \in P} \mathrm{MULT}(i, s_i^{(1)}, \ldots, s_i^{(d)}),$$

for any d secrets $s^{(1)}, \ldots, s^{(d)} \in \mathbb{F}$ and any d random strings $r^{(1)}, \ldots, r^{(d)} \in \mathcal{R}$, where $(s_i^{(j)})_{i \in P} = \varphi(s^{(j)}, r^{(j)})$ is a vector of shares for $s^{(j)}$. Furthermore, we say that a secret sharing scheme Σ is *linear* over $\mathbb{F}$ or $\mathbb{F}$-linear if $\mathcal{S}$ and $\mathcal{R}$ are vector spaces over $\mathcal{K} = \mathbb{F}$ and φ is a linear map over $\mathbb{F}$.

The most important application of d-multiplicative secret sharing is a construction of unconditionally secure MPC protocols to evaluate multivariate polynomials of total degree at most d.

Suppose that m clients C_j have secret inputs $x^{(j)} \in \mathbb{F}$ and want to evaluate a multivariate polynomial $f \in \mathbb{F}[X_1, \ldots, X_m]$ of total degree at most d. Furthermore, suppose that there are n servers indexed by P which help perform the computation. Let $\Sigma = (\mathcal{K} = \mathbb{F}, \mathcal{R}, \mathcal{S}, \varphi)$

be a d-multiplicative secret sharing scheme. Barkol et al. [1] construct an MPC protocol to obtain $f(x^{(1)}, \ldots, x^{(m)})$ by using Σ. For simplicity, we explain the protocol in the case of $m = d$ and $f = X_1 \ldots X_d$. Please refer to [1] for a general case.

- Round 1: Each client C_j generates shares $(s_i^{(j)})_{i \in P} = \varphi(x^{(j)}, r^{(j)})$ corresponding to his input $x^{(j)}$ and sends $s_i^{(j)} \in \mathcal{S}_i \subseteq \mathcal{S}$ to the server $i \in P$. In addition, C_j randomly chooses n field elements $z_i^{(j)}$, $i \in P$ conditioned on $\sum_{i \in P} z_i^{(j)} = 0$ and sends $z_i^{(j)} \in \mathbb{F}$ to the server $i \in P$.
- Round 2: Each server $i \in P$ computes $y_i = \mathrm{MULT}(i, s_i^{(1)}, \ldots, s_i^{(d)}) + \sum_{j \in [d]} z_i^{(j)}$ and sends $y_i \in \mathbb{F}$ to all clients.
- Output: Each client C_j computes $\sum_{i \in P} y_i$, which is equal to $\prod_{j \in [d]} x^{(j)}$.

Since the n servers can locally convert their shares into additive shares of the output, interaction is required only in Round 1 and the latter half of Round 2. The communication complexity of the protocol is

$$\sum_{j \in [m]} \sum_{i \in P} (\log |\mathcal{S}_i| + \log |\mathbb{F}|) + \sum_{i \in P} \sum_{j \in [m]} \log |\mathbb{F}| = m(\sigma(\Sigma) + 2n) \log |\mathbb{F}|.$$

Therefore, it is important to cut down the information ratio and the size of the base field to design a communication-efficient MPC protocol.

Let $\Delta \subseteq 2^P$ be a family of subsets of the n servers. We assume a passive adversary who can corrupt a set of servers A such that $A \in \Delta$. If the underlying secret sharing scheme Σ is Δ-private, then the adversary cannot learn anything about the inputs of the clients other than what follows from the output.

2.3 Examples of d-Multiplicative Secret Sharing Schemes

2.3.1 Shamir Secret Sharing Schemes

The (k, n)-*Shamir secret sharing scheme* [20] is a well-known $\mathcal{T}_k^n$-private linear secret sharing scheme. Let $\mathbb{F}$ be a finite field such that $|\mathbb{F}| > n$ and take n distinct nonzero elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$. Let $s \in \mathbb{F}$ be a secret to be shared. First, choose a random polynomial f over $\mathbb{F}$ of degree at most k conditioned on $f(0) = s$. Second, assign $f(\alpha_i)$ to $i \in P$ as a share. Note that the (k, n)-Shamir secret sharing scheme has the information ratio n, which is known to be optimal [16].

It can be seen that the scheme is d-multplicative if $n > dk$. Indeed, let $s^{(1)}, \ldots, s^{(d)}$ be d secrets and $f^{(1)}, \ldots, f^{(d)}$ be the corresponding d polynomials over $\mathbb{F}$ of degree at most k which are used to share $s^{(1)}, \ldots, s^{(d)}$, respectively. Since the degree of $g := \prod_{j \in [d]} f^{(j)}$ is at most $dk < n$, we can compute the product of the secrets from the shares $f^{(j)}(\alpha_i)$, $i \in \widetilde{P}$ for any subset $\widetilde{P} \subseteq P$ of size dk by Lagrange interpolation:

$$\prod_{j \in [d]} s^{(j)} = g(0) = \sum_{i \in \widetilde{P}} \lambda_i^{\widetilde{P}} g(\alpha_i) = \sum_{i \in \widetilde{P}} \lambda_i^{\widetilde{P}} \prod_{j \in [d]} f^{(j)}(\alpha_i),$$

where $\lambda_i^{\widetilde{P}}$ is a Lagrange coefficient, i.e., $\lambda_i^{\widetilde{P}} = \prod_{m \in \widetilde{P} \setminus \{i\}} \alpha_m / (\alpha_m - \alpha_i)$.

2.3.2 Weighted Threshold Secret Sharing Schemes

Weighted threshold adversary structures proposed in [20] are natural generalizations of the threshold ones to the multipartite setting. Let $\Pi = (P_1, \ldots, P_\ell)$ be a partition of P, $\boldsymbol{w} \in \mathbb{Z}_+^\ell$, and $t \in \mathbb{Z}_+$. Define $\mathcal{W}_{\boldsymbol{w},t}^{\Pi}$ as the Π-partite adversary structure such that

$$\Phi^{\Pi}(\mathcal{W}_{\boldsymbol{w},t}^{\Pi}) = \{\boldsymbol{x} \in \Phi^{\Pi}(2^P) : \boldsymbol{w} \cdot \boldsymbol{x} \leq t\}.$$

Note that, if $\boldsymbol{w} = \mathbf{1} \in \mathbb{Z}_+^\ell$, then $\mathcal{W}_{\mathbf{1},t}^\Pi$ is equal to $\mathcal{T}_t^n$.

It is possible to construct $\mathcal{W}_{\boldsymbol{w},t}^\Pi$-private linear secret sharing schemes by assigning multiple shares of Shamir's scheme to each player [20]. Let $\mathbb{F}$ be a finite field such that $|\mathbb{F}| > N := \boldsymbol{w} \cdot \Phi^\Pi(P)$ and take N distinct nonzero elements $\alpha_{ijk} \in \mathbb{F}$ for $i \in [\ell]$, $j \in P_i$ and $k \in [\boldsymbol{w}(i)]$. Let $s \in \mathbb{F}$ be a secret to be shared. First, choose a random polynomial f over $\mathbb{F}$ of degree at most t with $f(0) = s$. Second, assign $\boldsymbol{w}(i)$ shares $(f(\alpha_{ijk}))_{k \in [\boldsymbol{w}(i)]}$ to the player $j \in P$, where i is the unique index such that $j \in P_i$. Note that the information ratio of the scheme is $N = \boldsymbol{w} \cdot \Phi^\Pi(P)$. It can be shown that the scheme is d-multiplicative if $N > dt$ in the same manner as Shamir's scheme. In summary, the following proposition holds.

▶ **Proposition 1** ([20]). *Let $\Pi = (P_1, \dots, P_\ell)$ be a partition of P and $\mathcal{W}_{\boldsymbol{w},t}^\Pi$ be the Π-partite weighted threshold adversary structure with weight $\boldsymbol{w} \in \mathbb{Z}_+^\ell$ and threshold $t \in \mathbb{Z}_+$. Let $\mathbb{F}$ be a finite field such that $|\mathbb{F}| > \boldsymbol{w} \cdot \Phi^\Pi(P)$. Then there exists a $\mathcal{W}_{\boldsymbol{w},t}^\Pi$-private $\mathbb{F}$-linear secret sharing scheme Σ with information ratio $\sigma(\Sigma) = \boldsymbol{w} \cdot \Phi^\Pi(P)$. Furthermore, if $\boldsymbol{w} \cdot \Phi^\Pi(P) > dt$, then Σ is d-multiplicative.*

2.3.3 CNF Secret Sharing Schemes

For any adversary structure Δ of type Q_d, it is possible to construct a Δ-private d-multiplicative secret sharing scheme [15]. Let Δ^+ be the set of all maximal subsets in Δ. Let s be a secret to be shared. First, choose $|\Delta^+|$ random field elements r_A, $A \in \Delta^+$ such that $s = \sum_{A \in \Delta^+} r_A$. Second, assign $(r_A)_{A \in \Delta_i^+}$ to each $i \in P$, where $\Delta_i^+ = \{A \in \Delta^+ : i \notin A\}$. The information ratio of the scheme is $\sum_{i \in P} |\Delta_i^+|$. It can be seen that the scheme is d-multiplicative as long as Δ is of type Q_d.

3 d-Multiplicative Secret Sharing for Any Multipartite Q_d-Adversary Structure

In this section, we propose an explicit construction of a Δ-private d-multiplicative secret sharing scheme for any ℓ-partite adversary structure Δ of type Q_d. Our scheme achieves an information ratio $n|\max \Phi^\Pi(\Delta)| = O(n^{\ell+1})$. First, we restate the definition of Q_d-adversary structures in the multipartite setting.

▶ **Proposition 2.** *Let $\Pi = (P_1, \dots, P_\ell)$ be a partition of P and Δ be a Π-partite adversary structure on P. Then Δ is of type Q_d if and only if $\boldsymbol{x}_1 + \cdots + \boldsymbol{x}_d \not\succeq \Phi^\Pi(P)$ for any (not necessarily distinct) d points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_d \in \Phi^\Pi(\Delta)$.*

Proof. First, assume that P is covered by d subsets $B_1, \dots, B_d \in \Delta$. Since Δ is monotone decreasing, we may assume that the B_i's are pairwise distinct. Then it holds that $\Phi^\Pi(B_i) \in \Phi^\Pi(\Delta)$ and $\Phi^\Pi(B_1) + \cdots + \Phi^\Pi(B_d) = \Phi^\Pi(P)$.

Second, assume that there exists d points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_d \in \Phi^\Pi(\Delta)$ such that $\boldsymbol{x}_1 + \cdots + \boldsymbol{x}_d \succeq \Phi^\Pi(P)$. Since $\boldsymbol{c}' \in \Phi^\Pi(\Delta)$ if $\boldsymbol{c} \in \Phi^\Pi(\Delta)$ and $\mathbf{0} \preceq \boldsymbol{c}' \preceq \boldsymbol{c}$, we can replace the $\boldsymbol{x}_i$'s with d points $\boldsymbol{b}_1, \dots, \boldsymbol{b}_d \in \Phi^\Pi(\Delta)$ such that $\boldsymbol{b}_i \preceq \boldsymbol{x}_i$ and $\boldsymbol{b}_1 + \cdots + \boldsymbol{b}_d = \Phi^\Pi(P)$. Then there exist pairwise disjoint d subsets $B_1, \dots, B_d$ in Δ such that $\boldsymbol{x}_i = \Phi^\Pi(B_i)$. We have that $B_1 \cup \cdots \cup B_d = P$. ◀

Next, we explain the well-known decomposition technique [21], which is fundamental to our construction of d-multiplicative schemes. Roughly speaking, if an adversary structure Δ is decomposed into several adversary structures Δ_i, then a Δ-private secret sharing scheme can be obtained from secret sharing schemes each of which is Δ_i-private. Specifically, let

$\Delta_1, \ldots, \Delta_N$ be N adversary structures on P and set $\Delta^\forall = \bigcap_{i\in[N]} \Delta_i$ and $\Delta^\exists = \bigcup_{i\in[N]} \Delta_i$. Suppose that, for each $i \in [N]$, we are given a Δ_i-private secret sharing scheme $\Sigma_i = (\mathcal{K}_i, \mathcal{R}_i, \mathcal{S}_i, \varphi_i)$. We assume that all domains of secrets $\mathcal{K}_i$ are identical to a finite field $\mathbb{F}$.

In the following, we construct two secret sharing schemes $\Sigma^\forall$ and $\Sigma^\exists$ with domain of secrets $\mathbb{F}$ which are private with respect to $\Delta^\forall$ and $\Delta^\exists$, respectively. Set $\mathcal{R} = \mathcal{R}_1 \times \cdots \times \mathcal{R}_N$ and $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_N$. Let $s \in \mathbb{F}$ be a secret to be shared. First, $\Sigma^\forall$ randomly chooses $r = (r_1, \ldots, r_N)$ from $\mathcal{R}$ and sets a vector of shares as $\varphi(s, r) = (\varphi_1(s, r_1), \ldots, \varphi_N(s, r_N)) \in \mathcal{S}^n$. Equivalently, $\Sigma^\forall$ runs N secret sharing schemes Σ_i in parallel with secret s. Second, $\Sigma^\exists$ randomly chooses $r = (r_1, \ldots, r_N)$ from $\mathcal{R}$ and $N-1$ elements $s_1, \ldots, s_{N-1}$ from $\mathbb{F}$. Then it sets $s_N = s - \sum_{i\in[N-1]} s_i$ and a vector of shares as $\varphi(s, (r, s_1, \ldots, s_{N-1})) = (\varphi_1(s_1, r_1), \ldots, \varphi_N(s_N, r_N)) \in \mathcal{S}^n$. In other words, $\Sigma^\exists$ randomly splits s into $s_1, \ldots, s_N$ and runs Σ_i in parallel with secret s_i for each $i \in [N]$. Then the following proposition holds.

▶ **Proposition 3** ([21]). *Let $\mathbb{F}$ be a finite field. Let $\Delta_1, \ldots, \Delta_N$ be N adversary structures on P and set $\Delta^\forall = \bigcap_{i\in[N]} \Delta_i$ and $\Delta^\exists = \bigcup_{i\in[N]} \Delta_i$. Suppose that, for each $i \in [N]$, a Δ_i-private secret sharing scheme $\Sigma_i = (\mathbb{F}, \mathcal{R}_i, \mathcal{S}_i, \varphi_i)$ is given. Then there exist a $\Delta^\forall$-private secret sharing scheme $\Sigma^\forall$ and a $\Delta^\exists$-private secret sharing scheme $\Sigma^\exists$ both of which have information ratios at most $\sum_{i\in[N]} \sigma(\Sigma_i)$.*

Now, we are ready to provide our construction of d-multiplicative schemes. Let $\Pi = (P_1, \ldots, P_\ell)$ be a partition of P and Δ be a Π-partite adversary structure. Write $\max \Phi^\Pi(\Delta) = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N\}$. We decompose Δ into N adversary structures $\Delta_1, \ldots, \Delta_N$ as $\Delta = \bigcup_{j\in[N]} \Delta_j$, where Δ_j is the Π-partite adversary structure such that $\Phi^\Pi(\Delta_j) = \{\boldsymbol{x} \in \Phi^\Pi(2^P) : \boldsymbol{x} \preceq \boldsymbol{a}_j\}$. Furthermore, we decompose each Δ_j into ℓ adversary structures $\Delta_{j1}, \ldots, \Delta_{j\ell}$ as $\Delta_j = \bigcap_{k\in[\ell]} \Delta_{jk}$, where Δ_{jk} is the Π-partite adversary structure such that $\Phi^\Pi(\Delta_{jk}) = \{\boldsymbol{x} \in \Phi^\Pi(2^P) : \boldsymbol{x}(k) \le \boldsymbol{a}_j(k)\}$.

For each Δ_{jk}, we can construct a Δ_{jk}-private scheme Σ_{jk} based on the $(\boldsymbol{a}_j(k), |P_k|)$-Shamir secret sharing scheme. Indeed, let $\mathbb{F}$ be a finite field with $|\mathbb{F}| > n$ and fix n distinct nonzero elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$. For a secret $s \in \mathbb{F}$, choose a random polynomial f_{jk} of degree at most $\boldsymbol{a}_j(k)$ conditioned on $f_{jk}(0) = s$ and assign $f_{jk}(\alpha_i) \in \mathbb{F}$ to each player $i \in P_k$.

In view of Proposition 3, we obtain a Δ_j-private secret sharing scheme Σ_j from $\Sigma_{j1}, \ldots, \Sigma_{j\ell}$. Since Σ_{jk} does not assign any share to players in $P_{k'}$ for $k' \neq k$, the information ratio of Σ_j is n. Again, from Proposition 3, we obtain a Δ-private secret sharing scheme Σ with information ratio nN from $\Sigma_1, \ldots, \Sigma_N$.

We show that the scheme Σ constructed in this way is d-multiplicative if Δ is of type Q_d.

▶ **Theorem 4.** *Let $\mathbb{F}$ be a finite field with $|\mathbb{F}| > n$. Let $\Pi = (P_1, \ldots, P_\ell)$ be a partition of P and Δ be a Π-partite Q_d-adversary structure on P. Write $\max \Phi^\Pi(\Delta) = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N\}$. Then there exists a Δ-private d-multiplicative $\mathbb{F}$-linear secret sharing scheme Σ such that $\sigma(\Sigma) = nN$.*

Proof. First, we define some notations. For each $i \in P$, let $\ell_i \in [\ell]$ be the unique index such that $i \in P_{\ell_i}$. Since Δ is of type Q_d, it holds that $\boldsymbol{a}_{j_1} + \cdots + \boldsymbol{a}_{j_d} \not\succeq \Phi^\Pi(P)$ for any $\boldsymbol{j} = (j_1, \ldots, j_d) \in [N]^d$. In particular, there exists an index k such that $\boldsymbol{a}_{j_1}(k) + \cdots + \boldsymbol{a}_{j_d}(k) < |P_k|$. Thus, we can define a map $\psi : [N]^d \to [\ell]$ such that $\boldsymbol{a}_{j_1}(\psi(\boldsymbol{j})) + \cdots + \boldsymbol{a}_{j_d}(\psi(\boldsymbol{j})) < |P_{\psi(\boldsymbol{j})}|$ for any $\boldsymbol{j} = (j_1, \ldots, j_d) \in [N]^d$. For each $\boldsymbol{j} = (j_1, \ldots, j_d) \in [N]^d$, we fix a subset $\widetilde{P}_{\boldsymbol{j}} \subseteq P_{\psi(\boldsymbol{j})}$ of size $\boldsymbol{a}_{j_1}(\psi(\boldsymbol{j})) + \cdots + \boldsymbol{a}_{j_d}(\psi(\boldsymbol{j})) + 1$. Note that $\ell_i = \psi(\boldsymbol{j})$ if $i \in \widetilde{P}_{\boldsymbol{j}}$. Furthermore, we define $J_i := \{\boldsymbol{j} \in [N]^d : i \in \widetilde{P}_{\boldsymbol{j}}\}$ for $i \in P$.

Let Σ be the above secret sharing scheme. From the construction of Σ, any share assigned to $i \in P$ for a secret s has the form of $(f_{1\ell_i}(\alpha_i), \ldots, f_{N\ell_i}(\alpha_i))$, where each $f_{j\ell_i}$ is a polynomial of degree at most $\boldsymbol{a}_j(\ell_i)$ and $s = f_{1\ell_i}(0) + \cdots + f_{N\ell_i}(0)$. For any $k \neq k' \in [\ell]$, the schemes Σ_{jk} and $\Sigma_{jk'}$ have the same secret as inputs and hence it holds that $f_{jk}(0) = f_{jk'}(0)$.

Let $s^{(1)}, \dots, s^{(d)}$ be any d secrets. For $m \in [d]$, let $(f^{(m)}_{1\ell_i}(\alpha_i), \dots, f^{(m)}_{N\ell_i}(\alpha_i))$ be a share assigned to $i \in P$ for the secret $s^{(m)}$. Since $f^{(m)}_{jk}(0)$ have the same value for all $k \in [\ell]$, we denote the common value by $s^{(m)}_j$. Then it holds that $s^{(m)} = s^{(m)}_1 + \cdots + s^{(m)}_N$.

Note that, for any $\boldsymbol{j} = (j_1, \dots, j_d) \in [N]^d$ and $k = \psi(\boldsymbol{j})$, the product $\prod_{m\in[d]} f^{(m)}_{j_m k}$ is a polynomial of degree at most $\sum_{m\in[d]} \boldsymbol{a}_{j_m}(k)$ and $\widetilde{P}_{\boldsymbol{j}}$ is a subset of size $\sum_{m\in[d]} \boldsymbol{a}_{j_m}(k) + 1$. Thus, by Lagrange interpolation, for any $\boldsymbol{j} = (j_1, \dots, j_d) \in [N]^d$, it holds that

$$s^{(1)}_{j_1} \cdots s^{(d)}_{j_d} = f^{(1)}_{j_1,\psi(\boldsymbol{j})}(0) \cdots f^{(d)}_{j_d,\psi(\boldsymbol{j})}(0) = \sum_{i\in\widetilde{P}_{\boldsymbol{j}}} \lambda_i^{\widetilde{P}_{\boldsymbol{j}}} f^{(1)}_{j_1\ell_i}(\alpha_i) \cdots f^{(d)}_{j_d\ell_i}(\alpha_i),$$

where $\lambda_i^{\widetilde{P}_{\boldsymbol{j}}}$ is a lagrange coefficient, i.e., $\lambda_i^{\widetilde{P}_{\boldsymbol{j}}} = \prod_{k\in\widetilde{P}_{\boldsymbol{j}}\setminus\{i\}} \alpha_k/(\alpha_k - \alpha_i)$.

Now we have

$$\begin{aligned} s^{(1)} \cdots s^{(d)} &= \sum_{\boldsymbol{j}=(j_1,\dots,j_d)\in[N]^d} s^{(1)}_{j_1} \cdots s^{(d)}_{j_d} \\ &= \sum_{\boldsymbol{j}\in[N]^d} \sum_{i\in\widetilde{P}_{\boldsymbol{j}}} \lambda_i^{\widetilde{P}_{\boldsymbol{j}}} f^{(1)}_{j_1\ell_i}(\alpha_i) \cdots f^{(d)}_{j_d\ell_i}(\alpha_i) \\ &= \sum_{i\in P} \sum_{\boldsymbol{j}\in J_i} \lambda_i^{\widetilde{P}_{\boldsymbol{j}}} f^{(1)}_{j_1\ell_i}(\alpha_i) \cdots f^{(d)}_{j_d\ell_i}(\alpha_i). \end{aligned}$$

For $i \in P$ with $J_i = \emptyset$, the corresponding sum is assumed to be 0. Therefore, the d-multiplicativity follows by defining $\mathrm{MULT} : P \times \mathcal{S}^d \to \mathbb{F}$ as

$$\mathrm{MULT}(i, (\gamma^{(1)}_1, \dots, \gamma^{(1)}_N), \dots, (\gamma^{(d)}_1, \dots, \gamma^{(d)}_N)) = \sum_{\boldsymbol{j}=(j_1,\dots,j_d)\in J_i} \lambda_i^{\widetilde{P}_{\boldsymbol{j}}} \gamma^{(1)}_{j_1} \cdots \gamma^{(d)}_{j_d}. \qquad \blacktriangleleft$$

Since $N = |\max \Phi^{\Pi}(\Delta)|$ is clearly at most $|\Phi^{\Pi}(2^P)| = O(n^\ell)$, we can obtain a Δ-private d-multiplicative secret sharing schemes with information ratio $O(n^{\ell+1})$ for any ℓ-partite Q_d-adversary structure Δ. Although the CNF scheme for Δ is also d-multiplicative, the information ratio $\sum_{i\in P} |\Delta_i^+|$ is exponential in n. Indeed, the set of all maximal subsets Δ^+ contains at least all subsets A such that $\Phi^{\Pi}(A) = \boldsymbol{a}_1$. Therefore, we have

$$|\Delta_i^+| \geq \binom{|P_{\ell_i}| - 1}{\boldsymbol{a}_1(\ell_i)} \prod_{k\in[\ell]\setminus\{\ell_i\}} \binom{|P_k|}{\boldsymbol{a}_1(k)}.$$

▶ **Example 5.** We apply Theorem 4 to a family of bipartite adversary structures. Suppose that the number of servers an adversary can corrupt is at most k. Equivalently, suppose that the adversary structure Δ corresponding to the adversary satisfies $\Delta \subseteq \mathcal{T}^n_k$. If $n > dk$, Shamir's scheme can tolerate the maximal adversary $\Delta = \mathcal{T}^n_k$. In the case of $d(k-1) < n \leq dk$, it is no longer possible to make d-multiplicative schemes secure against an adversary who can corrupt *any* k servers since $\mathcal{T}^n_k$ is not of type Q_d. On the other hand, d-multiplicative secret sharing can still tolerate any $k-1$ corrupted servers. It is natural to ask how many subsets of size k we can add to Δ under the condition that $\Delta \supseteq \mathcal{T}^n_{k-1}$. Then, for $n = dk - r$ with $0 \leq r < d-1$,[1] we define the following $(S, P\setminus S)$-partite adversary structure $\mathcal{B}^n_k(S)$ for a subset $S \subseteq P$ of size $(d-r)k - 1$:

$$\mathcal{B}^n_k(S) = \mathcal{T}^n_{k-1} \cup \{A \subseteq P : |A| = k \text{ and } A \subseteq S\}.$$

[1] When $r = d-1$ and $n = d(k-1)+1$, $\mathcal{T}^n_{k-1} \cup \{B\}$ is not of type Q_d for any subset B of size k. In other words, $\mathcal{T}^n_{k-1}$ is the only Q_d-adversary structure Δ such that $\Delta \supseteq \mathcal{T}^n_{k-1}$.

This adversary structure corresponds to the situation in which the adversary can corrupt any $k-1$ servers in P and any k servers in S. It can be shown that $\mathcal{B}_k^n(S)$ is actually of type Q_d and is maximal in the sense that $\mathcal{B}_k^n(S) \cup \{B\}$ is not of type Q_d for any subset $B \notin \mathcal{B}_k^n(S)$ of size k (see Appendix A).

It can be seen that $\max \Phi^{\Pi}(\mathcal{B}_k^n(S)) = \{(k,0)\} \cup \{(x, k-1-x) : x = 0, 1, \ldots, k-2\}$ if $0 < r < d-1$ and $\max \Phi^{\Pi}(\mathcal{B}_k^n(S)) = \{(k,0), (k-2,1)\}$ if $r = 0$. From Theorem 4, we obtain a $\mathcal{B}_k^n(S)$-private d-multiplicative secret sharing scheme whose information ratio is kn if $d(k-1)+1 < n < dk$ and $2n$ if $n = dk$. The scheme can be defined over any finite field $\mathbb{F}$ with $|\mathbb{F}| > n$.

4 d-Multiplicative Secret Sharing for ℓ-Partite Adversary Structures with Good Geometric Properties

Let Δ be a Π-partite adversary structure. We show that there exists a more efficient Δ-private d-multiplicative scheme than the above general construction if the associated set of integer points $\Phi^{\Pi}(\Delta)$ has some good geometric property.

Let $C = \mathrm{Conv}(\Phi^{\Pi}(\Delta))$ be the *convex hull* of $\Phi^{\Pi}(\Delta)$ in $\mathbb{R}^{\ell}$. The convex hull of a finite set $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ is defined by

$$\mathrm{Conv}(S) = \left\{ \sum_{j=1}^{N} \alpha_j \boldsymbol{x}_j : \forall j \in [N],\ \alpha_j \geq 0 \text{ and } \sum_{j=1}^{N} \alpha_j = 1 \right\}.$$

Set $\boldsymbol{p} := (1/d)\Phi^{\Pi}(P)$. Assume that $\boldsymbol{p} \notin C$. The hyperplane separation theorem [13, Theorem 4.4] states that there exists a vector $\boldsymbol{h}$ such that $\boldsymbol{h} \cdot (\boldsymbol{c} - \boldsymbol{p}) < 0$ for any $\boldsymbol{c} \in C$. Then there do not exist $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_d \in \Phi^{\Pi}(\Delta)$ such that $\boldsymbol{x}_1 + \cdots + \boldsymbol{x}_d = \Phi^{\Pi}(P)$ since, otherwise, it would hold $\sum_{i \in [d]} \boldsymbol{h} \cdot (\boldsymbol{x}_i - \boldsymbol{p}) = 0$. In short, an adversary structure Δ is of type Q_d if $\boldsymbol{p} \notin \mathrm{Conv}(\Phi^{\Pi}(\Delta))$. Furthermore, since C is closed, there exists some $\epsilon > 0$ such that $\|\boldsymbol{p} - \boldsymbol{c}\| \geq \epsilon$ for any $\boldsymbol{c} \in C$.

For an adversary structure Δ with $\mathrm{dist}(\boldsymbol{p}, C) \geq \epsilon > 0$, we construct a Δ-private d-multiplicative secret sharing scheme whose information ratio is at most $O(\ell n^2/\epsilon)$ based on weighted threshold secret sharing. For example, if ϵ is a constant independent of n, the information ratio is much smaller than those of the schemes from the above general construction, which is $O(n^{\ell+1})$.

To begin with, we show that the convex hull C is also monotone decreasing. Specifically, for any $\boldsymbol{c} \in C$, the hyperrectangle determined by $\boldsymbol{0}$ and $\boldsymbol{c}$ is included in C.

▶ **Lemma 6.** *Let $\Pi = (P_1, \ldots, P_{\ell})$ be a partition of P and Δ be a Π-partite adversary structure. Let $C = \mathrm{Conv}(\Phi^{\Pi}(\Delta))$ be the convex hull of $\Phi^{\Pi}(\Delta)$ in $\mathbb{R}^{\ell}$. If $\boldsymbol{c} \in C$ and $\boldsymbol{0} \preceq \boldsymbol{c}' \preceq \boldsymbol{c}$, then $\boldsymbol{c}' \in C$.*

Proof. Let $\boldsymbol{c} \in C$ and represent it as a convex combination of some points of $\Phi^{\Pi}(\Delta)$: $\boldsymbol{c} = \sum_i \alpha_i \boldsymbol{x}_i$, where $\sum_i \alpha_i = 1$ and $\alpha_i \geq 0$. Then, for any $A \subseteq [\ell]$, $\boldsymbol{c}(A) = \sum_i \alpha_i \boldsymbol{x}_i(A)$ and hence $\boldsymbol{c}(A) \in C$. Here, for a vector $\boldsymbol{v} \in \mathbb{R}^{\ell}$ and $A \subseteq [\ell]$, $\boldsymbol{v}(A)$ denotes the vector whose entries indexed by A are the same as those of $\boldsymbol{v}$ and 0 otherwise. If we set $X := \{\boldsymbol{c}(A) : A \in 2^{[\ell]}\}$, it holds that $\mathrm{Conv}(X) \subseteq C$ since C is convex.

We finish the proof by showing that $\mathrm{Conv}(X) = \{\boldsymbol{x} \in \mathbb{R}^{\ell} : \boldsymbol{0} \preceq \boldsymbol{x} \preceq \boldsymbol{c}\}$. Let $\boldsymbol{x} \in \mathbb{R}^{\ell}$ with $\boldsymbol{0} \preceq \boldsymbol{x} \preceq \boldsymbol{c}$. Set $Z = \{i \in [\ell] : \boldsymbol{c}(i) \neq 0\}$ and write $\{\boldsymbol{x}(i)/\boldsymbol{c}(i) : i \in Z\} = \{v_1, \ldots, v_m\}$, where $v_0 := 0 \leq v_1 < \cdots < v_m \leq 1$. Furthermore, for $j \in [m]$, set $I_j = \{i \in Z : \boldsymbol{x}(i)/\boldsymbol{c}(i) = v_j\}$ and $A_j = I_j \cup I_{j+1} \cup \cdots \cup I_m$. Let $\boldsymbol{y}$ be a vector such that $\boldsymbol{y}(i) = \boldsymbol{x}(i)/\boldsymbol{c}(i)$ for $i \in Z$ and otherwise $\boldsymbol{y}(i) = 0$. Then we have $\boldsymbol{y} = \sum_{j=1}^{m} (v_j - v_{j-1}) \boldsymbol{1}(A_j)$, from which we obtain $\boldsymbol{x} = \sum_{j=1}^{m} (v_j - v_{j-1}) \boldsymbol{c}(A_j) + (1 - v_m)\boldsymbol{c}(\emptyset)$. Thus, $\boldsymbol{x} \in \mathrm{Conv}(X)$. The other inclusion clearly holds. ◀

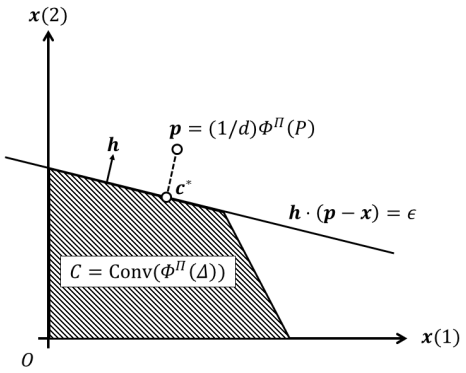


Figure 1 The convex hull of $\Phi^{\Pi}(\Delta)$ in $\mathbb{R}^{\ell}$.

Next, we show that if we set $\boldsymbol{h}$ as the unit vector parallel to $\boldsymbol{p}-\boldsymbol{c}^*$ for the closest point $\boldsymbol{c}^* \in C$ to $\boldsymbol{p}$, then $\boldsymbol{h}\cdot(\boldsymbol{p}-\boldsymbol{x}) \geq \mathrm{dist}(\boldsymbol{p}, C)$ for any $\boldsymbol{x} \in \Phi^{\Pi}(\Delta)$. Although the existence of $\boldsymbol{h}$ easily follows from the hyperplane separation theorem, we additionally show that $\boldsymbol{h} \in \mathbb{R}^{\ell}_{+}$ using the fact that C is monotone decreasing. The vector $\boldsymbol{h}$ is used to find a weight vector $\boldsymbol{w}$ such that $\Delta \subseteq \mathcal{W}^{\Pi}_{\boldsymbol{w},t}$.

▶ **Lemma 7.** *Let $\Pi = (P_1, \ldots, P_\ell)$ be a partition of P and Δ be a Π-partite adversary structure. Set $\boldsymbol{p} = (1/d)\Phi^{\Pi}(P)$. Let $C = \mathrm{Conv}(\Phi^{\Pi}(\Delta))$ be the convex hull of $\Phi^{\Pi}(\Delta)$ in $\mathbb{R}^{\ell}$. Suppose that $\mathrm{dist}(\boldsymbol{p}, C) \geq \epsilon > 0$, i.e., $\|\boldsymbol{c}-\boldsymbol{p}\| \geq \epsilon$ for any $\boldsymbol{c} \in C$. Then there exists a vector $\boldsymbol{h} \in \mathbb{R}^{\ell}_{+}$ with $\|\boldsymbol{h}\| = 1$ such that $\boldsymbol{h}\cdot(\boldsymbol{p}-\boldsymbol{x}) \geq \epsilon$ for any $\boldsymbol{x} \in \Phi^{\Pi}(\Delta)$.*

Proof. Let $\boldsymbol{c}^* = \mathrm{argmin}_{\boldsymbol{c}\in C}\|\boldsymbol{p}-\boldsymbol{c}\|$ and set $\boldsymbol{h}_0 = \boldsymbol{p}-\boldsymbol{c}^*$. Note that $\|\boldsymbol{h}_0\| \geq \epsilon$.

Then $\boldsymbol{h}_0\cdot(\boldsymbol{p}-\boldsymbol{c}) \geq \|\boldsymbol{h}_0\|^2$ for any $\boldsymbol{c} \in C$. Indeed, let $\boldsymbol{c}$ be any point in C. For λ with $0 < \lambda < 1$, we define a point $\boldsymbol{c}_\lambda$ as $\boldsymbol{c}_\lambda = \lambda\boldsymbol{c} + (1-\lambda)\boldsymbol{c}^*$. It follows from the definition of $\boldsymbol{c}^*$ that $\|\boldsymbol{c}_\lambda - \boldsymbol{p}\|^2 \geq \|\boldsymbol{c}^* - \boldsymbol{p}\|^2$. This implies that $0 \geq -\lambda\|\boldsymbol{c}-\boldsymbol{c}^*\|^2 + 2(\boldsymbol{c}^*-\boldsymbol{p})\cdot(\boldsymbol{c}^*-\boldsymbol{c})$. By making λ approach to 0, we obtain $\boldsymbol{h}_0\cdot(\boldsymbol{c}^*-\boldsymbol{c}) \geq 0$, which implies that

$$\boldsymbol{h}_0\cdot(\boldsymbol{p}-\boldsymbol{c}) = \boldsymbol{h}_0\cdot(\boldsymbol{p}-\boldsymbol{c}^*) + \boldsymbol{h}_0\cdot(\boldsymbol{c}^*-\boldsymbol{c}) \geq \|\boldsymbol{h}_0\|^2.$$

We show that $\boldsymbol{h}_0 \in \mathbb{R}^{\ell}_{+}$. Assume that $\boldsymbol{c}^* \not\preceq \boldsymbol{p}$. Then there is an index $j \in [\ell]$ with $\boldsymbol{c}^*(j) > \boldsymbol{p}(j)$. Set $\boldsymbol{c}' = \boldsymbol{c}^* - (\boldsymbol{c}^*(j) - \boldsymbol{p}(j))\boldsymbol{e}_j$. Since $\boldsymbol{0} \preceq \boldsymbol{c}' \preceq \boldsymbol{c}^*$, $\boldsymbol{c}'$ is in C. However, it holds that

$$\|\boldsymbol{c}'-\boldsymbol{p}\|^2 - \|\boldsymbol{c}^*-\boldsymbol{p}\|^2 = -(\boldsymbol{c}^*(j)-\boldsymbol{p}(j))^2 < 0,$$

which contradicts the definition of $\boldsymbol{c}^*$.

Set $\boldsymbol{h} = \boldsymbol{h}_0/\|\boldsymbol{h}_0\| \in \mathbb{R}^{\ell}_{+}$. Then $\|\boldsymbol{h}\| = 1$ and $\boldsymbol{h}\cdot(\boldsymbol{p}-\boldsymbol{x}) \geq \|\boldsymbol{h}_0\| \geq \epsilon$ for any $\boldsymbol{x} \in \Phi^{\Pi}(\Delta) \subseteq C$. ◀

To obtain a weight vector $\boldsymbol{w}$, we approximate $\boldsymbol{h}$ by a vector of rational numbers $\boldsymbol{h}+\boldsymbol{\delta}$ for a small vector $\boldsymbol{\delta}$ and set $\boldsymbol{w} = q(\boldsymbol{h}+\boldsymbol{\delta})$ for some integer q. Since $\boldsymbol{h}\cdot(\boldsymbol{p}-\boldsymbol{x}) \geq \epsilon$ for a finite number of vectors $\boldsymbol{x} \in \Phi^{\Pi}(\Delta)$, we can choose q to be $q = O(\ell n/\epsilon)$.

▶ **Lemma 8.** *In the setting of Lemma 7, let $\boldsymbol{h}$ be a vector of $\mathbb{R}^{\ell}_{+}$ with $\|\boldsymbol{h}\| = 1$ such that $\boldsymbol{h}\cdot(\boldsymbol{p}-\boldsymbol{x}) \geq \epsilon$ for any $\boldsymbol{x} \in \Phi^{\Pi}(\Delta)$. Then there exists a vector $\boldsymbol{w} \in \mathbb{Z}^{\ell}_{+}$ such that $\boldsymbol{w}\cdot(\boldsymbol{x}-\boldsymbol{p}) < 0$ for any $\boldsymbol{x} \in \Phi^{\Pi}(\Delta)$ and $0 \leq \boldsymbol{w}(j) \leq (\ell n/\epsilon) + 1$ for any $j \in [\ell]$.*

Proof. Write $\Phi^{\Pi}(\Delta) = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. Define a continuous function $f_j : \mathbb{R}^\ell \to \mathbb{R}$ as $f_j(\boldsymbol{w}) = \boldsymbol{w} \cdot (\boldsymbol{x}_j - \boldsymbol{p})$ for $j \in [N]$. Observe that $f_j(\boldsymbol{h}) \leq -\epsilon$ and that for any $\boldsymbol{\delta} \in \mathbb{R}^\ell$,

$$|f_j(\boldsymbol{h} + \boldsymbol{\delta}) - f_j(\boldsymbol{h})| \leq \|\boldsymbol{x}_j - \boldsymbol{p}\| \cdot \|\boldsymbol{\delta}\| \leq \sqrt{\ell} n \|\boldsymbol{\delta}\|.$$

Thus, $f_j(\boldsymbol{h} + \boldsymbol{\delta}) < 0$ for any $\boldsymbol{\delta} \in \mathbb{R}^\ell$ with $\|\boldsymbol{\delta}\| < \epsilon/(\sqrt{\ell} n)$.

Let q be the smallest positive integer satisfying $q > \ell n/\epsilon$. Set $p_j = \lceil q\boldsymbol{h}(j) \rceil$ for each $j \in [\ell]$. Since $\|\boldsymbol{h}\| = 1$, we have $0 \leq q\boldsymbol{h}(j) \leq q$ and hence $0 \leq p_j \leq q$.

Let $\boldsymbol{\delta} \in \mathbb{R}^\ell$ be a vector such that

$$0 \leq \boldsymbol{\delta}(j) = \frac{p_j}{q} - \boldsymbol{h}(j) \leq \frac{1}{q}.$$

Then $\|\boldsymbol{\delta}\| \leq \|q^{-1}\mathbf{1}\| < \epsilon/(\sqrt{\ell} n)$. Set $\boldsymbol{w} = q(\boldsymbol{h} + \boldsymbol{\delta}) = (p_1, \ldots, p_\ell) \in \mathbb{Z}_+^\ell$. It holds that $f_j(\boldsymbol{w}) = q f_j(\boldsymbol{h} + \boldsymbol{\delta}) < 0$ for any $j \in [N]$ and $0 \leq \boldsymbol{w}(j) \leq q \leq (\ell n/\epsilon) + 1$. ◀

Now, we construct a d-multiplicative scheme using the weight vector $\boldsymbol{w}$ in Lemma 8.

▶ **Theorem 9.** *Let $\Pi = (P_1, \ldots, P_\ell)$ be a partition of P and Δ be a Π-partite adversary structure. Let C be the convex hull of $\Phi^{\Pi}(\Delta)$ in $\mathbb{R}^\ell$. Set $\boldsymbol{p} = (1/d)\Phi^{\Pi}(P)$. Suppose that $\mathrm{dist}(\boldsymbol{p}, C) \geq \epsilon > 0$. If $\mathbb{F}$ is a finite field with $|\mathbb{F}| > (\ell n^2/\epsilon) + n$, then there exists a Δ-private d-multiplicative $\mathbb{F}$-linear secret sharing scheme whose information ratio is at most $(\ell n^2/\epsilon) + n$.*

Proof. From Lemma 8, we have $\boldsymbol{w} \in \mathbb{Z}_+^\ell$ such that $\boldsymbol{w} \cdot (\boldsymbol{x} - \boldsymbol{p}) < 0$ for any $\boldsymbol{x} \in \Phi^{\Pi}(\Delta)$ and $0 \leq \boldsymbol{w}(j) \leq (\ell n/\epsilon) + 1$ for any $j \in [\ell]$. Set $t := \max_{\boldsymbol{x} \in \Phi^{\Pi}(\Delta)}\{\boldsymbol{w} \cdot \boldsymbol{x}\}$. Clearly, $\boldsymbol{w} \cdot \boldsymbol{x} \leq t$ for any $\boldsymbol{x} \in \Phi^{\Pi}(\Delta)$. Furthermore, $dt < \boldsymbol{w} \cdot \Phi^{\Pi}(P)$ since $\boldsymbol{w} \cdot \boldsymbol{x} < \boldsymbol{w} \cdot \boldsymbol{p}$ for any $\boldsymbol{x} \in \Phi^{\Pi}(\Delta)$. Let $\mathbb{F}$ be a finite field with $|\mathbb{F}| > \boldsymbol{w} \cdot \Phi^{\Pi}(P)$. Then, applying Proposition 1 to the weighted threshold adversary structure $\mathcal{W}^{\Pi}_{\boldsymbol{w},t}$, we obtain Δ-private d-multiplicative secret sharing scheme over $\mathbb{F}$ whose information ratio is $\boldsymbol{w} \cdot \Phi^{\Pi}(P) \leq (\ell n^2/\epsilon) + n$. ◀

The construction of Theorem 9 is a natural generalization of Shamir's threshold schemes. Indeed, the convex hull $C = \mathrm{Conv}(\Phi^{\Pi}(\Delta))$ for $\Delta = \mathcal{T}_k^n$ is $\{\boldsymbol{x} \in \mathbb{R}^\ell : \mathbf{1} \cdot \boldsymbol{x} \leq k\}$. Thus, the condition $\boldsymbol{p} \notin C$ holds if and only if $n > dk$. Then we can set $\boldsymbol{w} \in \mathbb{Z}_+^\ell$ in the proof of Theorem 9 as $\mathbf{1} \in \mathbb{Z}_+^\ell$ and $t = \max_{\boldsymbol{x} \in \Phi^{\Pi}(\mathcal{T}_k^n)}\{\boldsymbol{w} \cdot \boldsymbol{x}\} = k$. The weighted threshold secret sharing for $\mathcal{W}^{\Pi}_{\boldsymbol{w},t} = \mathcal{W}^{\Pi}_{\mathbf{1},k}$ obtained from Proposition 1 is nothing but the (k, n)-Shamir secret sharing scheme.

▶ **Example 10.** We consider the bipartite adversary structure $\mathcal{B}_k^n(S)$ in Example 5 again.

If $n = dk - r$ and $0 < r < d - 1$, then the convex hull $C = \mathrm{Conv}(\Phi^{\Pi}(\mathcal{B}_k^n(S)))$ is

$$C = \{(x, y) \in \mathbb{R}^2 : x \geq 0,\ y \geq 0,\ (k-1)x + ky \leq k(k-1)\}.$$

The closest point $\boldsymbol{c}^* \in C$ to $\boldsymbol{p}$ is on the line $(k-1)x + ky = k(k-1)$. In particular, $\boldsymbol{c}^* - \boldsymbol{p}$ is parallel to $(k-1, k)$. Therefore, we can set $\boldsymbol{w} \in \mathbb{Z}_+^2$ in the proof of Theorem 9 as $(k-1, k)$. If we set $t = \max_{\boldsymbol{x} \in \Phi^{\Pi}(\mathcal{B}_k^n(S))}\{\boldsymbol{w} \cdot \boldsymbol{x}\} = k(k-1)$, then $\mathcal{B}_k^n(S) = \mathcal{W}^{\Pi}_{\boldsymbol{w},t}$. As a result, we obtain a $\mathcal{B}_k^n(S)$-private d-multiplicative secret sharing scheme whose information ratio is $\boldsymbol{w} \cdot \Phi^{\Pi}(P) = dk^2 - dk + 1$. The scheme can be defined over any finite field $\mathbb{F}$ with $|\mathbb{F}| > dk^2 - dk + 1$.

If $n = dk$, then the convex hull $C = \mathrm{Conv}(\Phi^{\Pi}(\mathcal{B}_k^n(S)))$ is

$$C = \{(x, y) \in \mathbb{R}^2 : x \geq 0,\ 0 \leq y \leq 1,\ x + 2y \leq k\}.$$

Now, we can set $\boldsymbol{w} \in \mathbb{Z}_+^2$ as $(1, 2)$ and $t = \max_{\boldsymbol{x} \in \Phi^{\Pi}(\mathcal{B}_k^n(S))}\{\boldsymbol{w} \cdot \boldsymbol{x}\} = k$. Therefore, we obtain a $\mathcal{B}_k^n(S)$-private d-multiplicative secret sharing scheme whose information ratio is $\boldsymbol{w} \cdot \Phi^{\Pi}(P) = dk + 1 = n + 1$. The scheme can be defined over any finite field $\mathbb{F}$ with $|\mathbb{F}| > n + 1$.

In both cases, the information ratios are smaller than those of the d-multiplicative schemes in Example 5, which are $kn = dk^2 - rk$ if $0 < r < d-1$ and $2n$ if $r = 0$. However, the schemes in Example 5 can be defined over a smaller field $\mathbb{F}$ such that $|\mathbb{F}| > n$.

Moreover, we provide another example of adversary structures which arise in real-world scenarios and apply Theorem 9 to them. In a naive approach based on weighted threshold secret sharing, the obtained information ratio would grow infinitely depending on the description of an adversary structure.

▶ **Example 11.** Suppose that a Π-partite adversary structure Δ is described as $\Phi^{\Pi}(\Delta) = \{\boldsymbol{x} \in \Phi^{\Pi}(2^P) : \boldsymbol{a} \cdot \boldsymbol{x} \leq b\}$ for some $\boldsymbol{a}$ and b and that $\boldsymbol{a} \cdot \boldsymbol{p} > b$. In a real-world setting, such Δ corresponds to a situation in which each value $\boldsymbol{a}(j)$ is the cost required to corrupt a single server in the j-th part and b specifies the maximum tolerable cost. Note that there are infinitely many pairs $(\boldsymbol{a}', b')$ such that $\Phi^{\Pi}(\Delta) = \{\boldsymbol{x} \in \Phi^{\Pi}(2^P) : \boldsymbol{a}' \cdot \boldsymbol{x} \leq b'\}$ since $\Phi^{\Pi}(\Delta)$ is a finite subset in $\mathbb{R}^{\ell}$.

If $\boldsymbol{a} \in \mathbb{Z}_+^{\ell}$, then we have $\Delta \subseteq \mathcal{W}^{\Pi}_{\boldsymbol{a},t}$ and $dt < \boldsymbol{a} \cdot \Phi^{\Pi}(P)$ for $t = \max_{\boldsymbol{x} \in \Phi^{\Pi}(\Delta)}\{\boldsymbol{a} \cdot \boldsymbol{x}\}$. Hence, we immediately obtain a Δ-private d-multiplicative scheme whose information ratio is $\boldsymbol{a} \cdot \Phi^{\Pi}(P)$. However, $\boldsymbol{a}$ is not necessarily a vector of non-negative integers. One may have a good rational approximation $\widetilde{\boldsymbol{a}}$ of $\boldsymbol{a}$ and set $\boldsymbol{w} = N\widetilde{\boldsymbol{a}} \in \mathbb{Z}_+^{\ell}$ for some N. Nevertheless, as the complexity of $\boldsymbol{a}$ increases, N would grow infinitely, which results in a large information ratio.

From Theorem 9, we can construct a Δ-private d-multiplicative scheme whose information ratio depends on the distance between the point $\boldsymbol{p}$ and the hyperplane $H = \{\boldsymbol{x} \in \mathbb{R}^{\ell} : \boldsymbol{a} \cdot \boldsymbol{x} = b\}$ rather than the complexity of the coefficient vector $\boldsymbol{a}$. Let $C = \{\boldsymbol{x} \in \mathbb{R}^{\ell} : \boldsymbol{a} \cdot \boldsymbol{x} \leq b\}$. Since C is convex, C includes $\mathrm{Conv}(\Phi^{\Pi}(\Delta))$. Thus, the distance between $\boldsymbol{p}$ and $\mathrm{Conv}(\Phi^{\Pi}(\Delta))$ is lower bounded by $\mathrm{dist}(\boldsymbol{p}, H)$. In other words, for any $\boldsymbol{c} \in \mathrm{Conv}(\Phi^{\Pi}(\Delta))$, it holds that $\|\boldsymbol{c} - \boldsymbol{p}\| \geq \mathrm{dist}(\boldsymbol{p}, H) = (\boldsymbol{a} \cdot \boldsymbol{p} - b)/\|\boldsymbol{a}\|$. From Theorem 9, we have a Δ-private d-multiplicative secret sharing scheme Σ such that

$$\sigma(\Sigma) \leq \frac{\ell n^2}{\mathrm{dist}(\boldsymbol{p}, H)} + n = \frac{\ell n^2 \|\boldsymbol{a}\|}{\boldsymbol{a} \cdot \boldsymbol{p} - b} + n.$$

To explicitly obtain secret sharing schemes from Theorem 9, we have to find the vector $\boldsymbol{h} = (\boldsymbol{p} - \boldsymbol{c}^*)/\|\boldsymbol{p} - \boldsymbol{c}^*\|$ in Lemma 7, where $\boldsymbol{c}^*$ is the closest point in $C = \mathrm{Conv}(\Phi^{\Pi}(\Delta))$ to $\boldsymbol{p}$. For that purpose, we can make use of (hard margin) *support vector machine* [22]. Note that, since C is convex, the vector $\boldsymbol{h}$ is characterized by the condition $\boldsymbol{h} \cdot (\boldsymbol{p} - \boldsymbol{x}) \geq \epsilon$, $\forall \boldsymbol{x} \in C$, where $\epsilon = \mathrm{dist}(\boldsymbol{p}, C)$. Set a training data set $\mathcal{D}$ as $\mathcal{D} = \{(\boldsymbol{x}, -1) : \boldsymbol{x} \in \Phi^{\Pi}(\Delta)\} \cup \{(\boldsymbol{p}, 1)\}$. Consider the following quadratic programming problem with respect to $\boldsymbol{w} \in \mathbb{R}^{\ell}$ and $b \in \mathbb{R}$:

$$\begin{array}{ll} \text{minimize} & \|\boldsymbol{w}\|^2 \\ \text{subject to} & y(\boldsymbol{w} \cdot \boldsymbol{x} - b) \geq 1, \qquad \forall (\boldsymbol{x}, y) \in \mathcal{D}. \end{array}$$

It holds that $\boldsymbol{h} \cdot \boldsymbol{x} - (\boldsymbol{h} \cdot \boldsymbol{p} - \epsilon/2) \leq -\epsilon/2$ for any $\boldsymbol{x} \in \Phi^{\Pi}(\Delta)$, which means that $((2/\epsilon)\boldsymbol{h}, (2/\epsilon)\boldsymbol{h} \cdot \boldsymbol{p} - 1)$ is a feasible solution to the problem. On the other hand, for any feasible solution $(\boldsymbol{w}, b)$, we have $C \subseteq \{\boldsymbol{x} \in \mathbb{R}^{\ell} : \boldsymbol{w} \cdot \boldsymbol{x} - b \leq -1\}$ and hence $\epsilon \geq 2/\|\boldsymbol{w}\|$, that is, $\|\boldsymbol{w}\| \geq \|(2/\epsilon)\boldsymbol{h}\|$. Therefore, the optimal solution $(\boldsymbol{w}^*, b^*)$ is given by $((2/\epsilon)\boldsymbol{h}, (2/\epsilon)\boldsymbol{h} \cdot \boldsymbol{p} - 1)$. Then we obtain the vector $\boldsymbol{h}$ by computing $\boldsymbol{h} = \boldsymbol{w}^*/\|\boldsymbol{w}^*\|$.

References

1 Omer Barkol, Yuval Ishai, and Enav Weinreb. On d-multiplicative secret sharing. *Journal of cryptology*, 23(4):580–593, 2010.

2 Amos Beimel. Secret-sharing schemes: A survey. In *Coding and Cryptology*, pages 11–46, 2011.

3 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 1–10, 1988.

4 G. R. Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pages 313–318, 1979.

5 Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under ddh. In *Advances in Cryptology — CRYPTO 2016*, pages 509–539, 2016.

6 Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without fhe. In *Advances in Cryptology — EUROCRYPT 2019*, pages 3–33, 2019.

7 I. Cascudo, R. Cramer, and C. Xing. The arithmetic codex. In *2012 IEEE Information Theory Workshop*, pages 75–79, 2012.

8 Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *Advances in Cryptology — CRYPTO 2006*, pages 521–536, 2006.

9 Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *Advances in Cryptology — EUROCRYPT 2000*, pages 316–334, 2000.

10 Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *Advances in Cryptology — CRYPTO 2007*, pages 572–590, 2007.

11 Oriol Farràs, Jaume Martí-Farré, and Carles Padró. Ideal multipartite secret sharing schemes. *Journal of cryptology*, 25(3):434–463, 2012.

12 Oriol Farràs and Carles Padró. Ideal secret sharing schemes for useful multipartite access structures. In *Coding and Cryptology*, pages 99–108, 2011.

13 P. M. Gruber. *Convex and Discrete Geometry*. Springer-Verlag, 2007.

14 Martin Hirt and Daniel Tschudi. Efficient general-adversary multi-party computation. In *Advances in Cryptology — ASIACRYPT 2013*, pages 181–200, 2013.

15 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.

16 E. Karnin, J. Greene, and M. Hellman. On secret sharing systems. *IEEE Transactions on Information Theory*, 29(1):35–41, 1983.

17 Emilia Käsper, Ventzislav Nikov, and Svetla Nikova. Strongly multiplicative hierarchical threshold secret sharing. In *International Conference on Information Theoretic Security*, pages 148–168, 2007.

18 M. Liu, L. Xiao, and Z. Zhang. Multiplicative linear secret sharing schemes based on connectivity of graphs. *IEEE Transactions on Information Theory*, 53(11):3973–3978, 2007.

19 Ueli Maurer. Secure multi-party computation made simple. *Discrete Applied Mathematics*, 154(2):370–381, 2006.

20 A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

21 D. R. Stinson. Decomposition constructions for secret-sharing schemes. *IEEE Transactions on Information Theory*, 40(1):118–125, 1994.

22 Vladimir Vapnik. Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780, 1963.

A The Bipartite Adversary Structure $\mathcal{B}_k^n(S)$

▶ **Proposition 12.** *Let k, d, r be integers such that $k \geq 1$, $d \geq 2$, and $0 \leq r < d-1$, respectively. Set $n = dk - r$. For any subset $S \subseteq P$ of size $(d-r)k-1$, the bipartite adversary structure $\mathcal{B}_k^n(S)$ is of type Q_d.*

Proof. Assume that P is covered by some pairwise disjoint d subsets $A_1, \ldots, A_d$ in $\mathcal{B}_k^n(S)$. Since $n = (d-r)k + r(k-1)$ and $|A_i| \leq k$ for any $i \in [d]$, we may assume that the first $d-r$ subsets $A_1, \ldots, A_{d-r}$ are of size k and the other subsets $A_{d-r+1}, \ldots, A_d$ are of size $k-1$. From the definition of $\mathcal{B}_k^n(S)$, $A_1, \ldots, A_{d-r}$ are pairwise disjoint subsets of S and, in particular, $|S| \geq |A_1| + \cdots + |A_{d-r}|$. However, S is a set of size $(d-r)k-1$, which is a contradiction. ◀

▶ **Proposition 13.** *Continuing the notation of Proposition 12, $\mathcal{B}_k^n(S) \cup \{B\}$ is not of type Q_d for any subset $B \notin \mathcal{B}_k^n(S)$ of size k.*

Proof. If B is a subset of size k such that $B \notin \mathcal{B}_k^n(S)$, then $|S \setminus B| \geq (d-r-1)k$. Then we can partition $S \setminus B$ into pairwise disjoint $d-r-1$ subsets $B_2, \ldots, B_{d-r}$ of size k. By partitioning $P \setminus (B \cup B_2 \cup \cdots \cup B_{d-r})$ into r subsets $B_{d-r+1}, \ldots, B_d$ each of size $k-1$, we obtain $B \cup B_2 \cup \cdots \cup B_d = P$, which means $\mathcal{B}_k^n(S) \cup \{B\}$ is not Q_d. ◀

Efficient MPC with a Mixed Adversary

Martin Hirt
ETH Zurich, Switzerland
hirt@inf.ethz.ch

Marta Mularczyk
ETH Zurich, Switzerland
mumarta@inf.ethz.ch

Abstract

Over the past 20 years, the efficiency of secure multi-party protocols has been greatly improved. While the seminal protocols from the late 80's require a communication of $\Omega(n^6)$ field elements per multiplication among n parties, recent protocols offer linear communication complexity. This means that each party needs to communicate a constant number of field elements per multiplication, independent of n.

However, these efficient protocols only offer active security, which implies that at most $t < n/3$ (perfect security), respectively $t < n/2$ (statistical or computational security) parties may be corrupted. Higher corruption thresholds (i.e., $t \geq n/2$) can only be achieved with degraded security (unfair abort), where one single corrupted party can prevent honest parties from learning their outputs.

The aforementioned upper bounds ($t < n/3$ and $t < n/2$) have been circumvented by considering mixed adversaries (Fitzi et al., Crypto' 98), i.e., adversaries that corrupt, at the same time, some parties actively, some parties passively, and some parties in the fail-stop manner. It is possible, for example, to achieve perfect security even if 2/3 of the parties are faulty (three quarters of which may abort in the middle of the protocol, and a quarter may even arbitrarily misbehave). This setting is much better suited to many applications, where the crash of a party is more likely than a coordinated active attack.

Surprisingly, since the presentation of the feasibility result for the mixed setting, no progress has been made in terms of efficiency: the state-of-the-art protocol still requires a communication of $\Omega(n^6)$ field elements per multiplication.

In this paper, we present a perfectly-secure MPC protocol for the mixed setting with essentially the same efficiency as the best MPC protocols for the active-only setting. For the first time, this allows to tolerate faulty majorities, while still providing optimal efficiency. As a special case, this also results in the first fully-secure MPC protocol secure against any number of crashing parties, with optimal (i.e., linear in n) communication. We provide simulation-based proofs of our construction.

2012 ACM Subject Classification Security and privacy → Network security

Keywords and phrases Multi-party Computation, Communication Cost

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.3

Related Version A full version of the paper is available at `https://eprint.iacr.org/2020/356`.

Funding *Marta Mularczyk*: Research supported by the Zurich Information Security and Privacy Center (ZISC).

1 Introduction

In this work, we consider the problem of secure multi-party computation (MPC), where n mutually distrusted parties want to jointly perform some computation, represented by a circuit over a finite field, with input, output, multiplication, affine and randomness gates. We assume that each pair of parties is connected by a secure channel and that the communication is synchronous.

Intuitively, a protocol executed by the parties is secure if it is "as good as" an ideal trusted third party who performs the computation for the parties. This is formalized in the so-called real-world/ideal-world paradigm by requiring that anything an adversary can do in the real-world protocol execution can also be achieved in the ideal world with the trusted party. In this work, we focus on perfect security against a central, static adversary, who can corrupt up to t parties of its choice.

Corruption Types

In the literature, several types of corruption are considered, in particular, active, passive (also called semi-honest) and fail-stop corruption[1]. Since the seminal works [9, 15], it is known that a perfectly-secure protocol can only tolerate $t < n/3$ active corruptions, or $t < n/2$ passive corruptions.[2] We note that one can tolerate $t < n$ active corruptions, at the cost of degrading security (i.e., give up on guaranteed output delivery), but in this work we focus on full security.

Fitzi et al. [26] pointed out a trade-off between the adversary's capabilities and the overall number of corruptions. For example, considering passive adversaries has the advantage of being able to tolerate as many as $t < n/2$ corruptions (in the fail-stop setting this is even $t < n$). However, these weak corruption types are not practical – if a single party misbehaves, protocols for such settings give no security guarantees at all. On the other hand, considering only active corruptions may be too pessimistic, since it comes at the cost of lowering the threshold to $n/3$. [26] showed that there is room in between the above "pure" settings and introduced the mixed-adversary setting, in which the adversary can simultaneously corrupt up to t_a parties actively, up to t_p parties passively, and up to t_f parties in a fail-stop manner.[3] Perfectly secure MPC in the mixed setting is possible if and only if $3t_a + 2t_p + t_f < n$.

The mixed setting strictly generalizes the pure settings and, in addition, offers flexibility, since it allows to trade off a few corruptions of a strong type for many corruptions of a weaker type. For example, decreasing t_a by one allows to tolerate three more fail-stop corruptions. A protocol for the mixed setting can tolerate, for example, $n/2$ fail-stop corrupted parties, in addition to slightly less than $1/6$ actively corrupted parties (which gives dishonest majority).

Communication-Efficient MPC

The first protocols proving that MPC is possible in the pure settings [9, 15] and in the mixed setting [26] were quite inefficient, with communication costs of evaluating one multiplication gate as high as $\Omega(n^6)$ field elements. Since then, great improvements have been achieved in the pure settings [19, 38, 22, 8, 10, 29, 33], leading to protocols with complexity linear in n. However, for the (from a practical viewpoint) more relevant mixed setting, no practically efficient protocols are known. Providing such protocols would immediately yield efficient solutions for all pure settings, and, additionally, all settings "in between".

---

[1] Here the adversary is only allowed to crash parties, and it does not learn their internal states. Sometimes this setting is relaxed by making the sending operation atomic – a party can only crash after sending all messages in a given round. Since this seems unrealistic, we do not consider this relaxation.

[2] It is also possible to achieve higher thresholds, for example $t < n/2$ active corruptions, if one considers additional assumptions (such as a broadcast channel). In this paper we do not consider these settings.

[3] We stress that a fail-corrupted party can be simultaneously passively-corrupted, in which case it colludes with other corrupted parties.

1.1 Contributions

We present the first multi-party computation protocol with perfect security against mixed adversaries, and with the communication cost of computing one multiplication gate linear in the number of parties. We achieve the optimal resiliency of $3t_a + 2t_p + t_f < n$. Moreover, we provide simulation-based proofs of security of our constructions.

This immediately yields, as special cases, protocols with optimal communication complexity for all pure settings. In particular, we get a protocol that tolerates $t_f < n$ fail-corruptions, a realistic setting that has so far received surprisingly little attention in the MPC literature. Moreover, our result covers all settings in between, for instance, one with many fail-corruptions and few active corruptions, as long as $t_f + 3t_a < n$. We believe that considering such settings makes sense, since in many applications it may be more likely that a party crashes, rather than that it engages in a malicious, coordinated attack.

Perhaps surprisingly, many techniques used in the settings with only active or only passive corruptions fail in the setting with additional fail-corruptions. We briefly summarize a couple of problems:

- The efficient secret reconstruction protocol of [22] requires the number of correct parties (that is, neither actively nor fail-corrupted) to be in $\Omega(n)$. Intuitively, in order to reconstruct k sharings, these are expanded to n sharings (using an error-correcting code). These n sharings are then reconstructed (robustly), one sharing to one party, who then forwards the secret to all other parties. The error-correcting code allows to correct faults introduced by corrupted parties. In the active-only setting, error correction is possible as long as $k \leq n - 2t_a$, so one can reconstruct $k = n/3$ sharings at costs $O(n^2)$. In the mixed setting, error-correction (with erasures) requires $k \leq n - 2t_a - t_f$, which might be as small as 1. So the quadratic costs cannot be amortized. To deal with this problem, we develop a more efficient, but non-robust, version of the protocol from [22], and employ an extended version of the player-elimination framework to make it robust.
- A standard technique for evaluating an input gate is to reconstruct an existing sharing of a random value towards the inputting party, who then broadcast the difference to its input. This approach breaks down in the mixed setting, if the inputting party crashes. This is because known broadcast protocols for the mixed setting [27] provide no guarantees for a corrupted sender, so the received value can be arbitrary (even if the sender is only fail-corrupted). In fact, a rushing adversary may even be able to change the sender's input to a related value.[4]

  We introduce a simple generic technique to deal with such situations – We execute the standard broadcast protocol that guarantees correctness only if the sender does not crash, and afterwards check if the party crashed, by executing a special heartbeat protocol. If so, its input is set to a default value (note that allowing this is unavoidable, e.g. if the party crashes before sending any messages).
- Hyper-invertible matrices [8] are used to generate big bunches of sharings of random values. Every party shares one input to the matrix. Then the parties verify the consistency of all outputs by reconstructing and verifying t_a of them. In the active setting, this is achieved by reconstructing $2t_a$ sharings (each to a different party). In the mixed setting one would need to reconstruct $2t_a + t_f$ sharings. This destroys the efficiency of the construction.

  To deal with this, we use the same approach as above – we change the semantics of

[4] Note that the fail-corrupted parties do not reveal their inputs, unless they are also passively or actively corrupted.

the protocol from [8] and give guarantees analogous to those of the broadcast protocol. Specifically, correctness is guaranteed only if no party crashes. In case of crashes, inconsistent outputs can go through undetected – this will be handled in the extended player-elimination framework, using the heartbeat subroutine.

- In the player-elimination framework [35], the actual computation is performed by a very efficient, so-called "detectable" protocol, that does not guarantee correctness, but that guarantees that an inconsistency is detected by a correct party. After the protocol, the parties agree on whether any inconsistencies were detected. If so, they identify the corrupted parties, eliminate them, and repeat the process. Several problems occur in the mixed setting. First, since we modified the semantics of hyper-invertible matrices, in our computation faults may not be observed by any correct party. Second, parties crashing during the identification process cause additional headache. Finally, it is not known how to identify a conflict between an honest and a corrupted party. In the active setting ($3t_a < n$), one can simply eliminate both of them, but in the mixed setting, eliminating a crashed party with an honest party would violate the threshold ($3t_a + 2t_p + t_f < n$). We deal with all these problems with a simple and elegant trick: We use the (almost) standard fault detection from [35]. In case of crashes, we might identify wrong parties. So, once we know who should be eliminated, we check whether some party has crashed and, if so, we eliminate it (alone). Otherwise, the parties were identified correctly and can be eliminated.

We prove the security of our protocol in a simulation-based framework, which allows to simplify and modularize the proof. That is, for most of our subprotocols we define an ideal functionality and prove that it is realized by the construction. We can then use this idealized functionality in the higher-level protocol, and the security of the overall construction follows by the composition theorem. We note, however, that modularization of protocols executed within the player-elimination framework is nontrivial. Roughly, idealizing a detectable protocol makes it impossible to identify corrupted parties in case of a disruption (this usually involves publicly replaying the disrupted execution, and hence depends on the actual implementation of the protocol). Therefore, for some subprotocols, instead of defining ideal functionalities, we only prove that they satisfy certain properties, which are then used in further proofs.

1.2 Related Work

Communication-Efficient Active MPC

Since the seminal feasibility results [45, 32, 9, 15, 43, 5], the goal of reducing the communication complexity of actively-secure multi-party computation has received a lot of attention. The considerable (although polynomial) complexity of the first protocols was reduced to linear cost per multiplication gate in the cryptographic setting [19, 38], in the unconditional setting (that is, allowing negligible error probability) [22, 20, 10, 29, 28], and in the setting with perfect security [8, 33].

There also exist efficient protocols that can tolerate a dishonest majority [24, 21, 42, 4] (where the online phase has linear complexity). We stress that such protocols can only offer degraded security [17], in particular, no fairness or guaranteed output delivery. In this work, we focus on full security.

On the negative side, it has been proved [23] that linear per-multiplication gate complexity is inherent for unconditionally secure protocols.

Communication Cost Independent of the Circuit Depth

Most protocols with linear communication complexity, including ours, communicate (over the whole execution) additional $O(D \cdot p(n))$ field elements, where D is the multiplicative depth of the circuit[5] and p is a small polynomial. This caveat was recently removed by Goyal et al. [33], who construct a perfectly-secure protocol for the active setting with linear complexity and no dependency on the circuit depth. We expect that our techniques can be applied to the protocol of [33], resulting in a more efficient (for certain circuits) protocol for the mixed setting. We leave proving this claim as an important open question.

Mixing Different Types of Corruptions

Protocols providing several guarantees, depending on the number and the type of corruptions, have been proposed [14, 39, 40]. These papers consider adversaries that *either* corrupt a number of parties passively, *or* corrupt a (smaller) number of parties actively. In contrast, in the mixed setting, an adversary *simultaneously* corrupts some parties passively and some actively.

The mixed setting was considered by Badrinarayanan et al. [3] in the context of round-efficient protocols. The authors present a constant-round protocol for the cryptographic setting, and assuming setup. They also give a simulation-based proof, in a model similar to ours. We note that their scheme is not communication-efficient.

The mixed setting has also been studied in the context of secure message transmission over an incomplete network [25, 16, 2], and in the context of degraded security [34].

Ghodosi and Pieprzyk [30] strengthen the mixed-adversary setting without fail-stop corruptions (with $t_f = 0$) to the setting with a so-called omnipresent adversary. The mixed setting can also be generalized to deal with so-called general adversaries [7, 36] (we note that these general protocols are very inefficient in the threshold setting).

1.3 Protocol Overview

Our protocol follows the preprocessing model – it consists of two phases: the preparation (offline) phase, executed even before the inputs are specified, and the evaluation (online) phase, executed once the circuit and the inputs are known.

We employ circuit randomization [6], which means that the goal of the preparation phase is to generate a number of shared multiplication triples and sharings of random values. This is done using hyper-invertible matrices [8], modified for the mixed setting. Then, in the evaluation phase, the circuit is evaluated gate by gate: input gates are evaluated with the help of existing sharings of random values, affine gates are evaluated locally using linearity of the secret sharing, and a bunch of multiplication gates is evaluated simultaneously with the help of multiplication triples (one per gate) and a new protocol for reconstructing a bunch of secrets towards all parties (using a non-robust version of the protocol from [22]). Evaluating an output gate corresponds to secret reconstruction.

Both phases employ the player-elimination framework [35], modified for the mixed setting, where some parties are eliminated whenever the adversary disrupts the protocol. The eliminated parties are excluded from further computation, with the exception of providing

[5] The multiplicative depth of a circuit is the maximal number of multiplication gates on any path in the circuit from an input or random gate to an output gate.

inputs and receiving outputs. The preparation phase starts with the original party set, and the evaluation phase continues with the set resulting from the preparation phase. (Note that this is different than in [8], where only the preparation phase requires player elimination.)

The communication complexity of the preparation phase is $O(|C|n\kappa + n^3\kappa)$, where $|C|$ is the size of the circuit. The communication complexity of the evaluation phase is $O(|C|n\kappa + n^3\kappa + Dn^3\kappa + c_i n^2\kappa)$, where D is the multiplicative and c_i is the number of input gates.[6]

1.4 Outline of the Paper

Section 2 presents the stand-alone simulation-based model with mixed adversaries. Section 3 contains some essential preliminaries. Section 4 considers byzantine agreement protocols needed in our setting. Section 5 extends the player-elimination framework to the mixed setting. Section 6 treats Shamir secret sharing and various reconstruction protocols used in this paper. Our main protocol is presented in Sections 7 (the preparation phase) and 8 (the evaluation phase). Finally, Section 9 contains conclusions and addresses universal composition.

2 Model

We consider a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of n parties, who want to compute a function, represented as a circuit over a finite field $\mathbb{F}$ with $|\mathbb{F}| \geq 2n$. The circuit contains c_i input, c_o output, c_m multiplication, c_a affine and c_r random gates.

Mixed Adversaries

A mixed adversary can corrupt up to t_a parties actively, up to t_p parties passively, and up to t_f parties in a fail-stop manner. We denote by $\mathcal{P}_a, \mathcal{P}_p$ and $\mathcal{P}_f$ the sets of actually corrupted parties. The adversary gains full control over actively corrupted parties, and it sees the whole internal state of passively corrupted parties. Moreover, at any point in the protocol, it can make a fail-stop corrupted party crash. Once a party is crashed, it does not send any messages. The adversary cannot see the inputs nor the messages processed by fail-stop corrupted parties (unless they are simultaneously actively or passively corrupted). A party which is not actively corrupted and has not crashed yet is called *correct*.

Stand-alone Security

We consider the standard stand-alone model of [12] with static adversary and synchronous communication over perfectly-secure channels (for details of this model, see [31] or [12]). We extend it to include mixed adversaries as follows. First, note that allowing multiple types of corruption simultaneously is straightforward. Passive and active corruptions are already modeled, so now we focus on fail corruptions.

Intuitively, we model fail corruptions as a very weak form of active corruptions, where the adversary (1) does not see the secret state or the messages received by a fail-corrupted party, and (2) only specifies the moment when such party stops sending messages in a given

[6] One can easily get linear dependency on the number of input gates, using the techniques of [8]. However, since the number of input gates is usually insignificant compared to the number of multiplication and affine gates, we do not include this optimization in this paper.

execution. Formally, the real-world execution starts with all fail-corrupted parties being correct. Then, each round proceeds as follows. First, the correct parties generate their messages, as in the protocol. The messages addressed to the actively- and passively-corrupted parties, together with the randomness used by the latter, are given to the adversary. She then specifies: (1) the messages sent by actively-corrupted parties, and (2) the set of fail-corrupted parties that crash in this round. For each party P_i crashed in this round, the adversary specifies a set of parties who still receive the message from P_i in this round. The crashed parties are no longer considered correct in this execution. Finally, all parties receive their messages.[7] The view of a party consists of its input, randomness and all received messages. The view of the adversary consists of the views of passively- and actively-corrupted parties.

In the ideal world, the ideal-world adversary (the simulator) interacts with the ideal functionality $\mathcal{F}$. As usual, he receives the inputs of passively-corrupted parties and modifies the inputs of the actively-corrupted ones. Moreover, for fail-corrupted parties, he chooses whether the input or a special symbol $\perp$ should be given to $\mathcal{F}$, where $\perp$ means that the party gives no input (note that this is inherent). In our model, fail-corrupted parties always receive outputs (this strong guarantee is achieved by our protocols).

Formally, we consider functionalities $\mathcal{F}$ with domain $(X \cup \{\perp\})^n$, where $\perp \notin X$. The semantic of $\perp$ is left to the functionality, e.g., it can be replaced by a default value. The ideal world is defined as follows: The parties that are not actively corrupted input values from X, and the simulator receives the inputs of passively-corrupted parties. He then chooses the inputs of actively-corrupted parties from $X \cup \{\perp\}$ and specifies a set D of fail-corrupted parties whose inputs are set to $\perp$. $\mathcal{F}$ is evaluated using the above inputs, and outputs are sent to all parties.

The standard security requirement is that for every (unbounded) real-world adversary $\mathcal{A}$, there exists an ideal-world adversary $\mathcal{S}$, such that the joint distribution of the view of $\mathcal{A}$ and the outputs of the correct parties executing the protocol is equal to the joint distribution of the output of $\mathcal{S}$ and the outputs of the correct parties computed by the ideal functionality $\mathcal{F}$.

Modular Composition

The hybrid world with fail corruptions is the standard one (note that fail corruptions do not affect ideal functionalities). Then, replacing ideal evaluation calls by subroutine calls is straightforward: a subroutine is called with the set of actually crashed parties reset (note that the set of fail-corrupted parties is constant). Observe that this means that crashed parties may "come back to life" in subroutines. This is a strong type of corruption, and our protocols are secure in this setting.[8]

Corruption-aware functionalities

The model of [12] was extended by Asharov and Lindell [1] to allow corruption-aware functionalities, whose code depends on the corrupted set. We generalize this to the mixed setting in the straightforward way. Specifically, our functionalities receive as inputs: (1) the inputs from the parties, and (2) the sets $\mathcal{P}_a, \mathcal{P}_p, \mathcal{P}_f$ of corrupted parties. Then, each functionality receives, upon initialization, the same set of parties actually controlled by the

[7] In particular, this means that the crashed parties still receive their messages.

[8] An alternative solution would be to enforce a consistent set of crashed parties. However, modeling this would require techniques from the adaptive setting, such as the environment providing the crashed set. Our solution is simpler and cleaner.

adversary. Note that this is well defined in the static setting. We refer to [12] for a formal description. Finally, we note that the corruption-aware functionalities were used in [1] only as a tool to allow modular proofs. This is the same in our case. In particular, the final MPC functionality both in this paper and in [1] is not corruption aware.

3 Preliminaries

We assume familiarity with hyper-invertible matrices and circuit randomization (we explain these concepts in the full version [37]).

The Current Party Set

In this paper we execute protocols within the player-elimination framework, where some parties are eliminated and the original party set $\mathcal{P}$ is reduced to a smaller set, which we denote by $\mathcal{P}'$. Accordingly, we use t'_a, t'_p, t'_f and n' to denote the corruption thresholds in $\mathcal{P}'$ and $|\mathcal{P}'|$, respectively. A tuple $(\mathcal{P}', t'_a, t'_p, t'_f)$ is called a *configuration*.

The current configuration is an input to most of our protocols. Security is guaranteed only if the inputted configuration is *valid*. Intuitively, we need that $3t'_a + 2t'_p + t'_f < n'$ holds in $\mathcal{P}'$. However, since we use Shamir sharings with degree $d = t_a + t_p$ (independent of $\mathcal{P}'$), we also require that such sharings are possible to reconstruct by the parties in $\mathcal{P}'$. This means that $t_a + t_p < n' - 2t'_a - t'_f$, which implies $3t'_a + 2t'_p + t'_f < n'$.

▶ **Definition 1.** *A configuration* $(\mathcal{P}', t'_a, t'_p, t'_f)$ *is* valid *if (1)* $\mathcal{P}' \subseteq \mathcal{P}$*, (2)* $t_a + t_p < n' - 2t'_a - t'_f$*, and (3)* $|\mathcal{P}' \cap \mathcal{P}_{\mathsf{x}}| \leq t'_{\mathsf{x}}$ *for all* $\mathsf{x} \in \{a, p, f\}$*, where* $\mathcal{P}_{\mathsf{x}}$ *denotes the actually corrupted parties.*

Complete Break Down

In some cases (which will never occur in the real execution) our functionalities provide no security at all. This happens, for example, when the parties input inconsistent sets $\mathcal{P}'$. In this case, we say that the functionality executes *Complete Break Down*, meaning that it immediately stops execution, sends all inputs to the adversary and allows her to set all outputs (see also [18]). Simulation is trivial in this case, and we omit this part in the code of simulators.

Fixed Matrices

At different times in the protocols we use hyper-invertible and Vandermonde matrices. We require that whenever in a protocol a matrix is chosen, all parties choose the same matrix. So, for example, a fixed matrix of maximal needed size can be chosen as the parameter of the protocol and the parties can use submatrices of appropriate sizes.

4 Byzantine Agreement

We consider two types of Byzantine-agreement protocols, namely consensus and broadcast. A consensus protocol allows a set of parties, each holding an input x_i, to reach agreement on a common value y, where $y = x$ if all correct parties have input $x_i = x$. The guarantees of consensus in the mixed setting were formalized by Garay and Perry [27]. They include

consistency – the outputs of all parties that are not actively corrupted are equal, and persistence – if inputs of all correct parties are equal to x, then the outputs are equal to x.[9]

A broadcast protocol allows a sender to publicly announce his value to all parties, where it is guaranteed that indeed all parties receive the same value. Broadcast can be trivially constructed from consensus: the sender sends his value to every party and then parties invoke consensus on the received values. Formally, in the mixed setting, we require the same consistency as in consensus – the outputs of all parties that are not actively corrupted are equal, and validity – if the sender stays correct until the end of the protocol and his input is x, then all parties output x.

A consensus protocol for the setting with active and fail-stop corruptions (that is, for the mixed setting with $t_p = 0$) was given by Garay and Perry [27]. Their protocol achieves one-bit consensus, assuming that $3t_a + t_f < n$. As the protocol is perfectly secure, it is also secure in the presence of (any number of) passively corrupted parties. The communication complexity of the protocol in [27] is $O(n^3)$. However, by applying the king-simulation technique of [11], this complexity can easily be reduced to $O(n^2)$.

When we execute protocols within the player-elimination framework, Byzantine agreement is invoked among the parties in the reduced party set $\mathcal{P}'$, but all parties in $\mathcal{P}$ should learn the output.[10] Hence, we employ the following consensus protocol, where every party $P_i \in \mathcal{P}'$ has input x_i, and every party in $\mathcal{P}$ receives output. The protocol internally invokes a standard consensus protocol, and it is secure assuming that $3t'_a + 2t'_p + t'_f < n'$.

Protocol Consensus($\{x_i\}_{P_i \in \mathcal{P}'}$)

1: The parties in $\mathcal{P}'$ invoke a standard consensus protocol (for example, [27, 11]), where the input of P_i is x_i.
2: Every party in $\mathcal{P}'$ sends the output to every party in $\mathcal{P} \setminus \mathcal{P}'$.
3: Every party in $\mathcal{P}'$ outputs the result of the consensus, and every party in $\mathcal{P} \setminus \mathcal{P}'$ determines its output using the majority rule.

Moreover, all parties in $\mathcal{P}$ should be able to act as a sender in broadcast.[11] Such broadcast can easily be constructed by having a sender $P_s \in \mathcal{P}$ send his value to all parties in $\mathcal{P}'$, who then invoke Consensus on the received values.

The cost of Consensus instantiated with the protocol of [27, 11] is $O(n^2)$ (note that n is the size of the initial set $\mathcal{P}$). We denote by $\mathcal{BA}(\kappa)$ the complexity of broadcasting or reaching consensus on a κ-bit message in the mixed setting. With the above protocol, $\mathcal{BA}(\kappa) = O(n^2\kappa)$.

▶ Remark. Our broadcast protocol guarantees no validity in case a fail-corrupted sender crashes. This means that while the adversary can only prevent a fail-corrupted party from sending messages, she can modify the messages broadcasted by such party. In this paper we only consider such weak guarantees.

As a side remark, we note that an alternative solution would be to strengthen the broadcast. This can be done by invoking after the weak broadcast our Heartbeat protocol (defined in Section 5) on the sender. If Heartbeat succeeds (meaning that the sender is correct at the end), then the value from the first broadcast can be used. Otherwise (the sender crashed during or before Heartbeat), the broadcast fails and parties output $\perp$.

[9] In terms of [27], we require agreement and frangible validity. We do not consider strong validity in this paper.
[10] All parties must know the current state of the protocol.
[11] The reason is that eliminated parties can still provide inputs to the MPC protocol.

5 Player-Elimination Framework for the Mixed Setting

The player-elimination framework was introduced in [35] with the goal of designing efficient actively secure multiparty protocols. The central idea is that every time the adversary actively disrupts the protocol execution, at least one malicious party is "eliminated", and the disrupted part of the protocol is repeated.

The player-elimination framework reduces the problem of designing a resilient protocol to the problem of designing a *detectable* protocol for the same task. In a nutshell, a detectable protocol can give incorrect outputs, but only if this is noticed by at least one honest party. Privacy needs to be preserved even if the adversary disrupts the protocol execution. Since detectability is a much weaker requirement than resilience, detectable protocols for a given task are usually much more efficient. The player-elimination framework transforms a detectable protocol into a resilient one, essentially without complexity overhead.

A protocol executed within the player-elimination framework is evaluated in segments. The size of a segment can be arbitrary, but it influences the overall complexity. For each segment, four phases are executed: detectable computation, fault detection, fault localization and player elimination. In the first phase the parties evaluate the segment, using the (very efficient) detectable protocol. Then, in fault detection, they detect whether any of them noticed a disruption in the first phase. If this is *not* the case, they proceed to the next segment. Otherwise, in fault localization they localize a set of two disputing parties (that is, two parties such that each of them claims that the other is corrupted). Both these parties are eliminated in player elimination, and the current segment is repeated. The eliminated parties no longer take part in the computation, but they can still provide inputs and receive outputs.

Unfortunately, the player-elimination framework for the active setting [35] does not work in the mixed setting. One reason is that a detectable protocol only guarantees that a fault is noticed by a non-actively corrupted party. This means that if all parties who noticed it crash right after the protocol, the fault may not be detected. Moreover, fault detection and localization subprotocols for the active-only setting break down in the mixed setting. For example, the localized set of disputing parties may contain a crashed and an honest party. If such set was eliminated, the condition $3t_a + 2t_p + t_f < n$ would no longer hold.

We therefore enhance the player-elimination framework to cope with mixed adversaries as follows. First, we modify the notion of a detectable protocol. Roughly, if a party crashes during (or before) the execution of a detectable protocol, then the output can be incorrect even without a correct party noticing this. Accordingly, in fault detection the parties detect not only whether a correct party noticed a disruption, but also whether a party crashed. Then, in Fault Localization, two sets of parties are localized, such that unless a party in the first set has crashed, at least one of the two parties in the second set is actively corrupted. Afterwards, the protocol Heartbeat is invoked on each party in the first set, in order to determine whether it crashed. Finally, in player elimination the parties eliminate either the crashed parties in the first set or, if there are no such parties, all parties in the second set.

Protocol Convention

In the description of detectable protocols, we say that a (correct) party who notices a fault becomes *unhappy*. Formally, each party stores a binary value, which we call a happy-bit, and which can take values "happy" and "unhappy". When a party becomes unhappy, it sets its happy-bit to "unhappy". The state of the happy-bits is local to each party, but it is persistent between protocols. Furthermore, we adopt the convention that during a detectable protocol, an unhappy party does not send any messages.

Detectability in the Mixed Setting

We extend the definition of a detectable protocol of [35] to the mixed setting. Informally, we require that a protocol may produce incorrect outputs, but only if this is noticed by a correct party, *or if any party crashed*. Moreover, we require (1) completeness – a protocol should give correct outputs if there are no faults during the execution and (2) privacy, which must be preserved always, no matter if there were crashes or malicious behavior.

▶ **Definition 2.** *A passively secure protocol* Π *for a set of parties* $\mathcal{P}'$ *is* detectable *if after the execution of* Π *at least one of the following is true: either (1) the outputs of all correct parties are correct, or (2) at least one correct party in* $\mathcal{P}'$ *is unhappy, or (3) at least one fail-stop corrupted party in* $\mathcal{P}'$ *crashed. Moreover, if all parties are honest, then no (correct) party becomes unhappy. Furthermore,* Π *guarantees privacy in all cases.*

Fault Detection

The protocol FaultDetect is executed by the parties in $\mathcal{P}'$ during the phase of fault detection. Its goal is to unify the happy-bits of the parties, such that *all* parties become unhappy whenever any correct party is unhappy or any party crashed. We also require that FaultDetect is complete, by which we mean that at the end of FaultDetect all correct parties are happy whenever (1) all parties enter the protocol happy, (2) all parties behave according to the protocol specification, and (3) all fail-corrupted parties *finish* FaultDetect alive. In particular, this means that if conditions (1) and (2) are fulfilled, and no party crashed before FaultDetect, but *there was a crash during FaultDetect*, we do not put any requirements on the final values of the happy-bits, except that they are the same for all parties.

Protocol FaultDetect$(\mathcal{P}', t'_a, t'_p, t'_f)$

1: Every $P_i \in \mathcal{P}'$ sends its happy-bit to every other party and sets its happy-bit to "unhappy" if from at least one party it does not receive a bit at all or if the bit it receives is "unhappy" (or if it was unhappy before).
2: The parties in $\mathcal{P}'$ run consensus on their happy-bits.
3: Every $P_i \in \mathcal{P}'$ sets its happy-bit to the result of the consensus.

▶ **Lemma 3.** *Assuming that* $3t'_a + t'_f < n'$*, the protocol FaultDetect gives the following guarantees:*

(Consistency) *At the end of FaultDetect the values of the happy-bits are the same for all parties.*

(Correctness) *If any party starts FaultDetect being correct and unhappy, or if any party starts FaultDetect crashed, then at the end all parties are unhappy.*

(Completeness) *If all parties are honest and start FaultDetect happy, then at the end all parties are happy.*

The communication complexity of FaultDetect is $O(n^2 + \mathcal{BA}(1))$.

Proof. Consistency follows by consistency of consensus. For correctness, note that both an unhappy party and a party which starts FaultDetect crashed make every correct party set its happy-bit to "unhappy" in Step 1. By persistence of consensus, every correct party is unhappy at the end of FaultDetect. Finally, if there is no crashed party and no unhappy party at the start of FaultDetect, and if all parties behave according to the protocol specification and no party crashes, then the value of happy-bit at the beginning of consensus is "happy" for all correct parties. By persistence, every correct party is happy at the end of the protocol. ◀

Heartbeat

The protocol Heartbeat allows the parties in $\mathcal{P}'$ to reach agreement on whether a given party $P_h \in \mathcal{P}'$ is alive. Formally, every party $P_i \in \mathcal{P}'$ outputs a binary value, equal to "alive" or "crashed".

Protocol Heartbeat$(P_h, (\mathcal{P}', t'_a, t'_p, t'_f))$

1: P_h sends the value 1 to every party $P_j \in \mathcal{P}'$.
2: The parties invoke consensus, where the input of P_j is 1 if it received the value 1 from P_h and 0 otherwise. The parties output "alive" if the output of the consensus is 1 and "crashed" otherwise.

▶ **Lemma 4.** *Assuming that $3t'_a + t'_f < n'$, the protocol Heartbeat gives the following guarantees:*
(Consistency) *All parties in $\mathcal{P}'$ output the same value "alive" or "crashed".*
(Correctness) *If P_h starts Heartbeat crashed, then the output of Heartbeat is "crashed".*
(Completeness) *If P_h finishes Heartbeat correct, then the output of Heartbeat is "alive".*
The communication complexity of Heartbeat is $O(n + \mathcal{BA}(1))$.

Proof. Consistency follows directly from consistency of consensus, while correctness and completeness follow from persistence of consensus. ◀

6 Secret Sharing

We employ the Shamir secret sharing [44], where each party P_i has associated with it a unique value $\alpha_i \in \mathbb{F} \setminus \{0\}$. A value $s \in \mathbb{F}$ is correctly shared with degree d among the parties in $\mathcal{P}$ if every correct party $P_i \in \mathcal{P}$ holds a value $s_i \in \mathbb{F}$, such that all points (α_i, s_i) lie on a polynomial g of degree at most d with $g(0) = s$. A situation, in which s is d-shared among $\mathcal{P}$ is called a d-sharing of s and denoted by $[s]_d$. Sometimes we require a value to be simultaneously d-shared and d'-shared. Such a sharing is called a (d, d')-sharing of s and denoted $[s]_{d,d'}$. The Shamir secret sharing is linear, that is, affine operations on shared values can be performed directly on the respective shares, without any communication.

There are two protocols associated with a secret-sharing scheme: share and reconstruct. In this paper, we do not consider the share protocol, and use instead a protocol that generates sharings of random values (as explained in Section 7). On the other hand, we consider three reconstruction protocols, the guarantees of which we describe below (the details are given in the full version [37]). All reconstruction protocols take as input the sharing degree d and the current configuration $(\mathcal{P}', t'_a, t'_p, t'_f)$.

- The private reconstruction protocol RecPriv$(P_r, d, (\mathcal{P}', t'_a, t'_p, t'_f), [s]_d)$ *detectably* reconstructs the sharing $[s]_d$ towards $P_r \in \mathcal{P}$, assuming that $d' < n - t'_a$ and $(\mathcal{P}', t'_a, t'_p, t'_f)$ is valid. The communication cost is $O(n'\kappa)$.
- The private reconstruction protocol RecPrivRobust$(P_r, d, (\mathcal{P}', t'_a, t'_p, t'_f), [s]_d)$ *robustly* reconstructs the sharing $[s]_d$ towards $P_r \in \mathcal{P}$, assuming that $d' < n - 2t'_a - t'_f$ and $(\mathcal{P}', t'_a, t'_p, t'_f)$ is valid. The communication cost is $O(n'\kappa)$.
- The public reconstruction RecPub$(d, \ell, (\mathcal{P}', t'_a, t'_p, t'_f), [s_1]_d, \ldots, [s_\ell]_d)$ *detectably* reconstructs ℓ sharings towards all parties in $\mathcal{P}'$, assuming $d' < n - t'_a$ and $(\mathcal{P}', t'_a, t'_p, t'_f)$ is valid. The communication cost is $O(\ell n'\kappa + n'^2\kappa)$. Technically, the protocol is a non-robust version of the protocol of [22].

7 Preparation Phase

Recall that the goal of the preparation phase is to robustly generate a number of multiplication triples, as formally defined by $\mathcal{F}_{\text{prepPhase}}$. To realize $\mathcal{F}_{\text{prepPhase}}$, we proceed as follows. First, we define an intermediate functionality $\mathcal{F}_{\text{triples}}$, which is essentially a non-robust version of $\mathcal{F}_{\text{prepPhase}}$ (it also includes fault localization) and realize $\mathcal{F}_{\text{prepPhase}}$ in the $\mathcal{F}_{\text{triples}}$-hybrid model. The rest of this section is then devoted to realizing $\mathcal{F}_{\text{triples}}$.

7.1 The Functionality $\mathcal{F}_{\text{prepPhase}}$

The parties input to the functionality the desired number L of triples and the current configuration $(\mathcal{P}', t'_a, t'_p, t'_f)$. Since the protocol employs player elimination, the resulting triples are shared among the parties in a smaller set $\mathcal{P}''$. The resulting configuration $(\mathcal{P}'', t''_a, t''_p, t''_f)$ is outputted to all parties. Moreover, all parties in $\mathcal{P}''$ receive the shares of L triples, where the sharing degree is $d = t_a + t_p$. We model the worst-case scenario, where the adversary is allowed to choose $(\mathcal{P}'', t''_a, t''_p, t''_f)$, as long as it is valid and decides on the shares of passively and actively corrupted parties.

Functionality $\mathcal{F}_{\text{prepPhase}}$

The functionality receives sets of corrupted parties $\mathcal{P}_a, \mathcal{P}_p, \mathcal{P}_f$. Let $d = t_a + t_p$.

1: Receive from each party a number L and a valid configuration $(\mathcal{P}', t'_a, t'_p, t'_f)$.
2: If these values (ignoring $\perp$) are not consistent among parties, or if $(\mathcal{P}', t'_a, t'_p, t'_f)$ is not valid, execute Complete Break Down. Else, send $(\text{Ok}, L, \mathcal{P}', t'_a, t'_p, t'_f)$ to the adversary and proceed as follows.
3: The adversary sends:
- A valid set $\mathcal{P}'' \subseteq \mathcal{P}'$ with thresholds t''_a, t''_p, t''_f.
- For each $P_j \in (\mathcal{P}_a \cup \mathcal{P}_p)$, shares $((a_j^{(k)}, b_j^{(k)}, c_j^{(k)}))_{k=1\dots L}$.

(Set $\mathcal{P}'' = \mathcal{P}'$, $t''_{\mathsf{x}} = t'_{\mathsf{x}}$ and $(a_j^{(k)}, b_j^{(k)}, c_j^{(k)}) = (0,0,0)$ if valid values are not received.)
4: Send $(\mathcal{P}'', t''_a, t''_p, t''_f)$ to all parties.
5: Generate L triples shared among the parties in $\mathcal{P}''$ as follows. For each triple k, choose random $a^{(k)}$ and $b^{(k)}$, and let $c^{(k)} = a^{(k)}b^{(k)}$. Then, for each $k = 1, \dots, L$ and $\mathsf{x} \in \{a, b, c\}$, choose the polynomial $g_{\mathsf{x}}^{(k)}$ at random from the set of all polynomials of degree at most d, going through the point $(0, \mathsf{x}^{(k)})$ and all points in $\{(\alpha_j, \mathsf{x}_j^{(k)}) \mid P_j \in (\mathcal{P}_a \cup \mathcal{P}_p)\}$. Send to each $P_i \in \mathcal{P}''$ its shares $(g_a^{(k)}(\alpha_i), g_b^{(k)}(\alpha_i), g_c^{(k)}(\alpha_i))$.

7.2 The Functionality $\mathcal{F}_{\text{triples}}$

On a high level, the functionality non-robustly generates a number of triples, and, in case of failure, identifies a set containing a significant number of actively- or fail-corrupted parties.

The parties input to $\mathcal{F}_{\text{triples}}$ the desired number of triples ℓ and the configuration $(\mathcal{P}', t'_a, t'_p, t'_f)$. Then, the adversary decides on one of three outcomes: (1) The detectable computation succeeds, and ℓ triples shared with degree $d = t_a + t_p$ among the parties in $\mathcal{P}'$ are generated; (2) The adversary disrupted the protocol and a set containing an active party is identified and outputted to all parties; (3) The adversary disrupted the protocol and a set of fail-corrupted parties is outputted. In the latter two cases, the adversary chooses the outputted set, but a sufficient fraction of parties in the set must be actually corrupted (if this is not satisfied, the parties receive the shared triples).

Functionality $\mathcal{F}_{\text{triples}}$

The functionality receives sets of corrupted parties $\mathcal{P}_a, \mathcal{P}_p, \mathcal{P}_f$. Let $d = t_a + t_p$.

1: Receive from each party a number ℓ and a valid configuration $(\mathcal{P}', t'_a, t'_p, t'_f)$.
2: If these values (ignoring $\perp$) are not consistent among parties, or if $(\mathcal{P}', t'_a, t'_p, t'_f)$ is not valid, execute Complete Break Down. Else, send $(\text{OK}, \ell, \mathcal{P}', t'_a, t'_p, t'_f)$ to the adversary and proceed as follows.
3: Receive from the adversary a message M, which is processed as follows:
- If $\text{M} = (\text{TRIPLES}, ((a_j^{(k)}, b_j^{(k)}, c_j^{(k)}))_{k=1\dots\ell, P_j \in (\mathcal{P}_a \cup \mathcal{P}_p)})$, then generate ℓ triples shared among the parties in $\mathcal{P}'$ as follows. For each triple k, choose random $a^{(k)}$ and $b^{(k)}$, and let $c^{(k)} = a^{(k)} b^{(k)}$. Then, for each $k = 1, \dots, \ell$ and $\mathsf{x} \in \{a, b, c\}$, choose the polynomial $g_{\mathsf{x}}^{(k)}$ at random from the set of all polynomials of degree at most d, going through the point $(0, \mathsf{x}^{(k)})$ and all points in $\{(\alpha_j, \mathsf{x}_j^{(k)}) \mid P_j \in (\mathcal{P}_a \cup \mathcal{P}_p)\}$. Send to each $P_i \in \mathcal{P}'$ its shares $(g_a^{(k)}(\alpha_i), g_b^{(k)}(\alpha_i), g_c^{(k)}(\alpha_i))$.
- If $\text{M} = (\text{ACTIVESET}, E)$, where $E \subseteq \mathcal{P}'$ is a set such that $|E \cap \mathcal{P}_a| \geq |E|/2$, then send $(\text{ACTIVESET}, E)$ to the parties.
- If $\text{M} = (\text{CRASHSET}, E)$, where $E \subseteq \mathcal{P}'$, $E \subseteq \mathcal{P}_f \cup \mathcal{P}_a$ and $E \neq \emptyset$, then send $(\text{CRASHSET}, E)$ to the parties.
- Any other message is treated as $(\text{TRIPLES}, ((0,0,0))_{k=1\dots\ell, P_j \in (\mathcal{P}_a \cup \mathcal{P}_p)})$.

7.3 Realizing $\mathcal{F}_{\text{prepPhase}}$ in the $\mathcal{F}_{\text{triples}}$-hybrid Model

We now present the protocol PreparationPhase that realizes $\mathcal{F}_{\text{prepPhase}}$ in the $\mathcal{F}_{\text{triples}}$-hybrid model. The protocol divides the L into $t_a + t_f$ segments of length ℓ and sequentially calls $\mathcal{F}_{\text{triples}}$ with input ℓ. It starts with $\mathcal{P}'' = \mathcal{P}'$ and in case of a disruption, eliminates parties from $\mathcal{P}''$. The outputs of all parties is the resulting set $\mathcal{P}''$ and, for parties in $\mathcal{P}''$, the shares of the triples.

Protocol PreparationPhase $(L, (\mathcal{P}', t'_a, t'_p, t'_f))$

Let $\ell = \lceil \frac{L}{t_a + t_f} \rceil$ and $d = t_a + t_p$.
Set $\mathcal{P}'' = \mathcal{P}'$, $t''_a = t'_a$, $t''_p = t'_p$ and $t''_f = t'_f$. For each segment $k = 1 \dots (t_a + t_f)$ do:

1: Send ℓ and $(\mathcal{P}'', t''_a, t''_p, t''_f)$ to $\mathcal{F}_{\text{triples}}$.
2: If the output is $(\text{ACTIVESET}, E)$, set $\mathcal{P}'' = \mathcal{P}'' \setminus E$ and $t''_a = t''_a - |E|/2$, and repeat Step 1.
3: Else if the output is $(\text{CRASHSET}, E)$, then set $\mathcal{P}'' = \mathcal{P}'' \setminus E$ and $t''_f = t''_f - |E|$, and repeat Step 1.
4: Else, store the sharings received from $\mathcal{F}_{\text{triples}}$ and continue to the next segment.

Every P_i outputs the configuration $(\mathcal{P}'', t''_a, t''_p, t''_f)$. Moreover, every $P_i \in \mathcal{P}''$ outputs the first L triples of shares received from $\mathcal{F}_{\text{triples}}$.

▶ **Theorem 5.** *Assuming that $3t_a + 2t_p + t_f < n$, the protocol PreparationPhase securely realizes $\mathcal{F}_{\text{prepPhase}}$ in the $\mathcal{F}_{\text{triples}}$-hybrid model, in the presence of a static mixed adversary.*

Proof. The simulator $\mathcal{S}_{\text{prepPhase}}$ only needs to simulate the interaction with the ideal functionality $\mathcal{F}_{\text{triples}}$. This is done by simply executing the code of $\mathcal{F}_{\text{triples}}$.

Throughout the execution in the real world, we have the following invariant: $(\mathcal{P}'', t''_a, t''_p, t''_f)$ is valid and the shares stored by parties in $\mathcal{P}''$ form correct d-sharings of triples. The former follows from the observation that $n'' - 2t''_a - t''_f$ is preserved when parties are eliminated (this is trivially guaranteed by $\mathcal{F}_{\text{triples}}$). This also implies that $3t''_a + 2t''_p + t''_f < n''$. For the latter, notice that a d-sharing in $\mathcal{P}'$ is also a correct d-sharing in $\mathcal{P}''$, since d is constant. The above invariant shows that the outputs are the same in the real and in the ideal world. [12] ◀

[12] Observe that we do not need that in case $\mathcal{F}_{\text{triples}}$ sends $(\text{CRASHSET}, E)$, the parties in E are actually not correct. It is enough that they are in $\mathcal{P}_f$.

7.4 Realizing $\mathcal{F}_{\text{triples}}$

We first construct two auxiliary protocols: The first protocol generates a number of double sharings of random values, using hyper-invertible matrices. This double-sharing protocol can then be used in the second protocol to detectably generate a number of multiplication triples. The triple-generation protocol, together with fault detection and fault localization, can be used to realize $\mathcal{F}_{\text{triples}}$.

Generating Double Sharings

The goal of the protocol DoubleShareRandom is to detectably generate a number of (d, d')-sharings of uniformly random values, unknown to the adversary, assuming that $3t'_a+2t'_p+t'_f < n'$. The degrees d and d' of the outputted sharings can be arbitrary.

We use the trivial protocol Share, which, on input a sharing degree d and a value s from a party P_i, generates a (possibly inconsistent) d-sharing of s.

Protocol Share$(P_i, d, s, (\mathcal{P}', t'_a, t'_p, t'_f))$

P_i chooses a random polynomial g of degree d and sends to every party $P_j \in \mathcal{P}'$ its share $s_j = g(\alpha_j)$.

The protocol generates the (d, d')-sharings in buckets of size at most $n' - 2t'_a - t'_p - \min(t'_a, t'_p)$. The idea is to, for each bucket, use Share to generate n' double-sharings of random values, each value chosen by a different party. Up to t'_a of these sharings might be inconsistent and up to $t'_a + t'_p$ of them might be known to the adversary. Then, we apply to this vector of sharings a (fixed) hyper-invertible matrix. First, this ensures that at least $n' - t_a - t_p$ of the resulting sharings contain uniformly random values, unknown to the adversary. Moreover, hyper-invertibility guarantees that if there exists a set of t'_a resulting sharings which are consistent, then all sharings are consistent. We exploit this fact by reconstructing $2t'_a$ of the resulting sharings, each towards a different party, who then checks the consistency of the sharing it received. If no correct party notices an inconsistency, then all sharings must be consistent. This verification step reveals to the adversary additional $\min(2t'_a, t'_a + t'_p)$ shared values, hence, only $n' - t_a - t_p - \min(2t'_a, t'_a + t'_p) = n' - 2t'_a - t'_p - \min(t'_a, t'_p)$ remain private. To generate any number ℓ of double sharings, the procedure sketched above is invoked a number of times (in parallel).

Protocol DoubleShareRandom$(d, d', \ell, (\mathcal{P}', t'_a, t'_p, t'_f))$

The ℓ sharings are generated in buckets of size $n' - 2t'_a - t'_p - \min(t'_a, t'_p)$ (with the last bucket possibly smaller). Generate each bucket of size l using the following procedure:

1: Every party $P_i \in \mathcal{P}'$ chooses s_i at random and double-shares it among $\mathcal{P}'$ by invoking Share$(P_i, d, s_i, (\mathcal{P}', t'_a, t'_p, t'_f))$ and Share$(P_i, d', s_i, (\mathcal{P}', t'_a, t'_p, t'_f))$.

2: The parties compute locally $([r_1]_{d,d'}, \ldots, [r_{n'}]_{d,d'})^T = M([s_1]_{d,d'}, \ldots, [s_{n'}]_{d,d'})^T$, using the shares of $s_1, \ldots, s_{n'}$, where M is a fixed hyper-invertible matrix of size $n' \times n'$.

3: For every $P_i \in \mathcal{P}'$ and $j \in \{1, \ldots, 2t'_a\}$, P_i sends its shares of $[r_j]_{d,d'}$ to P_j.

4: Every P_j with $j \in \{1, \ldots, 2t'_a\}$ verifies that the values it got define a correct (d, d')-sharing, that is, that all shares of the d-sharing lie on a polynomial g of degree d, that all shares of the d'-sharing lie on a polynomial g' of degree d', and that $g(0) = g'(0)$. If any of these conditions does not hold, P_j gets unhappy.

5: The l sharings generated in this bucket are $[r_{n'-l+1}]_{d,d'}, \ldots, [r_{n'}]_{d,d'}$.

Output all sharings generated in all buckets (that is, ℓ sharings in total).

The communication cost of DoubleShareRandom can be seen to be $O(\ell n'\kappa + n'^2\kappa)$. Hence, for large enough ℓ, the amortized complexity of generating one double-sharing is $O(n'\kappa)$.

▶ **Lemma 6.** *Assuming that $(\mathcal{P}', t'_a, t'_p, t'_f)$ is valid and the values inputted by the parties are consistent, DoubleShareRandom detectably generates ℓ correct (d, d')-sharings, where for each sharing, the shared value is uniformly random given the view of the adversary. The communication complexity of DoubleShareRandom is $O(\ell n'\kappa + n'^2\kappa)$.*

Proof. Consider one bucket, in which l (d, d')-sharings are generated, where $l \leq n' - 2t'_a - t'_p - \min(t'_a, t'_p)$.

Correctness: Assume that no crash occurred and that after the protocol all correct parties are happy. All values s_i generated by the correct parties in Step 1 are double-shared correctly. Thus, at least $n' - t'_a$ sharings $[s_i]_{d,d'}$ are correct. Moreover, at least t'_a out of $2t'_a$ sharings $[r_i]_{d,d'}$ verified in Step 4 are verified by correct parties and, thus, must be correct (as otherwise a correct party would become unhappy). Together, this gives n' correct sharings. Since M is hyper-invertible, any other sharing can be written as an affine combination of these correct sharings. Any affine combination of correct (d, d')-sharings is also a correct (d, d')-sharing, hence all involved sharings are correct.

Secrecy: The values known to the adversary are at most: $t'_a + t'_p$ values s_i (those chosen by passively or actively corrupted parties) and $\min(2t'_a, t'_a + t'_p) = t'_a + \min(t'_a, t'_p)$ values r_i (those possibly revealed to such parties). With these $2t'_a + t'_p + \min(t'_a, t'_p)$ values fixed, there is a bijective mapping between the actual $l \leq n' - 2t'_a - t'_p - \min(t'_a, t'_p)$ values r_i whose sharings are generated by the protocol and any other l values s_i, generated by honest or only fail-stop corrupted parties. Therefore, the values contained in the generated sharings are uniform random and independent of the view of the adversary.

Completeness: If all parties behave according to the protocol, correctness of DoubleShareRandom is trivial and security follows from the above argument.

Complexity: In each bucket the parties communicate $O(n'^2\kappa)$ bits, which follows by inspection of the protocol. Moreover, each bucket (but the last one) contains $n' - 2t'_a - t'_p - \min(t'_a, t'_p)$ double-sharings, which can be seen to be at least $\frac{1}{5}n'$ [13], hence, there are at most $5\ell/n' + 1$ buckets. Therefore, the overall communication complexity of DoubleShareRandom is $O((5\ell/n' + 1)n'^2\kappa) = O(\ell n'\kappa + n'^2\kappa)$. ◀

Detectably Generating Multiplication Triples

The goal of the protocol GenerateTriples is to detectably generate a number of random multiplication triples, shared in an arbitrary degree d, assuming that $3t'_a + 2t'_p + t'_f < n'$. The idea is that GenerateTriples invokes DoubleShareRandom, in order to generate random double-sharings $[a_i]_{d,d'}, [b_i]_{d,d'}$ and $[r_i]_{d,2d'}$, where $d' = t'_a + t'_p$. The parties then use the d'-sharings of a_i and b_i, and the $2d'$-sharings of r_i to locally compute $2d'$-sharings of the blinded products $e_i = a_i b_i - r_i$ as $[e_i]_{2d'} = [a_i]_{d'} [b_i]_{d'} - [r_i]_{2d'}$. These blinded products are then publicly reconstructed using RecPub and the d-sharings of the products c_i are computed as $[c_i]_d = [r_i]_d + e_i$, using the d-sharings of r_i.

[13] If we set $m = 3t'_a + 2t'_p$, then $t'_a \geq m/5$ or $t'_p \geq m/5$. It follows that $n' - 2t'_a - t'_p - \min(t'_a, t'_p) = n' - m + t'_a + t'_p - \min(t'_a, t'_p) = n' - m + \max(t'_a, t'_p) + \min(t'_a, t'_p) - \min(t'_a, t'_p) \geq n'/5$.

Protocol GenerateTriples$(d, \ell, (\mathcal{P}', t'_a, t'_p, t'_f))$

1: The parties invoke DoubleShareRandom$(d, d', 2\ell, (\mathcal{P}', t'_a, t'_p, t'_f))$ and DoubleShareRandom$(d, 2d', \ell, (\mathcal{P}', t'_a, t'_p, t'_f))$, where $d' = t'_a + t'_p$, to generate random double sharings $[a_1]_{d,d'}, \ldots, [a_\ell]_{d,d'}$, $[b_1]_{d,d'}, \ldots, [b_\ell]_{d,d'}$ and $[r_1]_{d,2d'}, \ldots, [r_\ell]_{d,2d'}$.
2: For $i \in \{1, \ldots, \ell\}$, the parties compute locally a $2d'$-sharing of $e_i = a_i b_i - r_i$ as $[c_i]_{2d'} = [a_i]_{d'} [b_i]_{d'} - [r_i]_{2d'}$.
3: The parties invoke RecPub$(2d', \ell, (\mathcal{P}', t'_a, t'_p, t'_f), [e_1]_{2d'}, \ldots, [e_\ell]_{2d'})$.
4: For $i \in \{1, \ldots, \ell\}$, the parties compute locally a d-sharing of $c_i = a_i b_i$ as $[c_i]_d = [r_i]_d + e_i$.
5: Output the ℓ triples $([a_1]_d, [b_1]_d, [c_1]_d), \ldots, ([a_\ell]_d, [b_\ell]_d, [c_\ell]_d)$.

▶ **Lemma 7.** *Assuming that $(\mathcal{P}', t'_a, t'_p, t'_f)$ is valid and the values inputted by the parties are consistent, GenerateTriples detectably generates ℓ triples of sharings, where each sharing is a correct d-sharing, and for each triple $([a]_d, [b]_d, [c]_d)$, the shared values a and b are uniformly random given the view of the adversary, and $c = ab$. The communication complexity of GenerateTriples is $O(\ell n' \kappa + n'^2 \kappa)$.*

Proof. Since $3t'_a + 2t'_p + t'_f < n'$, we get that the degree $2d'$ of the sharings used in Step 3 fulfills $2d' = 2t'_a + 2t'_p < n' - t'_a$. Hence, correctness and secrecy of outputted triples follow from the correctness of RecPub and 6. The complexity follows by inspection. ◀

Realizing $\mathcal{F}_{\text{triples}}$

The following protocol Triples realizes $\mathcal{F}_{\text{triples}}$.

Protocol Triples $(\ell, (\mathcal{P}', t'_a, t'_p, t'_f))$

1: Every party in $\mathcal{P}'$ sets its happy-bit to "happy".
▷ Detectable Computation
2: The parties invoke GenerateTriples$(d, \ell, (\mathcal{P}', t'_a, t'_p, t'_f))$ for $d = t_a + t_p$.
▷ Fault Detection
3: The parties invoke FaultDetect. If, as a result, they are all happy, they output the generated triples. Otherwise, they do the following.
▷ Fault Localization
4: Let P_r be the party with the smallest index in $\mathcal{P}'$. Every $P_i \in \mathcal{P}'$ sends to P_r the randomness R_i it used in the two previous steps and the messages M_i it received in those steps. If P_r does not receive values from some parties, it uses instead the default values and proceeds.[14]
5: For each P_i, P_r reproduces all messages that P_i should have sent, using R_i and M_i, and, for each P_j, compares them with the messages P_j claims to have received from P_i (as specified in M_j). Then, P_r broadcasts a tuple (l, P_i, P_j, x, x'), such that the l^{th} message P_j claims to have received from P_i was x', while according to P_i it should have been x.
6: P_i broadcasts whether it agrees with P_r (i.e., whether the l^{th} message it sent to P_j was x). P_j broadcasts whether it agrees with P_r (i.e., whether if the l^{th} message it received from P_i was x').
7: If P_i disagrees, every party sets $E = \{P_i, P_r\}$. If P_j disagrees, every party sets $E = \{P_j, P_r\}$. Otherwise, every party sets $E = \{P_i, P_j\}$.
8: For every $P_h \in \{P_i, P_j, P_r\}$, the parties in $\mathcal{P}'$ invoke Heartbeat(P_h).
9: If the output of every invocation of Heartbeat is "alive", output E. Otherwise, output the set of parties, for whom the output of Heartbeat was "crashed".

[14] If P_r does not get the values from some party P_i, it could, instead of ignoring it, broadcast the index of such party. However, such solution would make the description of the protocol more involved.

▶ **Theorem 8.** *Assuming that* $3t_a + 2t_p + t_f < n$*, the protocol* Triples *securely evaluates* $\mathcal{F}_{\mathsf{triples}}$ *in the presence of a static mixed adversary.*

Proof. We present the simulator $\mathcal{S}_{\mathsf{triples}}$. Roughly, since there are no private inputs to the protocol (only private outputs), $\mathcal{S}_{\mathsf{triples}}$ simulates the execution towards the adversary $\mathcal{A}$ by executing the protocol. The key point is to make sure that the outputs are distributed correctly.

Simulator $\mathcal{S}_{\mathsf{triples}}$

The simulator has black-box access to the adversary $\mathcal{A}$. It outputs whatever $\mathcal{A}$ outputs.

1: Receive $(\textsc{Ok}, \ell, \mathcal{P}', t'_a, t'_p, t'_f)$ from $\mathcal{F}_{\mathsf{triples}}$.
2: Execute $\mathsf{GenerateTriples}(d, \ell, (\mathcal{P}', t'_a, t'_p, t'_f))$ and $\mathsf{FaultDetect}$ on behalf of the correct parties in $\mathcal{P}' \setminus \mathcal{P}_a$: in each round, send the messages generated by the protocol to $\mathcal{A}$, receive the messages from corrupted parties and information about crashes, and compute the next messages accordingly.
3: If the parties do not agree on the output of $\mathsf{FaultDetect}$, abort.
4: Otherwise, if this output is "happy", do as follows: For each triple $k = 1 \dots \ell$, compute the shares of the corrupted parties in $\mathcal{P}'$ that would result from the protocol (this is fully determined by the exchanged messages) and choose random shares for the corrupted parties not in $\mathcal{P}'$. Send these shares, together with the command $\textsc{Triples}$, to $\mathcal{F}_{\mathsf{triples}}$.
5: Otherwise (i.e., the output is "unhappy"), continue by executing fault localization. If the correct parties do not agree on the resulting set E, abort. Otherwise, send to $\mathcal{F}_{\mathsf{triples}}$ $(\textsc{ActiveSet}, E)$ or $(\textsc{CrashSet}, E)$, depending on the result of fault localization.

Assume that the honest parties input consistent values ℓ, $\mathcal{P}'$, t'_a, t'_p and t'_f, and that the set $\mathcal{P}'$ is valid (otherwise, the simulation is trivial). This means that the preconditions for Lemmas 3, 4 and 7 are met. By consistency of $\mathsf{FaultDetect}$ (Lemma 3), $\mathcal{S}_{\mathsf{triples}}$ does not abort in Step 3. Then, $\mathcal{S}_{\mathsf{triples}}$ sends the command $\textsc{Triples}$ if and only if the result of $\mathsf{FaultDetect}$ is "happy", which is the same as in the real world. Hence, we can consider two cases: (1) the simulator sends $\textsc{Triples}$, and (2) he sends $\textsc{ActiveSet}$ or $\textsc{CrashSet}$.

Case Triples. Consider any triple and the corresponding sharing polynomials g_a, g_b and g_c. In the real world, by correctness of $\mathsf{FaultDetect}$, no party is crashed and no correct party is unhappy at the end of $\mathsf{GenerateTriples}$. Therefore, the secrecy stated in Lemma 7 guarantees that the free coefficients a and b of g_a and g_b are uniformly random and independent of the view of the adversary. Moreover, correctness stated in Lemma 7 guarantees that the free coefficient of g_c is $c = ab$. Therefore, the free coefficients are distributed identically as in the ideal world.

By correctness (Lemma 7), the degree of all polynomials is at most d. It is easy to see that in both worlds the polynomials are uniformly random among those consistent with the shares that would be outputted by the corrupted parties and with the free coefficients distributed as above.

Case ActiveSet or CrashSet. In this case it is enough to argue correctness (note that here the parties have no secret inputs and the ideal functionality is deterministic). The set E sent by $\mathcal{S}_{\mathsf{triples}}$ is the same as in the protocol.

The simulator sends $\textsc{CrashSet}$ if and only if the output of $\mathsf{Heartbeat}$ for some parties in $\mathcal{P}'$ was not "alive". In this case, E contains no correct party, by completeness of $\mathsf{Heartbeat}$ (Lemma 4).

Now consider the other case, where $\mathcal{S}_{\mathsf{triples}}$ sends $\textsc{ActiveSet}$. By correctness of $\mathsf{Heartbeat}$, none of P_i, P_j and P_k was crashed at the end of Step 7. It follows that if in Step 4 P_r received no messages from P_i (or P_j), then P_i (or P_j) is malicious and could have sent the

default values taken by P_r anyway. Consider how the set E is chosen in Step 7. If P_i (or P_j) disagrees, then clearly it makes a conflicting claim with P_r. On the other hand, if none of P_i and P_j disagrees with P_r, then one of P_i and P_j must be malicious, because they disagree on the message sent from P_i to P_j. Hence, E contains two parties making conflicting claims. One of these parties must be actively corrupted. ◀

7.5 Complexity

Consider first the complexity of the protocol Triples. By Lemmas 7 and 3, executing GenerateTriples and FaultDetect requires communicating $O(\ell n\kappa + n^2\kappa + \mathcal{BA}(1))$ bits. For the complexity of fault localization, observe that the total number of bits needed to send the messages M_i is exactly the communication complexity of GenerateTriples and FaultDetect, and that, asymptotically, sending the values R_i does not add to this complexity. Together with broadcasts and Heartbeat, this results in $O(\ell n\kappa + n^2\kappa + \mathcal{BA}(\kappa))$ bits for the whole protocol Triples.

For the final protocol PreparationPhase, notice that the adversary can make the parties repeat a segment at most $t_a + t_f$ times. This means that Triples is executed at most $2(t_a + t_f)$ times. Hence, the protcol PreparationPhase communicates PreparationPhase is $O(Ln\kappa + (t_a + t_f)(n^2\kappa + \mathcal{BA}(\kappa)))$, which amounts to $O(Ln\kappa + n^3\kappa)$.

8 Evaluating Any Circuit

In the full version of this paper [37] we show how to use the preprocessing functionality constructed in the previous section to evaluate any circuit. In order to do so, we first define a multiplication functionality $\mathcal{F}_{\mathsf{mult}}$, which can be used to evaluate a number of multiplication gates on the same depth. $\mathcal{F}_{\mathsf{mult}}$ expects as input a number of shared multiplication triples. We then construct a protocol that realizes the circuit-evaluation functionality $\mathcal{F}_{\mathsf{mpc}}$ in the $(\mathcal{F}_{\mathsf{prepPhase}}, \mathcal{F}_{\mathsf{mult}})$-hybrid model. Finally, we show how to realize $\mathcal{F}_{\mathsf{mult}}$, using circuit randomization, the detectable public reconstruction protocol RecPub and player elimination.

The final result is the following functionality $\mathcal{F}_{\mathsf{mpc}}$. We note that since the adversary can always prevent fail-corrupted parties from giving input, $\mathcal{F}_{\mathsf{mpc}}$ is defined on an extended input domain, where a party can input $\perp$, meaning that it gives no input. $\mathcal{F}_{\mathsf{mpc}}$ replaces the inputs $\perp$ by default inputs 0 and sends to all parties the set of all parties who inputted $\perp$ (this way, parties know what circuit was actually evaluated).

Functionality $\mathcal{F}_{\mathsf{mpc}}$

1: Receive from each party a circuit C with a topological order on the gates and the party's inputs to C. (Execute Complete Break Down if different values C are received, or if C cannot be evaluated.)
2: For each party whose input was $\perp$, set all inputs to C to 0.
3: Evaluate C and send outputs to the corresponding parties. Send to all parties the set of parties who inputted $\perp$.

9 Conclusions

In this paper we present a perfectly-secure protocol which allows to securely evaluate any circuit in the presence of a mixed adversary. Our protocol has linear communication complexity per multiplication gate. Furthermore, our protocol is secure as long as $3t_a +$

$2t_p + t_f < n$, which is optimal. Previous results for specific types of corruption can be seen special cases of our result: we immediately get a linear protocol for the active setting (with $t_p = t_f = 0$ and $3t_a < n$), a linear protocol for the passive setting (with $t_a = t_f = 0$ and $2t_p < n$) and a linear protocol for a fail-stop adversary (for $t_f < n$). Moreover, our result is much more general and implies protocols with linear complexity for any combination of these settings. For example, we achieve a protocol, where overall 2/3 of the parties are corrupted.

Furthermore, we present a precise, simulation-based proof of security. As a special case, this also yields the first actively-secure protocol with linear complexity and a sound simulation-based proof (note the the proof in [8] is property-based).

Universal Composition

Katz et el. [41] formalize synchronous computation with secure channels and guaranteed termination in the UC framework [13] and prove that any protocol that realizes a functionality F in the stand-alone model and has a straightline black-box simulator, immediately yields a protocol (with one additional synchronization round) that UC-realizes F. We note that they consider the active-only setting, so their result cannot be readily applied to our protocol for the mixed setting. However, we believe that with minor extensions, a similar result can be phrased for our formulation of the mixed setting. We remark that we chose to use the stand-alone model, since it naturally models the setting of synchronous computation over secure channels, which results in simpler proofs. The communication in the UC framework, on the other hand, is asynchronous and over insecure channels. Hence, to prove our protocol secure in the UC framework, we would need techniques similar to those of [41], where the authors use the hybrid model with clock and secure channels functionalities.

Future Work

An interesting direction for future work is to consider efficient protocols tolerating a mixed adversary in the cryptographic and in the statistically-secure setting (hence, with larger thresholds). Another important problem is to construct a protocol for the mixed setting, that does not have the additive factor $D \cdot p(n)$ in the communication complexity. This can potentially be achieved by combining our techniques with those of [33].

References

1 Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 30(1):58–151, January 2017. doi:10.1007/s00145-015-9214-4.

2 B. V. Ashwinkumar, Arpita Patra, Ashish Choudhary, Kannan Srinathan, and C. Pandu Rangan. On tradeoff between network connectivity, phase complexity and communication complexity of reliable communication tolerating mixed adversary. In Rida A. Bazzi and Boaz Patt-Shamir, editors, *27th ACM PODC*, pages 115–124. ACM, August 2008. doi:10.1145/1400751.1400768.

3 Saikrishna Badrinarayanan, Aayush Jain, Nathan Manohar, and Amit Sahai. Secure MPC: Laziness leads to GOD. Cryptology ePrint Archive, Report 2018/580, 2018. URL: https://eprint.iacr.org/2018/580.

4 Assi Barak, Martin Hirt, Lior Koskas, and Yehuda Lindell. An end-to-end system for large scale P2P MPC-as-a-service and low-bandwidth MPC for weak participants. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 695–712. ACM Press, October 2018. doi:10.1145/3243734.3243801.

5 Donald Beaver. Multiparty protocols tolerating half faulty processors. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 560–572. Springer, Heidelberg, August 1990. doi:10.1007/0-387-34805-0_49.

6 Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 420–432. Springer, Heidelberg, August 1992. doi:10.1007/3-540-46766-1_34.

7 Zuzana Beerliová-Trubíniová, Matthias Fitzi, Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE: Perfect security in a unified corruption model. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 231–250. Springer, Heidelberg, March 2008. doi:10.1007/978-3-540-78524-8_14.

8 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.

9 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.

10 Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 663–680. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_39.

11 Piotr Berman, Juan A Garay, and Kenneth J Perry. Bit optimal distributed consensus. In *Computer science*, pages 313–321. Springer, 1992.

12 Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000. doi:10.1007/s001459910006.

13 Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. doi:10.1109/SFCS.2001.959888.

14 David Chaum. The spymasters double-agent problem: Multiparty computations secure unconditionally from minorities and cryptographically from majorities. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 591–602. Springer, Heidelberg, August 1990. doi:10.1007/0-387-34805-0_52.

15 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.

16 Ashish Choudhary, Arpita Patra, B. V. Ashwinkumar, K. Srinathan, and C. Pandu Rangan. Perfectly reliable and secure communication tolerating static and mobile mixed adversary. In Reihaneh Safavi-Naini, editor, *ICITS 08*, volume 5155 of *LNCS*, pages 137–155. Springer, Heidelberg, August 2008. doi:10.1007/978-3-540-85093-9_15.

17 Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 466–485. Springer, Heidelberg, December 2014. doi:10.1007/978-3-662-45608-8_25.

18 Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015. doi:10.1017/CBO9781107337756.

19 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.

20 Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 445–465. Springer, Heidelberg, May / June 2010. doi:10.1007/978-3-642-13190-5_23.

21 Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013*, volume 8134 of *LNCS*, pages 1–18. Springer, Heidelberg, September 2013. doi:10.1007/978-3-642-40203-6_1.

22 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.

23 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.

24 Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_38.

25 Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, January 1993.

26 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.

27 Juan A. Garay and Kenneth J. Perry. A continuum of failure models for distributed computing. In Adrian Segall and Shmuel Zaks, editors, *Distributed Algorithms: 6th International Workshop, WDAG '92 Haifa, Israel*, pages 153–165, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

28 Daniel Genkin, Yuval Ishai, and Antigoni Polychroniadou. Efficient multi-party computation: From passive to active security via secure SIMD circuits. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 721–741. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7_35.

29 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.

30 Hossein Ghodosi and Josef Pieprzyk. Multi-party computation with omnipresent adversary. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 180–195. Springer, Heidelberg, March 2009. doi:10.1007/978-3-642-00468-1_11.

31 Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

32 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.

33 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.

34 Martin Hirt, Christoph Lucas, Ueli Maurer, and Dominik Raub. Graceful degradation in multi-party computation (extended abstract). In Serge Fehr, editor, *ICITS 11*, volume 6673 of *LNCS*, pages 163–180. Springer, Heidelberg, May 2011. doi:10.1007/978-3-642-20728-0_15.

35 Martin Hirt, Ueli M. Maurer, and Bartosz Przydatek. Efficient secure multi-party computation. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 143–161. Springer, Heidelberg, December 2000. doi:10.1007/3-540-44448-3_12.

36 Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE: Unconditional and computational security. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 1–18. Springer, Heidelberg, December 2008. doi:10.1007/978-3-540-89255-7_1.

37 Martin Hirt and Marta Mularczyk. Efficient MPC with a Mixed Adversary. Cryptology ePrint Archive, Report 2020/356, 2020. (full version of this paper). URL: https://eprint.iacr.org/2020/356.

38 Martin Hirt and Jesper Buus Nielsen. Robust multiparty computation with linear communication complexity. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 463–482. Springer, Heidelberg, August 2006. doi:10.1007/11818175_28.

39 Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 483–500. Springer, Heidelberg, August 2006. doi:10.1007/11818175_29.

40 Jonathan Katz. On achieving the "best of both worlds" in secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 11–20. ACM Press, June 2007. doi:10.1145/1250790.1250793.

41 Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 477–498. Springer, Heidelberg, March 2013. doi:10.1007/978-3-642-36594-2_27.

42 Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 830–842. ACM Press, October 2016. doi:10.1145/2976749.2978357.

43 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.

44 Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

45 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.

Practical Relativistic Zero-Knowledge for NP

Claude Crépeau
School of Computer Science, McGill University, Montréal, Québec, Canada
crepeau@cs.mcgill.ca

Arnaud Y. Massenet
École Normale Supérieure Paris-Saclay, Gif-sur-Yvette, France
arnaudyoh@gmail.com

Louis Salvail
Département d'Informatique et de R.O., Université de Montréal, Montréal, Québec, Canada
salvail@iro.umontreal.ca

Lucas Shigeru Stinchcombe
Bloomberg L.P., Tokyo, Japan
lucas.stinchcombe@mail.mcgill.ca

Nan Yang
Canadian Centre for Cyber Security, Ottawa, Ontario, Canada
Concordia University, Montréal, Québec, Canada
nan.yang@cyber.gc.ca

Abstract

In a Multi-Prover environment, how little spatial separation is sufficient to assert the validity of an **NP** statement in Perfect Zero-Knowledge ? We exhibit a set of two novel Zero-Knowledge protocols for the 3-COLorability problem that use two (*local*) provers or three (*entangled*) provers and only require exchanging one edge and two bits with two trits per prover. This greatly improves the ability to prove Zero-Knowledge statements on very short distances with very basic communication gear.

2012 ACM Subject Classification Theory of computation → Quantum information theory

Keywords and phrases Multi-Prover Interactive Proofs, Relativistic Commitments, 3-COLorability, Quantum Entanglement, Non-Locality

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.4

Funding *Claude Crépeau*: Supported in part by Québec's FRQNT (INTRIQ) and Canada's NSERC. *Louis Salvail*: Supported in part by an NSERC discovery grant and acceleration to discovery grant. *Nan Yang*: This work was completed at Concordia University and supported in part by Professors Jeremy Clark, and Claude Crépeau.

Acknowledgements We would like to thank P. Alikhani, N. Brunner, A. Chailloux, S. Designolle, A. Leverrier, W. Shi, T. Vidick, and H. Zbinden for various discussions about earlier versions of this work. We would also like to thank Jeremy Clark for his insightful comments. We are grateful to ITC's reviewers for several useful comments and corrections. Most of these have been implemented.

1 Introduction

The idea of using distance and special relativity (a theory of motion justifying that the speed of light is a sort of asymptote for displacement) to prevent communication between participants to multi-prover proof systems can be traced back to Kilian[16]. Probably, the original authors (Ben Or, Goldwasser, Kilian and Wigderson) of [1] had that in mind already, but it is not explicitly written anywhere. Kent was the first author to venture into *sustainable* relativistic commitments [15] and introduced the idea of arbitrarily prolonging their life span by playing

some ping-pong protocol between the provers (near the speed of light). This idea was made considerably more practical by Lunghi et al. in [19] who made commitment sustainability much more efficient. This culminated into an actual implementation by Verbanis et al. in [23] where commitments were sustained for more than a day!

As nice as this may sound, such *long-lasting* commitments have found so far very little practical use. Consider for instance the zero-knowledge proof for Hamiltonian Cycle as introduced by Chailloux and Leverrier[3]. Proving in Zero-Knowledge that a 500-vertex graph contains a Hamiltonian cycle would require transmitting 250 000 bit commitments (each of a couple hundreds of bits in length) and eventually sustaining them before the verifier can announce his choice of unveiling the whole adjacency matrix or just the Hamiltonian cycle. For a graph of $|V|$ vertices, this would require an estimated $200|V|^2$ bits of communication before the verifier can announce his choice *chall* (see Fig. 1). This makes the application prohibitively expensive. If you use a larger graph, you will need more time to commit, leading to more distance to implement the protocol of [3]. Either a huge separation is necessary between the provers (so that one of them can unveil according to the verifier's choice *chall* before he finds out the committal information B used by the other prover while the former must commit all the necessary information before he can find out the verifier's choice *chall*) or we must achieve extreme communication speeds between prover-verifier pairs. This would only be possible by vastly parallelizing communications between them at high cost. Modern (expensive) top-of-line communication equipment may reach throughputs of roughly 1Tbits/sec. A back of the envelope calculation estimates that the distance between the verifiers must be at least 100 km to transmit 250 000 commitments at such a rate.

In this work we consider the following problem: in a Multi-Prover environment, how little spatial separation is sufficient to assert the validity of an **NP** statement in Perfect Zero-Knowledge ? We exhibit a set of two novel Zero-Knowledge protocols for the 3-COLorability problem that use two (*local*) provers or three (*entangled*) provers and only require them to communicate two trits each after having each received an edge and two bits each from the verifier. This greatly improves the ability to prove Zero-Knowledge statements on very short distances with very minimal communication equipment. In comparison, the protocol of [3] would require transmitting millions of bits between a prover and his verifier before the latter may disclose what to unveil or not. This implies the provers would have to be very far from each other because all of these must reach the verifier *before* the formers can communicate.

Although certain algebraic zero-knowledge multi-prover interactive proofs for **NP** and **NEXP** using explicitly no commitments at all have been presented before in [18], [10] (sound against local provers) and [5],[12] (sound against entangled provers), in the local cases making these protocols entanglement sound is absolutely non-trivial, whereas in the entangled case the multi-round structure and the amount of communication in each round makes implementing the protocol completely impractical as well. (To their defense, the protocols were not designed to be *practical*).

The main technical tool we use in this work is a general Lemma of Kempe, Kobayashi, Matsumoto, Toner, and Vidick[13] to prove soundness of a three-prover protocol when the provers are *entangled* based on the fact that a two-prover protocol version is sound when the provers are only *local*. More precisely, they proved this when the three-prover version is the same as the two-prover version but augmented with an extra prover who is asked exactly the same questions as one of the other two at random and is expected to give the same exact answers.

Our protocols build on top of the earlier protocol due to Cleve, Høyer, Toner and Watrous[7] who presented an extremely simple and efficient solution to the 3-COL problem that uses only two provers, each of which is queried with either a vertex from a common edge,

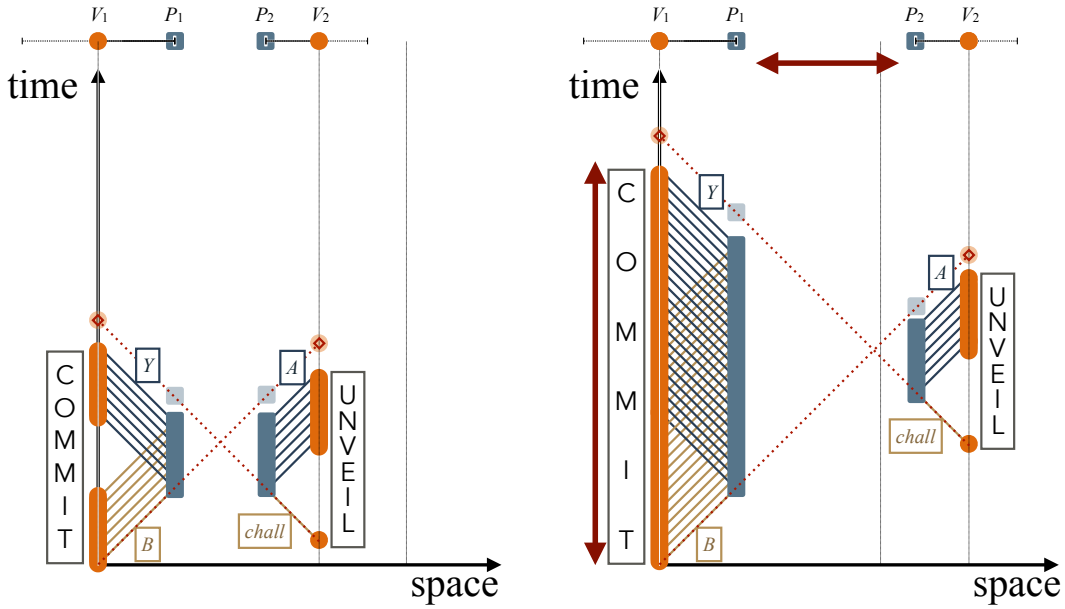


Figure 1 Space-Time diagrams of [3]'s ZK-MIP* for **NP**. (45° diagonals are the speed of light.) In the above two diagrams, V_1 at a first location sends a random matrix $\boxed{B}$ to P_1 who uses each entry to commit an entry of the adjacency matrix $\boxed{Y}$ of G. At another location, V_2 sends a random challenge $\boxed{chall}$ to P_2 who unveils all or some commitments as $\boxed{A}$. At all times, V_1 and V_2 must make sure that the answers they get from P_1 and P_2 come early enough that the direct communication line between V_1 and V_2 (even at the speed of light) is not crossed. The transition from left to right shows that increasing the number of vertices (and thus increasing the total commit time) pushes the verifiers further away from each other. In [3] the distance must increase quadratically with the number of vertices in the graph.

or twice the same vertex. In the former case, the verifier checks that the two ends of the selected edge are of distinct colours, while in the latter case, he check only that the provers answer the same colour given the same vertex. On the bright side, their protocol did not use commitments at all but unfortunately it did not provide Zero-Knowledge either. Moreover, it is a well established fact that this protocol cannot possibly be sound against entangled provers, because certain graph families have the property that they are not 3-colourable while having entangled-prover pairs capable of winning the game above with probability one. This was already known at the time when they introduced their protocol. The reason this protocol is not zero-knowledge follows from the undesirable fact that dishonest verifiers can discover the (random) colouring of non-edge pairs of vertices in the graph, revealing if they are of the same colour or not in the provers' colouring.

We are able to remedy to the zero-knowledge difficulty by allowing the provers to use commitments for the colour of their vertices. However they use these commitments in an innovative way that we call the *unveil-via-commit principle* (of independent interest) explained below. For this purpose we use commitments similar to those of Lunghi et al.[19] but in their simplest form possible, over the field $\mathbb{F}_3$ (or $\mathbb{F}_4$ if you insist working in binary), and thus with extremely weak binding property but also minimal in communication cost: a complete execution of the basic protocol transmits a question Q of exactly one edge number (using only $\log|E|$ bits) and two bits from verifiers to provers and an answer A of two trits back from the provers to verifiers (see Fig. 2). This implies that for a fixed communication

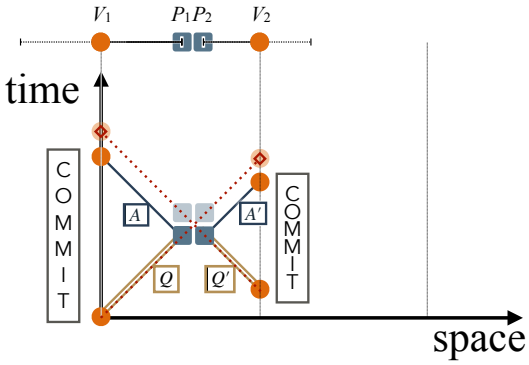


Figure 2 Space-Time diagram of our ZK-MIP* for **NP**. (45° diagonals are the speed of light.)

speed, the minimal distance of the provers in our protocol increases logarithmically with the number of vertices whereas the same parameter grows quadratically in [3]. Nevertheless, this is good enough to obtain a zero-knowledge version of the protocol that remains sound against *local* pairs of provers. The main idea being that the provers will each commit to the colours of two requested vertices only if they form an edge of the graph. To unveil the colour of any vertex, the verifiers must request commitment of the *same* vertex by *both* provers but using different randomizations. This way the verifiers may compute the colour of a vertex from the *linear system* established by the two commitments and not by explicitly requesting anyone to unveil. This is the unveil-via-commit principle (very similar to the double-spending detection mechanism of the untraceable electronic cash of Chaum, Fiat and Naor[4]). We then use the Lemma of [13] to prove soundness of the three-prover version of this protocol even when the provers are *entangled*. A positive side of the protocol of [3], however, is the fact that only two provers are necessary while we use three. Zero-Knowledge follows from the fact that only two edge vertices can be unveiled by requesting the same edge to both provers. Otherwise only a single vertex may be unveiled. Finally, we show that even the three-prover version of this protocol retains the zero-knowledge property: requesting any three edges from the provers may allow the dishonest verifiers to unveil the colours of a triangle in the graph but never two end-points that do not form an edge (going to four provers would however defeat the zero-knowledge aspect).

An actual physical implementation of this protocol is currently being developed.

1.1 Implementations Issues

Traditionally in the setup of Multi-Prover Interactive Proofs, there is a single verifier interacting with the many provers. However, when implementing no-communication via spatial separation (the so called relativistic setting) it is standard to break the verifier in a number of verifiers equal to the number of provers, each of them interacting at very short distance from their own prover. The verifiers can use the timing of the replies of their respective provers to judge their relative distance. In practice, this means that we can implement MIPs under relativistic assumptions if the verifier are "split" into multiple verifiers, each locally interacting with its corresponding prover. The verifiers use the distance between *themselves* to enforce the impossibility of the provers to communicate: no message from a verifier can be used to reply to another verifier faster than the speed of light *wherever the provers are located*.

Moreover, multi-prover interactive proof systems may have several rounds in addition to several provers. In general, protocols with several rounds may cause a threat to the inherent assumption that the provers are not allowed to communicate during the protocol's execution. Nevertheless, most of the existing literature resolves this issue by providing an honest verifier that is *non-adaptive*. To simplify this task, most of the protocols are actually single-round. We stick to these guidelines in this work. Moreover, in order to prove soundness of our protocols against entangled provers, we use a theorem that is currently only proven for single-round protocols. The protocols we describe are indeed single-round and non-adaptive.

2 Preliminaries

2.1 Notations

Random variables $A, B \in \Gamma$ are said to be equivalent, denoted $A = B$, if for all $x \in \Gamma$, $\Pr\{(A = x)\} = \Pr\{(B = x)\}$. The class of probabilistic polynomial-time Turing machines will be denoted PPT in the following. A PPT Turing machine is one having access to a fresh infinite read-only tape of random values (uniform values from the set of input symbols) at the outset of the computation. In the following, adversaries will also be allowed (in some cases) to be quantum machines. The precise ways quantum and classical machines are defined is not important in the following.

For M a Turing machine, we denote by $M(x)$ it execution with x on its input tape (x being a string of the tape alphabet symbols). A Turing machine (quantum or classical) augmented with read-only auxiliary-input tapes and write-only auxiliary-output tapes is called an *interactive Turing machine* (ITM). Read-only input tapes provide incoming messages while the write-only output tapes allow to send messages. Interactive Turing machine M_1 and M_2 are said to *interact* when for each of them, one of its write-only auxiliary-output tape corresponds to one read-only auxiliary-input tape of the other Turing machine. An execution of interactive Turing machines $M_1, \ldots, M_k$ on common input x is denoted $[M_1 \ldots M_k](x)$. For $1 \leq i \leq k$, machine M_i *accepts* the interactive computation on input x if it stops in state accept after the execution $[M_1 \ldots M_k](x)$. When the ITM M_i that accepts a computation is clear from the context, we say that $[M_1 \ldots M_k](x)$ accepts when M_i's final state is accept. In this scenario, $\Pr\{([M_1 \ldots M_k](x) = \mathsf{accept})\}$ denotes the probability that M_i terminates in state accept upon common input x. Quantum machines are also interacting through communication tapes the same way than for classical machines. When a quantum machine M_1 interacts with a classical machine M_2, we suppose that the write-only auxiliary tape and the reade-only auxiliary tape of M_1 used to communicate with M_2 are classical. This is the situation we will be addressing almost all the time in the following. A quantum machine M is also allowed to have a quantum auxiliary read-only input tape that may contain a part of a quantum state shared with other machines. This allows to model machines sharing entanglement at the outset of an interactive computation. Henceforth, we suppose that the (main) input tape of all machines (quantum or classical) is classical.

In the following, $G = (V, E)$ denotes an undirected graph with vertices V and edges E. If $n = |V|$ then we denote the set of vertices in G by $V = \{1, 2, \ldots, n\}$. We suppose that $(i, i) \notin E$ for all $1 \leq i \leq n$ (i.e. G has no loop). We denote uniquely each edge in E as (i, j) with $j > i$. For $i \in V$, let $\mathsf{Edges}(i) := \{(j, i) \in E\}_{j<i} \cup \{(i, j) \in E\}_{j>i}$ be the set of edges connecting vertex i in G. For $e, e' \in E$, we define $e \cap e' = i \in V$ if e and e' have only one vertex $i \in V$ in common. When e and e' have four distinct vertices in V, we set $e \cap e' = 0$. Finally, when $e = e'$, we set $e \cap e' := \infty$. For readability, we use the following special notations: $(a, b) \neq\!\!\!\!\!\!\cdot\;\; (c, d)$ means $a \neq c$ **and** $b \neq d$, while as always, $(a, b) \neq (c, d)$ simply means $a \neq c$ **or** $b \neq d$.

2.2 Non-local Games, and Relativistic Multi-Prover Interactive Proofs

Multi-provers interactive protocols are protocols involving a set of *provers* modelled by interactive Turing machines, each of them interacting with an interactive PPT Turing machine called the verifier V. Although all provers may share an infinite read-only auxiliary input tape at the outset of their computation, they do not interact with each other. When the provers are quantum, an extra auxiliary read-only quantum input tape is given and can be entangled with other provers at the beginning.

▶ **Definition 1.** *Let* $\mathsf{P}_1, \ldots, \mathsf{P}_k$ *be computationally unbounded interactive Turing machines and let* V *be an interactive* PPT *Turing machine. The* P_i*'s share a joint, infinitely long, read-only random tape (and an auxiliary reads-only quantum input tape if the provers are quantum). Each* P_i *interacts with* V *but cannot interact with* P_j *for any* $1 \leq j \neq i \leq k$*. We call* $[\mathsf{P}_1, \ldots, \mathsf{P}_k, \mathsf{V}]$ *a* k-prover interactive protocol (k–prover IP).

A $[\mathsf{P}_1, \ldots, \mathsf{P}_k, \mathsf{V}]$ k-prover interactive protocol is a *multi-prover interactive proof system for* L if it can be used to show V that a public input x is such that $x \in L$. At the end of its computation, V concludes $x \in L$ if and only if it ends up in state accept. We restrict our attention to interactive proof systems with perfect completeness since all our protocols have this property.

▶ **Definition 2.** *The* k*–prover interactive protocol* $\Pi = (\mathsf{P}_1, \ldots, \mathsf{P}_k, \mathsf{V})$ *is said to be a* k-prover interactive proof system with perfect completeness for L *if there exists* $q(n) < 1 - \frac{1}{\mathsf{poly}(n)}$ *such that following holds:*
perfect completeness: $(\forall x \in L)\left[\Pr\{\left([\mathsf{P}_1, \ldots, \mathsf{P}_k, \mathsf{V}](x) = \mathsf{accept}\right)\} = 1\right]$,
soundness: $(\forall x \notin L)(\forall \widetilde{\mathsf{P}}_1, \ldots, \widetilde{\mathsf{P}}_k)\left[\Pr\left\{\left([\widetilde{\mathsf{P}}_1, \ldots, \widetilde{\mathsf{P}}_k, \mathsf{V}](x) = \mathsf{accept}\right)\right\} \leq q(|x|)\right]$.
The parameter $q(|x|)$ *is called the* soundness error *of* Π*. Soundness can hold against classical provers or against quantum provers sharing entanglement. The former case is called* sound against classical provers *while to latter is called* sound against entangled provers.

Consider a k–prover interactive proof system $\Pi(x)$ (with or without perfect completeness) for L executed with public input $x \notin L$. In this situation, $\Pi(x)$ defines a so-called *quantum game*. The minimum $q(|x|)$ such that for all $\mathsf{P}'_1, \ldots, \mathsf{P}'_k$, $\Pr\{\left([\mathsf{P}'_1, \ldots, \mathsf{P}'_k, \mathsf{V}](x) = \mathit{accept}\right)\} \leq q(|x|)$ is called the *classical value of game* $\Pi[x]$ and is denoted $\omega(\Pi(x))$ when the provers are restricted to be classical and unable to communicate with each other upon public input x. When the provers, still unable to communicate with each other, are allowed to carry their computation quantumly and share entanglement, we denote by $\omega^*(\Pi(x)) \geq \omega(\Pi(x))$ the minimum $q(|x|)$ such that for all such quantum provers $\mathsf{P}'_1, \ldots, \mathsf{P}'_k$, $\Pr\{\left([\mathsf{P}'_1, \ldots, \mathsf{P}'_k, \mathsf{V}](x) = \mathit{accept}\right)\} \leq q(|x|)$. In this case, $\omega^*(\Pi(x))$ is called the *quantum value of game* $\Pi(x)$. A k–prover interactive proof system for L is said to be *symmetric* if V can permute the questions to all provers without changing their distribution.

The following result of Kempe, Kobayashi, Matsumoto, Toner, and Vidick[13] shows that the classical value of a symmetric one-round classical game cannot be too far from the quantum value of a *modified* game. Given a symmetric one-round two-prover game Π, one can always add a third prover P_3 and V asks P_3 the same question than P_1 with probability $\frac{1}{2}$ or the same question than P_2 with probability $\frac{1}{2}$. Then, V accepts if P_1 and P_2 would be accepted in $\Pi(x)$ and if P_3 returns the same answer as the one issued by the prover it emulates. We call $\Pi'(x)$ the modified game obtained that way from $\Pi(x)$.

▶ **Lemma 3** ([13], Lemma 17)**.** *Let* $\Pi(x)$ *be a two-prover one-round symmetric game and let* $\Pi'(x)$ *be its modified version with three provers. If* $\omega^*(\Pi'(x)) > 1 - \varepsilon$ *then we always get* $\omega(\Pi(x)) > 1 - \varepsilon - 12|Q|\sqrt{\varepsilon}$ *where* Q *is the set of* V*'s possible questions to a prover in* Π*.*

Lemma 3 remains true for non-symmetric two-prover one-round protocol by first making them symmetric at the cost of increasing the size of Q. This is always possible without changing the classical value of the game and by using at most twice the number of questions $|Q|$ of the original game (Lemma 4 in [13]).

The Zero-knowledge [11] version of MIPs were also defined in [1]:

Let $[\mathsf{P}_1, \ldots, \mathsf{P}_k, \mathsf{V}]$ be a k–prover IP. We denote by $\mathbf{view}(\mathsf{P}_1, \ldots, \mathsf{P}_k, \mathsf{V}, x)$ the random variable of V's outgoing and incoming messages with all provers according V's coin tosses.

▶ **Definition 4.** *Let* $[\mathsf{P}_1, \ldots, \mathsf{P}_k, \mathsf{V}]$ *be a k-prover interactive proof system for L. We say that* $[\mathsf{P}_1, \ldots, \mathsf{P}_k, \mathsf{V}]$ *is* perfect zero-knowledge *if for all* PPT *interactive Turing machines* $\widetilde{\mathsf{V}}$ *there exists a* PPT *machine* Sim *(i.e. the* simulator*) having blackbox access to* $\widetilde{\mathsf{V}}$ *such that for all* $x \in L$*,*

$$\mathbf{view}(\mathsf{P}_1, \ldots, \mathsf{P}_k, \widetilde{\mathsf{V}}, x) = \mathsf{Sim}(x) \ ,$$

and both random variables are equivalent. In the following, we allow $\widetilde{\mathsf{V}}$ *to be a quantum machine but our simulators will always be classical machines with blackbox access to* $\widetilde{\mathsf{V}}$*. If the zero-knowledge condition holds against quantum* $\widetilde{\mathsf{V}}$*, we say that the proof system is* perfect zero-knowledge against quantum verifiers.

2.3 Multi-Prover Commitments with Implicit Unveiling

Our multi-prover proof systems for 3COL use a simple 2-committer commitment scheme with a property allowing to guarantee perfect zero-knowledge. In this section, we give the description of this simple commitment scheme with its important properties for our purposes.

Assume that provers P_1 and P_2 share ℓ values $c_1, c_2, \ldots, c_\ell \in \mathbb{F}$ where $\mathbb{F}$ is a finite field. V wants to check that these values satisfy some properties without revealing the specific values.

Bit commitment schemes have been used in the multi-prover model ever since it was introduced in [1]. The original scheme was basically $w_i := b_i \cdot r_i + c_i$, a commitment w_i to value $c_i \in \mathbb{F}$ using pre-agreed random mask $b_i \in_R \mathbb{F}$ and randomness $r_i \in \mathbb{F}^*$ provided by V. Kilian[17] had a binary version where each bit $c_i := c_i^1 \oplus c_i^2 \oplus c_i^3$ is shared among provers P_1 and P_2 (and therefore $\mathbb{F}$ needs only to be a group). To commit c_i, V samples c_i^h from P_1 and c_i^j from P_2 at random. If $j = h$ but $c_i^j \neq c_i^h$, V immediately rejects the commitment. Otherwise either P_1 or P_2 may unveil by disclosing c_i^1, c_i^2, c_i^3 at a later time. Somehow, Crépeau's bad recollection of the scheme in [1] lead [2] to a similar but different scheme defining $w_i := c_i \cdot r_i + b_i$, a commitment w_i to bit $c_i \in \{0, 1\}$ using pre-agreed bit mask $b_i \in_R \{0, 1\}$ and binary randomness r_i provided by their corresponding verifiers. Although this latter form of commitment is intimately connected to the CHSH game [6] and the Popescu-Rohrlich box[20], this proximity is not relevant for the soundness and the completeness of our protocols, even against entangled provers. While the binding property of the latter scheme has been established in [15, 8, 23, 9, 19, 3] against entangled provers, it is still not clear how to get sound and complete proof systems against such provers. We shall rather get completeness and soundness against entangled provers using a different technique from [13] that uses a third prover.

For an arbitrary field $\mathbb{F}$, the commitment scheme produces commitment $w_i := c_i \cdot r_i + b_i$ to field element $c_i \in \mathbb{F}$ using pre-agreed field element mask b_i (specific to value $1 \leq i \leq \ell$) and random field element $r_i \in \mathbb{F}^*$ provided by their corresponding verifiers. Many results were proven for this specific form of the commitments. Notice however that the two versions discussed above, $w_i := b_i \cdot r_i + c_i$ in the former case and $w_i := c_i \cdot r_i + b_i$ in the latter have equivalent binding property(left as a simple exercice). Considering, the former as being

the degree-one secret sharing [22] of c_i hidden in the degree zero term, while the latter being the degree-one secret sharing of c_i hidden in the degree one term, we decided to use the former (original BGKW form) because all the known results about secret sharing are generally presented in this form. In particular, this form is more adapted to higher degree generalizations such as $w_i := a_i \cdot r_i^2 + b_i \cdot r_i + c_i$ being the degree-two secret sharing of c_i hidden in the degree zero term, and so on.

Moreover, this choice turns out to simplify our (perfect) zero-knowledge simulator. For the rest of this paper, we use $w_i := b_i \cdot r_i + c_i$ where $w_i, b_i, c_i \in \mathbb{F}_3$ and $r_i \in \mathbb{F}_3^*$. Provers therefore commit to trits, one value for each vertex corresponding to its colour in a 3–colouring of graph $G = (V, E)$. The values shared between P_1 and P_2 are therefore, for each vertex $i \in V$, the colour c_i of that vertex and a vertex specific random mask b_i.

Suppose that V asks P_1 to commit on the colour c_i of vertex $i \in V$ using randomness $r \in_R \mathbb{F}_3^*$. Let $w = b_i \cdot r + c_i$ be the commitment returned to V by P_1. Suppose V asks P_2 to commit on the colour c_j of vertex $j \in V$ using randomness $r' \in_R \mathbb{F}_3^*$. Let $w' = b_j \cdot r' + c_j$ be the commitment issued to V by P_2. The following 3 cases are possible depending on V's choices for i, j, r, and r':

1. *(forever hiding)* if $i \neq j$ then V learns nothing on neither c_i nor c_j since w and w' hide c_i and c_j with random and independent masks $b_i \cdot r$ and $b_j \cdot r'$ respectively. Even knowing $r, r' \in \mathbb{F}_3^*$, $b_i \cdot r$ and $b_j \cdot r'$ are uniformly distributed in $\mathbb{F}_3$.
2. *(consistency testing)* If $i = j$ and $r = r'$ then V can verify that $w = w'$. This corresponds to the immediate rejection of V in Kilian's two-prover commitment described above. It allows V to make sure that P_1 and P_2 are consistent when asked to commit on the same value.
3. *(implicit unveiling)* If $i = j$ and $r' \neq r$ then V can learn c_i (assuming $w = b_i \cdot r + c_i$ and $w' = b_i \cdot r' + c_i$) the following way. V simply computes $c_i := -(w + w')$ (Note that over an arbitrary field $c_i := (wr' - w'r)(r' - r)^{-1}$ whenever $r \neq r'$). Interpreting the meaning of this test can be done when considering a strategy for P_1 and P_2 that always passes the consistency test. In this case, $w - b_i \cdot r = c_i = w' - b_i \cdot r'$ are satisfied and V learns c_i.

As long as P_1 and P_2 are *local* (or *quantum non-local*) they cannot distinguish which option V has picked among the three. The consistency test makes sure that if P_1 and P_2 do not commit on identical values for some $i \in V$ then V will detect it when V picks the consistency test for commitment w and w' in position i.

3 Classical Two-Prover Protocol

First, consider a small variation over the protocol of Cleve et al. presented in [7]. In their protocol, when P_1 and P_2 both know and act upon the same valid 3-colouring of G, V asks each prover for the colour of a vertex in $G = (V, E)$. Consistency is verified when V asks the same vertex to each prover and compares that the same colour has been provided. The colorability is checked when the provers are asked for the colour of two connected vertices in G. This way of proceeding is however problematic for the zero-knowledge condition. V could be asking two vertices that do not form an edge for which their respective colour will be unveiled. This certainly allows V to learn something about P_1's and P_2's colouring. Indeed, repeating this many times will allow V to efficiently reconstruct a complete colouring. To remedy partially this problem, V is instead asking each prover the colouring of an entire edge of G. The colouring is (only) checked when both provers are asked the same edge, while consistency is checked when two intersecting edges are asked to the provers.

3.1 Distribution of questions

Let $G = (V, E)$ be a connected undirected graph. Let us define the probability distribution $\mathcal{D}_G = \{(p(e, e'), (e, e'))\}_{e,e' \in E}$ for the pair $(e, e') \in E \times E$ that V picks with probability $p(e, e')$ before announcing e to P_1 and e' to P_2. For $e, e' \in E$ such that $e \cap e' = 0$, we set $p(e, e') := 0$ so that V never asks two disconnected edges in G (this would be useless).

The first thing to do is to pick $e = (i, j) \in E$ uniformly at random. With probability ϵ (to be selected later), we set $e' = e$, which allows for an edge-verification test. With probability $1-\epsilon$, we perform a well-definition test as follows. With probability $\frac{1}{2}$, $e' \in \mathsf{Edges}(i)$ uniformly at random and with probability $\frac{1}{2}$, $e' \in \mathsf{Edges}(j)$ uniformly at random. In other words, the well-definition test picks the second edge e' with probability $\frac{1}{2}$ among the edges connecting $i \in V$ and with probability $\frac{1}{2}$ among the edges connecting $j \in V$. It follows that for $e = (i, j) \in E$ and $e' \in (\mathsf{Edges}(i) \cup \mathsf{Edges}(j)) \setminus \{e\}$, we have

$$p(e, e') = \frac{1-\epsilon}{2|E|} \left(\frac{|\{e'\} \cap \mathsf{Edges}(i)|}{|\mathsf{Edges}(i)|} + \frac{|\{e'\} \cap \mathsf{Edges}(j)|}{|\mathsf{Edges}(j)|} \right) . \tag{1}$$

We also get

$$p(e, e) = \frac{\epsilon}{|E|} + \frac{1-\epsilon}{2|E|} \left(\frac{1}{|\mathsf{Edges}(i)|} + \frac{1}{|\mathsf{Edges}(j)|} \right) \geq \frac{\epsilon}{|E|} . \tag{2}$$

It is easy to verify that $\mathcal{D}_G$ is a properly defined probability distribution over pairs of edges.

3.2 A Variant Over the Two-Prover Protocol of Cleve et al.

Distribution $\mathcal{D}_G$ produces two edges where the first one is provided to P_1 while the second one is provided to P_2. Each prover then returns the colour of each vertex of the edge to V. We denote the resulting protocol $\Pi^{(2)}_{\text{std}}$.

Protocol $\Pi^{(2)}_{\text{std}}[G]$: Two-prover, 3-COL.
Provers $\mathsf{P}_1, \mathsf{P}_2$ pre-agree on a random 3-colouring of G:
$\{(i, c_i) | c_i \in \mathbb{F}_3\}_{i \in V}$ such that $(i, j) \in E \implies c_j \neq c_i$.
Interrogation phase:

- V picks $((i, j), (i', j')) \in_{\mathcal{D}_G} E \times E$, sends (i, j) to P_1 and (i', j') to P_2.
- If $(i, j) \in E$ then P_1 replies with c_i, c_j.
- If $(i', j') \in E$ then P_2 replies with $c_{i'}, c_{j'}$.

Check phase:

- **Edge-Verification Test:**
  if $(i, j) = (i', j')$ then V accepts iff $c_i = c_{i'} \neq c_{j'} = c_j$.
- **Well-Definition Test:**
  if $(i, j) \cap (i', j') = h \in V$ then V accepts iff $c_h = c'_h$.

The perfect soundness of this protocol is not difficult to establish along the same lines of the proof of soundness for the original protocol in [7]. On the other hand, zero-knowledge does not even hold against honest verifiers. V learns the colour of each vertex contained in any two edges of G. This is certainly information about the colouring that V learns after the interaction. To some extend, the modifications we applied to the 2-prover interactive proof system of [7] leaks even more to V. In the next section, we show that the 2-prover commitment scheme, that we introduced in Sect. 2.3, can be used in protocol $\Pi^{(2)}_{\text{std}}$ to prevent this leakage completely.

4 Perfect Zero-Knowledge Two-Prover Protocol

We modify the protocol of section 3.2 to prevent V from learning the colours of more than two connected vertices in G. The idea is simple, P_1 and P_2 will return commitments for the colours of the vertices asked by V. The implicit unveiling of the commitment scheme described in section 2.3 will allow V to perform both the edge-verification and well-definition tests in a very similar way that in protocol $\Pi^{(2)}_{\text{std}}$. The commitments require V to provide a random nonzero trit for each vertex of the edge requested to a prover.

4.1 Distribution of questions

We now define the probability distribution $\mathcal{D}'_G$ for V's questions in protocol $\Pi^{(2)}_{\text{lhv}}[G]$ defined in the following section. It consists in one edge and two nonzero trits for each prover:

$$\mathcal{D}'_G = \{(p'(e,r,s,e',r',s'),((e,r,s),(e',r',s'))\}_{e,e'\in E,r,s,r',s'\in\mathbb{F}_3^*}$$

upon graph $G=(V,E)$ and where (e,r,s) is the question to P_1 and (e',r',s') is the question to P_2. $\mathcal{D}'_G$ is easily derived from the distribution $\mathcal{D}_G$ for the questions in $\Pi^{(2)}_{\text{std}}[G]$, as defined in section 3.1. First, an edge $e \in_R E$ is picked uniformly at random. Together with e, two nonzero trits $r,s \in_R \mathbb{F}_3^*$ are picked at random. Then, as in $\mathcal{D}_G$, with probability ϵ (to be selected later) the second edge $e'=e$, in which case we always set $r'=-r$ and $s'=-s$. This case allows for an edge-verification test. Finally, with probability $1-\epsilon$, we pick e' with probability $p(e,e')|E|$ so that the couple $((e,r,s),(e',r,t))$ is produced with probability $\frac{1}{8}p(e,e')$ for all $e,e' \in E$, and $r,s,t \in \mathbb{F}_3^*$. This will allow for a well-definition test. A consequence of (1) is that for $e=(i,j)\in E$, $e' \in \mathsf{Edges}(i)\cup\mathsf{Edges}(j)$

$$p'(e,r,s,e',r,t) \geq \frac{1-\epsilon}{16|E|}\left(\frac{|\{e'\}\cap\mathsf{Edges}(i)|}{|\mathsf{Edges}(i)|}+\frac{|\{e'\}\cap\mathsf{Edges}(j)|}{|\mathsf{Edges}(j)|}\right) \quad , \tag{3}$$

where the inequality results from $e=e'$ being possible. According to (2), we also get

$$p'(e,r,s,e,-r,-s) = \frac{p(e,e)}{4} \geq \frac{\epsilon}{4|E|} \quad . \tag{4}$$

4.2 The Protocol

The protocol is similar to $\Pi^{(2)}_{\text{std}}$ except that instead of returning to V the colour for each vertex of an edge in G, each prover returns commitments with implicit unveiling of these colours. If V asks two disjoint edges then V learns nothing about the values committed by the *forever-hiding* property of the commitment scheme. The resulting 2–prover one-round interactive proof system is denoted $\Pi^{(2)}_{\text{lhv}}$.

Protocol $\Pi^{(2)}_{\text{lhv}}[G]$: Two-prover, 3-COL

P_1 and P_2 pre-agree on random masks $b_i \in_R \mathbb{F}_3$ for each $i \in V$ and a random 3-colouring of G: $\{(i, c_i)|c_i \in \mathbb{F}_3\}_{i\in V}$ such that $(i, j)\in E \implies c_j \neq c_i$.

Commit phase:

- V picks $(((i, j), r, s), ((i', j'), r', s')) \in_{\mathcal{D}'_G} \left(E \times (\mathbb{F}^*_3)^2\right)^2$.
- V sends $((i, j), r, s)$ to P_1 and $((i', j'), r', s')$ to P_2.
- If $(i, j) \in E$ then P_1 replies $w_i = b_i \cdot r + c_i$ and $w_j = b_j \cdot s + c_j$.
- If $(i', j') \in E$ then P_2 replies $w'_{i'} = b_{i'} \cdot r' + c_{i'}$ and $w'_{j'} = b_{j'} \cdot s' + c_{j'}$.

Check phase:

Edge-Verification Test:

- if $(i, j) = (i', j')$ and $(r', s') \not\equiv (r, s)$ then V accept iff $w_i + w'_i \neq w_j + w'_j$.

Well-Definition Test:

- If $(i, j) = (i', j')$ and $(r', s') = (r, s)$ then V accepts iff $(w_i = w'_i) \wedge (w_j = w'_j)$.
- if $(i, j) \cap (i', j') = i$ and $r' = r$ then V accepts iff $w_i = w'_i$.
- If $(i, j) \cap (i', j') = j$ and $s' = s$ then V accepts iff $w_j = w'_j$.

Clearly, $\Pi^{(2)}_{\text{lhv}}$ satisfies perfect completeness. The following theorem establishes that in addition to perfect completeness, $\Pi^{(2)}_{\text{lhv}}$ is sound against classical provers.

▶ **Theorem 5.** *The two-prover interactive proof system* $\Pi^{(2)}_{\text{lhv}}$ *is perfectly complete with classical value* $\omega(\Pi^{(2)}_{\text{lhv}}[G]) \leq 1 - \frac{1}{9|E|}$ *upon any graph* $G = (V, E) \notin \mathsf{3COL}$.

Proof. Assume $G \notin \mathsf{3COL}$ and let us consider the probability δ that V detects an error in the check phase when interacting with two local dishonest provers $\widetilde{\mathsf{P}}_1$ and $\widetilde{\mathsf{P}}_2$. $\Pi^{(2)}_{\text{lhv}}$ is a one-round protocol where the provers cannot communicate directly with each other nor through V's questions since they are independent of the provers' answers. It follows that the strategy of $\widetilde{\mathsf{P}}_1$ and $\widetilde{\mathsf{P}}_2$ can be made deterministic without damaging the soundness error by letting each prover choosing the answer that maximizes her/his probability of success given her/his question. Therefore, consider a deterministic strategy as a pair of arrays $W^\ell[i, r, j, s] \in \mathbb{F}_3^2$ to be used by prover $\widetilde{\mathsf{P}}_\ell$ for $\ell\in\{1, 2\}$ (note: we only care about the entries where $(i, j)\in E$ upon question $((i, j), r, s)$ with $i < j$. V can always present edges in the same order)). For $z \in \{1, 2\}$, $W^\ell_z[\cdot, \cdot, \cdot, \cdot]$ is the z-th component of the output pair $W^\ell[\cdot, \cdot, \cdot, \cdot]$. We say that $W[i, r]$ for $[i, r] \in E \times \mathbb{F}^*_3$ is *well defined* if for all j, k such that $(i, j), (i, k)\in E$ and $\forall s, t \in \mathbb{F}^*_3$, one of the following 4 equalities is true depending on which of $j > i$ or $j < i$, $k > i$ or $k < i$ is correct

$$W^1_1[i, r, j, s] = W^2_1[i, r, k, t] = W^1_2[j, s, i, r], \text{ or } W^1_1[i, r, j, s] = W^2_2[k, t, i, r] = W^2_2[k, t, i, r] \quad (5)$$

When $W[i, r]$ is well defined for all $i \in V, r \in \mathbb{F}^*_3$, we say that W is well defined.

We now lower bound the probability $\delta_{\text{wdt}} > 0$ that, when $W[i, r]$ is not well-defined for some $i \in V$ and $r \in \mathbb{F}^*_3$, the well-definition test will detect it. When (5) is not satisfied , w.l.o.g. we have $W^1_1[i, r, j, s] \neq W^2_1[i, r, k, t]$ for some $(i, j), (i, k) \in E$. The other three cases are treated similarly. Let $e = (i, j)$ and $e' = (i, k)$ be these two edges. According to (3) (and (1) when $e = e'$), the well-definition test will then detect an error with probability

$$\Pr\{(\mathsf{V} \text{ picks } e \text{ and } e' \text{ with randmoness } r, s, t)\} = p'(e, r, s, e', r, t) \geq \frac{1 - \epsilon}{16|E||\mathsf{Edges}(i)|} \quad . \quad (6)$$

However, we can do much better: we observe that if $W[i, r]$ is not well defined, we can detect it in at least $2|\mathsf{Edges}(i)|$ places. Consider any $\ell > i$ such that $(i, \ell) \in E$ and $u \in \mathbb{F}^*_3$ (The case

where $\ell < i$ is treated similarly). It is obvious that one of the following three statements must be true:

$$W_1^1[i,r,j,s] \neq W_1^2[i,r,\ell,u],\ W_1^1[i,r,\ell,u] \neq W_1^2[i,r,\ell,u],\ \text{or } W_1^1[i,r,\ell,u] \neq W_1^2[i,r,k,t].$$

It follows that if $W[i,r]$ is not well defined then there are $2|\mathsf{Edges}(i)|$ ways for V to catch the provers and each of these has probability at least $\frac{1-\epsilon}{16|E|\cdot|\mathsf{Edges}(i)|}$ to be picked. It follows that,

$$\delta_{\text{wdt}} \geq \frac{(1-\epsilon)\cdot 2|\mathsf{Edges}(i)|}{16|E|\cdot|\mathsf{Edges}(i)|} = \frac{1-\epsilon}{8|E|}\ .$$

Now, assume that W is well-defined, which means that the commitment values produced by the provers satisfy the consistency test. As discussed in section 2.3, when the commitments are consistent, the unique colours committed upon are defined by $c_i := -\left(W[i,r] + W[i,-r]\right)$ for both values of r. Since $G \notin \mathsf{3COL}$, two of the vertices must be of the same colour at the end-points of at least one edge $(i^*,j^*) \in E$. In this case the edge-verification test will detect it when (i^*,j^*) is the edge announced to both provers and if randomness $(r,s) \in \mathbb{F}_3^* \times \mathbb{F}_3^*$ is announced to P_1 then $(-r,-s)$ is the randomness announced to $\widetilde{\mathsf{P}}_2$. Using (4), the probability δ_{evt} to detect such an edge when W is well defined satisfies

$$\delta_{\text{evt}} \geq \sum_{r,s} \min_{e\in E}(p'(e,r,s,e,-r,-s)) \geq \frac{\epsilon}{|E|}\ .$$

Therefore, the detection probability δ of any deterministic strategy for $G \notin \mathsf{3COL}$ satisfies

$$\delta \geq \min(\delta_{\text{wdt}}, \delta_{\text{evt}}) \geq \frac{1}{9|E|} \quad (\text{maximized at } \epsilon = 1/9)\ .$$

The result follows as the classical value of the game $\omega\left(\Pi_{\text{lhv}}^{(2)}[G]\right) \leq 1-\delta$. ◀

To prove (perfect) zero-knowledge, it suffices to show that if $((i,j),r,s)$ and $((i',j'),r',s')$ are selected arbitrarily, V can determine at most the colours of two vertices (that form an edge). The commitments prevent a dishonest prover $\widetilde{\mathsf{V}}$ to learn the colours of two vertices that are not connected by an edge in G. Proving this is not very hard and will be done in Section 5.3 for the three-prover case (although with three provers, $\widetilde{\mathsf{V}}$ may also learn the colour of three vertices that form a triangle). The addition of a third prover will allow, using lemma 3, to get soundness against entangled provers without compromising zero-knowledge. As shown in [7], their protocol is not necessarily sound against two entangled provers. We also do not know whether $\Pi_{\text{std}}^{(2)}$ is sound against two entangled provers.

5 Three-Prover Protocol Sound Against Entangled Provers

The three-prover protocol $\Pi_{\text{qnl}}^{(3)}$, defined below, is identical to $\Pi_{\text{lhv}}^{(2)}$ except that P_3 is asked to repeat exactly what P_1 or P_2 has replied. The prover that P_3 is asked to emulate is picked at random by V. An application of lemma 3 allows to conclude the soundness of $\Pi_{\text{qnl}}^{(3)}$ against entangled provers. Zero-knowledge remains since the only way to provide V with the colours of more than two connected vertices is if they form a complete triangle of G. This reveals nothing beyond the fact that $G \in \mathsf{3COL}$ to V, since all vertices will then show different colours.

5.1 Distribution of questions

The probability distribution $\widehat{\mathcal{D}}_G$ for V's questions to the three provers is easily obtained from the distribution $\mathcal{D}'_G$ for the questions in protocol $\Pi^{(2)}_{\text{lhv}}[G]$. V picks $((e,r,s),(e',r',s')) \in_{\mathcal{D}'_G} \left(E \times (\mathbb{F}_3^*)^2\right)^2$ and sets $\hat{e} = e$, $\hat{r} = r$, and $\hat{s} = s$ with probability $\frac{1}{2}$ or sets $\hat{e} = e'$, $\hat{r} = r'$, and $\hat{s} = s'$ also with probability $\frac{1}{2}$. Defined that way, $\widehat{\mathcal{D}}_G$ is a properly defined probability distribution for V's three questions, each one in $E \times (\mathbb{F}_3^*)^2$.

5.2 The Protocol

In protocol $\Pi^{(3)}_{\text{qnl}}$, after the three questions picked according $\widehat{\mathcal{D}}_G$ by V have been answered by the the provers, V accepts if and only if the replies of P_1 and P_2 are accepted in $\Pi^{(2)}_{\text{lhv}}$ and in addition, P_3 gave the same reply than the prover it emulates.

Protocol $\Pi^{(3)}_{\text{qnl}}[G]$: Three-prover, 3-COL.
Provers $\mathsf{P}_1, \mathsf{P}_2$, and P_3 pre-agree on random values $b_i \in_R \mathbb{F}_3$ for all $i \in V$ and a random 3-colouring of G: $\{(i, c_i) | c_i \in \{0,1,2\}\}_{i \in V}$ such that $(i,j) \in E \implies c_j \neq c_i$.
Commit phase:

- V picks $(((i,j),r,s),((i',j'),r',s'),((\hat{\imath},\hat{\jmath}),\hat{r},\hat{s})) \in_{\widehat{\mathcal{D}}_G} \left(E \times (\mathbb{F}_3^*)^2\right)^3$.
- V sends $((i,j),r,s)$ to P_1, $((i',j'),r',s')$ to P_2, and $((\hat{\imath},\hat{\jmath}),\hat{r},\hat{s})$ to P_3.
- If $(i,j) \in E$ then P_1 replies $w_i = b_i \cdot r + c_i$ and $w_j = b_j \cdot s + c_j$.
- If $(i',j') \in E$ then P_2 replies $w'_{i'} = b_{i'} \cdot r' + c_{i'}$ and $w'_{j'} = b_{j'} \cdot s' + c_{j'}$.
- If $(\hat{\imath},\hat{\jmath}) \in E$ then P_3 replies $\hat{w}_{\hat{\imath}} = b_{\hat{\imath}} \cdot \hat{r} + c_{\hat{\imath}}$ and $\hat{w}_{\hat{\jmath}} = b_{\hat{\jmath}} \cdot \hat{s} + c_{\hat{\jmath}}$.

Check phase:

Consistency Test:
- If $((\hat{\imath},\hat{\jmath}),\hat{r},\hat{s}) = ((i,j),r,s)$ then V rejects if $(w_i, w_j) \neq (\hat{w}_{\hat{\imath}}, \hat{w}_{\hat{\jmath}})$.
- If $((\hat{\imath},\hat{\jmath}),\hat{r},\hat{s}) = ((i',j'),r',s')$ then V rejects if $(w'_{i'}, w'_{j'}) \neq (\hat{w}_{\hat{\imath}}, \hat{w}_{\hat{\jmath}})$.

Edge-Verification Test:
- if $(i,j) = (i',j')$ and $(r',s') \not\equiv (r,s)$ then V accept iff $w_i + w'_i \neq w_j + w'_j$.

Well-Definition Test:
- If $(i,j) = (i',j')$ and $(r',s') = (r,s)$ then V accepts iff $(w_i = w'_i) \wedge (w_j = w'_j)$.
- if $(i,j) \cap (i',j') = i$ and $r = r'$ then V accepts iff $w_i = w'_i$.
- If $(i,j) \cap (i',j') = j$ and $s = s'$ then V accepts iff $w_j = w'_j$.

The soundness of protocol $\Pi^{(3)}_{\text{qnl}}$ against entangled provers can easily be shown a direct consequence of the soundness of protocol $\Pi^{(2)}_{\text{lhv}}$ against classical provers, by an application of Lemma 3. Indeed, the soundness error corresponds to the quantum value of the game when $G = (V,E) \notin \mathsf{3COL}$ provided $\Pi^{(2)}_{\text{lhv}}$ is symmetric. As defined in Sect. 4.1 however, the distribution of questions $\mathcal{D}'_G$ is not necessarily symmetric since the first edge e is picked uniformly at random in E while the second edge $e' \in E$ is picked from e in a way that the marginal may not be uniform. However, $\Pi^{(2)}_{\text{lhv}}$ can easily be turned into a symmetric protocol by picking $(e,r,s),(e',r',s')$ according $\mathcal{D}'_G$ and announcing (e,r,s) to P_1 and (e',r',s') to P_2 with probability $\frac{1}{2}$ while announcing (e,r,s) to P_2 and (e',r',s') to P_1 with probability $\frac{1}{2}$. The resulting symmetric protocol is equivalent to $\Pi^{(2)}_{\text{lhv}}$ and therefore shares its classical value upper bounded in Theorem 5 and the set of questions Q to each player remains the same as for $\Pi^{(2)}_{\text{lhv}}$. In the symmetric version, Q is thus the same for every prover and $|Q| = 4|E|$.

▶ **Theorem 6.** *The three-prover interactive proof system* $\Pi^{(3)}_{\text{qnl}}$ *is perfectly complete and has quantum value*

$$\omega^*\left(\Pi^{(3)}_{\text{qnl}}[G]\right) \leq 1 - \left(\frac{1}{9|E| + 432|E|^2}\right)^2 \leq 1 - \left(\frac{1}{21|E|}\right)^4 \tag{7}$$

upon any graph $G = (V, E) \notin$ 3COL.

Proof. Assume $G = (V, E) \notin$ 3COL. The contrapositive of Lemma 3 indicates any one-round symmetric game $\Pi^{(2)}_{\text{lhv}}[G]$ with classical value $\omega\left(\Pi^{(2)}_{\text{lhv}}[G]\right) \leq 1 - \delta - 12|Q|\sqrt{\delta}$ is such that the modified game $\Pi^{(3)}_{\text{qnl}}[G]$ has quantum value $\omega^*\left(\Pi^{(3)}_{\text{qnl}}[G]\right) \leq 1 - \delta$. The set Q of questions to each player satisfies $|Q| = 4|E|$. Theorem 5 establishes that $\delta + 12|Q|\sqrt{\delta} \geq \frac{1}{9|E|}$, which implies $\sqrt{\delta} \geq \frac{1}{\left(\sqrt{\delta}+12|Q|\right)\cdot 9|E|} \geq \frac{1}{(1+12|Q|)\cdot 9|E|} = \frac{1}{9|E|+432|E|^2} \geq \frac{1}{441|E|^2}$, and the result follows. ◀

As an immediate consequence of Theorem 6, $\Omega(|E|^4)$ sequential repetitions of $\Pi^{(3)}_{\text{qnl}}$ produces an interactive proof system for 3COL with negligible soundness error. Although the resulting proof system can be implemented on short distances, these many sequential rounds need to be performed at high rate for a given proof to be concluded in reasonable time. A few executions of $\Pi^{(3)}_{\text{qnl}}$ could be ran in parallel without having to greatly increase the distances while reducing the number of sequential rounds. However, we don't know how the soundness error decreases when $\Pi^{(3)}_{\text{qnl}}$ is ran only a few times in parallel, even though the results of Kempe and Vidick, a quantum version of Raz's parallel repetition theorem[21], indicate that $\Omega(|E|^4)$ runs in parallel produces a proof system with negligible soundness error[14].

5.3 Proof of Perfect Zero-Knowledge

In this section, we prove that protocol $\Pi^{(3)}_{\text{qnl}}$ is perfect zero-knowledge. As a consequence, $\Pi^{(2)}_{\text{lhv}}$ is also zero-knowledge since everything $\widetilde{\mathsf{V}}$ sees in $\Pi^{(2)}_{\text{lhv}}$ can also be observed in $\Pi^{(3)}_{\text{qnl}}$. The proof of zero-knowledge proceeds using the fact that a vertex must appear at least twice to have its colour unveiled. This is the *forever hiding property* of the commitment scheme described in Section 2.3. Notice that this would be enough for $\widetilde{\mathsf{V}}$ to learn something about the colouring if no extra condition on these three vertices is observed. In fact, we can easily show that only a few cases of colour disclosure are possible and in each of these cases, $\widetilde{\mathsf{V}}$ learns nothing about the colouring that it could not have computed on its own. $\widetilde{\mathsf{V}}$ can only learn the colour of two connected vertices in G and nothing else or the colours of three vertices forming a triangle in G. In each of these cases, $\widetilde{\mathsf{V}}$ learns random distinct colours for these vertices, which is to be expected by a valid 3-colouring of G. Let us show why this is enforced by the properties (see Section 2.3) of the commitment scheme. Remember that in order to learn the colour assigned to a vertex $i \in V$, $\widetilde{\mathsf{V}}$ must ask that vertex to at least 2 distinct provers. Otherwise, $\widetilde{\mathsf{V}}$ sees only random values returned by the provers. There are 8 cases of figure depending on how $\widetilde{\mathsf{V}}$ selects the 3 edges asked. Figure 3 shows all cases. The 3 edges indicated for each case are the one picked by $\widetilde{\mathsf{V}}$. The colours associated to white vertices remain hidden by the forever hiding property of the commitment scheme. For these vertices, the committed values received from the provers are just random and independent elements in $\mathbb{F}_3$. In each of the 8 cases, the unveiled colours of the vertices are displayed in shape and colour. We see that the only way to unveil the colour of two vertices (cases 4, 5, 6, and 7) is when they are connected by an edge, which means that the colours of both vertices

are random but distinct. The only way for $\widetilde{\mathsf{V}}$ to learn the colour of 3 distinct vertices is when they form a triangle (case 8). In this case, $\widetilde{\mathsf{V}}$ learns three random and distinct colours. Clearly, this is nothing more than something necessarily true when $G \in \mathsf{3COL}$.

These properties of the commitment scheme allow, for any quantum polynomial-time dishonest verifier $\widetilde{\mathsf{V}}$, an easy simulator for $\mathbf{view}(\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3, \widetilde{\mathsf{V}}, G)$ when $G \in \mathsf{3COL}$, thus establishing that $\Pi^{(3)}_{\text{qnl}}$ is perfect zero-knowledge.

▶ **Theorem 7.** *The three-prover interactive proof system $\Pi^{(3)}_{\text{qnl}}$ is perfect zero-knowledge against quantum verifiers.*

Proof. The simulator Sim is classical given blackbox access to $\widetilde{\mathsf{V}}$ (and $\widetilde{\mathsf{V}}$ can be quantum).

Consider an execution $\mathsf{Sim}(G)$ upon graph $G = (V, E)$. It first picks a random permutation $\textsc{col}[\cdot] : \mathbb{F}_3 \mapsto \mathbb{F}_3$ over three colours, each corresponding to a distinct element in $\mathbb{F}_3$. Table $\textsc{mark}[i, r] \in \{\mathsf{true}, \mathsf{false}\}$, for $i \in V$ and $r \in \mathbb{F}_3^*$, is initialized to false and will indicate if the output of a prover has already been simulated for vertex i with randomness r. Table $\textsc{count}[i]$, for $i \in V$, counts the number of times vertex i has been asked so far during the simulation. Variable $c \in \mathbb{F}_3$, initialized to 0, indicates the next colour index the simulator should use when a new colour must be unveiled during the simulation.

Simulator $\mathsf{Sim}(G)$: Simulator for $\widetilde{\mathsf{V}}$'s view upon graph G in $\Pi^{(3)}_{\text{qnl}}$.
All arithmetic below is performed in $\mathbb{F}_3$.

1. Let $\textsc{col}[\cdot]$ be a uniform permutation of $\mathbb{F}_3$ and let $c := 0$.
2. $\forall i \in V, \forall r \in \mathbb{F}_3^*$, let $\textsc{mark}[i, r] := \mathsf{false}$ and $\textsc{count}[i] := 0$.
3. Run $\widetilde{\mathsf{V}}$ until it returns $((i_1, j_1), r_1, s_1)$, $((i_2, j_2), r_2, s_2)$, $((i_3, j_3), r_3, s_3)$.
4. For each $\ell \in \{1, 2, 3\}$ do:
   - Whenever $(i_\ell, j_\ell) \in E$ is provided by $\widetilde{\mathsf{V}}$, output $(w^\ell_{i_\ell}, w^\ell_{j_\ell}) \in \mathbb{F}_3 \times \mathbb{F}_3$ to $\widetilde{\mathsf{V}}$, both computed as follows:
     a. If $\neg\textsc{mark}[i_\ell, r_\ell]$ then
        - If $\textsc{count}[i_\ell] = 0$ then pick $W[i_\ell, r_\ell] \in_R \mathbb{F}_3$.
        - If $\textsc{count}[i_\ell] = 1$ then set $W[i_\ell, r_\ell] := -\textsc{col}[c] - W[i_\ell, -r_\ell]$, $c := c + 1$.
        - $\textsc{count}[i_\ell] := \textsc{count}[i_\ell] + 1$.
     b. If $\neg\textsc{mark}[j_\ell, s_\ell]$ then
        - If $\textsc{count}[j_\ell] = 0$ then pick $W[j_\ell, s_\ell] \in_R \mathbb{F}_3$.
        - If $\textsc{count}[j_\ell] = 1$ then set $W[j_\ell, s_\ell] := -\textsc{col}[c] - W[j_\ell, -s_\ell]$, $c := c + 1$.
        - $\textsc{count}[j_\ell] := \textsc{count}[j_\ell] + 1$.
     c. $\textsc{mark}[i_\ell, r_\ell] := \mathsf{true}$, $\textsc{mark}[j_\ell, s_\ell] := \mathsf{true}$.
     d. $w^\ell_{i_\ell} := W[i_\ell, r_\ell]$, $w^\ell_{j_\ell} := W[j_\ell, s_\ell]$.

$\widetilde{\mathsf{V}}$ is then invoked to produce questions $((i_\ell, j_\ell), r_\ell, s_\ell)$ for all provers $\mathsf{P}_\ell, \ell \in \{1, 2, 3\}$. Sim now aims at setting the values $(w^\ell_{i_\ell}, w^\ell_{j_\ell})$ for P_ℓ's commitments. If $(i_\ell, j_\ell) \notin E$, Sim produces no value for $(w^\ell_{i_\ell}, w^\ell_{j_\ell})$, exactly as P_ℓ in $\Pi^{(3)}_{\text{qnl}}$.

When $(i_\ell, j_\ell) \in E$, Sim first produces P_ℓ's commitment $w^\ell_{i_\ell}$ for $i_\ell \in V$ and then produces P_ℓ's commitment $w^\ell_{j_\ell}$ for $j_\ell \in V$. We show how $w^\ell_{i_\ell}$, $w^\ell_{j_\ell}$ is computed similarly mutatis mutandis:

- if $\textsc{mark}[i_\ell, r_\ell]$ then Sim returns the value of $w^\ell_{i_\ell}$ already determined for the simulation of the commitment of an *earlier* prover P_h, $h < \ell$. This ensures that both the commitment's *consistency test* performed and the well-definition test are always successful, as in $\Pi^{(3)}_{\text{qnl}}$ with honest provers.

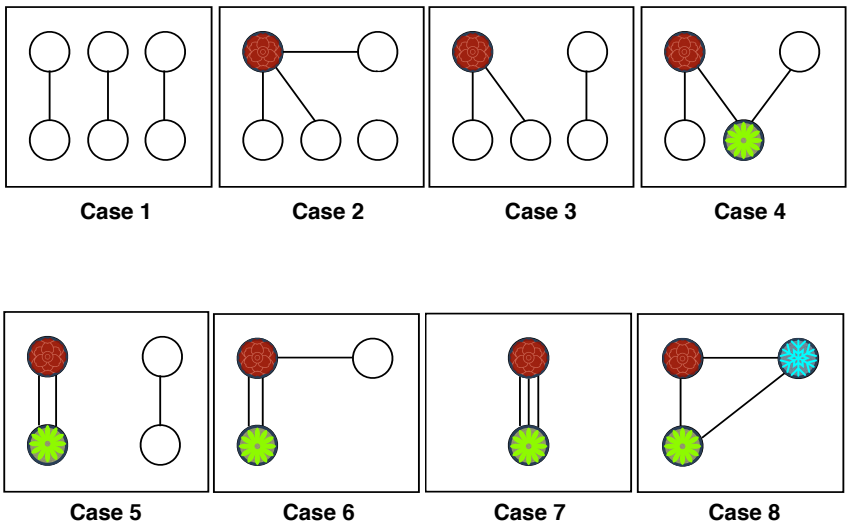


Figure 3 The 8 ways to unveil the colours of at most 3 vertices in $\Pi^{(3)}_{\text{qnl}}$.

- if $\neg\text{MARK}[i_\ell, r_\ell]$ then Sim has never simulated a commitment of the colour for vertex i_ℓ with randomness r_ℓ. The value $\text{COUNT}[i_\ell]$ indicates the number of times prior to this value for ℓ, vertex i_ℓ has been asked:
  - If $\text{COUNT}[i_\ell] = 0$ then $w^\ell_{i_\ell} \in_R \mathbb{F}_3$ is picked uniformly at random, as it should be when the commitment value for the colour of vertex i_ℓ is observed in isolation.
  - If $\text{COUNT}[i_\ell] = 1$ then the colour associated to vertex i_ℓ has been committed to value $w^h_{i_\ell}$ by an *earlier* simulated prover P_h, $h < \ell$ upon randomness $-r_\ell$ (otherwise, $\text{MARK}[i_\ell, r_\ell] = \mathsf{true}$). Sim sets $w^\ell_{i_\ell} = -\text{COL}[c] - w^h_{i_\ell}$, which satisfies the *implicit unveiling* of random colour $\text{COL}[c] = -w^\ell_{i_\ell} - w^h_{i_\ell}$. The current colour c is incremented.

  The value of $\text{COUNT}[i_\ell]$ is increased by one and $\text{MARK}[i_\ell, r_\ell] = \mathsf{true}$, as the colour of vertex i_ℓ with randomness r_ℓ has been committed upon by the simulated prover P_ℓ.

Let $(w^1_{i_1}, w^1_{j_1})$, $(w^2_{i_2}, w^2_{j_2})$, and $(w^3_{i_3}, w^3_{j_3})$ be all commitment values simulated by Sim. As discussed above and shown in Fig. 3, the colours of no more than 3 vertices are unveiled in the process. Sim always unveils as many different colours as there are colours unveiled to $\widetilde{\mathsf{V}}$. If Sim's simulated committed values unveils only the colour of one vertex then that colour is random, as it should in this case in $\Pi^{(3)}_{\text{qnl}}$.

If Sim's committed values unveils the colours of exactly 2 vertices then these 2 vertices form an edge in G and the colours are two different random colours, as it should be in $\Pi^{(3)}_{\text{qnl}}$. Finally, when Sim's committed values unveil the colours of exactly 3 vertices then these vertices form a triangle in G. The 3 colours unveiled by Sim to $\widetilde{\mathsf{V}}$ are different and assigned randomly to each of the 3 vertices, as it is in $\Pi^{(3)}_{\text{qnl}}$. Otherwise, if w^ℓ_i for $i \in V$ has been generated with only one random value then w^ℓ_i is random and uniform in $\mathbb{F}_3$, exactly as it is in $\Pi^{(3)}_{\text{qnl}}$ in the same situation. It is now clear that,

$$\mathbf{view}(\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3, \widetilde{\mathsf{V}}, G) = \mathsf{Sim}(G) \ ,$$

and $\Pi^{(3)}_{\text{qnl}}$ is perfect zero-knowledge. ◀

Note: Since no *rewinding* is used by our simulator, it is absolutely unnecessary to explicitly handle *auxiliary-inputs* or the fact that V is quantum. No special care is required to handle these considerations that become highly non-trivial in the case where rewinding is required.

6 Conclusion and Open Problems

We have provided a three-prover perfect zero-knowledge proof system for **NP** sound against entangled provers that is implementable in some well controlled environment. In order to make it fully practical, it would be better to find a protocol with smaller soundness error and requiring only two provers.

Our protocols are *proofs of membership* whereas in practice we would like to use them for identification purpose in which scenario *proofs of knowledge* is what we really need.

Moreover, we would like to extend our techniques to prove any language in **QMA** or **QCMA**, the natural quantum extensions of **NP**.

We would also want to prove whether $\Pi^{(2)}_{\text{std}}$ is sound against entangled provers. Finally, we seek a variant of $\Pi^{(2)}_{\text{std}}$ that would be sound against No-Signalling provers and variants of $\Pi^{(2)}_{\text{lhv}}$ and $\Pi^{(3)}_{\text{qnl}}$ that are both sound against No-Signalling provers and Zero-Knowledge.

References

1 Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 113–131, New York, NY, USA, 1988. ACM. doi:10.1145/62212.62223.

2 Gilles Brassard, Claude Crépeau, Dominic Mayers, and Louis Salvail. Defeating classical bit commitments with a quantum computer. arXiv:quant-ph/9806031, June 1998.

3 Andre Chailloux and Anthony Leverrier. Relativistic (or 2-prover 1-round) zero-knowledge protocol for NP secure against quantum adversaries. In *Advances in Cryptology – EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 – May 4, 2017, Proceedings, Part III*, pages 369–396. Springer International Publishing, 2017. doi:10.1007/978-3-319-56617-7_13.

4 D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *Proceedings on Advances in Cryptology*, CRYPTO '88, pages 319–327, Berlin, Heidelberg, 1990. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=88314.88969.

5 Alessandro Chiesa, Michael A. Forbes, Tom Gur, and Nicholas Spooner. Spatial isolation implies zero knowledge even in a quantum world. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:44, 2018. URL: https://eccc.weizmann.ac.il/report/2018/044.

6 John F. Clauser, Michael A. Horne, Abner Shimony, and Richard A. Holt. Proposed experiment to test local hidden-variable theories. *Phys. Rev. Lett.*, 23:880–884, October 1969. doi:10.1103/PhysRevLett.23.880.

7 Richard Cleve, Peter Høyer, Benjamin Toner, and John Watrous. Consequences and limits of nonlocal strategies. In *Proceedings of the 19th IEEE Annual Conference on Computational Complexity*, CCC '04, pages 236–249, Washington, DC, USA, 2004. IEEE Computer Society. doi:10.1109/CCC.2004.9.

8 Claude Crépeau, Louis Salvail, Jean-Raymond Simard, and Alain Tapp. Two provers in isolation. In *Advances in Cryptology – ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 407–430, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-25385-0_22.

9 Serge Fehr and Max Fillinger. Multi-prover commitments against non-signaling attacks. In *Advances in Cryptology – CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara,*

CA, USA, August 16-20, 2015, Proceedings, Part II, pages 403–421, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. doi:10.1007/978-3-662-48000-7_20.

10 Uriel Feige and Joe Kilian. Two prover protocols: low error at affordable rates. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 172–183. ACM, 1994. doi:10.1145/195058.195128.

11 S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM. J. Computing*, 18(1):186–208, February 1989.

12 Alex Bredariol Grilo, William Slofstra, and Henry Yuen. Perfect zero knowledge for quantum multiprover interactive proofs. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:86, 2019. URL: https://eccc.weizmann.ac.il/report/2019/086.

13 J. Kempe, H. Kobayashi, K. Matsumoto, B. Toner, and T. Vidick. Entangled games are hard to approximate. *SIAM Journal on Computing*, 40(3):848–877, 2011. doi:10.1137/090751293.

14 J. Kempe and T. Vidick. Parallel repetition of entangled games. In *Proceedings of 43rd ACM Symposium on Theory of Computing (STOC)*, pages 353–362, 2011.

15 Adrian Kent. Unconditionally secure bit commitment. *Phys. Rev. Lett.*, 83:1447–1450, August 1999. doi:10.1103/PhysRevLett.83.1447.

16 J Kilian. Strong separation models of multi prover interactive proofs. In *DIMACS Workshop on Cryptography*, 1990.

17 Joe Kilian. *Uses of randomness in algorithms and protocols*. MIT Press, 1990.

18 Dror Lapidot and Adi Shamir. A one-round, two-prover, zero-knowledge protocol for NP. *Combinatorica*, 15(2):204–214, 1995. doi:10.1007/BF01200756.

19 T. Lunghi, J. Kaniewski, F. Bussières, R. Houlmann, M. Tomamichel, S. Wehner, and H. Zbinden. Practical relativistic bit commitment. *Phys. Rev. Lett.*, 115:030502, July 2015. doi:10.1103/PhysRevLett.115.030502.

20 Sandu Popescu and Daniel Rohrlich. Quantum nonlocality as an axiom. *Foundations of Physics*, 24(3):379–385, 1994. doi:10.1007/BF02058098.

21 Ran. Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998. doi:10.1137/S0097539795280895.

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

23 Ephanielle Verbanis, Anthony Martin, Raphaël Houlmann, Gianluca Boso, Félix Bussières, and Hugo Zbinden. 24-hour relativistic bit commitment. *Phys. Rev. Lett.*, 117:140506, September 2016. doi:10.1103/PhysRevLett.117.140506.

Use Your Brain! Arithmetic 3PC for Any Modulus with Active Security

Hendrik Eerikson
Cybernetica AS, Tartu, Estonia
hendrik.eerikson@cyber.ee

Marcel Keller
CSIRO's Data61, Eveleigh, Australia
mks.keller@gmail.com

Claudio Orlandi
Department of Computer Science, DIGIT, Aarhus University, Denmark
orlandi@cs.au.dk

Pille Pullonen
Cybernetica AS, Tartu, Estonia
pille.pullonen@cyber.ee

Joonas Puura[1]
Institute of Computer Science, University of Tartu, Estonia
joonas.puura@gmail.com

Mark Simkin
Department of Computer Science, DIGIT, Aarhus University, Denmark
simkin@cs.au.dk

Abstract

Secure multiparty computation (MPC) allows a set of mutually distrustful parties to compute a public function on their private inputs without revealing anything beyond the output of the computation. This paper focuses on the specific case of actively secure three-party computation with an honest majority. In particular, we are interested in solutions which allow to evaluate arithmetic circuits over real-world CPU word sizes, like 32- and 64-bit words. Our starting point is the novel compiler of Damgård et al. from CRYPTO 2018. First, we present an improved version of it which reduces the online communication complexity by a factor of 2. Next, we replace their preprocessing protocol (with arithmetic modulo a large prime) with a more efficient preprocessing which only performs arithmetic modulo powers of two. Finally, we present a novel "postprocessing" check which replaces the preprocessing phase. These protocols offer different efficiency tradeoffs and can therefore outperform each other in different deployment settings. We demonstrate this with benchmarks in a LAN and different WAN settings. Concretely, we achieve a throughput of 1 million 64-bit multiplications per second with parties located in different continents and 3 million in one location.

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

Keywords and phrases Secure Multiparty Computation, Information Theoretic Security

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.5

Related Version `https://eprint.iacr.org/2019/164`

Funding *Hendrik Eerikson*: European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation program grant agreement No 778615 (BiggerDecisions).
Claudio Orlandi: Danish Independent Research Council under Grant-ID DFF-6108-00169 (FoCC), the European Research Council (ERC) under grant No 803096 (SPEC).

[1] Work done while at Cybernetica AS

Pille Pullonen: Estonian Research Council under grant IUT27-1, ERDF through the Estonian Centre of Excellence in ICT Resrearch (EXCITE).
Mark Simkin: Danish Independent Research Council under Grant-ID DFF-6108-00169 (FoCC), European Research Council (ERC) under grant No 669255 (MPCPRO), No 803096 (SPEC).

1 Introduction

Secure Multiparty Computation (MPC) is an umbrella term for a broad range of cryptographic techniques and protocols that enable a set of parties $\mathrm{P}_1, \ldots, \mathrm{P}_n$ to compute some function f of their private inputs $x_1, \ldots, x_n$ without revealing anything beyond the output $f(x_1, \ldots, x_n)$ of the computation. Most importantly, an actively misbehaving participant should not be able to bias the outcome of the computation (except by choosing their input) or learn anything about the inputs of the honest parties (except for what is leaked by the output itself). MPC started out as a purely theoretical research field in the 90ies, but has developed into a science on the brink of practical deployment. The number of of real-world use cases, MPC framework implementations, and startups is constantly increasing (see [4] for a survey).

The landscape of MPC protocols is broad and diverse, and protocols differ greatly in many parameters such as the number of involved parties, the corruption threshold, the adversarial model, and the network setting. We focus on a popular model of *three-party computation with an honest majority*. This model has been used in different real-world applications [15, 14, 10, 11, 2], often in the so-called *client-server* scenario where a possibly large number of clients secret share their inputs to three computation servers who then perform the computations [33] and return the result to the clients. A major advantage of the honest majority setting is that one can obtain protocols which do not rely on computationally expensive cryptographic operations (e.g. exponentiations, oblivious transfer), but typically only use light-weight arithmetic operations and achieve information theoretic security.

Existing implementations of three-party computation protocols for the honest-majority case fall into two broad categories. VIFF [24] and its successors [40] only support arithmetic computations over prime order fields. Sharemind's protocol suite [12, 13] can be used to evaluate arithmetic circuits with arbitrary word sizes, but is only secure against passive adversaries that follow the protocol faithfully. This means that one has to either settle for rather weak security guarantees or to develop applications specifically tailored to rather unnatural word sizes instead of using the common 32- and 64-bit word sizes that dominate real-world system architectures. In particular, this means that a developer has to match the needs of the MPC framework rather than the framework meeting the needs of the developer.

The main barrier to constructing actively secure protocols for evaluating arithmetic circuits with arbitrary word sizes lies in the fact that known approaches to achieving active security, like *information checking* techniques [39], require prime order fields. Up until recently it has been an open question to design protocols for arithmetic circuits with active security for arbitrary word sizes. In a recent work Damgård et al. [27] addressed this question by presenting an information theoretically secure protocol compiler that transforms passively secure protocols into actively secure ones that can tolerate up to $\mathcal{O}(\sqrt{n})$ corruptions and only have a *constant* overhead in storage and computational work.

Our Contributions. We consider the class of protocols produced by compiler of Damgård et al. [27], and we improve such protocols in several ways. The main idea behind Damgård et al.'s compiler is to let the *real parties* "emulate" *virtual parties* that execute the desired computation on behalf of the real parties. The crucial point is that the virtual parties can

execute[2] a passively secure protocol in a way that prevents any real party from actively misbehaving. Every time that a virtual party $\mathbb{P}_i$ is supposed to send a message to another virtual party $\mathbb{P}_j$ in the passively secure protocol, every real party that is emulating $\mathbb{P}_i$ computes the same message redundantly and sends it to every real party emulating $\mathbb{P}_j$. Each real party emulating $\mathbb{P}_j$ therefore receives a set of messages and aborts in case the received messages are not all equal. Intuitively this approach ensures active security as long as there is at least one honest real party in every virtual party, since any malicious party either follows the protocol (in which case we effectively only have passive corruptions) or sends a message that disagrees with the message that is sent by at least one honest party (in which case the honest receiving party and consequently all other parties abort the protocol). This approach heavily relies on the fact that *all* messages are sent redundantly, thus incurring a multiplicative blow-up in the bandwidth overhead of the protocol.

We present an improved compiler that significantly reduces the number of redundant messages that need to be sent during the execution. The idea is to elect *one* real party in each virtual party to be the "brain", which sends all messages on behalf of its virtual party to *all* real parties in the receiving virtual party. The other real parties, the "pinkies", still receive messages from the brains and thus can locally follow the protocol execution. At the end of the protocol, right before the output is released, we let all parties perform a single check that guarantees that all messages sent by the brains during the protocol are consistent with the messages all the pinkies would have sent. It is clear that if any of the brains cheated during the protocol execution, then it must have sent a message that is inconsistent with the view of at least one pinky, thus the protocol would abort during the checking phase. On the downside our new compiler now imposes a stronger security requirement on the protocol it starts with. Honest brains continue the protocol execution up to the checking phase even if a malicious brain misbehaves, which means that we need a protocol that does not leak any private information even if cheating during the computation phase occurs, e.g. protocol with weak privacy [32]. Thankfully, most passively secure secret sharing based protocols provide such security guarantees. More concretely, these protocols follow a compute-then-open structure, where the output of the computation is only revealed in the last round and any cheating during the preceding computation can only affect the correctness of the output, but not the privacy of the inputs. Thus, performing the consistency check at the end of the computation phase and *before* the output phase, ensures that no information is leaked. We formally present our new compiler and prove its security in Section 3. For the specific three-party case, our compiler produces a protocol, which is roughly twice as efficient as the protocol produced by the compiler of Damgård et al., since in the three party case each virtual party is emulated by one pinky and one brain.

Our second contribution is an improved preprocessing protocol for generating secret-shared multiplication triples. Damgård et al. generate both triples modulo a prime and triples modulo a power of 2, followed by a check-and-sacrifice step. We replace this by a preprocessing phase which does not perform any arithmetic in the larger prime field and solely uses computation modulo a slightly larger power of 2, thus improving on efficiency. While the sacrifice step is not performed in a field anymore, security follows using similar arguments as in the recent work on SPDZ over rings [21].

We show that it is possible to completely avoid the preprocessing phase if one wishes to do so. Recall that our underlying protocols are assumed to preserve privacy until the outputs are opened. We exploit this security property by running the multiplication protocols

[2] Note that virtual parties do not physically exist. "Virtual parties execute a protocol" means that the real parties simulate the virtual parties protocol execution.

optimistically and then, prior to opening the outputs, perform a single combined check. The protocols with preprocessing and with postprocessing therefore offer different efficiency tradeoffs. The protocol with preprocessing has a leaner online phase, whereas the protocol with postprocessing has a better overall performances. Descriptions of our protocols are given in Sections 4, 5, 6. In Section 7, we provide extensive performance benchmarks of our framework, both in the LAN as well as different WAN settings. Our protocols have been integrated in two of the leading MPC frameworks, namely the Sharemind MPC protocol suite and MP-SPDZ. As described in Section 7, we achieve the most efficient implementation of a three-party computation protocol for arithmetic circuits modulo 2^{64} with active security.

Other Related Work. The SPDZ family of protocols [8, 28, 25] efficiently implements MPC with active security in the *dishonest majority* setting. These protocols are split up into a slower, computationally secure *offline phase* in which correlated randomness (Beaver's triples) is generated and a faster, information-theoretically secure *online phase* in which these triples are consumed to compute the desired functionality. Active security in the online phase is achieved using information theoretic message authentication codes (MACs), which until recently limited the SPDZ approach to computation over fields. In a recent work [21], this limitation has been lifted, allowing to perform computation modulo 2^k (by defining the MACs modulo to be $2^{k+\lambda}$ where λ is the security parameter, thus introducing an overhead proportional to the security parameter). An implementation (and optimizations) of [21] was presented in [23]. In addition, [18] follows up [21] with a two-party protocol that uses homomorphic encryption and efficient zero-knowledge proofs in the precomputation phase.

Other recent works have considered active security in the three-party setting. [31] uses correlated random number generation to achieve efficient preprocessing and replication to achieve security. The protocol was originally presented only for Boolean circuits, but it was then noticed that the approach generalizes to general rings [36]. They mention actively secure protocols in this setting, but do not give detailed protocol descriptions and only implement semi-honest versions of their protocols. For finite fields, [20] achieves active security by running two copies of the computation, respectively with real and random inputs, and uses the latter to verify correctness (this approach can be used for more than three parties). Boyle et al. [16] recently presented a protocol that achieves no asymptotic communication overhead over a semi-honest protocol in the same setting. However, their benchmarks suggest that the computation of their protocol might be rather limiting in some network settings, see Section 7.3 for more information. After the first version of our paper appeared online, a very different protocol for the same three party honest majority setting was presented in [19]. They combine two linear secret sharing schemes, one between two and other between three parties where the former is used to share a component of the latter sharing. This allows them to create a circuit dependent precomputation phase where all the two party sharings of random values are precomputed based on the circuit structure. The online phase focuses on computing modifiers to turn the random precomputed sharings to real outputs. Moreover, novel techniques for honest-majority MPC over rings have very recently been deployed in [1]. It is however still unclear whether this can lead to protocols which are efficient in practice.

2 Preliminaries

We write $v \leftarrow \mathcal{X}$ to denote the sampling of a uniformly random value v from set $\mathcal{X}$. Throughout the paper λ denotes the security parameter. Given n parties $\mathrm{P}_1, \ldots, \mathrm{P}_n$, we write P_{i+1} to denote the party after P_i and we implicitly assume a wrap around of the party's index. That is $\mathrm{P}_{n+1} = \mathrm{P}_1$ and $\mathrm{P}_0 = \mathrm{P}_n$.

We define security using the UC framework [17]. In particular, we require the notion of *"Weak Privacy" against active adversaries* introduced in [32, Definition 5.11]. We include background definitions in Appendix A. Throughout the paper, we assume a synchronous communication network, a rushing adversary, and secure point-to-point channels.

2.1 Auxiliary Ideal Functionalities

We will make use of the following basic auxiliary ideal functionalities in this paper: The broadcast with *individual* abort functionality $\mathcal{F}_{\mathsf{bcast}}$ (Figure 1) allows a sender S to send a value v to a set of parties $\mathbb{P}$. The functionality guarantees that either a party aborts or it agrees on a consistent value with the other parties. Such a functionality is weaker than detectable broadcast [30], which requires that either all players agree on the same value or that all players unanimously abort. The functionality can easily be instantiated by letting the sender S send v to all parties in $\mathbb{P}$. Every party in $\mathbb{P}$ echoes the received value to all other parties in $\mathbb{P}$. Parties that receive consistent values output that value, parties that receive inconsistent values abort.

Functionality $\mathcal{F}_{\mathsf{bcast}}$ Functionality with sender S, who has input v, parties P_1, $\ldots$, P_n, and adversary $\mathcal{A}$.

1. S sends $(v, \mathbb{P})$ to $\mathcal{F}_{\mathsf{bcast}}$, where $v \in \{0,1\}^*$ and $\mathbb{P} \subset \{\mathrm{P}_1 \ldots \mathrm{P}_n\}$.
2. If either S or a party from $\mathbb{P}$ is corrupt, then $\mathcal{A}$ receives v and can decide which parties from $\mathbb{P}$ abort and which receive the output by sending a $|\mathbb{P}|$ long bit-vector b to the ideal functionality. For $\mathrm{P}_i \in \mathbb{P}$:
   a. If $b_i = 1$, then $\mathcal{F}_{\mathsf{bcast}}$ sends v to P_i.
   b. If $b_i = 0$, then $\mathcal{F}_{\mathsf{bcast}}$ sends $\perp$ to P_i.

Figure 1 Broadcast functionality.

The message checking functionality $\mathcal{F}_{\mathsf{check}}$ (Figure 2) allows a receiver R, who holds a vector of messages, to check whether all other parties $\mathrm{P}_1, \ldots, \mathrm{P}_n$ hold the same vector of messages. The functionality can be instantiated by letting each party P_i send its input to R. However, in this case the communication overhead is $\Theta(n\ell)$ messages, where ℓ is the number of messages in a vector. Assuming the existence of collision-resistant hash functions, one can obtain a more communication efficient solution by simply letting all parties hash their message vectors into small digests before sending them to R. The communication overhead of this is $\Theta(n\lambda)$ bits if we assume that the output length of the hash function is $\Theta(\lambda)$.

2.2 Additive Secret Sharing

We recall what additive secret sharing is and how to perform some basic operations on it. We will use this type of secret sharing in our three-party protocol in Section 4 and the modulus 2^m defines the word size over which computations will be performed. For example, for arithmetic computations over 64-bit integers, one can set $m = 64$. For the sake of concreteness, we restrict our attention to the three-party case.

If party P_i wants to share a value $a \in \mathbb{Z}_{2^m}$, it picks uniformly random $a_1, a_2 \leftarrow \mathbb{Z}_{2^m}$, sets $a_3 = a - a_1 - a_2 \mod 2^m$, and sends a_j to P_j. We use $[a]_m$ to denote additive secret sharing of a modulo 2^m. For a prime p, we will abuse notation and use $[a]_p$ to denote a secret sharing of a modulo p. To open a value $[a]_m$, every party P_i sends its value a_i to P_{i-1} and P_{i+1}.

Functionality $\mathcal{F}_{\mathsf{check}}$ The functionality runs with receiver R, parties $\mathrm{P}_1, \ldots, \mathrm{P}_n$, and adversary $\mathcal{A}$. Party $\mathrm{P}_i \in \{\mathrm{P}_1, \ldots, \mathrm{P}_n\}$ has input $\left(m_{(1,i)}, \ldots, m_{(\ell,i)}\right)$ and receiver R has $(m_1, \ldots, m_\ell)$.

---

1. All parties send their inputs to the $\mathcal{F}_{\mathsf{check}}$.
2. $\mathcal{A}$ can decide to continue or to abort.
   a. If $\mathcal{A}$ continues, then $\mathcal{F}_{\mathsf{check}}$ checks whether all inputs are the identical. It outputs same if this is the case, and different otherwise, to the receiver R (in the latter case, the functionality gives the inputs of all honest parties to $\mathcal{A}$).
   b. If $\mathcal{A}$ aborts, then $\mathcal{F}_{\mathsf{check}}$ sends $\perp$ to all parties.

Figure 2 Message checking functionality.

To add constant c to $[a]_m$, i.e., compute $[b]_m$ with $b = c + a \mod 2^m$, P_1 locally computes $b_1 = a_1 + c \mod 2^m$, while P_2 and P_3 just set $b_i = a_i$. To compute $[c]_m$, where $c = a + b \mod 2^m$, every party P_i locally adds its shares, i.e., computes $c_i = a_i + b_i \mod 2^m$.

Given a secret shared multiplication triple $([x]_m, [y]_m, [z]_m)$ with $z = x \cdot y \mod 2^m$ and two secret shared values $[a]_m$ and $[b]_m$, we compute $[c]_m$ with $c = a \cdot b \mod 2^m$ as follows:

1. Open $e = [x]_m + [a]_m$ and $d = [y]_m + [b]_m$
2. Compute $[c]_m = [z]_m + e \cdot [b]_m + d \cdot [a]_m - ed$

2.3 Additive Replicated Secret Sharing

We will use additive replicated secret sharing in our preprocessing protocol in Section 5 because it allows for efficient multiplication. Since our preprocessing protocol focuses on the three-party case, we will also restrict our attention to this case here.

If party P_i wants to share a value $a \in \mathbb{Z}_{2^m}$, it sets $a_i = 0$ and samples $a_{i+1}, a_{i-1} \leftarrow \mathbb{Z}_{2^m}$ under the constraint that $a = a_1 + a_2 + a_3 \mod 2^m$. It then sends a_{j-1} and a_{j+1} to P_j.[3] We write $[\![a]\!]_m$ to denote an additive replicated secret sharing of a modulo 2^m. We will abuse notation and write $[\![a]\!]_p$ to denote the additive replicated secret sharing modulo a prime p.

Generating a random shared value is a subroutine which will be useful in later protocols. If the parties want to generate shares of a random value they can do it in the following two ways. For unconditionally secure randomness each party P_i picks a random $s_{i-1} \in \mathbb{Z}_{2^m}$ and sends it to P_{i+1} while at the same time receiving s_{i+1} from P_{i-1}. For computationally secure randomness the parties run the unconditionally secure version, at the beginning of the protocol, once and for all, and interpret their shares as PRF keys K_1, K_2, K_3 such that party P_i knows K_{i-1} and K_{i+1}. When they want to generate the j-th random share, the parties define their shares $s^j_{i-1} = F_{K_{i-1}}(j)$ and $s^j_{i+1} = F_{K_{i+1}}(j)$.

To reveal a secret shared value $[\![a]\!]_m$, each party P_i sends a_{i-1} to P_{i-1} and a_{i+1} to P_{i+1}. Each P_j receives a_j from P_{j-1} and P_{j+1}, checks consistency of the received values, and outputs $a = a_1 + a_2 + a_3 \mod 2^m$ if the check passed. *Computational Security:* Opening

[3] This is a small yet non-trivial optimization of the protocol of [27], where a_i is also a random share and the other two parties have to check consistency of this value. By setting $a_i = 0$ we save communication of 4 ring elements per input gate, and one additional round of communication. Note that this change has no impact on security. If P_i is corrupt we need that the two other parties receive the same value of a_i, and this is trivially achieved by setting the value to 0. If one of the other two parties is corrupt, they would learn a_i anyway so whether it is random or a constant value has no security impact.

several values $a^{(1)}, \ldots, a^{(n)}$ can be optimized as follows: Each P_j only receives $a_j^{(l)}$ from P_{j-1} and computes $a^{(l)} = a_1^{(l)} + a_2^{(l)} + a_3^{(l)} \mod 2^m$. In addition the parties broadcast hashes of $(a^{(1)}, \ldots, a^{(n)})$ and abort in case of a mismatch.

To add a public constant c to a secret shared value $[\![a]\!]_m$, i.e., to compute $[\![b]\!]_m$, where $b = c + a \mod 2^m$, we set $b_1 = a_1 + c$, $b_2 = a_2$, and $b_3 = a_3$. To add $[\![a]\!]_m$ and $[\![b]\!]_m$, i.e., to compute $[\![c]\!]_m$, where $c = a + b \mod 2^m$ every party P_i locally adds their shares. It computes $c_{i-1} = a_{i-1} + b_{i-1} \mod m$ and $c_{i+1} = a_{i+1} + b_{i+1} \mod 2^m$. To multiply $[\![a]\!]_m$ by constant c, i.e., to obtain $[\![b]\!]_m$ with $b = c \cdot a \mod 2^m$, every party P_i computes $b_{i-1} = c \cdot a_{i-1} \mod 2^m$ and $b_{i+1} = c \cdot a_{i+1} \mod 2^m$.

Given $[\![a]\!]_m$ and $[\![b]\!]_m$, we can compute $[\![c]\!]_m$, with $c = a \cdot b \mod 2^m$, optimistically (i.e. with potential error in case of cheating) as follows:

1. The parties generate a random value $[\![s]\!]_m$;
2. Each P_i computes $u_{i+1} = a_{i+1}b_{i+1} + a_{i+1}b_{i-1} + a_{i-1}b_{i+1} + s_{i-1}$ and sends u_{i+1} to P_{i-1};
3. P_i receives u_{i-1}, thus defining $[\![u]\!]_m$;
4. The parties compute $[\![c]\!]_m = [\![u]\!]_m - [\![s]\!]_m$.

2.4 Additive Replicated Secret Sharing with Redundant Shares

In some of our protocols we use a different kind of replicated secret sharing, which we denote as $[\![x]\!]_{m,\lambda}$. Those are sharing of values in $\mathbb{Z}_{2^m}$ but represented with shares in $\mathbb{Z}_{2^{m+\lambda}}$, where λ is a security parameter. Those shares work as the regular additive replicated secret sharings described in the previous sections (e.g., all basic commands are unchanged), but employ shares in a larger ring – this is useful for checking correctness of multiplication triples as we shall see. We describe some basic protocols that can be run with this kind of shares:

To convert $[\![x]\!]_m$ to $[\![x]\!]_{m,\lambda}$ each party simply interprets their shares as elements of the larger ring (in other words, the shares are padded with 0s in the λ most significant positions). Note that in general $\sum_i x_i \mod 2^{m+\lambda} \neq x$, that is the sum can be either equal to x or to $x + 2^m$ depending on the magnitude of the shares. However, since the semantic of our sharing is that the shared value is $\sum_i x_i \mod 2^m$ the protocol is indeed correct (and this notation allows us a simpler description of more advanced protocols).

To convert $[\![x]\!]_{m,\lambda}$ down to $[\![x]\!]_m$ each party P_i reduces their shares modulo 2^m as $x'_{i+1} = x_{i+1} \mod 2^m$ and $x'_{i-1} = x_{i-1} \mod 2^m$. Both conversions preserve the shared value because computing modulo 2^m and modulo $2^{m+\lambda}$ are commutative as 2^m divides $2^{m+\lambda}$ and both operations trivially preserve the replication of shares.

2.5 Additive Replicated Secret Sharing over the Integers

Finally, we recall the replicated secret sharing over integers from [27]. The authors observed that one can secret share a value $a \in \mathbb{Z}_{2^m}$ over the integers using shares with bit-length $m + \lambda$. The λ extra bits ensure that the statistical distance between the distributions of shares for any two values in $\mathbb{Z}_{2^m}$ is negligible in λ.

To share a value $a \in \mathbb{Z}_{2^m}$, P_i picks $a_1, a_2 \leftarrow \{0, \ldots, 2^{m+\lambda} - 1\}$ and sets $a_3 = a - a_1 - a_2$. The shares are distributed among the parties as above. We write $[\![a]\!]_{\mathbb{Z}}$ to denote an additive replicated secret sharing of a over the integers.

Optimistic multiplication of $[\![a]\!]_{\mathbb{Z}}$ and $[\![b]\!]_{\mathbb{Z}}$ is similar to its counterpart modulo p. Let B be a bound on the share amplitude. Optimistically compute $[\![c]\!]_{\mathbb{Z}}$ with $c = a \cdot b$ as follows:

1. The parties generate shares of a random value $[\![s]\!]_{\mathbb{Z}}$ as described above, but s_i are chosen in $\{0, \ldots, 2^{2\lceil \log B \rceil + \lambda + 2} - 1\}$ (if the information-theoretic version is used, parties also check that the received shares s_i are in this range);

2. Each P_i computes u_{i+1} as before but over $\mathbb{Z}$;
3. P_i receives u_{i-1} and checks $|u_{i-1}| \leq 2^{2^{\lceil \log B \rceil + \lambda + 3}}$;
4. The parties compute $[\![c]\!]_m = [\![u]\!]_m - [\![s]\!]_m$.

Other operations are analogous to their counterparts modulo m. For a more details see [27].

3 Extension of the Compiler by Damgård et al.

The compiler $\mathsf{COMP}_{\mathsf{old}}$ by Damgård et al. [27] takes an n-party passively $\left(t^2+t\right)$-secure protocol Π and transforms it into a protocol $\mathsf{COMP}_{\mathsf{old}}(\Pi)$ that is secure with abort against t active corruptions[4]. For $t = 1$, the compiler transforms a passively two-secure three-party protocol into a protocol that is secure against one active corruption. The high-level idea of the compiler is to let virtual parties execute the passively secure protocol on behalf of the real parties. Each virtual party $\mathbb{P}_i$ is simulated by $t+1$ real parties $\mathrm{P}_i, \ldots, \mathrm{P}_{i+t}$ in a way that prevents an active adversary, who controls at most t real parties, from corrupting any of the virtual parties. Meaning that corrupting t real parties allows the adversary to see the view of at most $\left(t^2+t\right)$ virtual parties running the passive protocol. In the following we will write $\mathrm{P}_j \in \mathbb{P}_i$ to denote that real party P_j is simulating virtual party $\mathbb{P}_i$.

The workflow of their compiler can be split into two phases. In the first phase, for each virtual party $\mathbb{P}_i$, all real parties $\mathrm{P}_j \in \mathbb{P}_i$ agree on a common input and randomness that will be used by $\mathbb{P}_i$ during the execution of the passively secure protocol Π. Having the same input and the same randomness, every $\mathrm{P}_j \in \mathbb{P}_i$ will be able to redundantly compute the exact same messages that $\mathbb{P}_i$ is supposed to send during the execution of Π. In the second phase, the virtual parties run Π to compute the desired functionality from the inputs and randomness that the virtual parties have agreed upon. Whenever $\mathbb{P}_i$ is supposed to send a message to $\mathbb{P}_j$ according to Π, *every* real party simulating $\mathbb{P}_i$ will send a separate message to *every* real party simulating $\mathbb{P}_j$. Each real party verifies that it receives the same message from all sending real parties and aborts if this is not the case.

Intuitively, the resulting protocol is secure against t active corruptions, since an adversary cannot misbehave on behalf of a virtual party it is simulating, and at the same time be consistent with at least one other honest real party in the same virtual party. From an efficiency point of view, every message from one $\mathbb{P}_i$ to some other $\mathbb{P}_j$ is sent redundantly from $t+1$ to $t+1$ real parties. That is, if the passively secure protocol Π sends ℓ messages during a protocol execution, then $\mathsf{COMP}_{\mathsf{old}}(\Pi)$ will send roughly $\mathcal{O}\left(\ell \cdot t^2\right)$ messages.

3.1 A New Compiler for Protocols with Weak Privacy

We present a new compiler $\mathsf{COMP}_{\mathsf{new}}$, which makes slightly stronger assumptions about the starting protocol Π, but compiles it into an actively secure protocol in a more communication efficient manner. $\mathsf{COMP}_{\mathsf{new}}$ takes as input a $\left(t^2+t\right)$-weakly private protocol Π and outputs a compiled protocol $\mathsf{COMP}_{\mathsf{new}}(\Pi)$ that is secure against t active corruptions. If Π sends ℓ messages in total, then our compiled protocol will only send $\mathcal{O}\left(\ell \cdot t + t^2\right)$ messages.

Our new compiler follows the approach of $\mathsf{COMP}_{\mathsf{old}}$. However, instead of verifying the validity of every single message between virtual parties as soon as it is sent, we will let the real parties simulate the virtual parties in a more optimistic and communication efficient fashion, where the correctness of all communicated messages is only verified once at the end of the computation phase, right before the opening phase of Π. Pushing the whole

[4] The authors also show how to achieve active security with guaranteed output delivery, but we focus on security with abort.

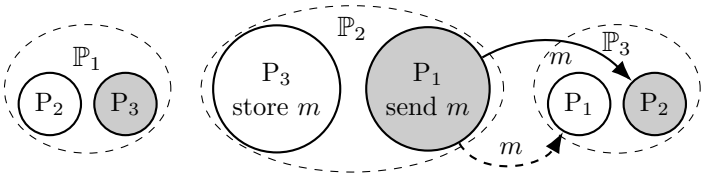


Figure 3 Simulation strategy for three parties with one active corruption. Dashed ellipses represent virtual parties and solid circles represent the real parties simulating it (brains are gray). Virtual party $\mathbb{P}_2$ is sending a message to virtual party $\mathbb{P}_3$. The arrows indicate that P_1, the brain of $\mathbb{P}_2$, sends one message to P_2 and one to P_1, which is omitted in reality, since it is sending a message to itself. P_3 stores this message in its transcript.

verification to the end of the computation phase allows us to reduce the total number or redundant messages that are sent. This new simulation strategy crucially relies on the weak active privacy of Π, since we are now allowing the adversary to misbehave up to the opening phase without aborting the protocol execution.

The first phase of $\mathsf{COMP}_{\mathsf{new}}$, where all parties agree on their inputs and random tapes, is identical to that of $\mathsf{COMP}_{\mathsf{old}}$ and is thus equally efficient. In the second phase, our new simulation approach works by selecting one arbitrary real party P_i in each virtual party $\mathbb{P}_j$ to be the brain $\mathrm{B}_j := \mathrm{P}_i$ of that virtual party. The brains will act on behalf of their corresponding virtual parties in an optimistic fashion and execute the computation phase of Π up to the opening phase. All other real parties, the pinkies, will receive the messages that their corresponding virtual parties should receive, which enables them to follow the protocol locally. However, the pinkies will not send any messages during the computation phase. They will only become actively involved in the opening phase to ensure that all brains behaved honestly during the computation phase. Once correctness is ensured, all parties will jointly perform the opening phase of Π. During the computation phase of Π, whenever virtual party $\mathbb{P}_i$ is supposed to send a message to virtual party $\mathbb{P}_j$, we let B_i send one message to each real party in $\mathbb{P}_j$. The receiving real parties do not perform any checks at this moment and just store the message. B_j will optimistically continue the protocol execution on behalf of $\mathbb{P}_j$ according to Π and the received message. This simulation strategy is illustrated in Figure 3.

At the end of the computation phase, all real parties jointly make sure that for each pair $(\mathbb{P}_i, \mathbb{P}_j)$, the sending virtual party $\mathbb{P}_i$ always behaved honestly towards the receiving virtual party $\mathbb{P}_j$. This is accomplished by using a message checking protocol (that implements $\mathcal{F}_{\mathsf{check}}$). If any of these checks output $\mathsf{different}$, then the protocol execution is aborted.

In the opening phase, after passing the previous check, every virtual party is supposed to send its last opening message to all other virtual parties. For each pair $(\mathbb{P}_i, \mathbb{P}_j)$, all real parties in $\mathbb{P}_i$ send the last message to all real parties in $\mathbb{P}_j$. Every receiving party checks that all $t+1$ received messages are consistent and aborts if this is not the case.

In our formal description, let $f(x_1, \ldots, x_n)$ be the n-party functionality that we want to compute. For the sake of simplicity and without loss of generality, we assume that all parties learn the output of the computation. Let $\mathbb{P}_i$ be the virtual party that is simulated by real parties $\mathrm{P}_i, \ldots, \mathrm{P}_{i+t}$. Let $\mathbb{V}_i$ be the set of virtual parties in whose simulation P_i participates.

Let f' be a related n-party functionality that takes as input $\left(x_1^i, \ldots, x_n^i\right)$ from every P_i and outputs $f(\sum_{i=1}^n x_1^i, \ldots, \sum_{i=1}^n x_n^i)$. That is, every party inputs one secret share of every original input. The functionality f' reconstructs the original inputs for f from the secret shares and then evaluates f on those inputs. Let $\Pi_{f'}$ be a passively $\left(t^2+t\right)$-secure protocol with weak privacy that securely implements $\mathcal{F}_{f'}$. The formal description of our compiler is given in Figure 4. Throughout our description we assume that honest parties consider message that they do not receive as malicious and act accordingly.

$\mathsf{COMP}_{\mathsf{new}}(\Pi_{f'})$

Inputs: Each party P_i has input x_i.

---

1. **Input sharing:**
   a. Each P_i secret shares its input $x_i = x_i^1 + \cdots + x_n^i$.
   b. For $1 \leq j \leq n$, each P_i sends $\left(x_i^j, \mathbb{P}_j\right)$ to the broadcast functionality $\mathcal{F}_{\mathsf{bcast}}$.
   c. Each P_i receives $z_j := \left(x_1^j, \ldots, x_n^j\right)$ for each $\mathbb{P}_j \in \mathbb{V}_i$ from the broadcast functionality and aborts if any of the shares equals $\perp$.
2. **Randomness:** Each brain B_i chooses a uniformly random string r_i and sends $(r_i, \mathbb{P}_i)$ to $\mathcal{F}_{\mathsf{bcast}}$. The receiving real parties abort if they receive $\perp$.
3. **Computation phase:** All virtual parties jointly execute the computation phase of $\Pi_{f'}$, where each $\mathbb{P}_i$ uses input z_j and random tape r_i, as follows:
   - Whenever $\mathbb{P}_i$ is supposed to send message m to $\mathbb{P}_j$, the brain B_i sends m to all $\mathrm{P}_k \in \mathbb{P}_j$.
   - Whenever $\mathbb{P}_j$ receives message m, all parties store the message but only B_j continues sending messages according to $\Pi_{f'}$. The pinkies locally follow the protocol and compute the message that they would send.
4. **Check:** At the end of the computation phase, all parties, brains and pinkies, jointly check that the current transcript is valid. For each pair $(\mathbb{P}_i, \mathbb{P}_j)$, for each party $\mathrm{P}_k \in \mathbb{P}_j$, we invoke $\mathcal{F}_{\mathsf{check}}$, where P_k acts as the receiver and $\mathbb{P}_i$ act as the remaining parties. The input of P_k is the list of messages it received from $\mathbb{P}_i$ and the input of all parties from $\mathbb{P}_i$ is the list of messages that they would have sent. If any invocation outputs $\mathsf{different}$, then the protocol execution is aborted.
5. **Opening phase:**
   a. For each pair $(\mathbb{P}_i, \mathbb{P}_j)$, all real parties in $\mathbb{P}_i$ send the last message of $\Pi_{f'}$ to all real parties in $\mathbb{P}_j$.
   b. Every real party in $\mathbb{P}_j$ checks that all received messages are equal. If they are it obtains the output of the computation and otherwise it aborts.

Figure 4 Formal description of our compiler.

▶ **Theorem 1.** *Let $n \geq 3$. Assume $\Pi_{f'}$ implements n-party functionality $\mathcal{F}_{f'}$ with $\left(t^2 + t\right)$-weak privacy. Then, $\mathsf{COMP}_{\mathsf{new}}(\Pi_{f'})$ implements functionality $\mathcal{F}_f$ with active security under individual abort against t corruptions. If $\Pi_{f'}$ has a total bandwidth cost of ℓ messages, then $\mathsf{COMP}_{\mathsf{new}}(\Pi_{f'})$ has a total bandwidth cost of $\mathcal{O}\left(\ell \cdot t + t^2\right)$ messages.*

Proof. Our proof closely follows the proof of [27] for the $\mathsf{COMP}_{\mathsf{old}}$ compiler. Let $\mathbb{P}^*$ be the set of corrupted real parties and let $\mathbb{V}^*$ be the set of virtual parties that are simulated by at least one corrupt real party. Let $\mathcal{S}_{f'}$ be the simulator of the $(t^2 + t)$-weakly private protocol $\Pi_{f'}$. We will use this simulator to construct a simulator $\mathcal{S}$ for the overall actively secure protocol $\mathsf{COMP}_{\mathsf{new}}(\Pi_{f'})$. The simulator $\mathcal{S}$ works as follows:

1. For each party $\mathrm{P}_i \in \mathbb{P}^*$ and $j \in [n]$, the adversary $\mathcal{Z}$ sends $(x_i^j, \mathbb{P}_j)$ to the ideal functionality $\mathcal{F}_{\mathsf{bcast}}$, which is emulated by the simulator $\mathcal{S}$. For any invocation that involves a corrupted party, the environment decides which outputs are $\perp$ and which get delivered. For each $\mathbb{P}_j \in \mathbb{V}^*$ and each corrupt real party in $\mathbb{P}_j$, we send back $(x_1^j, \ldots, x_n^j)$, where x_i^j is either the share that was sent by $\mathcal{Z}$ if P_i is corrupt or otherwise a uniformly random share.

2. For each corrupted party $\mathrm{P}_i \in \mathbb{P}^*$, we reconstruct its input as $x_i = \sum_{j=1}^{n} x_i^j$.
3. $\mathcal{S}$ sends the inputs of the corrupted parties to $\mathcal{F}_f$ and receives back the output of the computation $z = f(x_1, \ldots, x_n)$.
4. For each $\mathbb{P}_i \in \mathbb{V}^*$ we consider two cases. If the brain B_i is corrupted, then it chooses a random tape r_i and sends it to $\mathcal{F}_{\mathsf{bcast}}$, which again is simulated by $\mathcal{S}$. If B_i is honest, then the simulator picks a uniformly random r_i and sends it back to $\mathcal{Z}$ on behalf of $\mathcal{F}_{\mathsf{bcast}}$. Again, the environment can decide that some of the outputs in this step will be $\perp$, which will then be handled accordingly by our simulator.
5. At this point, we know the inputs and the random tapes of all virtual parties $\mathbb{P}_i \in \mathbb{V}^*$. We can therefore compute the exact messages that we would expect from an honest party following the protocol. We initialize the simulator $\mathcal{S}_{f'}$ with parties $\mathbb{P}_1 \ldots \mathbb{P}_n$ and the set of corrupted players $\mathbb{V}^*$.
6. When $\mathcal{S}_{f'}$ queries $\mathcal{F}_{f'}$ for the inputs of the corrupted parties, we give it $(x_1^i, \ldots, x_n^i)$ for each $\mathbb{P}_i \in \mathbb{V}^*$.
7. We now describe how to simulate the computation phase of the protocol.
   - $\mathcal{S}$ queries $\mathcal{S}_{f'}$ for the messages that the honest brains send to the corrupted virtual parties. For each message m to some $\mathbb{P}_i \in \mathbb{V}^*$, we send m to each corrupted real party in $\mathbb{P}_i$ (unless the sender received $\perp$ in one of step 1 or 4 of this simulator in which case it sends nothing).
   - $\mathcal{Z}$ outputs the messages that the corrupt parties send to the honest ones. Since we know the input and random tape of each corrupted party, we can see which messages are honestly generated and which are not. Forward the message of the sending brain to $\mathcal{S}_{f'}$ as the message of $\mathbb{P}_i$.
8. At the end of the computation phase, we simulate the check protocol as follows. For each pair $(\mathbb{P}_i, \mathbb{P}_j)$, for each real party $R \in \mathbb{P}_j$, we have one invocation of the functionality $\mathcal{F}_{\mathsf{check}}$. The simulator $\mathcal{S}$ needs to simulate the ideal functionality towards the corrupted parties in each invocation that involves a corrupted party. Note that the inputs of all honest parties to each check are known from the previous part of the simulation. We, at this point, also know whether any of the corrupted brains cheated or not and if so which check invocation should fail. Furthermore, whenever a corrupted party sends a value to the check functionality, we know whether it's the correct one or not. Using the above observations it follows that the simulator always knows how to simulate each invocation of $\mathcal{F}_{\mathsf{check}}$ and when to return $\mathsf{different}$, when to return same, and when to abort the computation.
9. If all checks passed, meaning that the adversary did not misbehave at any point in time, then we continue the simulation. The simulator $\mathcal{S}$ knows all the last messages of each corrupted party and it knows the output of the functionality z. Since the opening phase is a linear reconstruction of the last messages, the simulator picks a uniformly last message for each honest party under the condition that the linear combination of all last messages results in z. The simulator faithfully executes the last step of the protocol compiler with the corrupted parties. For any simulated honest real party that receives incorrect messages from $\mathcal{Z}$, we will instruct $\mathcal{F}_f$ to make this party abort. For any honest real party that receives the correct last round messages, we instruct $\mathcal{F}_f$ to deliver the output of the computation.

The simulation of the first protocol phase (steps 1-4) is perfect. The adversary sees uniformly random shares, random tapes, or the things it sent itself just like in a real execution. The simulation of $\mathcal{F}_{\mathsf{bcast}}$ is identical to a real execution. The indistinguishability of the simulation in step 6 directly follows from the security guarantees of $\mathcal{S}_{f'}$. As in the real execution, we do

not send anything from real honest parties that may have aborted during the first phase. Otherwise, in both the real and the ideal world, the protocol does not abort during the computation phase. During the computation phase $\mathcal{S}$ has access to the random tapes and inputs of the corrupted parties, thus always knows when and where cheating occurred. This enables to correctly determine when and where the protocol would abort and simulate the outcome of the check phase in step 8 correctly. ◀

Similar to [27], we proved our result for the case of active security with individual abort, where some honest parties may terminate, while some may not. As in their work, our result easily extends to unanimous abort with one additional round of secure broadcast.

4 Efficient Three-Party Computation

All of our protocols are fundamentally based on the seminal work of Beaver [5], who presented a conceptually simple and clean approach for passively secure circuit evaluation. We present two flavors of protocols. One with preprocessing and two different instantiations of the preprocessing phase and one without preprocessing, but some light postprocessing. The protocols that involve preprocessing have a larger *total* runtime, but a leaner online phase, whereas the postprocessing protocol has a smaller overall runtime. Our protocol with preprocessing is presented in this section and the two different preprocessing protocols are presented in Section 5. Protocol with postprocessing is presented in Section 6.

4.1 Beaver's Circuit Evaluation Approach

The circuit evaluation approach by Beaver [5] enables, in our case, three parties to evaluate an arithmetic circuit f over arbitrary rings $\mathbb{Z}_{2^m}$ with security against two passive corruptions. The protocol is split into a preprocessing and an online phase. During the preprocessing phase the parties jointly generate some function-independent correlated randomness in the form of additively secret shared multiplication triples $[a_i]_m$, $[b_i]_m$, $[c_i]_m$, where $c_i = a_i \cdot b_i \mod 2^m$. In the online phase these triples are then consumed to securely evaluate some desired function f. Beaver's online phase works in three steps. First, all parties additively secret share their input among the other parties. Then, all parties jointly evaluate the circuit in a gate-by-gate fashion on the secret shared values. Additions are performed locally, and multiplications require interaction as well as correlated randomness as explained in Section 2.2. In the last step, the parties jointly reconstruct the secret shared values of the output wires of the circuit. Note that the reconstruction phase is just a linear function of the messages received during the opening phase.

▶ **Proposition 2.** *Let f be an arithmetic circuit with N multiplication gates. Given N preprocessed multiplication triples, the three-party protocol* Beaver_f*, implements functionality $\mathcal{F}_f$ with 2-weak privacy and has linear reconstruction.*

4.2 Our Protocol with Preprocessing

We focus on the popular setting with three parties and one active corruption and obtain our protocol by applying Theorem 1 to Beaver's circuit evaluation approach. Let f be the three-party functionality that shall be computed, where each party P_i has an input $x_i \in \mathbb{Z}_{2^m}$. As before, let f' be the related three-party functionality that first recomputes the original inputs from the additive secret shares and then evaluates f. Let N be the number of multiplication gates in f and assume for the moment that all real parties have

$\mathsf{COMP}_{\mathsf{new}}(\mathsf{Beaver}_{f'})$

Inputs: Each party P_i has input $x_i \in \mathbb{Z}_{2^m}$ and they all share preprocessed triples $[\![a_j]\!]_m, [\![b_j]\!]_m, [\![c_j]\!]_m$ for $j \in \{1, \ldots, N\}$.

1. **Input sharing:**
   a. Each P_i picks $x_i^1, x_i^2 \leftarrow \mathbb{Z}_{2^m}$ and sets $x_i^3 = x_i - x_i^1 - x_i^2 \mod 2^m$.
   b. For $1 \le j \le 3$, each P_i sends $\left(x_i^j, \mathbb{P}_j\right)$ to the broadcast functionality $\mathcal{F}_{\mathsf{bcast}}$.
   c. Each P_i receives $z_{i-1} := \left(x_1^{i-1}, x_2^{i-1}, x_3^{i-1}\right)$ and $z_{i+1} := \left(x_1^{i+1}, x_2^{i+1}, x_3^{i+1}\right)$ via $\mathcal{F}_{\mathsf{bcast}}$ and aborts if any of the shares equals $\perp$.
2. **Randomness:** Each brain B_i chooses a uniformly random string r_i and sends $(r_i, \mathbb{P}_i)$ to $\mathcal{F}_{\mathsf{bcast}}$. The receiving real parties abort if they receive $\perp$.
3. **Computation phase:** All virtual parties now evaluate $\mathsf{Beaver}_{f'}$ in a gate-by-gate fashion, where each $\mathbb{P}_i$ uses input z_i as follows:
   - Multiplication gates are evaluated using correlated randomness.
   - Other gates are executed locally.
4. **Check:** For each pair $(\mathbb{P}_i, \mathbb{P}_j)$, we use $\mathcal{F}_{\mathsf{check}}$ to verify the correctness of the messages sent from $\mathbb{P}_i$ to $\mathbb{P}_j$.
5. **Opening phase:**
   a. For each output wire w, for each pair $(\mathbb{P}_i, \mathbb{P}_j)$, all real parties in $\mathbb{P}_i$ send their secret share of w to all real parties in $\mathbb{P}_j$.
   b. For all $\mathbb{P}_j$, all $\mathrm{P}_k \in \mathbb{P}_j$ check that all received are equal. If not, abort.
   c. If all received shares are consistent, then reconstruct the output and terminate.

Figure 5 Three-party arithmetic circuit evaluation in $\mathbb{Z}_{2^m}$ with active security with abort against one active corruption.

already shared this many *replicated* secret shares of multiplication triples $[\![a_i]\!]_m$, $[\![b_i]\!]_m$, $[\![c_i]\!]_m$ in a preprocessing phase[5]. Our concrete preprocessing protocol will be described in detail in Section 5. Since for $1 \le i \le 3$, virtual party $\mathbb{P}_i$ will be simulated by P_{i-1} and P_{i+1}, it holds that real parties holding replicated shares is equivalent to virtual parties holding additive secret shares. This way, one can think of those replicated shares as parts of the real parties' inputs that have already been shared correctly among the virtual parties during preprocessing. We state the compiled protocol $\mathsf{COMP}_{\mathsf{new}}(\mathsf{Beaver}_{f'})$ in Figure 5.

We explicitly state the concrete communication complexity of the protocol. Addition gates require no communication. Evaluating a multiplication gate requires sending 6 words of m bit each. The opening phase, including the checking protocol, requires sending 5 hash values (we choose 256 as the output of the hash) as well as the output shares giving a total of $1280 + 4m \cdot |\mathrm{out}|$ bits for out output gates.

5 Preprocessing

During the preprocessing phase we generate replicated secret sharings of multiplication triples $c = a \cdot b \mod 2^m$. We describe two versions of this phase. The first is obtained combining the preprocessing of Damgård et al. [27] with the batch verification technique of

[5] The functionalities f and f' have equally many multiplication gates, since reconstructing the inputs from additive secret shares does not require any multiplications.

SingleVerify

Inputs: Shared triples $[\![a]\!]_{\mathbb{Z}}, [\![b]\!]_{\mathbb{Z}}, [\![c]\!]_{\mathbb{Z}}$ and $[\![x]\!]_p, [\![y]\!]_p, [\![z]\!]_p$ with prime $p > c$

Interpret $[\![a]\!]_{\mathbb{Z}}, [\![b]\!]_{\mathbb{Z}}, [\![c]\!]_{\mathbb{Z}}$ as a triple in $\mathbb{Z}_p$.

1. Parties generate a random $[\![r]\!]_p$ and open r.
2. Parties compute $[\![e]\!]_p = r[\![x]\!]_p + [\![a]\!]_p$.
3. Parties compute $[\![d]\!]_p = [\![y]\!]_p + [\![b]\!]_p$.
4. Open e, d and compute $[\![t]\!]_p = de - rd[\![x]\!]_p - e[\![y]\!]_p + r[\![z]\!]_p - [\![c]\!]_p$
5. Open $[\![t]\!]_p$ and output success if $t = 0$ and fail otherwise.

Figure 6 Verification of triple $([\![a]\!]_{\mathbb{Z}}, [\![b]\!]_{\mathbb{Z}}, [\![c]\!]_{\mathbb{Z}})$ sacrificing one triple to check the other.

Ben-Sasson et al. [7]. The generation of multiplication triples is split in three steps. First, based on Damgård et al., we optimistically generate secret shared multiplication triples over the integers. Next, we interpret them as triples in a field $\mathbb{Z}_p$, for some sufficiently large prime p, and perform the batch verification protocol of Ben-Sasson et al. to ensure that all triples are correct. Lastly, we reduce all integer shares modulo 2^m to obtain shares of multiplication triples in our desired ring $\mathbb{Z}_{2^m}$[6].

The second version is inspired by Cramer et al. [21], where MACs modulo some prime are replaced with MACs modulo $2^{m+\lambda}$ with 2^m being the "plaintext space" and λ being a statistical security parameter. Hence, we replace the computation modulo prime in the correctness check with a check performed over $2^{m+\lambda}$, which still guarantees security. This is efficient because we avoid the computational complexity of computing modulo primes.

5.1 [27]-style Preprocessing

Optimistic generation of a multiplication triple over the integers is straightforward. First each party P_i uses replicated secret sharing over the integers to share random values $a_i, b_i \in \mathbb{Z}_{2^m}$. All parties jointly compute $[\![a]\!]_{\mathbb{Z}} = \sum_{i=1}^{3} [\![a_i]\!]_{\mathbb{Z}}$ and $[\![b]\!]_{\mathbb{Z}} = \sum_{i=1}^{3} [\![b_i]\!]_{\mathbb{Z}}$ and then use the optimistic multiplication of replicated secret shares from Section 2.5 to compute $[\![c]\!]_{\mathbb{Z}}$.

Given an optimistically generated triple $[\![a]\!]_{\mathbb{Z}}, [\![b]\!]_{\mathbb{Z}}, [\![c]\!]_{\mathbb{Z}}$, the verification of [27] proceeds as follows. First, optimistically generate another multiplication triple in $\mathbb{Z}_p$, where p is a prime such that $p > c$. Then parties interpret the multiplication triple over the integers as a triple in $\mathbb{Z}_p$ and employ the standard technique of "sacrificing" one triple to check the other [26]. Concretely, the authors sacrifice the triple in $\mathbb{Z}_p$ to check $[\![a]\!]_{\mathbb{Z}}$, $[\![b]\!]_{\mathbb{Z}}$, $[\![c]\!]_{\mathbb{Z}}$. The check, SingleVerify, is detailed in Figure 6. The rationale behind this approach is that if the multiplicative relation $a \cdot b = c$ holds over the integers, then it also holds in $\mathbb{Z}_p$ and vice versa since $p > c$ and thus no wrap-around due to the modulo operation happens.

Given N optimistically generated multiplication triples $[\![a_i]\!]_{\mathbb{Z}}$, $[\![b_i]\!]_{\mathbb{Z}}$, $[\![c_i]\!]_{\mathbb{Z}}$ over the integers, we would like to efficiently check that, for all $i \in \{1, \dots N\}$, the multiplicative relationship $a_i \cdot b_i = c_i$ holds. Checking every multiplication triple separately, would require us to generate N additional multiplication triples in $\mathbb{Z}_p$ and perform N invocations of SingleVerify.

Instead, we use a clever verification idea of Ben-Sasson et al. [7] to verify N triples with N additional optimistic multiplications and a *single* invocation of SingleVerify. The idea is to encode all multiplication triples $(a_1, b_1, c_1), \dots, (a_N, b_N, c_N)$ as three polynomials (f, g, h),

[6] Valid multiplication triples over integers are valid modulo 2^m.

where the relation $f \cdot g = h$ will hold iff all multiplication triples are correct. Then we will verify that the polynomial relation $f(x) \cdot g(x) \equiv h(x)$ holds.

More concretely, let f and g be degree N-1 polynomials over $\mathbb{Z}_p$ uniquely defined as $f(i) = a_i$ and $g(i) = b_i$. Since, we expect h to be $f \cdot g$ and thus of degree $2N - 2$, we require $2N - 1$ points to uniquely define it. For $i \in \{1, \dots N\}$, we set $h(i) = c_i$. For $i \in \{N + 1, \dots 2N - 1\}$, we set $h(i) = f(i) \cdot g(i)$, where the multiplication is performed optimistically. If all multiplication triples and optimistic multiplications are correct, then $f \cdot g = h$ holds and an evaluation at a random point z will always fulfill $f(z) \cdot g(z) \equiv h(z) \mod p$. If, however, some multiplication triple is not valid, then $f \cdot g \neq h$ and the two polynomials $f \cdot g$ and h can agree on at most $2N - 2$ many points. This means that for a uniformly random point $z \in \mathbb{Z}_p$, we have $\Pr[f(z) \cdot g(z) = h(z) \mid f \cdot g \neq h] \leq \frac{2N-2}{|\mathbb{Z}_p|}$.

This algorithm relies on the fact that we can interpolate and evaluate additively secret shared polynomials. Given shares of points $[\![a_i]\!]_p$ for $i \in \{1, \dots, N\}$ of polynomial f, we would like to evaluate $f(z)$. Define an extension of Kronecker delta $\delta_i^N(x)$ as

$$\delta_i^N(x) := \prod_{j=1, j\neq i}^{N} \frac{x-j}{i-j} = \begin{cases} 1 & x = i \\ 0 & x \neq i, \ x \leq N \end{cases} \text{ giving } [\![f(z)]\!]_p = \sum_{i=1}^{N} \left(\delta_i^N(z) \cdot [\![a_i]\!]_p\right)$$

where $f(x)$ is evaluated locally. Batch verification protocol is formalized in Figure 7. Its security directly follows from the security of the preprocessing of Damgård et al. and the batch verification protocol of Ben-Sasson et al. Let Π_{Triple} be the resulting preprocessing protocol that first optimistically generates N triples over the integers, then executes the batch verification, and finally reduces all shares modulo 2^m.

5.2 [21]-style Preprocessing

As in the previous subsection, optimistic generation of a multiplication triple is straightforward. This time, the parties (using replicated secret sharing), generate random sharings $[\![x]\!]_{m,\lambda}$, $[\![y]\!]_{m,\lambda}$ and then use the optimistic multiplication protocol to compute $[\![z]\!]_{m,\lambda}$.

We present a verified multiplication protocol in $\mathbb{Z}_{2^m}$ where, in order to mitigate zero divisors, most of the computation is executed in $\mathbb{Z}_{2^{m+\lambda}}$ for a statistical parameter λ. The techniques used in this approach are inspired by the protocol SPD$\mathbb{Z}_{2^k}$ [21]. However, here they are used in a very different context, since SPD$\mathbb{Z}_{2^k}$is a protocol for the dishonest majority case (and therefore their preprocessing phase requires expensive public-key operations), while our honest majority protocol can be instantiated using only cheap arithmetic operations.

Figure 8 presents the core of our protocol with replicated sharing with redundant shares. The protocol uses the sacrifice step of the MASCOT protocol [34]. Note that generating the share of a and the value r can be done non-interactively using PRSS. We now evaluate the properties of our protocol. The correctness follows from the correctness of optimistic multiplication and that the fact that $rz + c - ey = rxy + ay - (rx + a) \cdot y = 0 \mod 2^{m+\lambda}$.

Assuming that all openings are verified using $\mathcal{F}_{\mathsf{check}}$ (which ensures that a corrupt party cannot send different shares to different parties), the corrupt party can only deviate by adding an additive error in the optimistic products. We define the (potential) errors as $z = xy + \epsilon_z$ and $c = ay + \epsilon_c$. If the input tuple $[\![x]\!]_{m,\lambda}, [\![y]\!]_{m,\lambda}, [\![z]\!]_{m,\lambda}$ is incorrect then we have that $\epsilon_z \neq 0 \mod 2^m$. Inserting into the check equation, we get $rxy + r\epsilon_z + ay + \epsilon_c - (rx + a)y = r\epsilon_z + \epsilon_c \mod 2^{m+\lambda}$. Here $r\epsilon_z \mod 2^{m+\lambda}$ is uniform in a set of at least size 2^λ as r is a uniformly random λ-bit number. Since the ϵ_c is chosen by the adversary before r is sampled, the adversary will be able to make the protocol accept an incorrect tuple with probability at most $2^{-\lambda}$. This argument corresponds to the proof of Claim 6 of [21].

BatchVerify

Inputs: N preprocessed triples $[\![a_i]\!]_{\mathbb{Z}}$, $[\![b_i]\!]_{\mathbb{Z}}$, $[\![c_i]\!]_{\mathbb{Z}}$ for $i \in \{1, \ldots, N\}$ over the integers. And a uniformly random triple $[\![x]\!]_p, [\![y]\!]_p, [\![z]\!]_p$ in $\mathbb{Z}_p$.

Interpret$[\![a_i]\!]_{\mathbb{Z}}$, $[\![b_i]\!]_{\mathbb{Z}}$, $[\![c_i]\!]_{\mathbb{Z}}$ as a triple in the field $\mathbb{Z}_p$ for a sufficiently large prime p.

1. For $i \in \{1, \ldots, N\}$, define $[\![f(i)]\!]_p := [\![a_i]\!]_p$ and $[\![g(i)]\!]_p := [\![b_i]\!]_p$.
2. For $i \in \{N+1, \ldots, 2N-1\}$, evaluate

$$[\![f(i)]\!]_p := \sum_{j=1}^{N} \left(\delta_j^N(i) \cdot [\![a_j]\!]_p\right), \text{ and } [\![g(i)]\!]_p := \sum_{j=1}^{N} \left(\delta_j^N(i) \cdot [\![b_j]\!]_p\right)$$

3. For $i \in \{1, \ldots, N\}$, define $[\![h(i)]\!]_p := [\![c_i]\!]_p$.
4. For $i \in \{N+1, \ldots, 2N-1\}$, compute $[\![h(i)]\!]_p = [\![f(i)]\!]_p \cdot [\![g(i)]\!]_p$ optimistically.
5. Parties generate a random $[\![z]\!]_p$ and open z.
6. Parties evaluate the polynomials at z:

$$[\![\alpha]\!]_p = [\![f(z)]\!]_p := \sum_{j=1}^{N} \left(\delta_j^N(z) \cdot [\![f(j)]\!]_p\right), \; [\![\beta]\!]_p = [\![g(z)]\!]_p := \sum_{j=1}^{N} \left(\delta_j^N(z) \cdot [\![g(j)]\!]_p\right)$$

$$[\![\gamma]\!]_p = [\![h(z)]\!]_p := \sum_{j=1}^{2N-1} \left(\delta_j^{2N-1}(z) \cdot [\![h(j)]\!]_p\right)$$

7. Test SingleVerify $([\![\alpha]\!]_p, [\![\beta]\!]_p, [\![\gamma]\!]_p, [\![x]\!]_p, [\![y]\!]_p, [\![z]\!]_p)$.

Figure 7 Batch verification of multiplication triples.

SPD$\mathbb{Z}_{2^k}$-like check for correct multiplication

Inputs: Shared triple $[\![x]\!]_{m,\lambda}$, $[\![y]\!]_{m,\lambda}$, $[\![z]\!]_{m,\lambda}$

---

1. Parties generate a random $[\![a]\!]_{m,\lambda}$ and execute an optimistic multiplication with $([\![a]\!]_{m,\lambda}, [\![y]\!]_{m,\lambda})$ to get $[\![c]\!]_{m,\lambda}$.
2. Parties jointly generate a random $r \in \mathbb{Z}_{2^\lambda}$.
3. Parties reveal $[\![e]\!]_{m,\lambda} = r[\![x]\!]_{m,\lambda} + [\![a]\!]_{m,\lambda}$.
4. Parties output the result of the equality check $r[\![z]\!]_{m,\lambda} + [\![c]\!]_{m,\lambda} - e[\![y]\!]_{m,\lambda} \stackrel{?}{=} 0$

Figure 8 Verification of a triple using redundant $[\![\cdot]\!]_{m,\lambda}$ sharing.

Similarly to other uses of this style verification, the check can be batched for an arbitrary number of triples. Batching makes the cost for generating the random number r negligible. Furthermore, the communication for the final step can be reduced: the parties hold $\{[\![x^{(j)}]\!]_m\}$ and would like to check if $x^{(j)} = 0$ for all j, every party P_i computes $x_i^{(j)} = 0 - x_{i-1}^{(j)} - x_{i+1}^{(j)}$. All parties then hash $\{x_0^{(j)}, x_1^{(j)}, x_2^{(j)}\}_{\forall j}$ and broadcast their result. If there is a mismatch, they abort. With these two optimizations, the asymptotic communication per triple is determined by the two optimistic multiplications and the opening of e. All involve sending one $m + \lambda$-bit value to one other party, so we arrive at $3(m + \lambda)$ bits per party and multiplication. If the

Protocol with Postprocessing

Input Each party P_i has input x_i.

1. Each party P_i secret shares its input x_i using the secret sharing with redundant shares as described in Section 2.4 to obtain $[\![x_i]\!]_{m,\lambda}$.
2. All parties jointly evaluate the circuit, where multiplications are performed optimistically as described in Section 2.3. Let $\{([\![x_i]\!]_{m,\lambda}, [\![y_i]\!]_{m,\lambda}, [\![z_i]\!]_{m,\lambda})\}_{i\in[N]}$ be the set of all the performed multiplications, where x_i, y_i are the left and right inputs and z_i is the output of the i-th multiplication.
3. For $i \in [N]$, compute a random $[\![a_i]\!]_{m,\lambda}$ and optimistically compute $[\![c_i]\!]_{m,\lambda} = [\![y_i \cdot a_i]\!]_{m,\lambda}$.
4. All parties generate a random value $r \in \mathbb{Z}_{2^\lambda}$.
5. Parties open $[\![e_i]\!]_{m,\lambda} = r[\![x_i]\!]_{m,\lambda} + [\![a_i]\!]_{m,\lambda}$
6. For each $i \in [N]$ the parties check $r[\![z_i]\!]_{m,\lambda} + [\![c_i]\!]_{m,\lambda} - e_i[\![y_i]\!]_{m,\lambda} \stackrel{?}{=} 0$ and abort if any of these checks fails.
7. If all checks passed, then the parties jointly open the outputs of the circuit.

Figure 9 Protocol for secure circuit evaluation that does not require a preprocessing phase.

protocol is used for preprocessing there are another m bits sent per party and multiplication during the online phase, otherwise $3(m + \lambda)$ is the total cost per multiplication. For the common choice of $m = 64$ and $\lambda = 40$, this gives a total of 312 bits.

6 Protocol with Postprocessing

Our postprocessing protocol is similar in spirit to our [21]-style preprocessing protocol in Section 5.2. We use similar building blocks, but in a different order, which allows us to reduce the total computation time at the cost of a slightly more expensive online phase. We describe our protocol in Figure 9 for the three party setting. For the sake of simplicity we describe the protocol with separate checks for each multiplication gate, but optimizations like the batch verification described in Section 5.2 can be applied equally well to this protocol. The security proof for the protocol is completely analogous to the security proof in Section 5.2.

7 Implementation and Evaluation

To help adoption and accessibility of our protocols, we implemented them using Sharemind [9] and the MP-SPDZ framework [29]. We provide extensive benchmarks in both LAN and WAN settings for both implementations as well as a theoretical analysis of the asymptotic communication. Throughout this section, we use a statistical security parameter $\lambda = 40$.

Sharemind already supported semi-honest computation in $\mathbb{Z}_{2^{32}}$ and $\mathbb{Z}_{2^{64}}$. We added [27]-style preprocessing with BatchVerify from Section 5.1 and postprocessing from Section 6.

MP-SPDZ already supported replicated secret sharing in $\mathbb{Z}_{2^{64}}$ and $\mathbb{Z}_p$ as well as the protocol for $\mathbb{Z}_p$ by Lindell and Nof [35]. Its use of C++ templating easily allows to add new protocols reusing existing components, and it provides an efficient implementation of $\mathbb{Z}_{2^k}$ arithmetic for any k. It also uses Montgomery representation for arithmetic modulo a prime. We have added the following protocols: [27]-style preprocessing with SingleVerify from Section 5.1, cut-and-choose preprocessing of triples similar to Araki et al. [3] (simple version), and [21]-style preprocessing as well as postprocessing from Section 5.2.

Table 1 Communication bits per party for $\mathbb{Z}_{2^{64}}$ multiplication.

	Offline	Online	Total
DOS18 preprocessing (single)	992	128	1120
DOS18 preprocessing (batch)	464	128	592
ABF+17 preprocessing (simple)	448	128	576
CDE+18 preprocessing	312	128	440
Postprocessing	-	312	312
Semi-honest	-	64	64
Malicious ASTRA [19]	448	85	553

7.1 Communication

Table 1 shows the communication complexity per multiplication in $\mathbb{Z}_{2^{64}}$ with the various protocols for $\lambda = 40$. While the numbers are obtained from running the protocols in batches of at least one million with rounding, they match the asymptotic cost one would expect from a manual analysis. For comparison, we have added the figures reported in a recent concurrent work by Chaudhari et al. [19] (averaged over the parties because their protocol is asymmetric) that also considers honest majority three party case.

One optimistic multiplication in $\mathbb{Z}_{2^m}$ requires sending m bits, and using Beaver multiplication in the data-dependent phase requires opening two masked values, thus sending $2m$ bits. A CDE+18-style sacrifice [21] requires two optimistic multplications and one opening in $\mathbb{Z}_{2^{m+\lambda}}$, while simple ABF+17 [3] asymptotically requires three optimistic multiplications and two classic sacrifices that require two openings each.[7] This comes down to $7m$ bits.[8] Finally, DOS18 preprocessing [27] with SingleVerify requires two optimistic multiplications in $\mathbb{Z}_p$ and two openings in $\mathbb{Z}_p$ as well as sending two $(m + \lambda)$-bit values for sharing over the integers, totalling in $2(m + \lambda) + 3 \log p$ bits. It roughly holds that $\log p > 7 + 2m + 3\lambda$, so for our choice of parameters $\log p > 255$.[9] The slight difference to the figure in the table comes from rounding up to multiples of eight. Using BatchVerify in Figure 7 allows to avoid the openings in $\mathbb{Z}_p$, bringing the complexity to slightly more than ABF+17 preprocessing.

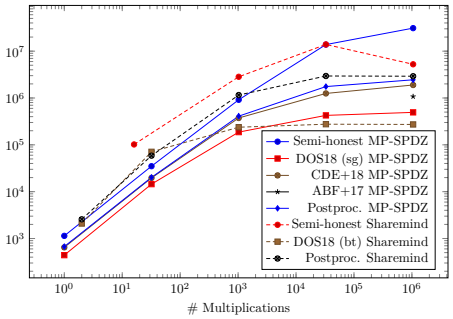
7.2 Benchmarks

We have run our implementations in SIMD fashion, that is, combining the communication of a varying number of multiplications in as few network messages as possible. All benchmarks in this section are averages over ten executions. Unsurprisingly up to a certain number the throughput increases. Figure 10a shows our benchmarks for various numbers of parallel multiplications in a LAN (AWS `c5.9xlarge` instances in the same region). We have 36 virtual CPUs, 72 GiB of RAM, and 10 Gbit/s network network connection. The figure for cut-and-choose is limited to 1048576 because the analysis by Araki et al. [3] mandates batches of at least this size. The plot shows that all malicious protocols perform similarly except the [27] protocol, and that the postprocessing protocol is slightly ahead as we expected.

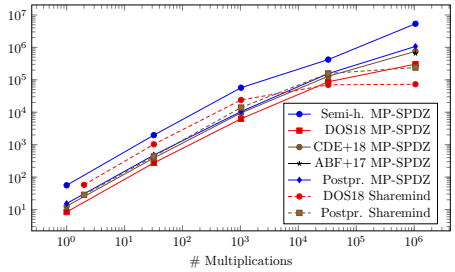
[7] Because of cut-and-choose we cannot use the trick used for DOS18-style sacrificing.

[8] The more sophisticated preprocessing of Araki et al. [3] would cost $5m$, which is still slightly more than a CDE+18-style sacrifice.

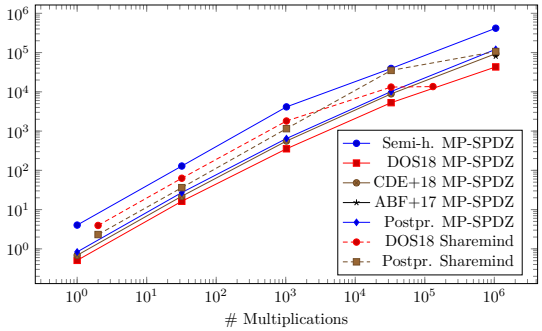
[9] According to Damgård et al. [27], $p > 100 \cdot 2^{2m+2\lambda}$, but a quick recalculation of $24 \cdot B^2 2^\lambda$ with $B = 2^{m+\lambda+1}$ shows that it should be 3λ instead of 2λ in the inequality for p.



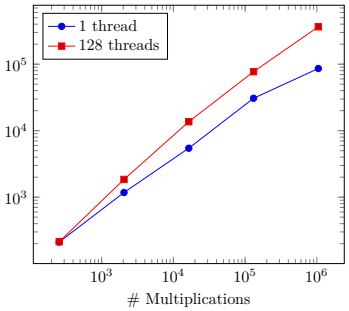
(a) Throughput in LAN.



(b) Throughput in continental WAN.



(c) Throughput in global WAN.



(d) Postprocessing on MP-SPDZ in global WAN.

Figure 10 Comparison of 64-bit multiplication throughput (multiplications/s).

Figure 10b shows our benchmarks for various numbers of parallel multplications in a continental WAN, that is one AWS `c5.9xlarge` instance in each of Frankfurt, London, and Paris. The results mirror the results in the LAN setting except for the fact that 2^{20} parallel multiplications perform better than 2^{15} for all protocols. This is most likely because of the increased network delay of up to 12 ms. Finally, Figure 10c shows benchmarks for a global WAN, that is one AWS `c5.9xlarge` instance in each of Frankfurt, Northern California, and Tokyo. The largest network latency we observed is 236 ms in this setting.

We generally found that our protocols do not use all bandwidth that is available. Figure 10d supports this by showing that increasing from a single thread to 128 increases the output while keeping same the number of parallel multiplications.

7.3 Comparison with Other Implementations

We provide a comparison with the most relevant previous implementations. The concurrent work of Choudhary et al. [19] does not provide throughput of multiplications for their offline phase, only for more complex computation such as AES evaluation. It is therefore hard to compare their implementation to ours. Furthermore, AES evaluation does not lend itself to computation in $\mathbb{Z}_{2^k}$ for $k > 1$, which makes a rather odd benchmark in this setting.

Three party honest majority actively secure multiplication with 61-bit Mersenne field is implemented in [20] where they measure that a circuit with 10^6 multiplication gates and depth 20 can be evaluated in 0.3 seconds in a single AWS region (presumably 10 Gbit/s networks). This amounts to a throughput of 3.3 million multiplications per second, while our postprocessing protocol with Sharemind in a LAN achieved 2.9 million for a slightly smaller batch size that $10^6/20$. This shows that our protocol is competitive despite the extra effort needed for rings as compared to fields.

For a 31-bit prime, [16] report a throughput of 1.7 million multiplications per second for the computation of their verification protocol on a single core of an AWS c5.9xlarge instance ($s = 128$ in Table 3 ibidem). In comparison, we achieve 2.9 million in a LAN setting including communicatio). In the WAN settings we benchmark below 1 million (continentally) or 0.2 million (globally). Also, note that their verification protocol for $\mathbb{Z}_{2^{64}}$ requires computation in the ring $(\mathbb{Z}_{2^{64}})[X]/f(X)$ with f being a polynomial of degree 47 while the benchmarked protocol for fields does not require an extension field. It is therefore likely that the throughput of their protocol for $\mathbb{Z}_{2^{64}}$ is significantly lower.

The batchwise multiplication verification is optimized in [37]. The authors estimate that their computation optimizations achieves up to 10^7 two-party verifications per second using multithreading for 64-bit primes and up to $5 \cdot 10^6$ with 128-bit prime. Their estimations are based on their implementation of the computations, they do not benchmark the protocol with communication. From a conceptual point of view, [37] uses similar verification as our batch verification, hence their work indicates that our implementation might also benefit from more optimized field arithmetic. However, we still need to use larger fields to accommodate the integer secret sharing meaning we need more communication to achieve triples modulo 2^k of the same length as their modulo prime triples.

References

1 Mark Abspoel, Ronald Cramer, Ivan Damgård, Daniel Escudero, and Chen Yuan. Efficient information-theoretic secure multiparty computation over $\mathbb{Z}/p^k\mathbb{Z}$ via galois rings. Cryptology ePrint Archive, Report 2019/872, 2019. URL: `https://eprint.iacr.org/2019/872`.

2 Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. MCMix: Anonymous messaging via secure multiparty computation. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1217–1234, Vancouver, BC, 2017. USENIX Association. URL: `https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/alexopoulos`.

3 Toshinori Araki, Assi Barak, Jun Furukawa, Tamar Lichter, Yehuda Lindell, Ariel Nof, Kazuma Ohara, Adi Watzman, and Or Weinstein. Optimized honest-majority MPC for malicious adversaries - breaking the 1 billion-gate per second barrier. In *2017 IEEE Symposium on Security and Privacy*, pages 843–862, San Jose, CA, USA, May 22–26 2017. IEEE Computer Society Press. `doi:10.1109/SP.2017.15`.

4 David W. Archer, Dan Bogdanov, Yehuda Lindell, Liina Kamm, Kurt Nielsen, Jakob Illeborg Pagter, Nigel P. Smart, and Rebecca N. Wright. From keys to databases - real-world applications of secure multi-party computation. *The Computer Journal*, 61(12):1749–1771, 2018. `doi:10.1093/comjnl/bxy090`.

5 Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432, Santa Barbara, CA, USA, August 11–15 1992. Springer, Heidelberg, Germany. `doi:10.1007/3-540-46766-1_34`.

6 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, May 2–4 1988. ACM Press. `doi:10.1145/62212.62213`.

7 Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 663–680, Santa Barbara, CA, USA, August 19–23 2012. Springer, Heidelberg, Germany. `doi:10.1007/978-3-642-32009-5_39`.

8 Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188, Tallinn, Estonia, May 15–19 2011. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-20465-4_11.

9 Dan Bogdanov. *Sharemind: programmable secure computations with practical applications*. PhD thesis, University of Tartu, 2013.

10 Dan Bogdanov, Marko Jõemets, Sander Siim, and Meril Vaht. How the estonian tax and customs board evaluated a tax fraud detection system based on secure multi-party computation. In Rainer Böhme and Tatsuaki Okamoto, editors, *FC 2015: 19th International Conference on Financial Cryptography and Data Security*, volume 8975 of *Lecture Notes in Computer Science*, pages 227–234, San Juan, Puerto Rico, January 26–30 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-47854-7_14.

11 Dan Bogdanov, Liina Kamm, Baldur Kubo, Reimo Rebane, Ville Sokk, and Riivo Talviste. Students and Taxes: a Privacy-Preserving Study Using Secure Computation. *PoPETs*, 2016(3):117–135, 2016. URL: http://www.degruyter.com/view/j/popets.2016.2016.issue-3/popets-2015-0019/popets-2016-0019.xml.

12 Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In Sushil Jajodia and Javier López, editors, *ESORICS 2008: 13th European Symposium on Research in Computer Security*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206, Málaga, Spain, October 6–8 2008. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-88313-5_13.

13 Dan Bogdanov, Margus Niitsoo, Tomas Toft, and Jan Willemson. High-performance secure multi-party computation for data mining applications. *Int. J. Inf. Secur.*, 11(6):403–418, November 2012. doi:10.1007/s10207-012-0177-2.

14 Dan Bogdanov, Riivo Talviste, and Jan Willemson. Deploying secure multi-party computation for financial data analysis - (short paper). In Angelos D. Keromytis, editor, *FC 2012: 16th International Conference on Financial Cryptography and Data Security*, volume 7397 of *Lecture Notes in Computer Science*, pages 57–64, Kralendijk, Bonaire, February 27 – March 2 2012. Springer, Heidelberg, Germany.

15 Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *FC 2009: 13th International Conference on Financial Cryptography and Data Security*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343, Accra Beach, Barbados, February 23–26 2009. Springer, Heidelberg, Germany.

16 Elette Boyle, Niv Gilboa, Yuval Ishai, and Ariel Nof. Practical fully secure three-party computation via sublinear distributed zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019.*, pages 869–886. ACM, 2019. doi:10.1145/3319535.3363227.

17 Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17 2001. IEEE Computer Society Press. doi:10.1109/SFCS.2001.959888.

18 Dario Catalano, Mario Di Raimondo, Dario Fiore, and Irene Giacomelli. Monza: Fast maliciously secure two party computation on z_{2^k}. Cryptology ePrint Archive, Report 2019/211, 2019. URL: https://eprint.iacr.org/2019/211.

19 Harsh Chaudhari, Ashish Choudhury, Arpita Patra, and Ajith Suresh. Astra: High throughput 3pc over rings with application to secure prediction. Cryptology ePrint Archive, Report 2019/429, 2019. URL: https://eprint.iacr.org/2019/429.

20 Koji Chida, Daniel Genkin, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Yehuda Lindell, and Ariel Nof. Fast Large-Scale Honest-Majority MPC for Malicious Adversaries. In Hovav

Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 34–64, Cham, 2018. Springer International Publishing.

21 Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD $\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 769–798. Springer, 2018. doi:10.1007/978-3-319-96881-0_26.

22 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.

23 Ivan Damgård, Daniel Escudero, Tore Kasper Frederiksen, Marcel Keller, Peter Scholl, and Nikolaj Volgushev. New primitives for actively-secure MPC over rings with applications to private machine learning, 2019. URL: https://eprint.iacr.org/2019/599.

24 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: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 160–179, Irvine, CA, USA, March 18–20 2009. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-00468-1_10.

25 Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013: 18th European Symposium on Research in Computer Security*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18, Egham, UK, September 9–13 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40203-6_1.

26 Ivan Damgård and Claudio Orlandi. Multiparty computation for dishonest majority: From passive to active security at low cost. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 558–576, Santa Barbara, CA, USA, August 15–19 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-14623-7_30.

27 Ivan Damgård, Claudio Orlandi, and Mark Simkin. Yet another compiler for active security or: Efficient MPC over arbitrary rings. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 799–829. Springer, 2018. doi:10.1007/978-3-319-96881-0_27.

28 Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662, Santa Barbara, CA, USA, August 19–23 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-32009-5_38.

29 Data61. MP-SPDZ - Versatile framework for multi-party computation. URL: https://github.com/data61/MP-SPDZ.

30 Matthias Fitzi, Nicolas Gisin, Ueli M. Maurer, and Oliver von Rotz. Unconditional byzantine agreement and multi-party computation secure against dishonest minorities from scratch. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 482–501, Amsterdam, The Netherlands, April 28 – May 2 2002. Springer, Heidelberg, Germany. doi:10.1007/3-540-46035-7_32.

31 Jun Furukawa, Yehuda Lindell, Ariel Nof, and Or Weinstein. High-throughput secure three-party computation for malicious adversaries and an honest majority. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part II*,

volume 10211 of *Lecture Notes in Computer Science*, pages 225–255, Paris, France, May 8–12 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-56614-6_8.

32 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 Annual ACM Symposium on Theory of Computing*, pages 495–504, New York, NY, USA, May 31 – June 3 2014. ACM Press. doi:10.1145/2591796.2591861.

33 Thomas P. Jakobsen, Jesper Buus Nielsen, and Claudio Orlandi. A framework for outsourcing of secure computation. In *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security, CCSW '14, Scottsdale, Arizona, USA, November 7, 2014*, pages 81–92, 2014. doi:10.1145/2664168.2664170.

34 Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 830–842, 2016. doi:10.1145/2976749.2978357.

35 Yehuda Lindell and Ariel Nof. A framework for constructing fast MPC over arithmetic circuits with malicious adversaries and an honest-majority. In *ACM CCS 17: 24th Conference on Computer and Communications Security*, pages 259–276. ACM Press, 2017. doi:10.1145/3133956.3133999.

36 Payman Mohassel and Peter Rindal. ABY^3: A mixed protocol framework for machine learning. In *ACM CCS 18: 25th Conference on Computer and Communications Security*, pages 35–52. ACM Press, 2018. doi:10.1145/3243734.3243760.

37 Peter Sebastian Nordholt and Meilof Veeningen. Minimising communication in honest-majority MPC by batchwise multiplication verification. In Bart Preneel and Frederik Vercauteren, editors, *Applied Cryptography and Network Security*, pages 321–339, Cham, 2018. Springer International Publishing.

38 Martin Pettai and Peeter Laud. Automatic proofs of privacy of secure multi-party computation protocols against active adversaries. In *IEEE 28th Computer Security Foundations Symposium, CSF 2015, Verona, Italy, 13-17 July, 2015*, pages 75–89, 2015. doi:10.1109/CSF.2015.13.

39 Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st Annual ACM Symposium on Theory of Computing*, pages 73–85, Seattle, WA, USA, May 15–17 1989. ACM Press. doi:10.1145/73007.73014.

40 Berry Schoenmakers. MPyC – Secure multiparty computation in Python. GitHub, 2018. URL: https://github.com/lschoe/mpyc.

A Security Definitions

We define security using the UC framework of Canetti [17]. Protocols proven secure in this framework retain security even when composed arbitrarily and executed concurrently. Concretely, we use a flavour of the classical UC framework, proposed in [22]. We provide a short summary of the security framework here and refer the reader to [22] for more details. Security is defined by comparing a real and an ideal interaction. In the ideal interaction, we have a trusted party, called the ideal functionality $\mathcal{F}$, that receives inputs from all parties, computes the desired function, and returns the result to the parties. In the real interaction, the parties do not have $\mathcal{F}$, but rather interact with each other according to some protocol description Π. The protocol Π may make use of some other auxiliary ideal functionality $\mathcal{G}$. In both interactions, the environment $\mathcal{Z}$ chooses the inputs of all parties and acts as an adversary that may corrupt some subset of the parties passively or actively. We say that Π securely realizes $\mathcal{F}$ if an adversary in the real world can not do "more harm" than an adversary in the ideal world. Concretely, we require the existence of a simulator $\mathcal{S}$, aka. ideal world adversary, that simulates $\mathcal{Z}$'s view of a real interaction. $\mathcal{S}$ simulates the views of the corrupted players, the interaction with auxiliary functionality $\mathcal{G}$, and it may interact

with $\mathcal{F}$. At the end of a protocol execution $\mathcal{Z}$ outputs a single bit. Let $\mathsf{IDEAL}_\lambda[\mathcal{Z}, S, \mathcal{F}]$ and $\mathsf{REAL}_\lambda[\mathcal{Z}, \Pi, \mathcal{G}]$ be the random variables that represent $\mathcal{Z}$'s output bit in the ideal and real execution, respectively. We say that Π securely realizes functionality $\mathcal{F}$, if $\mathcal{Z}$ cannot distinguish real interaction from communicating with the simulator $\mathcal{S}$.

▶ **Definition 3.** Π *securely implements functionality* $\mathcal{F}$ *with respect to a class of environments* Env *in the* $\mathcal{G}$*-hybrid model, if there exists a simulator* S *such that for all* $\mathcal{Z} \in Env$ *we have*

$$|\Pr[\mathsf{REAL}_\lambda[\mathcal{Z}, \Pi, \mathcal{G}] = 1] - \Pr[\mathsf{IDEAL}_\lambda[\mathcal{Z}, S, \mathcal{F}] = 1]| \ \leq \mathsf{negl}(\lambda) \ .$$

We capture different security notions by specifying the environments. For passive security the environment $\mathcal{Z}$ can corrupt up to t parties. $\mathcal{Z}$ gets full read-only access to the corrupted parties internal tapes. All parties follow the protocol honestly. The simulator $\mathcal{S}$ is allowed to ask the ideal functionality $\mathcal{F}$ for the inputs of the corrupted parties. For active security the environment $\mathcal{Z}$ is allowed to corrupt up to t parties. $\mathcal{Z}$ gets full control of the corrupted parties. Once the ideal functionality $\mathcal{F}$ received inputs from all parties, it computes the output and sends it to $\mathcal{Z}$. The environment sends back a bit indicating whether the parties should obtain the output or $\perp$. A slightly weaker notion known as active security with individual abort allows the adversary to specify which honest parties abort and which do not.

We use the definition of weak privacy against active adversaries [32, Definition 5.11] (a slight variant of the same property was defined under the name "active privacy" in [38]), which captures the security properties offered by many existing protocols [6, 5] that follow the compute-then-open paradigm. These protocols are split into a computation and opening phase. The computation phase consists of multiple rounds of interaction, whereas the opening phase requires a single round of communication. Intuitively, weak privacy says that an active adversary cannot learn anything until the opening phase, and this is captured saying that there exists a simulator that can simulate the truncated view of the protocol up to the opening phase without having access to the inputs or outputs of any honest parties. Finally, these protocols are "linear", meaning that the output of the parties in the protocol is a linear function of the messages sent in the opening phase.

Expander Graphs Are Non-Malleable Codes

Peter Michael Reichstein Rasmussen
Basic Algorithms Research Copenhagen, University of Copenhagen, Denmark
pmrr@di.ku.dk

Amit Sahai
UCLA, Los Angeles, CA, USA
sahai@cs.ucla.edu

Abstract

Any d-regular graph on n vertices with spectral expansion λ satisfying $n = \Omega(d^3 \log(d)/\lambda)$ yields a $O\left(\frac{\lambda^{3/2}}{d}\right)$-non-malleable code for single-bit messages in the split-state model.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives; Mathematics of computing → Spectra of graphs

Keywords and phrases Non-Malleable Code, Expander Graph, Mixing Lemma

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.6

Funding *Peter Michael Reichstein Rasmussen*: Supported in part by grant 16582, Basic Algorithms Research Copenhagen (BARC), from the VILLUM Foundation.
Amit Sahai: Supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, and NSF grant 1619348, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant rom Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C- 0205. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

Acknowledgements A significant effort was made to simplify our proof as much as possible, which eventually resulted in the approximately 2-page proof of our main result presented here; we thank Anders Aamand and Jakob Bæk Tejs Knudsen for suggestions and insights regarding the main theorem that helped simplify and improve the results presented. Furthermore, we thank Aayush Jain, Yuval Ishai, and Dakshita Khurana for early discussions regarding simple constructions of split-state non-malleable codes not based on expander graphs.

1 Introduction

A key goal in theoretical computer science is the identification of structures that exhibit resilience to adversarial tampering. The classical notion in this space is that of an error-detection or error-correction code, where we seek to ensure that tampering caused by an adversary that can modify a *bounded* number of symbols in a codeword can be detected or corrected.

But what if the number of errors that an adversary can introduce is unbounded? The objective of error detection or correction is clearly impossible to achieve in this setting – the adversary can simply replace the transmitted codeword with an encoding of some other fixed value. Thus, the main question of study in this context concerns the notion of *malleability*: informally speaking, our core goal must be to prevent the adversary from replacing an encoding of a value x with an encoding of some other related value $\tilde{x} \neq x$.

The central information-theoretic object in this setting is called a split-state non-malleable code [5]. Since their introduction in 2010 [5], split-state non-malleable codes have been the subject of intense study within theoretical computer science [5, 4, 1, 3, 2, 6]. Here,

we consider the most basic form of a split-state non-malleable code, namely a code for encoding a single bit. A split-state non-malleable code [5] for single-bit messages consists of randomized encoding and decoding algorithms $(\mathrm{enc}, \mathrm{dec})$. A message $m \in \{0,1\}$ is encoded as a pair of strings $(L, R) \in \{0,1\}^k \times \{0,1\}^k$, such that $\mathrm{dec}(L, R) = m$. An adversary then specifies an arbitrary pair of functions $g, h : \{0,1\}^k \to \{0,1\}^k$. The code is said to be non-malleable if, intuitively, the message obtained as $\mathrm{dec}(g(L), h(R))$ is "unrelated" to the original message m. In particular, to be ε-non-malleable, it is enough [4] to guarantee that when the message m is chosen uniformly at random and encoded into (L, R), the probability that $\mathrm{dec}(g(L), h(R)) = 1 - m$ is at most $\frac{1}{2} + \varepsilon$.

1.1 Previous Work

All known constructions and proofs of security for explicit split-state non-malleable codes have required complex mathematical proofs, and all known such proofs either directly or indirectly used the mathematics behind constructions of two-source extractors [4, 1, 3, 2, 6]. In fact, after constructing the first non-malleable code in the split-state model Dziembowski, Kazana, and Obremski wrote: "This brings a natural question if we could show some relationship between the extractors and the non-malleable codes in the split-state model. Unfortunately, there is no obvious way of formalizing the conjecture that non-malleable codes need to be based on extractors" [4].

1.2 Our Contribution

In this work, we seek to establish new, simpler, foundations for the construction of single-bit split-state non-malleable codes. We do so by answering in the negative the implicit conjecture of [4]; we show that it is not necessary to base constructions of non-malleable codes on the theory of extractors.

Specifically, we show that expander graphs immediately give rise to split-state non-malleable codes for single-bit messages. We prove that any d-regular graph on $n = 2^k$ nodes with spectral expansion λ satisfying $n = \Omega(d^3 \log(d)/\lambda)$ yields a $O\left(\frac{\lambda^{3/2}}{d}\right)$-non-malleable code for single-bit messages in the split-state model. Our proof is elementary, requiring a little more than two pages to prove, having at its heart two nested applications of the Expander Mixing Lemma. Furthermore, we only need expanders of high degree (e.g., $d = n^{1/3}$), which can be constructed and analyzed easily (see, e.g., [7] or Appendix C), yielding $2^{-\Omega(k)}$-non-malleable codes. It is worth noting that the manner in which we construct a single-bit code from an expander graphs is similar to how [4] constructs a single-bit code from a two-source extractor. Thus, our main discovery is that expander graphs suffice for such a construction to succeed.

Our construction of non-malleable codes from expander graphs thus opens up a new line of attack in the study of split-state non-malleable codes. It is important to keep in mind that current constructions of non-malleable codes supporting messages of arbitrary length use many ideas pioneered in the construction of [4], in particular the use of extractors. While we do not yet know how to generalize our results beyond single-bit messages, we speculate that further investigation building upon our work will reveal a deeper connection and more powerful simple constructions based on expanders.

It should be noted that two-source extractors are well-known to exhibit expansion properties; however, in all previous proofs, much more than mere expansion was used to argue non-malleability. Indeed previous proofs apply extractors repeatedly; for instance the proof of [4] uses the extractor property multiple times (e.g., in equation (22) and using

equation (43) in [4]). We also note that it is not surprising that 1-bit non-malleable codes will exhibit some sort of expansion properties. Our contribution is the converse: that good expansion is *sufficient* for the construction of non-malleable codes.

1.3 Parameters and a Comparison with DKO13

For completeness we include an analysis of the concrete parameters of our resulting code. Let $\gamma > 0$ be given. Our construction yields a 1-bit γ-non-malleable split-state code where each part of the message is a vertex of a d-regular graph G on n vertices. The graph G must have expansion λ and satisfy $n = \Omega(d^3 \log(d)/\lambda)$ and $\lambda^{3/2}/d = O(\gamma)$. Suppose that G has expansion $\Theta(\sqrt{d})$, which is the case for the instantiation in Appendix C. We may then set $d = \Theta((1/\gamma)^4)$ and $n = \tilde{\Theta}((1/\gamma)^{10})$. Thus, our code uses space $20\log(1/\gamma) + O(\log\log(1/\gamma))$ to encode a single bit. The instantiation from Appendix C is not able to choose n as flexibly as suggested here and uses space $24\log(1/\gamma)$. The time taken to encode and decode a message in this instantiation is $O(\log(1/\gamma))$. In comparison, the instantiation of [4] uses space around $90\log(1/\gamma)$ and the time to encode and decode is $O(\log(1/\gamma)\log^2(\log(1/\gamma)))$. It should be noted, however, that the construction of [4] supports leakage as well, something that we are not considering in this paper.

1.4 Intuition behind our construction and analysis

Every graph, $G = (V, E)$ yields a single-bit split-state code in the following straightforward manner: To encode 1, pick an edge $(v, u) \in E$ uniformly at random, and set the left encoding to be v and the right encoding to be u. To encode a 0, do the same with a uniformly random non-edge in the graph.

Our analysis proceeds in two parts. First, in Proposition 6, using only elementary manipulations, we give an exact characterization of the success probability of any particular tampering split-state adversary against the code associated with any graph. The split-state adversary uses two functions, g and h, to tamper with the left and right encodings, respectively. The significant term of the probability to be analyzed is the quantity

$$\sum_{(v,u)\in E} \left(\frac{d\left|g^{-1}(v)\right| \cdot \left|h^{-1}(u)\right|}{n} - \left|E(g^{-1}(v), h^{-1}(u))\right| \right).$$

To bound this expression, we make the following observations. First, sparsity of the graph allows us to bound many of the terms immediately. Second, the term in the parentheses above immediately suggests a bound using the Expander Mixing Lemma, applied to the number of edges from $g^{-1}(v)$ to $h^{-1}(u)$. Third, we observe that the sum itself is over edges $(v, u) \in E$, and furthermore, the remaining sum of problematic terms are a sum over edges of the form $(v, u) \in E \cap (T \times S)$ for some vertex subsets $S, T \subset V$. This allows us to apply the Expander Mixing Lemma a *second* time, effectively bounding the number of "error terms" that accumulate through the initial use of the Expander Mixing Lemma. The actual analysis of this bound is just over a page of calculation. The analysis follows the intuition above, modulo a partitioning of terms into sets of appropriate size for the analysis to work.

2 Preliminaries

We shall assume familiarity with the basics of codes and non-malleable codes. A cursory review of relevant definitions can be found in the appendix.

▶ **Notation 1** (Graphs). *A graph $G = (V, E)$ consists of vertices V and edges $E \subset V \times V$. In this exposition every graph is undirected and $n = |V|$ always denotes the number of vertices of the graph in question.*

- *For any $v \in V$ we denote by $N(v)$ the set of neighbors of v in G.*
- *For any two subsets $S, T \subseteq V$ we denote by $E(S, T)$ the set of (directed) edges from S to T in G. I.e. $E(S, T) = \{(v, u) \in S \times T \mid (v, u) \in E\}$.*

▶ **Definition 2** (Spectral Expander). *Let $G = (V, E)$ be a d-regular graph, A_G be its adjacency matrix, and $\lambda_1 \geq \cdots \geq \lambda_n$ be the eigenvalues of A_G. We say that G is a λ spectral expander if $\lambda \geq \max\{|\lambda_2|, \ldots, |\lambda_n|\}$.*

▶ **Theorem 3** (Expander Mixing Lemma). *Suppose that $G = (V, E)$ is a λ spectral expander. Then for every pair of subsets $S, T \subset V$ we have*

$$\left| |E(S,T)| - \frac{d \cdot |S| \cdot |T|}{n} \right| \leq \lambda \sqrt{|S| \cdot |T|}.$$

Our results will rely on the following characterization of 1-bit non-malleable codes by Dziembowski, Kazana, and Obremski found in [4].

▶ **Theorem 4.** *Let $(\mathrm{enc}, \mathrm{dec})$ be a coding scheme with $\mathrm{enc}\colon \{0,1\} \to \mathcal{X}$ and $\mathrm{dec}\colon \mathcal{X} \to \{0,1\}$. Further, let $\mathcal{F}$ be a set of functions $f\colon \mathcal{X} \to \mathcal{X}$. Then $(\mathrm{enc}, \mathrm{dec})$ is ε-non-malleable with respect to $\mathcal{F}$ if and only if for every $f \in \mathcal{F}$,*

$$\Pr_{b \xleftarrow{u} \{0,1\}} (\mathrm{dec}(f(\mathrm{enc}(b))) = 1 - b) \leq \frac{1}{2} + \varepsilon,$$

where the probability is over the uniform choice of b and the randomness of enc.

3 Results

We first formally introduce our candidate code and then prove that it is a non-malleable code.

3.1 Candidate Code

From a graph we can very naturally construct a coding scheme as follows.

▶ **Definition 5** (Graph Code). *Let $G = (V, E)$ be a graph. The associated* graph code, *$(\mathrm{enc}_G, \mathrm{dec}_G)$, consists of the functions*

$$\mathrm{enc}_G\colon \{0,1\} \to V \times V, \qquad \mathrm{dec}_G\colon V \times V \to \{0,1\}$$

which are randomized and deterministic, respectively, and given by

$$\mathrm{enc}_G(b) = \begin{cases} (u,v) \xleftarrow{u} (V \times V) \setminus E, & b = 0, \\ (u,v) \xleftarrow{u} E, & b = 1, \end{cases}$$

$$\mathrm{dec}_G(v_1, v_2) = \begin{cases} 0, & (v_1, v_2) \notin E, \\ 1, & (v_1, v_2) \in E. \end{cases}$$

3.2 Non-Malleability of Expander Graph Codes

Finally, arriving at the core of the matter, we first establish the following lemma casting the expression of Theorem 4 in terms of graph properties.

▶ **Proposition 6.** *Let $G = (V, E)$ be a graph, functions $g, h\colon V \to V$ be given, and $f = (g, h)\colon V \times V \to V \times V$ satisfy $f(u, v) = (g(u), h(v))$. For the probability that f flips a random bit encoded by enc_G, write*

$$T = \Pr_{b \xleftarrow{u} \{0,1\}} (\mathrm{dec}_G(f(\mathrm{enc}_G(b))) = 1 - b)$$

where the probability is taken over the randomness of enc_G and the sampling of b. Then

$$T = \frac{1}{2} + \frac{1}{2d(n-d)} \sum_{(v,u)\in E} \left(\frac{d \left|g^{-1}(v)\right| \cdot \left|h^{-1}(u)\right|}{n} - \left|E(g^{-1}(v), h^{-1}(u))\right| \right). \tag{1}$$

Proof. For $b \in \{0, 1\}$ denote by Q_b the probability

$$Q_b = \Pr(\mathrm{dec}_G(f(\mathrm{enc}_G(b))) = 1 - b)$$

taken over the randomness of enc_G. It is clear that $T = \frac{Q_0+Q_1}{2}$ and that by definition

$$Q_0 = \Pr_{(v,u) \xleftarrow{u} V\times V\setminus E} [(g(v), h(u)) \in E], \qquad Q_1 = \Pr_{(v,u) \xleftarrow{u} E} [(g(v), h(u)) \notin E].$$

First, for $b = 0$ we see that the number of non-edges that are mapped by f to any given $(v, u) \in E$ is given by $\left|g^{-1}(v)\right| \cdot \left|h^{-1}(u)\right| - \left|E(g^{-1}(v), h^{-1}(u))\right|$. There are $n(n-d)$ non-edges in G so it follows that

$$Q_0 = \frac{\sum_{(v,u)\in E} \left|g^{-1}(v)\right| \cdot \left|h^{-1}(u)\right| - \left|E(g^{-1}(v), h^{-1}(u))\right|}{n(n-d)}.$$

Second, for $b = 1$ the number of edges of G that are mapped to non-edges by f is given by $\sum_{(v,u)\notin E} \left|E(g^{-1}(v), h^{-1}(u))\right|$. Since there are dn edges of G to choose from when encoding the bit $b = 1$,

$$Q_1 = \frac{\sum_{(v,u)\notin E} \left|E(g^{-1}(v), h^{-1}(u))\right|}{dn}.$$

Now, observing that the number of (directed) edges in the graph is dn and that $\{g^{-1}(v)\}_{v\in V}$ and $\{h^{-1}(u)\}_{u\in V}$ are both partitions of V, we get

$$Q_1 = \frac{dn - \sum_{(v,u)\in E} \left|E(g^{-1}(v), h^{-1}(u))\right|}{dn} = 1 - \frac{\sum_{(v,u)\in E} \left|E(g^{-1}(v), h^{-1}(u))\right|}{dn}.$$

Putting it all together,

$$\begin{aligned} T &= \frac{\sum_{(v,u)\in E} \left|g^{-1}(v)\right| \cdot \left|h^{-1}(u)\right| - \left|E(g^{-1}(v), h^{-1}(u))\right|}{2n(n-d)} + \frac{1}{2} - \frac{\sum_{(v,u)\in E} \left|E(g^{-1}(v), h^{-1}(u))\right|}{2dn} \\ &= \frac{1}{2} + \frac{1}{2d(n-d)} \sum_{(v,u)\in E} \left(\frac{d \left|g^{-1}(v)\right| \cdot \left|h^{-1}(u)\right|}{n} - \left|E(g^{-1}(v), h^{-1}(u))\right| \right). \end{aligned}$$

◀

We proceed immediately with the main theorem, which concludes the exposition. In order to keep this presentation short and to the point, more elaborate calculations, which avoid the log-factors, have been placed in the appendix as Theorem 10.

▶ **Theorem 7.** *Let $G = (V, E)$ be d-regular with spectral expansion λ satisfying $n = \Omega(d^3 \log(d)^4/\lambda)$. Then $(\mathrm{enc}_G, \mathrm{dec}_G)$ is an $\tilde{O}\left(\frac{\lambda^{3/2}}{d}\right)$-non-malleable code in the split-state model.*

Proof. Let $f = (g, h) \colon V \times V \to V \times V$ be given. By Theorem 4 and Proposition 6 we just need to show that

$$R = \frac{1}{2d(n-d)} \cdot \sum_{(v,u)\in E} \left(\frac{d\left|g^{-1}(v)\right| \cdot \left|h^{-1}(u)\right|}{n} - \left|E(g^{-1}(v), h^{-1}(u))\right| \right)$$

is bounded by $\tilde{O}\left(\frac{\lambda^{3/2}}{d}\right)$. Define the sets

$$G_1 = \left\{ v \in V \mid \left|g^{-1}(v)\right| > \frac{n}{d^2} \right\}, \qquad H_1 = \left\{ u \in V \mid \left|h^{-1}(u)\right| > \frac{n}{d^2} \right\},$$
$$G_2 = \left\{ v \in V \mid \left|g^{-1}(v)\right| \leq \frac{n}{d^2} \right\}, \qquad H_2 = \left\{ u \in V \mid \left|h^{-1}(u)\right| \leq \frac{n}{d^2} \right\},$$

for $i, j \in \{1, 2\}$ write

$$R_{i,j} = \frac{1}{2d(n-d)} \sum_{(v,u)\in E\cap(G_i\times H_j)} \left(\frac{d\left|g^{-1}(v)\right| \cdot \left|h^{-1}(u)\right|}{n} - \left|E(g^{-1}(v), h^{-1}(u))\right| \right),$$

and observe that $R = \sum_{1\leq i,j\leq 2} R_{i,j}$.

Consider the case when $i = 2$. Simply bounding the terms of the form $\left|g^{-1}(v)\right| \cdot \left|h^{-1}(u)\right|$ by using that each vertex has only d neighbours, we get

$$\begin{aligned} R_{2,1} + R_{2,2} &\leq \frac{1}{2n(n-d)} \sum_{(v,u)\in E\cap(G_2\times V)} \left|g^{-1}(v)\right| \cdot \left|h^{-1}(u)\right| \\ &\leq \frac{1}{2n(n-d)} \cdot d \cdot \sum_{u\in V} \frac{n}{d^2} \cdot \left|h^{-1}(u)\right| \\ &= \frac{n}{2(n-d)d}. \end{aligned}$$

Thus, $R_{2,1} + R_{2,2} = O\left(d^{-1}\right)$. By symmetry, $R_{1,2} = O\left(d^{-1}\right)$. It only remains to show that $R_{1,1} = \tilde{O}\left(\frac{\lambda^{3/2}}{d}\right)$. To this end, partition G_1 and H_1, respectively, as

$$G_1^k = \left\{ v \in G_1 \mid \frac{n}{2^{k-1}} \geq \left|g^{-1}(v)\right| > \frac{n}{2^k} \right\}, \qquad H_1^l = \left\{ v \in H_1 \mid \frac{n}{2^{l-1}} \geq \left|h^{-1}(u)\right| > \frac{n}{2^l} \right\}$$

for $1 \leq k, l \leq \left\lceil \log_2\left(d^2\right) \right\rceil$. Now, focusing on each pair G_1^k and H_1^l, we write

$$S_{k,l} = \frac{1}{2d(n-d)} \sum_{(v,u)\in E\cap(G_1^k\times H_1^l)} \left(\frac{d\left|g^{-1}(v)\right| \cdot \left|h^{-1}(u)\right|}{n} - \left|E(g^{-1}(v), h^{-1}(u))\right| \right)$$

and apply first the mixing lemma then the Cauchy-Schwartz inequality to get

$$\begin{aligned} 2d(n-d)S_{k,l} &= \sum_{v\in G_1^k} \left(\frac{d\left|g^{-1}(v)\right| \cdot \sum_{u\in N(v)\cap H_1^l} \left|h^{-1}(u)\right|}{n} - \left| E\left(g^{-1}(v), \bigcup_{u\in N(v)\cap H_1^l} h^{-1}(u) \right) \right| \right) \\ &\leq \sum_{v\in G_1^k} \lambda \sqrt{|g^{-1}(v)| \cdot \sum_{u\in N(v)\cap H_1^l} |h^{-1}(u)|} \\ &\leq \lambda \sqrt{\frac{n}{2^{k-1}} \cdot \frac{n}{2^{l-1}}} \cdot \sum_{v\in G_1^k} \sqrt{\left|N(v)\cap H_1^l\right|} \\ &\leq 2\lambda n \cdot 2^{-\frac{l+k}{2}} \cdot \sqrt{\left|G_1^k\right|} \cdot \sqrt{\left|E(G_1^k, H_1^l)\right|}. \end{aligned}$$

We use the fact that $\left|G_1^k\right| \leq 2^k, \left|H_1^l\right| \leq 2^l$, apply the mixing lemma to the last factor, and wield Jensen's inequality on the arising square root to obtain

$$\begin{aligned} d(n-d)S_{k,l} &\leq \lambda n \cdot 2^{-\frac{l+k}{2}} \cdot \sqrt{\left|G_1^k\right|} \cdot \sqrt{\frac{d \cdot \left|G_1^k\right| \cdot \left|H_1^l\right|}{n}} + \lambda \sqrt{\left|G_1^k\right| \cdot \left|H_1^l\right|} \\ &\leq \lambda \sqrt{2^k dn} + 2^{\frac{k-l}{4}} \lambda^{3/2} n \leq \lambda \cdot \sqrt{d^3 n} + 2^{\frac{k-l}{4}} \lambda^{3/2} n. \end{aligned}$$

By symmetry of k and l, $d(n-d)S_{k,l} \leq \lambda \cdot \sqrt{d^3 n} + 2^{\frac{l-k}{4}} \lambda^{3/2} n$. Thus,

$$\begin{aligned} R_{1,1} &= \sum_{1 \leq k,l \leq \lceil \log_2(d^2) \rceil} S_{k,l} \\ &\leq O\left(\frac{\lambda \log(d)^2 \cdot \sqrt{d}}{\sqrt{n}}\right) + O\left(\frac{\lambda^{3/2}}{d}\right) \cdot \sum_{1 \leq k,l \leq \lceil \log_2(d^2) \rceil} 2^{-\frac{|k-l|}{4}} \\ &= O\left(\frac{\log(d)\lambda^{3/2}}{d}\right). \end{aligned}$$

◀

References

1 Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In *Symposium on Theory of Computing, STOC*, 2014.

2 Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In *Symposium on Theory of Computing, STOC*, 2016.

3 Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. In *Foundations of Computer Science, FOCS*, 2014.

4 Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In *CRYPTO*, 2013.

5 Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, 2010.

6 Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In *Symposium on Theory of Computing, STOC*, 2017.

7 Luca Trevisan. Luca trevisan's 'in theory' blog. `https://lucatrevisan.wordpress.com/2011/02/28/cs359g-lecture-16-constructions-of-expanders/`. Accessed: 2018-09-27.

A Definitions for Split-State Non-Malleable Codes

Here, we recall the basic definition of a split-state non-malleable code due to [5].

▶ **Definition 8** (Coding scheme). *We define a* coding scheme *to be a pair of functions* $(\mathrm{enc}, \mathrm{dec})$. *The encoding function* $\mathrm{enc}\colon \mathcal{M} \to \mathcal{X}$ *is randomized while the decoding function* $\mathrm{dec}\colon \mathcal{X} \to \mathcal{M} \cup \{\perp\}$ *is deterministic. Further, for all* $s \in \mathcal{M}$ *the pair satisfies*

$$\Pr[\mathrm{dec}(\mathrm{enc}(s)) = s] = 1$$

where the probability is taken over the randomness of enc.

▶ **Definition 9** (Split State Non-Malleable Code). *A coding scheme* (enc, dec), $\mathrm{enc}\colon \mathcal{M} \to \mathcal{L} \times \mathcal{R}$ *and* $\mathrm{dec}\colon \mathcal{L} \times \mathcal{R} \to \mathcal{M} \cup \{\perp\}$, *is* ε*-non-malleable in the split state model if for every pair of functions* $g\colon \mathcal{L} \to \mathcal{L}, h\colon \mathcal{R} \to \mathcal{R}$ *and writing* $f = (g, h)$ *there exists a distribution* D_f

supported on $\mathcal{M} \cup \{*, \perp\}$ *such that for every* $s \in \mathcal{M}$ *the two random variables defined by the experiments*

$$A_f^s = \left\{ \begin{smallmatrix} (L,R) \leftarrow \mathrm{enc}(s); \\ Output\ \ \mathrm{dec}(g(L), h(R)) \end{smallmatrix} \right\}$$
$$B_f^s = \left\{ \begin{smallmatrix} \tilde{s} \leftarrow D_f; \\ If\ \tilde{s} = *\ \ output\ s\ \ else\ \ output\ \tilde{s} \end{smallmatrix} \right\}$$

have statistical distance at most ε.

B Deliver Us from Log Factors

A more thorough analysis of the sums in the proof of Theorem 7 allows us to get slightly better bounds. The technicalities are of little interest to the big picture and were hence omitted in the body of the paper. The addition consists of an alternative ending to the proof of Theorem 7.

▶ **Theorem 10.** *Let* $G = (V, E)$ *be* d*-regular with spectral expansion* λ *satisfying* $n = \Omega(d^3 \log(d)/\lambda)$. *Then* $(\mathrm{enc}_G, \mathrm{dec}_G)$ *is an* $O\left(\frac{\lambda^{3/2}}{d}\right)$*-non-malleable code in the split-state model.*

Proof. At the very end of the proof of Theorem 7, we arrived at

$$d(n-d)S_{k,l} \leq 2^{-\frac{l+k}{2}} \lambda n \cdot \sqrt{\left|G_1^k\right|} \cdot \sqrt{\frac{d \cdot \left|G_1^k\right| \cdot \left|H_1^l\right|}{n} + \lambda \cdot \sqrt{\left|G_1^k\right| \cdot \left|H_1^l\right|}}.$$

Applying Jensen's inequality, we get

$$S_{k,l} \leq O\left(\frac{\lambda}{\sqrt{dn}}\right) \cdot 2^{-\frac{l+k}{2}} \cdot \left|G_1^k\right| \cdot \sqrt{\left|H_1^l\right|} + O\left(\frac{\lambda^{3/2}}{d}\right) \cdot 2^{-\frac{l+k}{2}} \cdot \sqrt[4]{\left|G_1^k\right|^3 \cdot \left|H_1^l\right|} \tag{2}$$

with the functions hidden by the O-notation being independent of k, l.

Now, note that

$$\left|g^{-1}(G_1^k)\right| \geq \frac{n \cdot \left|G_1^k\right|}{2^k} \qquad \left|h^{-1}(H_1^l)\right| \geq \frac{n \cdot \left|H_1^l\right|}{2^l} \tag{3}$$

and for all $k \leq \lceil \log_2(d^2) \rceil$ we have $\frac{\left|G_1^k\right|}{2^{k/2}} \leq 2d$. We shall bound each of the terms of (2) separately.

First, write

$$L = \sum_{1 \leq k,l \leq \lceil \log_2(d^2) \rceil} \left(2^{-\frac{l+k}{2}} \cdot \left|G_1^k\right| \cdot \sqrt{\left|H_1^l\right|}\right).$$

Using the Cauchy-Schwartz inequality in the second inequality,

$$\begin{aligned} L &\leq 2d \cdot \sum_{1 \leq l \leq \lceil \log_2(d^2) \rceil} \sqrt{2^{-l} \left|H_1^l\right|} \\ &\leq O\left(d \cdot \sqrt{\log(d)}\right) \cdot \sqrt{\sum_{1 \leq l \leq \lceil \log_2(d^2) \rceil} 2^{-l} \cdot \left|H_1^l\right|} \\ &\leq O\left(d \cdot \sqrt{\log(d)}\right) \cdot \sqrt{\sum_{1 \leq l \leq \lceil \log_2(d^2) \rceil} \frac{\left|h^{-1}(H_1^l)\right|}{n}} \\ &= O\left(d \cdot \sqrt{\log(d)}\right) \end{aligned}$$

since the H_1^l are disjoint subsets of V. In conclusion,

$$O\left(\frac{\lambda}{\sqrt{dn}}\right) \cdot \sum_{1 \leq k,l \leq \lceil \log_2(d^2) \rceil} 2^{-\frac{l+k}{2}} \cdot \left|G_1^k\right| \cdot \sqrt{\left|H_1^l\right|} = O\left(\frac{\lambda \cdot \sqrt{d \cdot log(d)}}{\sqrt{n}}\right) = O\left(\frac{\lambda^{3/2}}{d}\right).$$

Second, let $k \leq l$ and write $t = l - k$. We now bound the sum using (3). Write

$$K = \sum_{1 \leq k < l \leq \lceil \log_2(d^2) \rceil} 2^{-\frac{l+k}{2}} \cdot \sqrt[4]{\left|G_1^k\right|^3 \cdot \left|H_1^l\right|}.$$

Then

$$\begin{aligned} K &\leq \sum_{1 \leq k < l \leq \lceil \log_2(d^2) \rceil} \left(\frac{2^{\frac{k-l}{4}}}{n} \cdot \sqrt[4]{\left|g^{-1}(G_1^k)\right|^3 \cdot \left|h^{-1}(H_1^l)\right|} \right) \\ &\leq \sum_{t=0}^{\lceil \log_2(d^2) \rceil} \left(\frac{2^{-\frac{t}{4}}}{n} \sum_{l=t}^{\lceil \log_2(d^2) \rceil} \sqrt[4]{\left|g^{-1}(G_1^{l-t})\right|^3 \cdot \left|h^{-1}(H_1^l)\right|} \right) \\ &\leq \sum_{t=0}^{\lceil \log_2(d^2) \rceil} \left(\frac{2^{-\frac{t}{4}}}{n} \left(\sum_{l=t}^{\lceil \log_2(d^2) \rceil} \left|g^{-1}(G_1^{l-t})\right| \right)^{3/4} \cdot \left(\sum_{l=t}^{\lceil \log_2(d^2) \rceil} \left|h^{-1}(H_1^l)\right| \right)^{1/4} \right) \\ &\leq \sum_{t=0}^{\lceil \log_2(d^2) \rceil} 2^{-\frac{t}{4}} = O(1), \end{aligned}$$

where the third inequality is established using Hölder's inequalty.

It now follows that

$$\sum_{1 \leq k \leq l \leq \lceil \log_2(d^2) \rceil} S_{k,l} = O\left(\frac{\lambda^{3/2}}{d}\right).$$

By symmetry of k and l,

$$R_{1,1} = \sum_{1 \leq k,l \leq \lceil \log_2(d^2) \rceil} S_{k,l} = O\left(\frac{\lambda^{3/2}}{d}\right),$$

which completes the proof. ◀

C Instantiating Our Construction

Using our results to instantiate an efficient, secure split-state non-malleable code, we require a family of graphs $\{G_k\}_{k \in \mathbb{N}}$, where each $G_k = (V_k, E_k)$ is d_k-regular with spectral expansion λ_k, satisfying the following:

1. The function $\varepsilon(k) = \frac{\lambda_k^{3/2}}{d_k}$ is negligible.
2. We have $n_k = |V(G_k)| = \Omega(d_k^3 \log(d_k)/\lambda_k)$
3. Both sampling an edge $(u, v) \xleftarrow{u} E_k$ and sampling a non-edge $(u, v) \xleftarrow{u} (V_k \times V_k) \setminus E_k$ can be done in time polynomial in k.

4. Determining membership of a pair $(u, v) \in V \times V$ in $E(G_k)$ can be done deterministically in time polynomial in k.

Given such a family of graphs it is clear that the corresponding graph code $(\mathrm{enc}_{G_k}, \mathrm{dec}_{G_k})$ is an efficiently computable non-malleable code.

C.1 Instantiation with High-Degree Cayley Graphs

Explicit constructions of such families of graphs do indeed exist. We shall here give an example from [7] from the class of graphs known as Cayley graphs. The construction is as follows.

▶ **Definition 11.** *For p a prime and $1 \leq t < p$ let the graph $\mathrm{LD}_{p,t}$ have vertex set $\mathbb{F}_p^{t+1}$ and edge set*

$$E(\mathrm{LD}_{p,t}) = \left\{(x, x + (b, ab, a^2b, \ldots, a^tb)) \mid x \in \mathbb{F}_p^{t+1}, a, b \in \mathbb{F}_p\right\},$$

i.e. $x, y \in V(\mathrm{LD}_{p,T})$ are connected by an edge if and only if there exists $a, b \in \mathbb{F}_p$ such that $y = x + (b, ab, a^2b, \ldots, a^tb)$.

It is worth nothing that the graph $\mathrm{LD}_{p,t}$ is p^2-regular and that it is undirected as x is connected to y if and only if y is connected to x.

Now, let $t = 5$ and for each $k \in \mathbb{N}$ let p_k be some k-bit prime. We consider the family of graphs $\{\mathrm{LD}_{p_k,5}\}_{k\in\mathbb{N}}$ for our instantiation. In the following, we shall check the criteria from the beginning of the section point by point.

1. The family of graphs $\mathrm{LD}_{p,t}$ has great expander properties.
   ▶ **Theorem 12** (explicit in Trevisan [7])**.** *For $1 < t < p$, the graph $\mathrm{LD}_{p,t}$ is a pt-spectral expander.*
   This fact allows us to note that for our particular choice of graphs, $\varepsilon(k) = \frac{(p_k t)^{3/2}}{p_k^2} < \frac{12}{\sqrt{p}_k}$, which in fact is $2^{-\Omega(k)}$ and the representation size is $O(k)$ bits.
2. We have $\Omega\left(\frac{d_k^3 \log(d_k)}{\lambda_k}\right) = \Omega(p^5 \log(p))$ such that indeed,
$$n_k = |V(\mathrm{LD}_{p_k,5})| = p^6 = \Omega\left(\frac{d_k^3 \log(d_k)}{\lambda_k}\right).$$
3. Sampling an edge $(u, v) \xleftarrow{u} E(\mathrm{LD}_{p_k,t})$ is simply a question of picking $x \in \mathbb{F}_{p_k}^{t+1}, a, b \in \mathbb{F}_{p_k}$ uniformly at random and then outputting the edge $(x, x + (b, ab, a^2b, \ldots, a^tb))$.
   To pick a non-edge, simply sample two random vertices $u, v \in \mathbb{F}_{p_k}^{t+1}$ uniformly at random and check (with the procedure to be specified below) whether $(u, v) \in E(\mathrm{LD}_{p_k,t})$. Since for $t > 1$ the probability of hitting an edge with such a random choice is $\leq 1/p_k$, the expected number of repetitions is constant and hence the procedure takes expected polynomial time.
4. To test membership of some $(u, v) \in \left(\mathbb{F}_{p_k}^{t+1}\right)^2$ in $E(\mathrm{LD}_{p_k,t})$, perform the following operation: Compute $x = u - v$ and write $x = (x_0, \ldots, x_t)$. It is now trival to check whether $\left(1, \frac{x_1}{x_0}, \ldots, \frac{x_t}{x_0}\right)$ is of the form $(1, a, a^2, \ldots, a^t)$.

Leakage-Resilient Secret Sharing in Non-Compartmentalized Models

Fuchun Lin
Department of Electrical and Electronic Engineering, Imperial College London, UK
flin@ic.ac.uk

Mahdi Cheraghchi
EECS Department, University of Michigan, Ann Arbor, MI, USA
http://mahdi.ch
mahdich@umich.edu

Venkatesan Guruswami
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
venkatg@cs.cmu.edu

Reihaneh Safavi-Naini
Department of Computer Science, University of Calgary, CA
rei@ucalgary.ca

Huaxiong Wang
Division of Mathematical Sciences, School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore, SG
hxwang@ntu.edu.sg

Abstract

Leakage-resilient secret sharing has mostly been studied in the *compartmentalized* models, where a leakage oracle can arbitrarily leak bounded number of bits from all shares, provided that the oracle only has access to a bounded number of shares when the leakage is taking place. We start a systematic study of leakage-resilient secret sharing against *global* leakage, where the leakage oracle can access the full set of shares simultaneously, but the access is restricted to a special class of leakage functions. More concretely, the adversary can corrupt several players and obtain their shares, as well as applying a leakage function from a specific class to the full share vector. We explicitly construct such leakage-resilient secret sharing with respect to affine leakage functions and low-degree multi-variate polynomial leakage functions, respectively. For affine leakage functions, we obtain schemes with threshold access structure that are leakage-resilient as long as there is a substantial difference between the total amount of information obtained by the adversary, through corrupting individual players and leaking from the full share vector, and the amount that the reconstruction algorithm requires for reconstructing the secret. Furthermore, if we assume the adversary is non-adaptive, we can even make the secret length asymptotically equal to the difference, as the share length grows. Specifically, we have a threshold scheme with parameters similar to Shamir's scheme and is leakage-resilient against affine leakage. For multi-variate polynomial leakage functions with degree bigger than one, our constructions here only yield ramp schemes that are leakage-resilient against such leakage. Finally, as a result of independent interest, we show that our approach to leakage-resilient secret sharing also yields a competitive scheme compared with the state-of-the-art construction in the compartmentalized models.

2012 ACM Subject Classification Security and privacy → Cryptography; Security and privacy → Information-theoretic techniques; Theory of computation → Expander graphs and randomness extractors

Keywords and phrases Leakage-resilient cryptography, Secret sharing scheme, Randomness extractor

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.7

Related Version Full version at https://arxiv.org/abs/1902.06195.

1st Conference on Information-Theoretic Cryptography (ITC 2020).
Editors: Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs; Article No. 7; pp. 7:1–7:24

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

Funding *Fuchun Lin*: The research of Fuchun Lin was supported in part by EPSRC grant EP/S021043/1 and Singapore Ministry of Education under Research Grant MOE2016-T2-2-014(S) and RG133/17(S).
Venkatesan Guruswami: The research of Venkatesan Guruswami was supported in part by United States NSF grants CCF-1422045 and CCF-1563742.
Reihaneh Safavi-Naini: The research of Reihaneh Safavi-Naini was in part supported by Natural Sciences and Engineering Research Council of Canada, Discovery Grants Program.
Huaxiong Wang: The research of Huaxiong Wang was supported by Singapore Ministry of Education under Research Grant MOE2016-T2-2-014(S) and RG133/17(S).

Acknowledgements Part of this work was done when the first author was in Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University. We would like to thank Vipul Goyal for helpful discussions. We also would like to thank the anonymous reviewers for comments.

1 Introduction

Secret sharing, introduced independently by Blakley [11] and Shamir [42], is a fundamental cryptographic primitive with far-reaching applications; e.g., a major tool in secure multiparty computation (cf. [19]). The goal in secret sharing is to encode a secret s into a number of *shares* $\mathsf{c}_1, \ldots, \mathsf{c}_\mathsf{n}$ that are distributed among a set $[\mathsf{n}] = \{1, \ldots, \mathsf{n}\}$ of players such that the access to the secret through collaboration of players can be accurately controlled. An *authorized* subset of players is a set $A \subseteq [\mathsf{n}]$ such that the shares with indices in A can be pooled together to reconstruct the secret s. On the other hand, A is an *unauthorized* subset if the knowledge of the shares with indices in A reveals no information about the secret. The set of authorized and unauthorized sets define an access structure, where the most widely used is the so-called *threshold* structure. A threshold secret sharing scheme is defined with respect to a threshold t and satisfies the following property: Any set $A \subseteq [\mathsf{n}]$ with $|A| < \mathsf{t}$ is an unauthorized set and any set $A \subseteq [\mathsf{n}]$ with $|A| \geq \mathsf{t}$ is an authorized set. Threshold secret sharing with threshold t is also called t-out-of-n secret sharing. Any threshold secret sharing scheme sharing m-bit secrets necessarily requires shares of length (in bits) at least m, and Shamir's scheme attains this lower bound [44]. The *information ratio* defined as the ratio of the secret length to the maximum share length measures the storage efficiency of a secret sharing scheme.

Leakage-resilience of secret sharing considers an adversary that has certain form of extra information about the shares beyond the unauthorized sets of shares and studies how to keep the secret remain private. Dziembowski and Pietrzak [23] developed an n-out-of-n *intrusion-resilient* secret sharing scheme using methods from the *bounded retrieval model*. The shares of such secret sharing schemes are made artificially large so that protocols with bounded communication complexity can not retrieve them completely. The reconstruction procedure is interactive requiring the players to exchange r short messages, while the adversary can also attack in rounds but is restricted to at most $r - 1$ rounds. The idea is to exploit the fact that there exist functions which can be efficiently computed interactively using low communication complexity in r, but not in $r - 1$ rounds. Davì, Dziembowski and Venturi [20] constructed the first 2-out-of-2 secret sharing scheme that statistically hides the secret even after an *adaptive* adversary executes an arbitrary leakage protocol on the two shares with bounded communication. The relation of their scheme to a 2-out-of-2 intrusion-resilient secret sharing is that the reconstruction of intrusion-resilient secret sharing needs to access only small part of the share while their scheme does not have such restriction.

Study of leakage-resilience of secret sharing has recently gained general attention. Benhamouda, Degwekar, Ishai and Rabin [10] initiated a systematic investigation on the leakage-resilience of secret sharing in the *local leakage model*. The adversary in the local leakage model *non-adaptively* obtains a bounded number (less than $\mathsf{t}-1$) of full shares and leaks from all the other shares individually, each share using an *arbitrary* leakage function bounded only by its output length. The authors focus on investigating whether standard secret sharing schemes, such as additive secret sharing and Shamir's scheme are already leakage-resilient with respect to such local leakage. They showed that these t-out-of-n secret sharing schemes, if the base field is a large prime, are leakage-resilient for some limited parameter settings. This is in sharp contrast to the results of Guruswami and Wootters [29] from an orthogonal study of Reed-Solomon codes as self-repairable codes, which translated into the setting of leakage-resilient secret sharing effectively shows that by leaking one bit using a *linear* function from each share, the secret of Shamir's scheme over finite field with characteristic 2 can be completely reconstructed. While starting with standard secret sharing schemes has the advantage of inheriting the nice algebraic structure and optimality (information ratio 1) from those schemes, the large prime field requirement for the choice of base field and the limited available leakage parameters are limiting the applications of such leakage-resilient secret sharing schemes. Concurrently and independently, Goyal and Kumar [27], motivated by the task of constructing *non-malleable secret sharing*, a new primitive of tamper-resilient secret sharing they put forward (see more details in *Related works*), constructed a 2-out-of-n leakage-resilient secret sharing scheme in the local leakage model through building up from a 2-out-of-2 leakage-resilient secret sharing scheme, which is in turn constructed from the inner product function. The authors showed that the same 2-out-of-n leakage-resilient secret sharing scheme is in fact leakage-resilient against a slightly stronger leakage adversary than the one in the local leakage model, as they took their non-malleable secret sharing results to general access structures and hence need the strengthened local leakage model [28]. Since then leakage-resilient secret sharing in the local leakage model is widely studied mostly with connection to non-malleable secret sharing [7, 43, 1]. Most of the known constructions of leakage-resilient secret sharing take a compiler approach with various features to transform a secret sharing scheme into a leakage-resilient one. Note that in this approach, local leakage-resilience is enabled through pumping independent randomness into each share and hence the obtained schemes do not have *full reconstruction* [38], which is a property that requires shares that are enough to reconstruct the secret also uniquely reconstruct the full share vector. The advantage of secret sharing without full reconstruction is that one can pump unlimited independent randomness into each share and allow the adversary to leak unlimited amount of information except some finite amount (hence it is possible for the scheme to tolerate asymptotic leakage rate 1) [7, 43]. On the other hand, the extra independent randomness eventually results in a bigger share size than necessary (small information ratio).

The local leakage adversary considered by Srinivasan and Vasudevan [43] is partially adaptive in the sense that for t-out-of-n threshold schemes, the choice of each local leakage function can be based on the value of the $t-2$ shares. Kumar, Meka, and Sahai [31] proposed an *adaptive joint leakage model* as opposed to the non-adaptive and partially adaptive local leakage models. They defined a *bounded collusion protocol* as a communication complexity problem that is tailored for secret sharing. A $\mathsf{t}-1$-bounded collusion protocol is one that runs in multiple rounds, each round involves at most $\mathsf{t}-1$ players and the output of that round can depend on the inputs of the involved players and all previous transcript. This can be seen as a generalisation of the communication complexity problem for the 2-out-of-2 secret sharing of [20]. Faonio and Venturi[25] (under computational assumption) considered a *noisy* leakage model that, instead of bounding the output length of the leakage functions, bounds the min-entropy of the share conditioned on the output of the leakage function.

All the leakage models mentioned are more or less based on a compartmentalized assumption that assumes the adversary does not have access to the full share vector at one time, while nevertheless allows the adversary to apply arbitrary leakage functions to the set of shares that are accessed. In this work, we are mainly interested in leakage-resilience of secret sharing against *global* type of leakage, where a leakage adversary can access the full set of shares simultaneously, but is restricted in the type of access that is admissible. Bogdanov, Ishai, Viola and Williamson [12] obtained several instances of leakage-resilient secret sharing in such global leakage models (see more details in *Related works*). For example, it is shown that Shamir's scheme over finite fields of characteristic 2 is leakage-resilient against constant depth polynomial size circuits (AC^0) and sign polynomials of degree 2, unless the threshold t is at most polylogarithmic in n. But as mentioned before, these schemes are not leakage resilient against (local) linear leakage [29]). As far as we know, this is the only previous work that studies global leakage-resilience of secret sharing.

Our contributions. We start a systematic study of Leakage-Resilient Secret Sharing (LR-SS) with respect to a class $\mathcal{L}$ of global leakage functions. To stay compatible with the well established local leakage models, we define a $(\mathcal{L}, \beta, \theta, \varepsilon)$-leakage resilient t-out-of-n secret sharing scheme. The parameter β is the bound on the total amount of leakage (measured in bits) through leakage functions chosen from $\mathcal{L}$. The parameter θ is the bound on the number of corrupted players. The parameter ε is the leakage-resilience error (measured in statistical distance). We see that if we define a class $\mathcal{L}_{\mathsf{local}}$ of functions that mimic a local leakage adversary that leaks ℓ bits arbitrarily from each share, then we recover the local leakage model as a $(\mathcal{L}_{\mathsf{local}}, \ell(\mathsf{n}-\theta), \theta, \varepsilon)$-leakage resilient t-out-of-n secret sharing scheme. We focus on the case when the share is an element of a finite field $\mathbb{F}_q$ of characteristic 2, which is of greater practical importance. We also consider adaptive as well as non-adaptive adversary, which distinguishes between the cases where the adversary can or can not base the subsequent leakage on previously obtained transcript. The exposition is focused on t-out-of-n secret sharing and all schemes constructed satisfy the *full reconstruction* property, in particular, any t shares determine the value of the remaining $\mathsf{n}-\mathsf{t}$ shares. We place a special emphasize on achieving the best possible leakage tolerance as well as optimal information ratio, for some cases.

We start with the class $\mathcal{L}_{\mathsf{affine}}$ of affine leakage functions over $\mathbb{F}_2$. This class of leakage functions is interesting because of the following. In practice, the shares of a secret sharing scheme are to be sent to the players in private. Suppose we send the shares over a network and some of the packets sent by intermediate servers are leaked. If only routing is used, we have the compartmentalized model where individual packets are observed by the adversary. Suppose in a real-life application where the network is utilizing linear network coding [26], then effectively we are going to have global affine leakage. Another example is the type of leakage for Shamir's scheme over finite field of characteristic 2 implied in the result of [29]. It is in effect a subset of the linear leakage functions, which is in turn subsumed by affine leakage functions. Our results for affine type of leakage are summarized as follows.

▶ **Theorem** (informal summary of Theorem 14 and Theorem 17)**.** *Let $\mathcal{L}_{\mathsf{affine}}$ be the class of affine leakage functions over $\mathbb{F}_2$. Let $0 < \xi \leq 1$ be any real number. There is a t-out-of-n secret sharing scheme over $\mathbb{F}_q$ that is simultaneously $(\mathcal{L}_{\mathsf{affine}}, \beta, \theta, \varepsilon)$-leakage resilient for all $0 \leq \beta < (\mathsf{t}-\xi)\log q$ and $\theta \in \{0, 1, \ldots, \mathsf{t}-1\}$ such that $\theta + \frac{\beta}{\log q} \leq \mathsf{t}-\xi$, where the share size q is a large enough power of 2 and the secret length $m = \Omega(\log q)$. In particular, if the adversary is non-adaptive, the result can be strengthened to have secret length $m = \xi \log q - o(\log q)$.*

The leakage tolerance in the above construction is the best one can hope for in the following sense. The amount of information given to the adversary is $(\mathsf{t}-\xi)\log q$ bits in total (θ shares through corrupted players and β bits through global leakage) while with $\mathsf{t}\log q$ bits of information the secret (in fact the full share vector) can be completely reconstructed. Our results show that as long as $\xi > 0$, we can construct a t-out-of-n secret sharing scheme with constant information ratio that is universally leakage-resilient against affine leakage for all combinations of β and θ as long as the total amount satisfies $\theta\log q + \beta \le (\mathsf{t}-\xi)\log q$.

The strengthened result in the case of non-adaptive adversary is furthermore optimal in the sense that the asymptotic information ratio is exactly ξ. For example, let $\xi = 1$. We have a t-out-of-n secret sharing scheme with asymptotic information ratio 1. As a plain secret sharing scheme, this scheme is almost as good as the Shamir's secret sharing scheme, except that Shamir's scheme has perfect privacy $\varepsilon = 0$ and information ratio exactly 1. On the other hand, Shamir's scheme over finite field of characteristic 2 is not leakage-resilient against even leaking one bit from each share using a linear leakage function [29], while our scheme is universally leakage-resilient against affine leakage for all combinations of β and θ as long as the total amount of information given to the adversary satisfies $\theta\log q + \beta \le (\mathsf{t}-1)\log q$.

We venture a little bit further and explore leakage-resilience of secret sharing against stronger algebraic type of global leakage. These leakage models can occur in more sophisticated applications of secret sharing schemes such as the secure multiparty computation. Private inputs from the participants are protected by secret sharing schemes and algebraic computations on the private inputs are done through computations on the individual shares. Leakage that occurs throughout the duration of the computation is a good example for global multi-variate polynomial leakage model.

Let $\mathcal{L}_{d\text{-poly}}$ denote the class of multi-variate polynomials with degree at most d over $\mathbb{F}_2$. The LR-SS with respect to $\mathcal{L}_{d\text{-poly}}$ we construct is a slightly weaker form of t-out-of-n secret sharing called *ramp scheme*. There is a second threshold called privacy threshold $\mathsf{t}-g$ and the gap between the two thresholds is denoted by g. When $g = 1$, we recover the threshold scheme. For ramp schemes, it is possible to share a secret longer than the share length.

▶ **Theorem** (informal summary of Theorem 20). *Let $\mathcal{L}_{d\text{-poly}}$ be the class of multi-variate polynomials over $\mathbb{F}_2$ with degree at most d. Let g be an integer satisfying $\frac{g}{\mathsf{t}} > 1 - \frac{1}{c_d}$, where $c_d = \Theta(d^2 4^d)$ is a constant determined by d. There is a t-out-of-n secret sharing scheme over $\mathbb{F}_q$ with threshold gap g over the finite field $\mathbb{F}_q$ that is simultaneously $(\mathcal{L}_{d\text{-poly}}, \beta, \theta, \varepsilon)$-leakage resilient for all $\beta = (\log q)\phi$ and $\theta \le \mathsf{t}-g$ satisfying $0 \le \phi \le \mathsf{t}-g-\theta$, where the share size q is a large enough power of 2 and the secret length $m = \Omega(\log q)$.*

The threshold gap g in the above result is mostly bigger than 1. We note that smaller threshold gap can be achieved using our construction if a building block of better parameters is constructed in the future.

Finally, as a result of independent interest, we consider an instantiation of our generic construction for global leakage to obtain LR-SS in local leakage model with a partially adaptive adversary (currently strongest local leakage model) [43]. It is known that the number of full shares from corrupted players that a partially adaptive adversary has before the leakage functions are chosen is at most $\mathsf{t}-2$ [1]. Our LR-SS allow the number θ of

[1] As argued in [43], based on $\mathsf{t}-1$ shares, the partially adaptive adversary can invoke the reconstruction algorithm to define a leakage function that takes a new share (not among the $\mathsf{t}-1$ shares) as input, pooling with the $\mathsf{t}-1$ shares to reconstruct the secret, and outputs one bit of the secret.

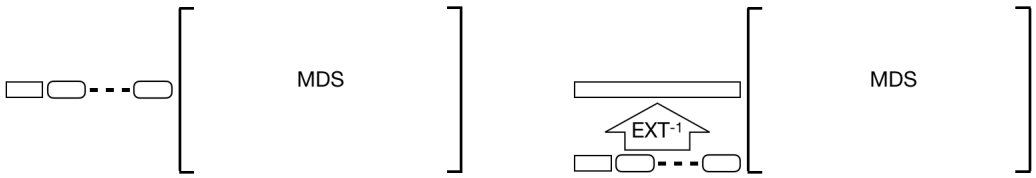


Figure 1 Shamir's scheme versus our generic construction.

corrupted players to be equal to $\mathsf{t}-1$. But due to the known impossibility result, when $\theta=\mathsf{t}-1$, the partially adaptive adversary is required to choose leakage functions base on at most $\mathsf{t}-2$ out of the $\mathsf{t}-1$ corrupted players.

▶ **Theorem** (informal summary of Theorem 24)**.** *Let $\mathcal{L}_{\mathsf{local}}$ be the class of local leakage functions. Let $\xi>0$ be a small real number. There is a t-out-of-n secret sharing scheme over $\mathbb{F}_q$ that is simultaneously $(\mathcal{L}_{\mathsf{local}},\beta,\theta,\varepsilon)$-leakage resilient against a partially adaptive adversary that corrupts θ players for $\theta\in\{0,1,\ldots,\mathsf{t}-2\}$ and, based on the shares of the θ players, chooses $\mathsf{n}-\theta$ arbitrary leakage functions each leaks $\ell=\frac{(\mathsf{t}-\theta-8\xi)\log q}{(\mathsf{n}-\theta)(1+5\xi)}$ bits for the remaining $\mathsf{n}-\theta$ shares, where the share size q is a large enough power of 2 and the secret length $m=\frac{\xi}{1+5\xi}\log q-o(\log q)$. When $\theta=\mathsf{t}-1$, the partially adaptive adversary can only choose the $\mathsf{n}-\mathsf{t}+1$ arbitrary leakage functions based on $\mathsf{t}-2$ shares*

Our scheme achieves positive information ratio, which is an absolute constant, while the scheme in [43] achieves information ratio $\Omega(1/\mathsf{n})$, which depends on n. On the other hand, when $\theta=\mathsf{t}-1$, our scheme achieves an asymptotic leakage rate of $\frac{\mathsf{t}-\theta}{\mathsf{n}-\theta}=\frac{1}{\mathsf{n}-\mathsf{t}+1}$, while the scheme in [43] achieves asymptotic leakage rate 1, which is only possible for LR-SS without full reconstruction. For LR-SS with full reconstruction, the asymptotic leakage rate of $\frac{1}{\mathsf{n}-\mathsf{t}+1}$ is close to the optimal $\frac{1}{\mathsf{n}-\mathsf{t}}$, according to the bound developed for local leakage model LR-SS that satisfy full reconstruction property [38].

Technical overview. All these results are obtained from a generic construction that can be interpreted as follows. We preprocess the secret and the randomness of the sharing algorithm of Shamir's scheme over finite field of characteristic 2 before inputting them into the sharing algorithm to enable leakage-resilience. As illustrated in **Figure 1**, Shamir's scheme can be seen as directly concatenating the secret (as a finite field element of $\mathbb{F}_{2^m}$) with $\mathsf{t}-1$ independent uniformly random finite field elements of $\mathbb{F}_{2^m}$ and multiply with a $\mathsf{t}\times\mathsf{n}$ Maximum Distance Separable (MDS) matrix over the finite field $\mathbb{F}_{2^m}$. Our construction is to input the m bits secret as binary string and the $m(\mathsf{t}-1)$ bits independent uniform randomness into the inverter of an invertible randomness extractor to obtain an $m\mathsf{t}$ bits output, which is then interpreted as a vector in $\mathbb{F}_{2^m}^{\mathsf{t}}$ and multiply with the $\mathsf{t}\times\mathsf{n}$ MDS matrix over the finite field $\mathbb{F}_{2^m}$.

A randomness extractor takes an entropy source as input and output a close to uniform distribution over a smaller space. For structured entropy source, for example, the source is a flat distribution over an affine subspace of the universal space $\{0,1\}^n$, there exists a single function that turns any such source distribution with enough entropy into a close to uniform distribution over $\{0,1\}^m$. Such extractors are called *seedless extractors*. On the other hand, for arbitrary entropy source, we need a family of functions that are labeled by *seeds* and uniformly choose one function among them to extract from such entropy source. These extractors are called *seeded extractors*. A seeded extractor is *strong* if given a source distribution, almost all functions in the family can extract close to uniform output from it. A

seedless extractor is called invertible if there exists an efficient function (called *inverter*) that takes a vector in $\{0,1\}^m$ and some randomness as input and outputs a random pre-image in $\{0,1\}^n$. A seeded extractor is invertible if all the functions in the family are invertible. The inverter of a seeded extractor first samples a uniform seed and then use the inverter of the function corresponding to the seed to invert the vector in $\{0,1\}^m$.

The intuition of our generic construction illustrated in **Figure 1** is that randomness extractors can make the secret and output of the leakage functions independent, hence provide privacy and leakage-resilience. Assume that the secret is uniform. Then the random pre-image outputted by the inverter has uniform distribution. As long as this uniformly distributed pre-image conditioned on the output of the leakage functions has enough entropy (and with the right structure in the case when a seedless extractor is used), the secret remains uniform.

Our results in this work are obtained by using different randomness extractors in the generic construction.

- Affine leakage, non-adaptive adversary.
  We use a *linear* strong seeded extractor that extracts all the randomness. A seeded extractor is linear if every function in the family is linear. The linearity of the extractor function together with the fact that the source distributions induced by affine leakage functions are flat over some affine subspaces allows us to claim that almost all functions in the family output exactly uniform distribution. This observation plays an important role in achieving optimal information ratio.
- Affine leakage, adaptive adversary.
  We use an invertible affine extractor that can extract from an arbitrary fraction of entropy and output length is a constant fraction of the input length with exponentially small extractor error. In fact, we additionally apply a series of sophisticated optimization techniques to improve the parameters of the obtained scheme.
- Low-degree multi-variate polynomial leakage.
  There are a few challenges that have to be overcome when the degree of multi-variate polynomials goes beyond 1. For an affine leakage function f, the number of solutions to $f(x) = a$ for any a that admits non-empty solutions is determined by f and independent of the value a. This gives a natural bound on the min-entropy of a random x conditioned on the value of $f(x)$. When the degree is bigger than 1, we no longer have such nice structure. We identify the *average conditional min-entropy* [21] as the "right" entropy measure and use it to derive a lower bound with respect to the output length of the leakage function. Average conditional min-entropy is usually used in combination with the *average-case* strong seeded extractors. We then define a seedless analogue of average-case strong seeded extractors and verify that the explicit seedless extractor we use in our construction is in fact an *average-case seedless* extractor. We think these might have independent interest in other application scenarios of seedless extractors. There is also the challenge of explicitly constructing such extractors with parameters as good as affine extractors, which we leave it as an interesting open question.
- Local leakage, partially adaptive adversary.
  We use an average-case strong seeded extractor with exponentially small error. Such extractors necessarily requires a long seed. That is one of the reasons that we no longer have optimal information ratio (in the instantiation for affine leakage non-adaptive adversary, the seed length is negligible compared to the extractor input). There is one more modification. The random seed chosen by the inverter of the seeded extractor is no longer directly appended to the random pre-image to be encoded using the MDS code.

We use a Shamir's scheme to share this random seed into n shares. The final share of our LR-SS consists of one MDS codeword component and one share of the random seed. Through this modification, we are able to prove that the partially adaptive adversary is not able to make the leakage depend on the seed, which then remains independent of the source and leakage-resilience follows from the definition of the seeded extractor. The fact that we are using a seeded extractor with seed length bigger than the output length may seem counter-intuitive. But since the goal here is to provide leakage-resilience instead of extracting randomness, we do not have to force the extractor seed to be shorter than the extractor output.

Related works. The works most related to ours is the following. Bogdanov, Ishai, Viola and Williamson [12] initiated the study of *t-wise indistinguishability* as a relaxation of the well studied notion of *t-wise independence* and considered a pair of such distributions as a statistical secret sharing scheme sharing one bit secret. Through discovering the fact that t-wise indistinguishability implies leakage-resilience against a class of global leakage functions with *approximate degree* smaller than a quantity determined by t, they obtained the first instance of LR-SS in the global leakage model. On one hand, the leakage-resilience is implied by the privacy of the secret sharing shceme (leakage resilience for free). On the other hand, the achieved leakage-resilience is restricted to very limited leakage functions.

As distant related works, we briefly discuss a large body of works (with many overlapping references) on tamper-resilience of secret sharing. Goyal and Kumar [27] initiated the systematic study of Non-Malleable Secret Sharing (NM-SS). A secret sharing is called a NM-SS against certain type of tampering if a tampering either results in the original secret or a random secret whose distribution depends only on the particular tampering function that was applied and independent of the original secret. The recent interests in constructing LR-SS is partially due to its role as an important building block in the constructions of NM-SS. The basic tampering model for NM-SS is the *independent* tampering model where each share is arbitrarily tampered independent of each other [27, 28, 43, 7, 1, 25]. The study of NM-SS is in turn closely related to Non-Malleable Codes (NMC) proposed by Dziembowski, Pietrzak and Wichs [24]. The independent tampering model of NM-SS is corresponding to the *split-state* tampering models studied for NMC. In particular, according to [4], 2-split state NMC's are also 2-out-of-2 NM-SS's [37, 22, 3, 4, 2, 15, 34, 35, 17]. Non-compartmentalized models are well studied in the literature of NMC. Agrawal et.al [5, 6] initiated the study of non-compartmentalized tampering models for NMC. They considered non-malleability against permutation composed with *bit-wise independent tampering*, and showed that non-malleable codes in such a tampering model transform non-malleable bit-commitments into a non-malleable string-commitment. There have been other non-compartmentalized tampering families studied for non-malleable codes: *bounded fan-in* circuits [14], affine functions [16], small-depth circuits [8] and decision tree [9]. Our global affine leakage model can be seen as a leakage-resilient analogue of the non-compartmentalized affine tampering model in NMC. There is not yet a tamper-resilient analogue for our low-degree multi-variate polynomial leakage model.

Paper organisation. The rest of the paper is organised as follows. Section 2 contains the definitions of various randomness extractors that appear in this work. Section 3 starts with a general definition of LR-SS that extends the local LR-SS to include the global leakage models. It is followed by a detailed description of our generic construction. In Section 4, we study the affine leakage models and obtain two sets of results for adaptive adversary and

non-adaptive adversary, respectively. Section 5 is devoted to the low-degree multi-variate polynomial leakage model. In Section 6, we extend our results to give a LR-SS in local leakage model. We conclude our results in Section 7.

2 Preliminaries

The *statistical distance* of two random variables (their corresponding distributions) is defined as follows. For $\mathsf{X}, \mathsf{Y} \leftarrow \Omega$,

$$\mathsf{SD}(\mathsf{X};\mathsf{Y}) = \frac{1}{2}\sum_{\omega\in\Omega} |\Pr(\mathsf{X}=\omega) - \Pr(\mathsf{Y}=\omega)|.$$

We say X and Y are ε-close (denoted $\mathsf{X} \overset{\varepsilon}{\sim} \mathsf{Y}$) if $\mathsf{SD}(\mathsf{X},\mathsf{Y}) \leq \varepsilon$.

We use various types of randomness extractors in our constructions. Randomness extractors extract close to uniform bits from input sequences that are not uniform but have some guaranteed entropy. See [39] and references there in for more information about randomness extractors.

A *randomness source* is a random variable with lower bound on its min-entropy, which is defined by $\mathsf{H}_\infty(\mathsf{X}) = -\log\max_{\mathsf{x}}\{\Pr[\mathsf{X}=\mathsf{x}]\}$. We say a random variable $\mathsf{X} \leftarrow \{0,1\}^n$ is a (n,k)-*source*, if $\mathsf{H}_\infty(\mathsf{X}) \geq k$.

For well structured sources, there exist deterministic functions that can extract close to uniform bits. An *affine* (n,k)-source is a random variable that is uniformly distributed on an affine translation of some k-dimensional sub-space of $\{0,1\}^n$. Let U_m denote the random variable uniformly distributed over $\{0,1\}^m$.

▶ **Definition 1.** *A function* $\mathsf{aExt}\colon \{0,1\}^n \to \{0,1\}^m$ *is an affine* (k,ε)*-extractor if for any affine* (n,k)*-source* X*, we have*

$$\mathsf{SD}(\mathsf{aExt}(\mathsf{X});\mathsf{U}_m) \leq \varepsilon.$$

We will use Bourgain's affine extractor (or the alternative [33] due to Li) in our constructions.

▶ **Lemma 2** ([13])**.** *For every constant* $0 < \mu \leq 1$*, there is an explicit affine extractor* $\mathsf{aExt}\colon \{0,1\}^n \to \{0,1\}^m$ *for affine* $(n, n\mu)$*-sources with output length* $m = \Omega(n)$ *and error at most* $2^{-\Omega(n)}$*.*

An algebraic set is a set of common zeros of one or more multivariate polynomials defined over a finite field. An *algebraic source* is a random variable distributed uniformly over an algebraic set. Algebraic sources are a natural generalization of affine sources that have been widely studied.

▶ **Definition 3.** *A function* $\mathsf{aExt}\colon \{0,1\}^n \to \{0,1\}^m$ *is an algebraic* (k,d,ε)*-extractor if for any degree-*d *algebraic source* $\mathsf{U}_\mathcal{V}$ *with algebraic set* $\mathcal{V} \subseteq \{0,1\}^n$ *and* $|\mathcal{V}| \geq 2^k$*, we have*

$$\mathsf{SD}(\mathsf{aExt}(\mathsf{U}_\mathcal{V});\mathsf{U}_m) \leq \varepsilon.$$

We will use Li and Zuckerman's [32] recent algebraic extractors in our constructions.

▶ **Lemma 4** ([32])**.** *For any positive integer* d*, there is an efficient* $\left((1-\frac{1}{c_d})n, d, 2^{-\Omega(\frac{n}{c_d})}\right)$*-extractor* $\mathsf{aExt}\colon \{0,1\}^n \to \{0,1\}^m$*, where* $c_d = \Theta(d^2 4^d)$ *and* $m = \Omega(\frac{n}{c_d})$*.*

More generally, *recognizable sources* are flat distributions over sets of the form $\{\mathbf{x}|f(\mathbf{x}) = \mathsf{v}\} \subseteq \{0,1\}^n$ for functions f coming from some specified class $\mathcal{C}$. The distribution $\mathsf{U}_{\{\mathbf{x}|f(\mathbf{x})=\mathsf{v}\}}$ is called the f-recognizable source. A $\mathcal{C}$-recognizable source is the set of f-recognizable sources for each $f \in \mathcal{C}$.

For general (n,k)-sources, there does not exist a deterministic function that can extract close to uniform bits from all of them simultaneously. A family of deterministic functions are needed.

▶ **Definition 5.** *A function* $\mathsf{Ext}\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ *is a strong seeded* (k,ε)*-extractor if for any* (n,k)*-source* X*, we have*

$$\mathsf{SD}(\mathsf{S}, \mathsf{Ext}(\mathsf{S},\mathsf{X}); \mathsf{S}, \mathsf{U}_m) \leq \varepsilon,$$

where S *is chosen uniformly from* $\{0,1\}^d$*. A seeded extractor* $\mathsf{Ext}(\cdot,\cdot)$ *is called linear if for any fixed seed* $\mathsf{S} = \mathsf{s}$*, the function* $\mathsf{Ext}(\mathsf{s},\cdot)$ *is a linear function.*

There are linear seeded extractors that extract all the randomness, for example, the Trevisan's extractor [45]. In particular, we use the following improvement of this extractor due to Raz, Reingold and Vadhan [40].

▶ **Lemma 6** ([40])**.** *There is an explicit linear strong* (k,ε)*-extractor* $\mathsf{Ext}\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ *with* $d = O(\log^3(n/\varepsilon))$ *and* $m = k - O(d)$*.*

We will need another explicit strong seeded extractor from universal hash family for our constructions.

▶ **Lemma 7** ([30])**.** *There is an explicit linear strong* $(3m, 2^{-m})$*-extractor* $\mathsf{Ext}\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ *with* $d = 5m$*.*

In the applications of randomness extractors, the source does not necessarily come in the form of one single random variable, but as a pair of random variables and we want to extract from one of them conditioned on the other one. In this general setting, we usually need the *average min-entropy* to measure the amount of available entropy for extraction.

▶ **Definition 8** ([21])**.** *The average conditional min-entropy* $\tilde{\mathsf{H}}_\infty(\mathsf{U}|\mathsf{V})$ *of two random variables* $\mathsf{U} \leftarrow \mathcal{U}$ *and* $\mathsf{V} \leftarrow \mathcal{V}$ *is defined as*

$$\tilde{\mathsf{H}}_\infty(\mathsf{U}|\mathsf{V}) = -\log\left(\sum_{\mathsf{v}\in\mathcal{V}} \mathsf{Pr}[\mathsf{V}=\mathsf{v}] \max_{\mathsf{u}\in\mathcal{U}} \{\mathsf{Pr}[\mathsf{U}=\mathsf{u}|\mathsf{V}=\mathsf{v}]\}\right).$$

The average conditional min-entropy satisfies the following property.

▶ **Lemma 9** ([21])**.** *Let* $\mathsf{V} \leftarrow \mathcal{V}$*. Then the average conditional min-entropy* $\tilde{\mathsf{H}}_\infty(\mathsf{U}|\mathsf{V})$ *is lower bounded as follows.*

$$\tilde{\mathsf{H}}_\infty(\mathsf{U}|\mathsf{V}) \geq \mathsf{H}_\infty(\mathsf{U}) - \log|\mathcal{V}|.$$

We need an *average-case* strong seeded extractor to extract the average conditional min-entropy from such a pair of random variables.

▶ **Definition 10** ([21])**.** *A function* $\mathsf{Ext}\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ *is an average-case strong seeded* (k,ε)*-extractor if for any random variables* $\mathsf{X} \leftarrow \{0,1\}^n$ *and* $\mathsf{V} \leftarrow \mathcal{V}$ *satisfying* $\tilde{\mathsf{H}}_\infty(\mathsf{X}|\mathsf{V}) \geq k$*, we have*

$$\mathsf{SD}(\mathsf{S}, \mathsf{V}, \mathsf{Ext}(\mathsf{S},\mathsf{X}); \mathsf{S}, \mathsf{V}, \mathsf{U}_m) \leq \varepsilon,$$

where S *is chosen uniformly from* $\{0,1\}^d$*.*

Explicit constructions of randomness extractors have efficient forward direction of extraction. In some applications, we usually need to efficiently invert the process: Given an extractor output, sample a random pre-image. This is not necessarily efficient if the extractor is not a linear function, in which case we need to explicitly construct an *invertible extractor*. If the extractor is linear, sampling a random pre-image can be done in polynomial time.

▶ **Definition 11** ([18]). *Let f be a mapping from $\{0,1\}^n$ to $\{0,1\}^m$. For $\mu \geq 0$, a function $\mathsf{Inv}\colon \{0,1\}^m \times \{0,1\}^r \to \{0,1\}^n$ is called a μ-inverter for f if the following conditions hold:*

- *(Inversion) Given $\mathsf{y} \in \{0,1\}^m$ such that its pre-image $f^{-1}(\mathsf{y})$ is nonempty, for every $\mathsf{r} \in \{0,1\}^r$ we have $f(\mathsf{Inv}(\mathsf{y},\mathsf{r})) = \mathsf{y}$.*
- *(Uniformity) $\mathsf{Inv}(\mathsf{U}_m, \mathsf{U}_r)$ is μ-close to U_n.*

A μ-inverter is called efficient if there is a randomized algorithm that runs in worst-case polynomial time and, given $\mathsf{y} \in \{0,1\}^m$ and r as a random seed, computes $\mathsf{Inv}(\mathsf{y},\mathsf{r})$. We call a mapping μ-invertible if it has an efficient μ-inverter, and drop the prefix μ from the notation when it is zero. We abuse the notation and denote the inverter of f by f^{-1}.

Finally, we need the following simple lemma whose proof can be found in the full version.

▶ **Lemma 12.** *Let V,V' be two random variables distributed over the set $\mathcal{V}$ and W,W' over $\mathcal{W}$ satisfying $\mathsf{SD}(\mathsf{V},\mathsf{W};\mathsf{V}',\mathsf{W}') \leq \varepsilon$. Let $\mathcal{E} \subset \mathcal{W}$ be an event. Then we have the following.*

$$\mathsf{SD}(\mathsf{V}|\mathsf{W} \in \mathcal{E}; \mathsf{V}'|\mathsf{W}' \in \mathcal{E}) \leq \frac{2\varepsilon}{\mathsf{Pr}[\mathsf{W}' \in \mathcal{E}]}.$$

3 Leakage Resilient Secret Sharing

In this section, we define a general leakage model for secret sharing, which can be viewed as an extension of the local leakage model proposed in [10, 27] to include non-compartmentalised leakage models.

▶ **Definition 13.** *Let t and n satisfying $2 \leq \mathsf{t} \leq \mathsf{n}$ be two integers. Let $\mathbb{F}_q$ be the finite field of q elements for q a power of 2. Let $\mathcal{L}$ be a set of Boolean functions of $\mathsf{n}\log q$ bits input. Let β be an integer denoting the bound on the total number of bits leaked. Let $\theta < \mathsf{t}$ be an integer and ε be a small positive real. A $(\mathcal{L},\beta,\theta,\varepsilon)$-leakage resilient t-out-of-n secret sharing scheme over the finite field $\mathbb{F}_q$ is defined by a pair of polynomial-time algorithms $(\mathsf{Share},\mathsf{Rec})$, where Share is a randomized mapping of an input $\mathbf{s} \in \{0,1\}^m$, for $m \leq \log q$, to a share vector $\mathbf{Sh} = (\mathsf{Sh}_1,\ldots,\mathsf{Sh}_\mathsf{n})$ and the reconstruction algorithm Rec is a deterministic function mapping a set $A \subseteq [\mathsf{n}]$ and the corresponding shares $\mathbf{Sh}_A = (\mathsf{Sh}_i)_{i\in A}$ to a secret in $\{0,1\}^m$, such that the following properties hold:*

- *Correctness: $\mathsf{Rec}(A,\mathbf{Sh}_A)$ outputs the secret $\mathbf{s}$ for all sets $A \subseteq [\mathsf{n}]$ where $|A| \geq \mathsf{t}$.*
- *Privacy and leakage resiliency:*
  - *Non-adaptive adversary: for any pair $\mathsf{s}^0,\mathsf{s}^1 \in \{0,1\}^m$ of secrets with share vectors $\mathbf{Sh}^0$ and $\mathbf{Sh}^1$, any $A \subseteq [\mathsf{n}]$ of size $|A| \leq \theta$, any β functions $f_i \in \mathcal{L}$, $i = 1,\ldots,\beta$,*

$$\mathsf{SD}\left(\mathsf{Leak}_{A,\beta}(\mathbf{Sh}^0), \mathbf{Sh}^0_A; \mathsf{Leak}_{A,\beta}(\mathbf{Sh}^1), \mathbf{Sh}^1_A\right) \leq \varepsilon, \tag{1}$$

where $\mathsf{Leak}_{A,\beta}(\mathbf{Sh}^b)$ denotes the output of the Boolean functions $f_1,\ldots,f_\beta$ on input $\mathbf{Sh}^b$ for $b \in \{0,1\}$. In the case when $\mathcal{L}$ is defined such that the input of these Boolean functions $f_1,\ldots,f_\beta$ are restricted to one share and otherwise not restricted, we recover the local leakage model.

- *Adaptive adversary: for any pair* $\mathsf{s}^0, \mathsf{s}^1 \in \{0,1\}^m$ *of secrets with share vectors* $\mathbf{Sh}^0$ *and* $\mathbf{Sh}^1$*,*

$$\mathsf{SD}\left(\mathsf{LEAK}(\mathbf{Sh}^0, \mathcal{L}, \beta, \theta); \mathsf{LEAK}(\mathbf{Sh}^1, \mathcal{L}, \beta, \theta)\right) \leq \varepsilon, \tag{2}$$

where $\mathsf{LEAK}(\mathbf{Sh}^b, \mathcal{L}, \beta, \theta)$ *denotes the transcript of the following interactive protocol between the adversary* $\mathcal{A}$ *and an oracle* $\mathcal{O}$ *holding* $\mathbf{Sh}^b$*. For* $i = 1, \ldots, \beta$ *and* $j = 1, \ldots, \theta$*,* $\mathcal{A}$ *chooses* $f_i \in \mathcal{L}$ *and* $\mathsf{n}_j \in [\mathsf{n}]$ *based on all previous communication, and* $\mathcal{O}$ *answers with* $f_i(\mathbf{Sh}^b)$ *and* $\mathbf{Sh}^b_{\mathsf{n}_j}$*, respectively.*

When $\beta = 0$ *and* $\theta = \mathsf{t} - 1$ *is achieved, we recover the statistical privacy of a threshold scheme. When* $\beta = 0$ *and only* $\theta < \mathsf{t} - 1$ *is achieved, the scheme is a ramp scheme with statistical privacy for privacy threshold* θ*.*

Generic Construction

$$\begin{cases} \mathsf{Share}(\cdot) & = \mathsf{ECCenc}(\mathsf{EXT}^{-1}(\cdot)); \\ \mathsf{Rec}(\cdot) & = \mathsf{EXT}(\mathsf{ECCdec}(\cdot)). \end{cases}$$

The ECC is an erasure correcting code with encoder/decoder pair $(\mathsf{ECCenc}, \mathsf{ECCdec})$ and EXT is an invertible randomness extractor with inverter EXT^{-1}.

In most of the instantiations, EXT is a seedless extractor. All the efficient extractors mentioned in the preliminary section (if not already invertible) can be transformed into one that is invertible, at the cost of increasing the input length, using the following method. Let $\mathsf{EXT}_0 \colon \{0,1\}^{n_0} \to \{0,1\}^m$ be an (n_0, k)-extractor with error ε. Let $n = n_0 + m$. Then $\mathsf{EXT} : \{0,1\}^{n_0+m} \to \{0,1\}^m$ defined as follows is a ε-invertible $(n, k+m)$-extractor with error ε.

$$\mathsf{EXT}(\mathsf{x}||\mathsf{y}) = \mathsf{EXT}_0(\mathsf{x}) + \mathsf{y},$$

where "||" denotes concatenation. The inverter of EXT is

$$\mathsf{EXT}^{-1}(\mathsf{s}) = (\mathsf{U}_{n_0}||\mathsf{EXT}_0(\mathsf{U}_{n_0}) + \mathsf{s}),$$

where the two copies of U_{n_0} denote the same random variable. In the case when EXT is a seeded extractor, its inverter EXT^{-1} uniformly chooses a seed and invert according to the function labeled by the seed. More concretely, let $\mathsf{Ext} \colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ be an invertible seeded extractor. We use the short hand $\mathsf{Ext}_{\mathsf{z}}(\cdot) = \mathsf{Ext}(\mathsf{z}, \cdot)$ and $\mathsf{EXT}(\cdot) = \mathsf{Ext}_{\mathsf{U}_d}(\cdot)$. Then the inverter $\mathsf{EXT}^{-1} \colon \{0,1\}^m \to \{0,1\}^n$ is defined as follows.

$$\mathsf{EXT}^{-1}(\mathsf{s}) = \mathsf{EXT}^{-1}_{\mathsf{U}_d}(\mathsf{s}),\ \mathsf{U}_d \xleftarrow{\$} \{0,1\}^d.$$

Though not explicitly reflected in the notations above, the uniform seed chosen in the process of inverting is also recorded so that the reconstruction algorithm can correctly recover the secret. For example, in the construction for global affine leakage (Theorem 14), the seed is directly appended to the n-bit pre-image.

In all the instantiations, ECC is a linear Maximum Distance Separable (MDS) code over a large enough finite field $\mathbb{F}_q$ of characteristic 2. For example, we require the share size q satisfy $q > \mathsf{n}$ if we use a $[\mathsf{n}, \mathsf{t}, \mathsf{n} - \mathsf{t} + 1]_q$ Reed-Solomon code and the input length $n = \mathsf{t} \log q$ of the invertible extractor EXT should be big enough to achieve the error ε.

4 Affine Leakage Models

In this section, we study LR-SS with respect to the class $\mathcal{L}_{\mathsf{affine}}$ of affine functions over $\mathbb{F}_2$. We have two instantiations of the generic construction for non-adaptive and adaptive adversary, respectively.

4.1 Affine Leakage Non-Adaptive Adversary

We begin with an instantiation that gives optimal parameters in terms of both leakage tolerance and information ratio, though we can only prove security against a non-adaptive adversary.

▶ **Theorem 14.** *Let $\mathcal{L}_{\mathsf{affine}}$ be the class of affine functions over $\mathbb{F}_2$. Let $0 < \xi \leq 1$ be any real number. There is a family of t-out-of-n secret sharing schemes over $\mathbb{F}_q$ (labeled by $\log q$) that is simultaneously $(\mathcal{L}_{\mathsf{affine}}, \beta, \theta, \varepsilon)$-leakage resilient against a non-adaptive adversary for $0 \leq \beta \leq (\mathsf{t} - \xi)\log q$ and $\theta \in \{0, 1, \ldots, \mathsf{t}-1\}$ such that $\theta + \frac{\beta}{\log q} \leq \mathsf{t} - \xi$, where the share size q (determined by ε and satisfies $q > \mathsf{n}$) is a large enough power of 2 and the secret length $m = \xi \log q - o(\log q)$.*

Proof. We instantiate the generic construction using a linear strong seeded extractor $\mathsf{Ext}\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ and a linear MDS code.

- $\mathsf{EXT} : \{0,1\}^n \to \{0,1\}^m$ is instantiated as follows.

  $$\mathsf{EXT}(\mathsf{x}) = \mathsf{Ext}(\mathsf{U}_d, \mathsf{x}),$$

  where Ext is the linear strong seeded extractor from Lemma 6 with parameters (k_E, ε_E). The uniform seed sampled during inverting is directly appended to the n-bit pre-image $\mathsf{EXT}^{-1}(\mathsf{s})$.
  According to Lemma 6, there is an explicit linear strong (k_E, ε_E)-extractor $\mathsf{Ext}\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ with $d = O(\log^3(n/\varepsilon_E))$ and $m = k_E - O(d)$.
- ECC is a linear MDS code with parameter $[\mathsf{n}, \mathsf{t}, \mathsf{n} - \mathsf{t} + 1]$ over $\mathbb{F}_q$, where $\log q = \frac{d+n}{\mathsf{t}}$. The output of $\mathsf{ECCenc}\colon \{0,1\}^{d+n} \to \mathbb{F}_q^{\mathsf{n}}$ is the share vector.

Reconstruction from any t shares follows from the functionality of ECC and the invertibility guarantee of the EXT, which insures that any correctly recovered pre-image is mapped back to the original secret.

We next prove privacy and leakage resiliency, which will follow naturally from Lemma 15. We first recall this general property of a linear strong extractor, which is proved in [36].

▶ **Lemma 15** ([36])**.** *Let $\mathsf{Ext}\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ be a linear strong (k, ε)-extractor, $f\colon \{0,1\}^{d+n} \to \{0,1\}^a$ be an affine function with output length $a \leq n - k$. For any $\mathsf{m}, \mathsf{m}' \in \{0,1\}^m$, let $(\mathsf{Z}, \mathsf{X}) = (\mathsf{U}_d, \mathsf{U}_n) | (\mathsf{Ext}(\mathsf{U}_d, \mathsf{U}_n) = \mathsf{m})$ and $(\mathsf{Z}', \mathsf{X}') = (\mathsf{U}_d, \mathsf{U}_n) | (\mathsf{Ext}(\mathsf{U}_d, \mathsf{U}_n) = \mathsf{m}')$. We have*

$$\mathsf{SD}(f(\mathsf{Z}, \mathsf{X}); f(\mathsf{Z}', \mathsf{X}')) \leq 8\varepsilon. \tag{3}$$

The inverter EXT^{-1} takes a secret, which is a particular extractor output $\mathsf{s} \in \{0,1\}^m$, and uniformly samples a seed $\mathsf{z} \in \{0,1\}^d$ of Ext before uniformly finds an $\mathsf{x} \in \{0,1\}^n$ such that $\mathsf{Ext}(\mathsf{z}, \mathsf{x}) = \mathsf{s}$. This process of obtaining (z, x) is the same as sampling uniformly and independently $(\mathsf{U}_d, \mathsf{U}_n) \xleftarrow{\$} \{0,1\}^{d+n}$ and then restricting to $\mathsf{Ext}(\mathsf{U}_d, \mathsf{U}_n) = \mathsf{s}$. We define the random variable pair

$$(\mathsf{Z}, \mathsf{X}) := (\mathsf{U}_d, \mathsf{U}_n) | (\mathsf{Ext}(\mathsf{U}_d, \mathsf{U}_n) = \mathsf{s}) \tag{4}$$

and refer to it as the pre-image of s.

Let $\Pi_A : \mathbb{F}_q^{\mathsf{n}} \to \mathbb{F}_q^{\theta}$ be the projection function that maps a share vector to the θ shares with index set $A \subseteq [\mathsf{n}]$ and $|A| = \theta$ chosen by the non-adaptive adversary. Observe that the combination $(\Pi_A \circ \mathsf{ECCenc}) : \{0,1\}^{d+n} \to \{0,1\}^{(\log q)\theta}$ is an affine function. Moreover, for any affine leakage function $l : \{0,1\}^{(\log q)\mathsf{n}} \to \{0,1\}^{\beta}$, the composition $(l \circ \mathsf{ECCenc}) : \{0,1\}^{d+n} \to \{0,1\}^{\beta}$ is also an affine function. So the view of the adversary is simply the output of the affine function $f = (\Pi_A \circ \mathsf{ECCenc} || l \circ \mathsf{ECCenc})$, where "||" denotes concatenation, applied to the random variable tuple (Z, X) defined in (4).

We want to prove that the statistical distance of the views of the adversary for a pair of secrets s and s' can be made arbitrarily small. The views of the adversary are the outputs of the affine function f with inputs $(\mathsf{Z}^0, \mathsf{X}^0)$ and $(\mathsf{Z}^1, \mathsf{X}^1)$ for the secret s^0 and s^1, respectively. Let $k_E = n - (\log q)\theta - \beta$. According to Lemma 15, we then have that the privacy and leakage resiliency error is $8\varepsilon_E$.

Finally, since $\theta + \frac{\beta}{\log q} < \mathsf{t} - \xi$, we have $\mathsf{t} - \theta - \frac{\beta}{\log q} > \xi$. Let $d + n$ be a multiple of t. We then have a family of schemes labeled by $\log q$. The privacy and leakage resiliency error $8\varepsilon_E$ is negligible in n, and hence is obviously negligible in $\log q$. The secret length can be chosen as follows.

$$m = n - (\log q)\theta - \beta - O(d) = (\log q)(\mathsf{t} - \theta - \frac{\beta}{\log q}) - o(\log q) = \xi \log q - o(\log q),$$

where the seed length is $d = O(\log^3(2n/\varepsilon))$ and $d + n = (\log q)\mathsf{t}$. ◀

▶ Remark 16. Let $\xi = 1$ and consider $\theta = \mathsf{t} - 1$ and $\beta = 0$. In this case, we recover a t-out-of-n secret sharing scheme with information ratio $\frac{m}{\log q} \approx 1$, for large enough q. This is almost as good as the Shamir's secret sharing scheme, except that Shamir's scheme has perfect privacy and information ratio exactly 1 while we only have statistical privacy with a negligible privacy error and information ratio close to 1. On the other hand, as demonstrated in [29], Shamir's scheme over finite field of characteristic 2 is completely vulnerable in the face of a non-standard leakage adversary, in particular, even leaking one bit from each share allows reconstruction of the complete secret. The t-out-of-n secret sharing scheme in Theorem 14, when we set $m = \log q - o(\log q)$, is leakage-resilient against a non-adaptive adversary who obtains any $\theta \leq \mathsf{t} - 1$ shares and leak up to $\beta = (\log q)\phi$ bits for any $0 \leq \phi \leq \mathsf{t} - 1 - \theta$ through applying affine leakage functions.

4.2 Affine Leakage Adaptive Adversary

The instantiation for adaptive adversary does not have optimal information ratio. We manage to maintain optimal leakage tolerance.

▶ **Theorem 17.** *Let $\mathcal{L}_{\mathsf{affine}}$ be the class of affine functions over $\mathbb{F}_2$. Let $0 < \xi \leq 1$ be any real number. There is a family of t-out-of-n secret sharing schemes over $\mathbb{F}_q$ (labeled by $\log q$) that is simultaneously $(\mathcal{L}_{\mathsf{affine}}, \beta, \theta, \varepsilon)$-leakage resilient against an adaptive adversary for $0 \leq \beta < (\mathsf{t} - \xi)\log q$ and $\theta \in \{0, 1, \ldots, \mathsf{t} - 1\}$ such that $\theta + \frac{\beta}{\log q} \leq \mathsf{t} - \xi$, where the share size q (determined by ε and satisfies $q > \mathsf{n}$) is a large enough power of 2 and the secret length $m = \Omega(\log q)$.*

We use an invertible seedless extractors that can extract from affine recognizable sources with any constant fraction of entropy with output length also a constant fraction of the input and with exponentially small error. There are known constructions [13, 33] of affine extractors that can extract from any constant fraction of entropy and output a constant fraction of random bits with exponentially small error. We can directly instantiate our generic construction with these extractors by transforming them into invertible extractors as described in the generic construction.

Here we exploit a classical approach to building affine extractors through composing with a seeded extractor and use the trick in [18] to make it invertible at a lower cost than the above method. But the obtained affine extractor does not have exponentially small error and we can not directly use it in the generic construction intuition due to the exponential grow of error when transforming from extractor-based security to secret sharing security. We then use a more delicate analysis of the error terms that circumvents the problem.

We first recall the classical framework of constructing seedless extractors from composing with seeded extractors. Seeded extractors are known to explicitly extract all the entropy and are not restricted by source structures. Moreover, there are known constructions of *linear* seeded extractors perform almost as well as the best seeded extractors. The elegant idea of this framework is to use a seedless extractor to extract a short output from the structured source, which then serves as the seed for a seeded extractor to extract all the entropy from the same source. For this idea to work, the dependence of the extracted seed on the source has to be carefully analyzed (and removed).

▶ **Lemma 18** ([41]). *Let $\mathcal{D}$ be a class of distributions over $\{0,1\}^n$. Let $\mathsf{E}:\{0,1\}^n\to\{0,1\}^d$ be a seedless extractor for $\mathcal{D}$ with error ϵ. Let $\mathsf{F}:\{0,1\}^d\times\{0,1\}^n\to\{0,1\}^m$. Let X be a distribution in $\mathcal{D}$ and assume that for every $\mathsf{z}\in\{0,1\}^d$ and $\mathsf{y}\in\{0,1\}^m$, the distribution $(\mathsf{X}|\mathsf{F}(\mathsf{z},\mathsf{X})=\mathsf{y})$ belongs to $\mathcal{D}$. Then*

$$\mathsf{SD}(\mathsf{E}(\mathsf{X}),\mathsf{F}(\mathsf{E}(\mathsf{X}),\mathsf{X});\mathsf{U}_d,\mathsf{F}(\mathsf{U}_d,\mathsf{X}))\le 2^{d+3}\epsilon.$$

An example of such a class of distributions is the affine source, in which case we can use an *affine* extractor $\mathsf{F}=\mathsf{aExt}$ and a *linear* seeded extractor $\mathsf{E}=\mathsf{Ext}$. An affine source X conditioned on $\mathsf{Ext}(\mathsf{z},\mathsf{X})=\mathsf{y}$, which amounts to a set of linear equations, is still an affine source for aExt. With appropriate choice of parameters, we obtain a better (than aExt) affine extractor $\mathsf{aExt}'(\mathsf{X}):=\mathsf{Ext}(\mathsf{aExt}(\mathsf{X}),\mathsf{X})$. With an increase of d bits (instead of m bits as described below generic construction) in the input, we have the following invertible affine extractor.

$$\mathsf{EXT}(\mathsf{Sd}||\mathsf{X}):=\mathsf{Ext}(\mathsf{aExt}(\mathsf{X})+\mathsf{Sd},\mathsf{X}),$$

whose inverter is $\mathsf{EXT}^{-1}(\mathsf{s}):=\left(\mathsf{aExt}(\mathsf{Ext}_{\mathsf{Z}}^{-1}(\mathsf{s}))+\mathsf{Z}||\mathsf{Ext}_{\mathsf{Z}}^{-1}(\mathsf{s})\right)$, where $\mathsf{Z}\xleftarrow{\$}\{0,1\}^d$.

Proof of Theorem 17. We instantiate the generic construction using an affine extractor $\mathsf{aExt}:\{0,1\}^n\to\{0,1\}^d$, a linear strong seeded extractor $\mathsf{Ext}:\{0,1\}^d\times\{0,1\}^n\to\{0,1\}^m$ and a linear MDS code.

- $\mathsf{EXT}:\{0,1\}^{d+n}\to\{0,1\}^m$ is instantiated as follows.

  $$\mathsf{EXT}(\mathsf{z}||\mathsf{x})=\mathsf{Ext}(\mathsf{aExt}(\mathsf{x})+\mathsf{z},\mathsf{x}),$$

  where Ext is the linear strong seeded extractor from Lemma 6 with parameters (k_E,ε_E) and $\varepsilon_E<\frac{1}{8}$; aExt is the seedless extractor from Lemma 2 with parameters (k_A,ε_A). The inverter

  $$\mathsf{EXT}^{-1}(\mathsf{s}):=\left(\mathsf{aExt}(\mathsf{Ext}_{\mathsf{Z}}^{-1}(\mathsf{s}))+\mathsf{Z}||\mathsf{Ext}_{\mathsf{Z}}^{-1}(\mathsf{s})\right),$$

  where $\mathsf{Z}\xleftarrow{\$}\{0,1\}^d$
  According to Lemma 6, there is an explicit linear strong (k_E,ε_E)-extractor $\mathsf{Ext}:\{0,1\}^d\times\{0,1\}^n\to\{0,1\}^m$ with $d=O(\log^3(n/\varepsilon_E))$ and $m=k_E-O(d)$.
  According to Lemma 2, for every constant $0<\mu\le 1$, there is an explicit affine extractor $\mathsf{aExt}:\{0,1\}^n\to\{0,1\}^d$ for affine (n,k_A)-sources, where $k_A=n\mu$, with output length $d=\Omega(n)$ and error ε_A at most $2^{-\Omega(n)}$.

- ECC is a linear MDS code with parameter $[\mathsf{n}, \mathsf{t}, \mathsf{n}-\mathsf{t}+1]$ over $\mathbb{F}_q$, where $\log q = \frac{d+n}{\mathsf{t}}$. The output of $\mathsf{ECCenc}\colon \{0,1\}^{d+n} \to \mathbb{F}_q^{\mathsf{n}}$ is the share vector.

Reconstruction from any t shares follows from the functionality of ECC and the invertibility guarantee of the EXT, which insures that any correctly recovered pre-image is mapped back to the original secret.

We next prove privacy and leakage resiliency. Consider a uniform secret U_m. By the uniformity guarantee of the inverter, we have $\mathsf{Share}(\mathsf{U}_m) = \mathsf{ECCenc}(\mathsf{Sd}||\mathsf{U}_n)$. Our analysis is done for any fixed $\mathsf{Sd} = \mathsf{sd}$. This captures a stronger adversary who on top of adaptively reading t shares, also has access to Sd through an oracle. It is easy to see that the fixing of $\mathsf{Sd} = \mathsf{sd}$ does not alter the distribution of the source U_n, which remains uniform over $\{0,1\}^n$. Let $\mathsf{V} \colon = \mathsf{LEAK}(\mathsf{ECCenc}(\mathsf{sd}||\mathsf{U}_n), \mathcal{L}_{\mathsf{affine}}, \beta, \theta)$ denote the view of the adversary on the encoding of a uniform source for the fixed $\mathsf{Sd} = \mathsf{sd}$. Let $\mathsf{Z} \colon = \mathsf{aExt}(\mathsf{U}_n) + \mathsf{sd}$ denote the seed of the strong linear extractor Ext. Finally, let $\mathsf{S} \colon = \mathsf{Ext}(\mathsf{Z}, \mathsf{U}_n)$. We study the random variable tuple $(\mathsf{V}, \mathsf{Z}, \mathsf{S})$ to complete the proof.

The pair $(\mathsf{Z}, \mathsf{S})|\mathsf{V} = \mathsf{v}$ for any fixed $\mathsf{V} = \mathsf{v}$ is by definition $(\mathsf{aExt}(\mathsf{U}_n) + \mathsf{sd}, \mathsf{Ext}(\mathsf{aExt}(\mathsf{U}_n) + \mathsf{sd}, \mathsf{U}_n))|\mathsf{V} = \mathsf{v}$. The distribution $(\mathsf{U}_n|\mathsf{V} = \mathsf{v})$ is an affine source with at least $n - (\log q)\theta - \beta$ entropy. Let $k_A = n - \theta \log q - \beta - m$. According to Lemma 18, we have

$$(\mathsf{Z}, \mathsf{S})|\mathsf{V} = \mathsf{v} \overset{2^{d+3}\varepsilon_A}{\sim} (\mathsf{U}_d, \mathsf{Ext}(\mathsf{U}_d, \mathsf{U}_n))|\mathsf{V} = \mathsf{v}.$$

Our concern is the relation between S and V, and therefore would like to further condition on values of Z. Let $k_E = n - (\log q)\theta - \beta - d$ and consider the linear strong extractor Ext. In this step, we crucially use the linearity of Ext and the underlying linear space structure of the affine source $\mathsf{U}_n|\mathsf{V} = \mathsf{v}$ to claim that there is a subset $\mathcal{G} \subset \{0,1\}^d$ of good seeds such that $\mathsf{Pr}[\mathsf{U}_d \in \mathcal{G}] \geq 1 - 4\varepsilon_E$ and for any $\mathsf{z} \in \mathcal{G}$, the distribution of $\mathsf{Ext}(\mathsf{z}, \mathsf{U}_n)|\mathsf{V} = \mathsf{v}$ is exactly uniform. This is true because $\mathsf{Ext}(\mathsf{z}, \mathsf{U}_n)|\mathsf{V} = \mathsf{v}$ is an affine source. If its entropy is m, then it is exactly uniform. If its entropy is less than m, its statistical distance $\varepsilon_E^{\mathsf{z}}$ from uniform is at least $\frac{1}{2}$. Using an averaging argument we have that at least $1 - 4\varepsilon_E$ fraction of the seeds should satisfy $\varepsilon_E^{\mathsf{z}} < \frac{1}{4}$, and hence $\varepsilon_E^{\mathsf{z}} = 0$. We then use Lemma 12 with respect to the event $\mathsf{Z} \in \mathcal{G}$ to claim that

$$(\mathsf{S}|(\mathsf{V} = \mathsf{v}, \mathsf{Z} \in \mathcal{G})) \overset{\frac{2^{d+4}\varepsilon_A}{1-4\varepsilon_E}}{\sim} (\mathsf{Ext}(\mathsf{U}_d, \mathsf{X})|(\mathsf{V} = \mathsf{v}, \mathsf{U}_d \in \mathcal{G})),$$

where the right hand side is exactly U_m. Note that the subset $\mathcal{G}$ is determined by the choice of the θ shares and by the leakage adversary, hence remains the same for any value of $\mathsf{V} = \mathsf{v}$. We then have

$$((\mathsf{V}, \mathsf{S})|\mathsf{Z} \in \mathcal{G}) \overset{\frac{2^{d+4}\varepsilon_A}{1-4\varepsilon_E}}{\sim} (\mathsf{V}, \mathsf{U}_m).$$

Another application of Lemma 12 with respect to the event $\mathsf{S} = \mathsf{s}$ gives

$$(\mathsf{V}|(\mathsf{Z} \in \mathcal{G}, \mathsf{S} = \mathsf{s})) \overset{\frac{2^{(m+1)+(d+4)}\varepsilon_A}{1-4\varepsilon_E}}{\sim} \mathsf{V}.$$

We finally bound the privacy and leakage resiliency error as follows.

$$\begin{aligned}
&\mathsf{SD}((\mathsf{V}|\mathsf{S} = \mathsf{s}_0); (\mathsf{V}|\mathsf{S} = \mathsf{s}_1)) \\
&\leq 2\mathsf{SD}((\mathsf{V}|\mathsf{S} = \mathsf{s}); \mathsf{V}) \\
&= 2\mathsf{Pr}[\mathsf{Z} \in \mathcal{G}] \cdot \mathsf{SD}((\mathsf{V}|(\mathsf{Z} \in \mathcal{G}, \mathsf{S} = \mathsf{s})); \mathsf{V}) + 2\mathsf{Pr}[\mathsf{Z} \notin \mathcal{G}] \cdot \mathsf{SD}((\mathsf{V}|(\mathsf{Z} \notin \mathcal{G}, \mathsf{S} = \mathsf{s})); \mathsf{V}) \\
&\leq 2\left(1 \cdot \tfrac{2^{(m+1)+(d+4)}\varepsilon_A}{1-4\varepsilon_E} + (4\varepsilon_E + \varepsilon_A) \cdot 1\right) \\
&< 2^{(m+1)+(d+4)+2}\varepsilon_A + 8\varepsilon_E.
\end{aligned}$$

Note that in the error bound above, the exponential term $2^{(m+1)+(d+4)+2}$ only appears as the multiplier of ε_A, the error of aExt. In order to cancel out the exponential multiplier $2^{(m+1)+(d+4)+2}$, we require aExt to have an exponentially small error $\varepsilon_A = \frac{\varepsilon/2}{2^{(m+1)+(d+4)+2}}$, which can be trivially done by setting $k_A = \Omega(n)$. If we want to have secret length $m = \Omega(\log q) = \Omega(\frac{d+n}{\mathsf{t}})$, we also need to set $k_E = m + O(d) = \Omega(n)$. Both are achieved by requiring

$$n - (\log q)\theta - \beta - d = \Omega(n),$$

which in turn, given that $d = O(\log^3(n/\varepsilon_E))$, can be achieved by requiring

$$n(1 - \frac{\theta}{\mathsf{t}} - \frac{\beta}{n}) = \Omega(n) \text{ or } 1 - \frac{\theta}{\mathsf{t}} - \frac{\beta}{t \log q} > 0.$$

This shows that for any $\theta + \frac{\beta}{\log q} < \mathsf{t}$, privacy and leakage resiliency error ε can be achieved through a large enough q and the secret length is $m = \Omega(\log q)$. This concludes the proof. ◀

▶ Remark 19. Directly using the affine extractor aExt and transform it using the method in the generic construction will result in an invertible affine extractor $\mathsf{EXT}\colon \{0,1\}^{m+n} \to \{0,1\}^m$ with exponentially small error. Instantiating the generic construction using this extractor also gives a secret sharing scheme that is simultaneously $(\mathcal{L}_{\mathsf{affine}}, \beta, \theta, \varepsilon)$-leakage resilient for any $\theta + \frac{\beta}{\log q} < \mathsf{t}$. But the information ratio of this instantiation is $\frac{m}{(m+n)/\mathsf{t}}$, while in Theorem 17, the information ratio is $\frac{m}{(d+n)/\mathsf{t}}$. Recall that $d = O(\log^3(n/\varepsilon_E))$ and $m = \Omega(n)$. We then have $\frac{m}{(d+n)/\mathsf{t}} \approx \frac{m}{n/\mathsf{t}} > \frac{m}{(m+n)/\mathsf{t}}$.

5 Low-degree Multi-variate Polynomial Leakage

We next consider the class $\mathcal{L}_{d\text{-}\mathsf{poly}}$ of global leakage functions that are multi-variate polynomials in binary variables $x_1, \ldots, x_{\mathsf{n}\log q}$ with degree at most d, as natural extension of the affine (degree 1) leakage functions $\mathcal{L}_{\mathsf{affine}}$. From now on, we assume the degree of the algebraic leakage functions are bigger than 1 (for $d = 1$, our constructions in previous section give better parameters).

We have seen in Theorem 14 that as long as the adversary is non-adaptive and restricted to affine leakage, we can instantiate the generic construction with *any* seeded extractor to obtain a LR-SS against global affine leakage. Unfortunately, we can already give an example of seeded extractor that is not sufficient for providing privacy and leakage resiliency against a non-adaptive adversary who globally leaks through degree 2 multi-variate polynomials. It is well known that the inner product function $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ gives a good seeded extractor. This function can be described by a degree 2 polynomial in $2n$ variables. Assuming our generic construction is instantiated using the inner product seeded extractor and a non-adaptive adversary chooses exactly the corresponding degree 2 multi-variate polynomial to leak, the single bit leakage is the secret itself and the scheme is not leakage-resilient. Note that this example does not rule out the possibility of obtaining non-adaptive LR-SS against algebraic leakage of degree $d > 1$ using a specially chosen seeded extractor, for example, not computable by degree d polynomials. In the standard applications of seeded extractors, we only require the extractor function to be efficient. Here we need extractor functions to at least have degree more than d. For simplicity, in this work, we only use seedless extractors (for algebraically recognizable sources), which by definition already takes the structure of the leakage functions into account. Instantiating with a seedless extractor has an advantage of providing security against both a non-adaptive adversary and an adaptive adversary.

▶ **Theorem 20.** *Let $\mathcal{L}_{d\text{-poly}}$ be the class of multi-variate polynomials over $\mathbb{F}_2$ with degree at most d. Let g be an integer satisfying $\frac{g}{\mathsf{t}} > 1 - \frac{1}{c_d}$, where $c_d = \Theta(d^2 4^d)$ is a constant determined by d. There is a family of t-out-of-n secret sharing scheme with threshold gap g over the finite field $\mathbb{F}_q$ (labeled by $\log q$) that is simultaneously $(\mathcal{L}_{d\text{-poly}}, \beta, \theta, \varepsilon)$-leakage resilient for all $\beta = (\log q)\phi$ and $\theta \leq \mathsf{t} - g$ satisfying $0 \leq \phi \leq \mathsf{t} - g - \theta$. The share size q (determined by ε and satisfying $q > \mathsf{n}$) is a large enough power of 2 and the secret length $m = \Omega(\log q)$.*

Before we prove Theorem 20 through instantiating our generic construction, we need to find a suitable measure on the amount of remaining entropy conditioned on the leaked information. The average min-entropy is usually used in combination with the so-called *average-case* extractors, which are strong seeded extractors that have the extractor guarantee averaging over an extra random variable that is related to the source. As far as we know, average-case extractors have not been defined in the seedless case. We then include a brief discussion here that might be of independent interest.

▶ **Definition 21.** *A (k, ε)-seedless extractor $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^m$ for $\mathcal{C}$-recognizable sources is called average-case, if for all pair of random variables (X, V), where X is a $\mathcal{C}$-recognizable source and V is a random variable arbitrarily related to X satisfying that $\mathsf{X}|\mathsf{V} = \mathsf{v}$ is a $\mathcal{C}$-recognizable source for any $\mathsf{V} = \mathsf{v}$ and $\tilde{\mathsf{H}}_\infty(\mathsf{X}|\mathsf{V}) \geq k$, we have*

$$\mathsf{SD}(\mathsf{V}, \mathsf{Ext}(\mathsf{X}); \mathsf{V}, \mathsf{U}_m) \leq \varepsilon.$$

While any strong seeded extractor can be trivially converted into an average-case extractor at the cost of an increase in the extractor error and a proportional strengthening on the min-entropy requirement, there are extractor constructions that directly provide an average-case extractor [21]. This is also the case for seedless extractors.

▶ **Lemma 22** (modified from [21])**.** *For any $\delta > 0$, if Ext is a $(k - \log 1/\delta, \varepsilon)$-seedless extractor for $\mathcal{C}$-recognizable sources, then Ext is also an average-case $(k, \varepsilon + \delta)$-seedless extractor for $\mathcal{C}$-recognizable sources.*

Sometimes the strengthening on entropy requirement can be quite costly ($\log 1/\delta$ can be a large value if the extractor error $\varepsilon + \delta$ has to be exponentially small). Luckily, by examining the construction of the extractor in Lemma 4, we assert that it is already an average-case seedless extractor.

▶ **Lemma 23.** *For any integer $d > 0$, there is an efficient average-case $\left((1 - \frac{1}{c_d})n, d, 2^{-\Omega(\frac{n}{c_d})}\right)$-extractor $\mathsf{aExt}\colon \{0,1\}^n \to \{0,1\}^m$, where $c_d = \Theta(d^2 4^d)$ and $m = \Omega(\frac{n}{c_d})$.*

We now instantiate our generic construction with the average case seedless extractor for low-degree multi-variate polynomials in Lemma 23 to give a proof for Theorem 20.

Proof for Theorem 20. We instantiate the generic construction with the average case seedless extractor for sources recognizable by low-degree multi-variate polynomial functions in Lemma 23.

- $\mathsf{EXT} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^m$ is instantiated as follows.

  $$\mathsf{EXT}(\mathsf{x}||\mathsf{y}) = \mathsf{aExt}(\mathsf{x}) + \mathsf{y},$$

  where $\mathsf{aExt}\colon \{0,1\}^n \to \{0,1\}^m$ is a seedless extractor for degree d multi-variate polynomials that can extract from $\frac{g}{\mathsf{t}} - \xi$ fraction of entropy, for an integer $1 \leq g \leq \mathsf{t} - 1$ and a small real $\xi > 0$, with extractor error ε_A. The inverter of EXT is

  $$\mathsf{EXT}^{-1}(\mathsf{s}) = (\mathsf{U}_n || \mathsf{aExt}(\mathsf{U}_n) + \mathsf{s}).$$

- ECC is a linear MDS code with parameter $[\mathsf{n}, \mathsf{t}, \mathsf{n}-\mathsf{t}+1]$ over $\mathbb{F}_q$, where $\log q = \frac{m+n}{\mathsf{t}}$. The output of $\mathsf{ECCenc}\colon \{0,1\}^{m+n} \to \mathbb{F}_q^{\mathsf{n}}$ is the share vector.

The composition of ECCenc and the leakage function $f \in \mathcal{L}_{d\text{-}\mathsf{poly}}$ remains a multi-variate polynomials over $\mathbb{F}_2$ with degree at most d. Since our analysis will be focusing on the extractor aExt, we derive the entropy bound on the first n bits of $\mathsf{EXT}^{-1}(\mathsf{U}_m)$. According to Lemma 9, the average (conditional) min-entropy of the random variable U_n conditioned on the view $\mathsf{V} = \mathsf{LEAK}(\mathsf{EXT}^{-1}(\mathsf{U}_m), \mathcal{L}_{d\text{-}\mathsf{poly}}, \beta, \theta)$ of an adaptive adversary corresponding to a leakage strategy $(\mathcal{L}_{d\text{-}\mathsf{poly}}, \beta, \theta)$ is bounded as follows.

$$\tilde{\mathsf{H}}_\infty(\mathsf{U}_n|\mathsf{V}) \geq n - (\log q)\theta - \beta.$$

Under the condition that $\theta + \frac{\beta}{\log q} \leq \mathsf{t} - g$, we have

$$\begin{aligned}\tilde{\mathsf{H}}_\infty(\mathsf{U}_n|\mathsf{V}) &\geq n - (\log q)(\mathsf{t}-g)\\ &= n - \tfrac{(m+n)(\mathsf{t}-g)}{\mathsf{t}}\\ &= \tfrac{n}{\mathsf{t}} - \tfrac{m(\mathsf{t}-g)}{\mathsf{t}},\end{aligned}$$

which is bigger than a $\frac{1}{\mathsf{t}} - \xi$ fraction of n for large enough n.

It then follows from the average case extractor property of aExt that

$$(\mathsf{V}, \mathsf{aExt}(\mathsf{U}_n)) \overset{\varepsilon_A}{\sim} (\mathsf{V}, \mathsf{U}_m).$$

From an application of Lemma 12, we obtain for a secret s,

$$(\mathsf{V}|\mathsf{aExt}(\mathsf{U}_n) = \mathsf{s}) \overset{2^m\varepsilon_A}{\sim} \mathsf{V}.$$

We then conclude that the scheme is a $(\mathcal{L}_{d\text{-}\mathsf{poly}}, \beta, \theta, 2^{m+2}\varepsilon_A)$-LR-SS.

According to Lemma 23, for any positive integer d, there is an efficient average case extractor $\mathsf{aExt}\colon \{0,1\}^n \to \{0,1\}^m$ for algebraic sources with parameters $\left((1-\frac{1}{c_d})n, d, 2^{-\Omega(\frac{n}{c_d})}\right)$, where $c_d = \Theta(d^2 4^d)$ and $m = \Omega(\frac{n}{c_d})$. We then require the threshold gap g to satisfy $\frac{g}{\mathsf{t}} > 1 - \frac{1}{c_d}$. In particular, once $c_d > 2$, we must have $g > \mathsf{t} - \frac{\mathsf{t}}{c_d} > \frac{\mathsf{t}}{2} \geq 1$ and do not have a threshold scheme. On the other hand, if we are satisfied with obtaining a leakage resilient ramp scheme, then privacy and leakage resiliency are guaranteed for all θ and β satisfying $\theta + \frac{\beta}{\log q} \leq \mathsf{t} - g < \frac{t}{c_d}$. The scheme shares a secret of $m = \Omega(\frac{n}{c_d})$ bits and each share contains $\log q = \frac{m+n}{\mathsf{t}}$ bits. So the information ratio is positive (a constant determined by c_d and t). Moreover, the leakage-resilience error $\varepsilon = 2^{m+2}\varepsilon_A$ can be made arbitrarily small. ◀

6 Local Leakage with Full Reconstruction

Srinivasan and Vasudevan considered a partial adaptive local leakage model that they called *strong local leakage* model and constructed such schemes for applications in leakage resilient secure multiparty computation. This model is the strongest local leakage model. We show an instantiation of our generic construction that yields competitive schemes in this model to illustrate the applicability of our generic construction to local leakage models.

▶ **Theorem 24.** *Let $\mathcal{L}_{\mathsf{local}}$ be the class of local functions. There is a family of t-out-of-n secret sharing schemes over $\mathbb{F}_q$ (labeled by $\log q$) that is simultaneously $(\mathcal{L}_{\mathsf{local}}, \ell(\mathsf{n}-\theta), \theta, \varepsilon)$-leakage resilient against a partially adaptive adversary that fully corrupts θ players for $\theta \in \{0, 1, \ldots, \mathsf{t}-2\}$ and, based on the shares of the θ players, chooses $\mathsf{n}-\theta$ arbitrary leakage*

functions each leaks $\ell = \frac{(\mathsf{t}-\theta-8\xi)\log q}{(\mathsf{n}-\theta)(1+5\xi)}$ *bits for the remaining* $\mathsf{n}-\theta$ *shares, where* $\xi > 0$ *is a small real number, the share size* q *(determined by* ε *and* $q > \mathsf{n}$*) is a large enough power of* 2 *and the secret length* $m = \frac{\xi}{1+5\xi}\log q - o(\log q)$*. When* $\theta = \mathsf{t}-1$*, the partially adaptive adversary can only choose the* $\mathsf{n}-\mathsf{t}+1$ *arbitrary leakage functions based on* $\mathsf{t}-2$ *shares*

Proof. We instantiate the generic construction using a linear strong seeded extractor $\mathsf{Ext}\colon \{0,1\}^d \times \{0,1\}^n \to \{0,1\}^m$ and a linear MDS code. We also use a t-out-of-n Shamir's secret sharing to protect the seed such that the leakage at an individual share is independent of the seed.

- $\mathsf{EXT} : \{0,1\}^n \to \{0,1\}^m$ is instantiated as follows.

  $$\mathsf{EXT}(\mathsf{x}) = \mathsf{Ext}(\mathsf{U}_d, \mathsf{x}),$$

  where Ext is the linear strong seeded extractor from 7 with parameters entropy requirement $3m$, extractor error $2^{-m}\varepsilon$ and seed length $d = 5m$. The uniform seed sampled during inverting is shared using t-out-of-n Shamir's secret sharing into $(\mathsf{SdSh}_1, \ldots, \mathsf{SdSh}_\mathsf{n}) \in \mathbb{F}_{2^{5m}}^\mathsf{n}$.
- ECC is a linear MDS code with parameter $[\mathsf{n}, \mathsf{t}, \mathsf{n}-\mathsf{t}+1]$ over $\mathbb{F}_{q_0}$, where $\log q_0 = \frac{d+n}{\mathsf{t}}$. The output of $\mathsf{ECCenc}\colon \{0,1\}^{d+n} \to \mathbb{F}_{q_0}^\mathsf{n}$ is $(\mathsf{SorSh}_1, \ldots, \mathsf{SorSh}_\mathsf{n})$. The final share vector is $((\mathsf{SdSh}_1||\mathsf{SorSh}_1), \ldots, (\mathsf{SdSh}_\mathsf{n}||\mathsf{SorSh}_\mathsf{n})) \in \mathbb{F}_q^\mathsf{n}$, where $q = 2^{5m}q_0$.

The leakage is independent of the seed. The seed is referring to the $5m$-bit uniform and independent seed sampled when inverting the extractor EXT. The seed is shared using the t-out-of-n Shamir's secret sharing and the shares are appended to the payload shares. The partially adaptive adversary corrupts arbitrary choice of θ players and obtains θ full shares. Since $\theta \le \mathsf{t}-2$, the adversary not only obtains no information about the seed from the θ shares of the Shamir's scheme, but also can not obtain any information about the seed even if one more share is given. It then follows that the choice of individual leakage functions for the remaining $\mathsf{n}-\theta$ shares are independent of the seed. Note that the output of these individual leakage functions can depend on the seed, since Shamir's scheme over finite field of characteristic 2 is known to be not leakage resilient to (even non-adaptive) local leakage. We only need the fact that the choice (made after obtaining θ shares and before receiving information about the rest of the $\mathsf{n}-\theta$ shares) of these functions is independent of the seed for our security proof. This is because we use a strong seeded extractor to provide privacy and leakage resiliency for the payload scheme and by definition the security of a strong seeded extractor holds even if the seed is revealed, as long as the source is independent of the seed. Here the source is the uniform n-bit string conditioned on the information about it contained in the corrupted θ shares and the outputs of the individual leakage functions.

Achievable parameters.

$$\mathsf{U}_n = \mathsf{EXT}^{-1}(\mathsf{U}_m).$$

Let V be the adversary's view. We have

$$\tilde{\mathsf{H}}_\infty(\mathsf{U}_n|\mathsf{V}) \ge n - \theta\log q_0 - (\mathsf{n}-\theta)\ell.$$

The total entropy in the uniform n-bit string U_n is

$$n = \mathsf{t}\log q_0 - d = \mathsf{t}\log q_0 - 5m.$$

The amount of information about U_n contained in the θ shares is $\theta \log q_0$. The amount of information about U_n contained in the outputs of the leakage functions is

$$(\mathsf{n}-\theta)\ell = (\mathsf{n}-\theta)\frac{(\mathsf{t}-\theta-8\xi)\log q}{(\mathsf{n}-\theta)(1+5\xi)} = (\mathsf{t}-\theta-8\xi)\log q_0.$$

The remaining *average conditional min-entropy* is

$$\mathsf{t}\log q_0 - 5m - \theta \log q_0 - (\mathsf{t}-\theta-8\xi)\log q_0 = 8\xi \log q_0 - 5m,$$

where $m = \xi \log q_0 - o(\log q_0)$. This asserts that the remaining entropy is at least $3m$, sufficient for the average case extractor Ext. ◀

▶ Remark 25. When we set $\theta = \mathsf{t}-1$, our scheme is comparable with the scheme constructed in [43]. Both schemes allow the partial adaptive adversary to choose the individual leakage functions according to $\mathsf{t}-2$ shares and fully leak $\mathsf{t}-1$ shares. Our scheme achieves positive information ratio, which is an absolute constant, while the scheme in [43] achieves information ratio $\Omega(1/\mathsf{n})$. On the other hand, our scheme achieves an asymptotic leakage rate of $\frac{\mathsf{t}-\theta}{\mathsf{n}-\theta} = \frac{1}{\mathsf{n}-\mathsf{t}+1}$, while the scheme in [43] achieves asymptotic leakage rate 1. Note that our scheme belong to the sub-class of t-out-of-n secret sharing schemes that have *full reconstruction* property [38], since any t shares uniquely determine the rest of the shares. It is shown in [38], for such schemes, it is required that

$$\log q \geq \frac{\ell(\mathsf{n}-\mathsf{t})}{\mathsf{t}-\theta}, \text{ for any } \ell \geq 1.$$

The above inequality implies an upper bound on the asymptotic leakage rate.

$$\frac{\ell}{\log q} \leq \frac{\mathsf{t}-\theta}{\mathsf{n}-\mathsf{t}}.$$

When $\theta = \mathsf{t}-1$, we have

$$\frac{\ell}{\log q} \leq \frac{1}{\mathsf{n}-\mathsf{t}},$$

which holds even for non-adaptive adversary. This indicates that our scheme is already close to optimal with respect to leakage rate.

7 Conclusion

We started a systematic study of leakage-resilient secret sharing against *global* leakage, where the leakage oracle can access the full set of shares simultaneously, but the access is restricted to a special class of leakage functions. We studied such leakage-resilient secret sharing with respect to affine leakage functions and low-degree multi-variate polynomial leakage functions. We explicitly constructed threshold schemes with best leakage tolerance against affine functions. If the adversary is non-adaptive, our scheme is optimal both in terms of leakage tolerance and information ratio. For multi-variate polynomial leakage functions with degree bigger than one, our construction only yielded ramp schemes. As a result of independent interest, we showed that our approach to leakage-resilient secret sharing also yielded a competitive scheme compared with the state-of-the-art construction in the compartmentalized models.

References

1 Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, Joao Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret-sharing schemes for general access structures. *IACR Cryptology ePrint Archive*, page https://eprint.iacr.org/2018/1147, 2018.

2 Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In *ACM SIGACT Symposium on Theory of Computing, STOC 2015*, pages 459–468, 2015.

3 Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. *SIAM J. Comput.*, 47(2):524–546, 2018.

4 Divesh Aggarwal, Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Leakage-resilient non-malleable codes. In *Theory of Cryptography Conference, TCC 2015*, pages 398–426, 2015.

5 Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In *Advances in Cryptology - CRYPTO 2015*, pages 538–557, 2015.

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 *Theory of Cryptography Conference, TCC 2015*, pages 375–397, 2015.

7 Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. *IACR Cryptology ePrint Archive*, page https://eprint.iacr.org/2018/1144, 2018.

8 Marshall Ball, Dana Dachman-Soled, Siyao Guo, Tal Malkin, and Li-Yang Tan. Non-malleable codes for small-depth circuits. In *IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 826–837, 2018.

9 Marshall Ball, Siyao Guo, and Daniel Wichs. Non-malleable codes for decision trees. Cryptology ePrint Archive, Report 2019/379, 2019.

10 Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In *Advances in Cryptology - CRYPTO 2018*, pages 531–561, 2018.

11 George R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the 1979 AFIPS National Computer Conference*, pages 313–317, 1979.

12 Andrej Bogdanov, Yuval Ishai, Emanuele Viola, and Christopher Williamson. Bounded indistinguishability and the complexity of recovering secrets. *Advances in Cryptology -CRYPTO*, pages 593–618, 2016.

13 Jean Bourgain. On the construction of affine extractors. *Geometric and Functional Analysis*, 17(1):33–57, 2007.

14 Nishanth Chandran, Bhavana Kanukurthi, and Srinivasan Raghuraman. Information-theoretic local non-malleable codes and their applications. In *Theory of Cryptography - TCC*, pages 367–392, 2016.

15 Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In *ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 285–298, 2016.

16 Eshan Chattopadhyay and Xin Li. Non-malleable codes and extractors for small-depth circuits, and affine functions. In *ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 1171–1184, 2017.

17 Eshan Chattopadhyay and Xin Li. Non-malleable extractors and codes in the interleaved split-state model and more. *CoRR*, page http://arxiv.org/abs/1804.05228, 2018.

18 Mahdi Cheraghchi, Frédéric Didier, and Amin Shokrollahi. Invertible extractors and wiretap protocols. *IEEE Trans. Information Theory*, 58(2):1254–1274, 2012. doi:10.1109/TIT.2011.2170660.

19 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`.

20 Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In *Security and Cryptography for Networks SCN*, pages 121–137, 2010.

21 Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008.

22 Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In *Advances in Cryptology - CRYPTO 2013*, pages 239–257, 2013.

23 Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *Foundations of Computer Science FOCS 2007*, pages 227–237, 2007.

24 Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. *J. ACM*, 65(4):20:1–20:32, 2018.

25 Antonio Faonio and Daniele Venturi. Non-malleable secret sharing in the computational setting: Adaptive tampering, noisy-leakage resilience, and improved rate. *IACR Cryptology ePrint Archive*, page https://eprint.iacr.org/2019/105, 2019.

26 Christina Fragouli and Emina Soljanin. *Network Coding Fundamentals*, volume 2. Foundations and Trends in Networking, 2007.

27 Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In *ACM SIGACT Symposium on Theory of Computing, STOC 2018*, pages 685–698, 2018.

28 Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing for general access structures. In *Advances in Cryptology - CRYPTO 2018*, pages 501–530, 2018.

29 Venkatesan Guruswami and Mary Wootters. Repairing reed-solomon codes. *IEEE Trans. Information Theory*, 63(9):5684–5698, 2017.

30 Masahito Hayashi and Toyohiro Tsurumaru. More efficient privacy amplification with less random seeds via dual universal hash function. *IEEE Transactions on Information Theory, Vol 62, Issue 4, 2213 - 2232.*, 2016.

31 Ashutosh Kumar, Raghu Meka, and Amit Sahai. Leakage-resilient secret sharing against colluding parties. *Foundations of Computer Science, FOCS 2019*, 2019.

32 Fu Li and David Zuckerman. Improved extractors for recognizable and algebraic sources. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 72:1–72:22, 2019.

33 Xin Li. A new approach to affine extractors and dispersers. *IEEE Conference on Computational Complexity, CCC 2011*, pages 137–147, 2011.

34 Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In *ACM SIGACT Symposium on Theory of Computing, STOC 2017*, pages 1144–1156, 2017.

35 Xin Li. Pseudorandom correlation breakers, independence preserving mergers and their applications. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:28, 2018.

36 Fuchun Lin, Mahdi Cheraghchi, Venkatesan Guruswami, Reihaneh Safavi-Naini, and Huaxiong Wang. Secret sharing with binary shares. In *Innovations in Theoretical Computer Science Conference, ITCS 2019*, pages 53:1–53:20, 2019.

37 Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In *Advances in Cryptology - CRYPTO*, pages 517–532, 2012.

38 Jesper Buus Nielsen and Mark Simkin. Lower bounds for leakage-resilient secret sharing. *IACR Cryptology ePrint Archive*, page https://eprint.iacr.org/2019/181, 2019.

39 Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 1(52):43–52, 1996.

40 Ran Raz, Omer Reingold, and Salil P. Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. *J. Comput. Syst. Sci.*, 65(1):97–128, 2002.

41 Ronen Shaltiel. How to get more mileage from randomness extractors. In *IEEE Conference on Computational Complexity (CCC) 2006*, pages 46–60, 2006.

42 Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

43 Akshayaram Srinivasan and Prashant Nalini Vasudevan. Leakage resilient secret sharing and applications. *Advances in Cryptology - CRYPTO 2019*, pages 480–509, 2019.

44 Douglas R. Stinson. An explication of secret sharing schemes. *Des. Codes Cryptography*, 2(4):357–390, 1992.

45 Luca Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, 2001.

Lower Bounds for Function Inversion with Quantum Advice

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

Tai-Ning Liao
National Taiwan University, Taipei, Taiwan
b04901168@ntu.edu.tw

Luowen Qian
Boston University, MA, USA
luowenq@bu.edu

Abstract

Function inversion is the problem that given a random function $f : [M] \to [N]$, we want to find pre-image of any image $f^{-1}(y)$ in time T. In this work, we revisit this problem under the preprocessing model where we can compute some auxiliary information or advice of size S that only depends on f but not on y. It is a well-studied problem in the classical settings, however, it is not clear how quantum algorithms can solve this task any better besides invoking Grover's algorithm [6], which does not leverage the power of preprocessing.

Nayebi et al. [9] proved a lower bound $ST^2 \geq \tilde{\Omega}(N)$ for quantum algorithms inverting *permutations*, however, they only consider algorithms with *classical* advice. Hhan et al. [8] subsequently extended this lower bound to fully quantum algorithms for inverting permutations. In this work, we give the same asymptotic lower bound to fully quantum algorithms for inverting functions for fully quantum algorithms under the regime where $M = O(N)$.

In order to prove these bounds, we generalize the notion of quantum random access code, originally introduced by Ambainis et al. [3], to the setting where we are given a list of (not necessarily independent) random variables, and we wish to compress them into a variable-length encoding such that we can retrieve a random element just using the encoding with high probability. As our main technical contribution, we give a nearly tight lower bound (for a wide parameter range) for this generalized notion of quantum random access codes, which may be of independent interest.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives; Theory of computation → Oracles and decision trees; Theory of computation → Quantum query complexity

Keywords and phrases Cryptanalysis, Data Structures, Quantum Query Complexity

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.8

Related Version A full version of the paper is available at `https://arxiv.org/abs/1911.09176`.

Funding *Kai-Min Chung*: Kai-Min Chung is partially supported by the 2019 Academia Sinica Career Development Award under Grant no. 23-17, and MOST QC project under Grant no. MOST 108-2627-E-002-001-.
Luowen Qian: Luowen Qian is supported by the DARPA SIEVE program.

Acknowledgements The authors would like to thank Nai-Hui Chia, Luca Trevisan, Xiaodi Wu, and Penghui Yao for their helpful insights during the discussions. We also thank the anonymous reviewers at QIP and ITC for pointing out various issues in the paper.

1 Introduction

Space-time trade-offs are a widely observed phenomenon in data structure complexity. In this work, we are interested in trade-offs between offline preprocessing advice length and online running time in inverting random functions, namely, the trade-off between the size S (in the number of bits) of pre-computed data structure (or advice) on the function (but not the image that we wish to invert) and the algorithm's running time T for computing the inverse of a certain image. Such trade-offs give lower bound for algorithms that inverts cryptographic functions without taking the specific structure of that family of functions.

Without pre-computed advice ($S = 0$), classical computers require $T = \Omega(\varepsilon N)$ for inverting a random image for a random function $f : [N] \mapsto [N]$ with probability ε, and quantum computers require $T = \Omega(\sqrt{\varepsilon N})$ [2] to do so. Both bounds are asymptotically tight, since we observe that exhaustive search and Grover's algorithm [6] on input range $[\varepsilon N]$ inverts an ε fraction of inputs, respectively. However, if we allow some pre-computed advice, classical computers can do much better. Hellman [7] showed that every function can be inverted with $S = T = \tilde{O}(N^{2/3})$ and every permutation can be inverted using only $S = T = \tilde{O}(N^{1/2})$. However, it is not known whether we can do better than Grover's algorithm or Hellman's algorithm, even if we allow quantum computers to come into play. Therefore motivated by post-quantum cryptanalysis, it is natural to ask whether these two algorithms are indeed the best that we can do. For classical computers, De et al. [5] (going back to ideas of Yao [14]) showed that $ST = \tilde{\Omega}(\varepsilon N)$ is required for both functions and permutations, and Corrigan-Gibbs and Kogan [4] gave some evidence that improving this lower bound seems to be difficult, by connecting function inversion problem to several other hard problems in complexity theory, communication complexity, etc. For quantum computers, Nayebi et al. [9] showed that $ST^2 = \tilde{\Omega}(\varepsilon N)$ is required, however, this result only applies to the case where the computation and the oracle queries are quantum but the pre-computed advice remains classical. However, they also noted that the advice given to a quantum computer can as well be quantum, and it remains open to prove a lower bound for computations in that model.

1.1 Our Contributions

In this work, we resolve this discrepancy by showing that $ST^2 = \tilde{\Omega}(\varepsilon N)$ is still required even if the inverter is allowed to use quantum advice. Formally,

▶ **Definition 1.** *A function (or permutation) inverter is a pair $(\alpha, \mathcal{A})$, where:*

1. *$\alpha = \alpha(f)$ is a pre-computed quantum advice of S qubits, which can depend on the function $f : [M] \mapsto [N]$; (for permutations, $M = N$)*
2. *$\mathcal{A}$ is a quantum oracle algorithm that takes advice α and an image $y \in [N]$, makes at most T quantum queries to the function as an oracle O_f, and outputs a supposed pre-image $x \in [M]$.*

▶ **Definition 2.** *Fix a function inverter $(\alpha, \mathcal{A})$.*

- *We say that "$(\alpha, \mathcal{A})$ inverts y for f" if*

$$\Pr[f(\mathcal{A}^f(\alpha, y)) = y] \geq 2/3,$$

where the probability is taken over the measurement results (internal randomness) of $\mathcal{A}$.

- *For any real ε, we say that "$(\alpha, \mathcal{A})$ inverts ε fractions of inputs" if*

$$\Pr_{y,f}[(\alpha, \mathcal{A}) \text{ inverts } y \text{ for } f] \geq \varepsilon,$$

where y and f is sampled uniformly from $[N]$ and S_N, respectively.

▶ **Theorem 3.** *(Lower bound for permutations) For any permutation inverter that invert ε fractions of inputs, assuming:*

1. $\varepsilon = \omega(1/N)$, (1)

that is, the inverter can succeed on more than a constant number of points;

2. $T = o(\varepsilon\sqrt{N})$, (2)

noting that $T = O(\sqrt{\varepsilon N})$ is the complexity of Grover's search algorithm;

3. $S \geq 1$. (3)

We have

$$ST^2 \geq \tilde{\Omega}(\varepsilon N)$$

for all sufficiently large N.

▶ **Theorem 4.** *(Lower bound for functions) For any function inverter that invert ε fractions of inputs, assuming:*

1. $M = O(N)$, (4)

2. $T = o(\varepsilon\sqrt{M}/\log^{10} N)$, (5)

noting that $T = O(\sqrt{\varepsilon M})$ is the complexity of Grover's search algorithm;

3. $\varepsilon \geq 1/N$, (6)

that is, the inverter performs no worse than a fixed point output inverter;

4. $S \geq 1$. (7)

We have

$$ST^2 \geq \tilde{\Omega}(\varepsilon M)$$

for all sufficiently large M.

Towards proving these two theorems, we also develop a lower bound for a natural generalization of quantum random access code (QRAC). We believe the notion of quantum random access code is a natural object to study in quantum information theory, and that our generalization has potential to find other applications in quantum information. In Section 4, we will explain the concept more thoroughly and prove the lower bound.

1.2 Related Work

Independently in [8], they considered a number of cryptographic applications of random functions under both classical advice (quantum query) model and quantum advice model, which they denote as AI-QROM and QAI-QROM respectively. Under quantum advice model, their Theorem 6 showed bounds for inverting random permutations using different techniques, namely, gentle measurements and semi-classical oracle.

However, in their work, they left the problem of proving bounds for random functions open and we partially give some answers to that open problem in this work. They noted that generalizing this to function inversion seems problematic – to use gentle measurement lemma, we need to boost the per-element success probability to $1 - O(1/N^4)$; however, in the function case, even with our idea of using 2-universal hash functions (which we outline in the technical overview section), we cannot hope to boost the per-element success probability

beyond $1 - o(1/N)$ as it would already make storing all the hash tags too expensive for an efficient encoding. In conclusion, it seems hopeless to combine gentle measurement technique with our 2-universal hash for adversaries with constant success probability on ε fractions of input. Our QRAC technique, on the other hand, works and gives non-trivial bound even if the per-element success probability is as low as $1 - O(1/\log N)$ under the exact same setting. This shows that our QRAC technique seems to be able to achieve some improvements compared to their approach. We also note that our proof technique does not involve internal measurements in the compress/decompress algorithm and is conceptually simpler.

2 Technical Overview

2.1 Permutations

We first show how to solve the permutation inversion problem, which is an easier argument.

Compression argument

In De et al. [5], the main idea in proving the lower bound is to leverage the inverter to produce an algorithm that compresses the permutation into a short string, and the information theoretic lower bound on the size of the string translates to our desired lower bound. However, as the inverter needs to make T adaptive queries, we need to produce the correct answer for the inverter so that she can successfully invert the image and we can extract the information from the inverter. The way to do this is to randomly remove a small enough subset of the image from the permutation. As we are picking a small independently random subset, the probability that the inverter hits this subset will be small. Therefore, we can use the advice and the permutation without the removed fraction as the encoding for the permutation, and since the length of the encoding is lower bounded by the entropy of all the permutations the encoding scheme is able to compress, this translates to a lower bound in the space-time trade-off for the permutation inversion problem. In the process, we "cheated" by using some shared randomness, but it turns out we can fix this since having shared randomness does not affect the information theoretic lower bound that we need in the end.

As shown by Nayebi et al. [9], this idea also holds similarly for algorithms that can make quantum queries to the permutation. Namely, if we change δ fraction of the input, by a similar argument to proving the optimality of Grover's algorithm [2], a quantum query algorithm is required to take $\Omega(\sqrt{1/\delta})$ queries to distinguish the change with constant probability. However, they also have shown that this approach has a fundamental limitation when one tries to adapt it to the case where the pre-processed information can be quantum. Recall that in order to invoke the inverter to recover a deleted entry, we need to invoke it with the pre-computed advice. If the advice is classical, we can simply repeat this process for every entry to recover the entire permutation table; but if the advice is quantum, we cannot hope to do this repeatedly as the previous copy would be destroyed by measurement, and we cannot hope to clone multiple copies of the advice for free due to no cloning theorem [13]. The only thing we can do is to produce multiple copies of the same advice in the encoding phase, however, if we work out the calculation, we can see that this encoding scheme is too inefficient for proving a meaningful lower bound for inverting permutations.

Avoiding repeated measurements

Approaching this challenge, our idea is to reduce the problem to a similar problem that does not require recovering the entire permutation table. Ambainis et al. [3] introduced the notion of Quantum Random Access Code with Shared Randomness, which is a two-player game where two players share some randomness R; the first player $\mathcal{A}$ gets a bit string X chosen uniformly at random and is asked to encode it into an encoding $Y \leftarrow \mathcal{A}(X, R)$; and the second player is asked to recover X_i given Y, R and some index $i \in [|X|]$ chosen uniformly at random. Assuming the two player succeeds with probability δ, the number of bits in Y is lower bounded by (with some very rough approximations when $\delta \to 1$) $|Y| \geq \delta|X|$. It can be shown that this lower bound is tight even when everything is classical, simply by observing that an algorithm that simply remembers a δ fraction of the input wins the game with probability δ. This game has found several applications in quantum information theory and quantum cryptography, for example [1].

Thus, a natural idea is to come up with a similar lower bound for quantum random access code with shared randomness for permutations and do the reduction. However, unlike in the case of bit strings, as there is correlations between each element of the permutation, our lower bound argument would need to proceed very carefully. Indeed, in this work we proved a lower bound on the expected number of qubits which is only related to the overall entropy, the average element entropy, and the recovery success probability. Furthermore, this holds even if there exists correlations between the elements. In general, this lower bound is weaker than the compression argument where the entire permutation is recovered. However, we note that if the success probability is high, say $\delta \geq 1 - O(1/N)$ for permutations, then the expected number of qubits needs to be at least $\log N! - O(\log N)$, which asymptotically matches the lower bound for compression argument in the classical case.

A direct encoding scheme would be using the encoding scheme of Nayebi et al. [9] and decode only the element in question. However, this direct idea does not work, since we are randomly removing entries from the permutation, the scheme only succeeds when the removed entries (determined by shared randomness R) does not affect the output of the inverter, which only happens with a small probability. This means that δ will be bounded away from 1. Recall that our encoding will need to remember $1 - o(1)$ fraction of the permutation, this gives us no meaningful bound. In fact, in order for this idea to succeed, we need to boost the success probability to also $1 - o(1)$.

We observe that in our proof for quantum random access code, the length of our encoding is ultimately bounded by the von Neumann entropy of the encoding. By using the variable length version of quantum source coding theorem, we can also use a variable length encoding that is still bounded by the von Neumann entropy of the encoding. Specifically, if the randomness will cause the encoding to err, we will simply use the entire permutation table as our encoding, which the decoder can decode any element directly. By repeating the advice poly-logarithmically many times, we can make the success probability sufficiently close to 1 for proving a meaningful bound.

2.2 Functions

To bootstrap the previous argument into an argument for function inverters, we can view the inverse function f^{-1} as a partition of $[M]$, and our goal is to design a random access code for querying this partition. In order to accommodate all possible adversaries, we only pick the pre-images that have high probability to be returned by the adversary. However, consider the following bad case, $f^{-1}(y) = \{x_1, x_2\}$, and the adversary uniformly returns

x_1, x_2 or a third bad output x'. In this case, majority vote will not work since (without loss of generality) assuming we removed x_1 from the encoding, the decoder cannot distinguish adversary returns x_1 or x' (assuming the adversary gets lucky so that x' is also removed from the encoding). To fix this, we use a 2-universal hash function (sampled from shared randomness) and use the hash tag to distinguish the correct output.

However, we need to choose the hash length very carefully, as choosing a length too short results in high error probability, and length too long results in inefficient coding (our goal is to achieve nontrivial savings for the random function). In particular, due to our QRAC bound, we must choose our length tag to be much shorter than $\log N$ to get a nontrivial bound for function inversion. It turns out that using a length of $\log \log N$ works in our case.

3 Preliminaries

We denote $[N]$ to be $\{k \in \mathbb{Z} : 1 \leq k \leq N\}$, and the set of all possible bijections from $[N]$ to itself to be S_N.

▶ **Definition 5.** *(Quantum oracle) For any classical function $f : X \mapsto Y$ where Y is some additive group, it naturally corresponds to a quantum oracle O_f such that for all $x \in X, y \in Y$,*

$$O_f(|x\rangle|y\rangle) = |x\rangle|y + f(x)\rangle.$$

Let $\mathcal{A}^O$ be a quantum oracle algorithm taking O as an oracle. In the rest of the paper, we will abuse the notation $\mathcal{A}^f$ to represent $\mathcal{A}^{O_f}$. For random oracles, it is equivalent to viewing oracle calls as the same as querying from an exponential sized truth table of the oracle.

▶ **Definition 6.** *The query magnitude at j of $|\phi\rangle = \sum_c \alpha_c |c\rangle$ is defined to be $q_j(|\phi\rangle) = \sum_{c \in C_j} |\alpha_c|^2$, where C_j is the set of all computational basis states that query position j.*

▶ **Definition 7.** *Given a quantum algorithm $\mathcal{A}$, the total query magnitude at j of $\mathcal{A}$ with (oracle access to) input x is defined to be $q_j(x) = \sum_{|\phi\rangle} q_j(|\phi\rangle)$, where the sum is taken over all the quantum queries produced by the algorithm.*

▶ **Lemma 8.** *(Swapping lemma) [12, Lemma 3.1] Let $|\phi_x\rangle$ and $|\phi_y\rangle$ be the final state of $\mathcal{A}$ on inputs x and y respectively. Let T be (the upper bound of) the number of queries $\mathcal{A}$ has made. Then:*

$$\||\phi_x\rangle - |\phi_y\rangle\| \leq \sqrt{T \sum_{j : x_j \neq y_j} q_j(x)},$$

where $\||\phi_x\rangle - |\phi_y\rangle\|$ denote the Euclidean distance between the two vectors.

▶ **Theorem 9.** *(Quantum Source Coding Theorem) [10] Let Σ be an alphabet, $\rho \in D(\mathbb{C}^\Sigma)$ be a density operator whose von Neumann entropy is $S(\rho)$.*

1. *If $L > S(\rho)$, then N independent samples of ρ can be losslessly compressed into LN qubits for all sufficiently large N;*
2. *If $L < S(\rho)$, then N independent samples of ρ can be losslessly compressed into LN qubits for at most finitely many N's.*

▶ **Theorem 10.** *(2-Universal Hashing) For every ε, there exists a 2-universal hash function family with error probability ε and output length $-\log \varepsilon$ (using some finite amount of randomness). [11, Chapter 3]*

4 Quantum Random Access Codes with Variable Length

Intuitively, quantum random access code looks at the following problem:

- A random function $f : [N] \mapsto X_N$ is sampled from an *arbitrary* distribution.
- At the offline phase, an unbounded algorithm gets access to the entire function and produces a quantum state $|\alpha\rangle$ of bounded size ℓ (therefore dimension at most 2^ℓ).
- At the online phase, a uniformly random challenge $x \in [N]$ is generated, and the algorithm given $|\alpha\rangle$ and x is asked to recover $f(x)$ with probability δ.

In this section, we want to prove that there is a trade-off between the expected encoding size $L := \mathop{\mathbb{E}}_f[\ell]$ and the success probability δ. This is a generalization of QRAC considered in previous works like [3] since we can view their QRAC equivalent to ours by making the following restrictions:

1. $X_N = \{0, 1\}$.
2. The function distribution is always the uniform distribution.
3. The quantum state length ℓ is fixed parameter that does not depend on the specific function f.

We formalize the problem above as quantum random access code with variable length, as given by the definition below.

▶ **Definition 11.** *Let F_N be a set of functions $f : [N] \to X_N$ for some finite set X_N. A quantum random access code with variable length (QRAC-VL) for F_N consists of two algorithms* $(\mathrm{Enc}, \mathrm{Dec})$.

1. $\mathrm{Enc} : F_N \times \mathcal{R} \to \mathbb{C}^*$. *The encoding algorithm encodes a function $f \in F_N$ with some fresh independent randomness in $\mathcal{R}$ to some qubits. The number of qubits denoted by $\ell = \ell(f)$ can depend on the function f.*
2. $\mathrm{Dec} : \mathbb{C}^* \times [N] \times \mathcal{R} \to X_N$. *The decoding algorithm compute $f(x)$ on some specific element $x \in [N]$ with the encoded message in $\mathbb{C}^{2^\ell}$, and it uses the same shared randomness for the encoding algorithm.*

The performance of the code is measured by two parameters L and δ. We define

$$L := \mathop{\mathbb{E}}_f[\ell(f)]$$

to be the average length of the coding scheme over uniform distribution on $f \in F_N$, and

$$\delta := \Pr_{f,x,R}[\mathrm{Dec}(\mathrm{Enc}(f; R), x; R) = f(x)]$$

to be the probability that our scheme correctly reconstructs the image of the function, where the probability is taken over uniform distribution on $f \in F_N$, $x \in [N]$, and the scheme's internal randomness.

First, we prove a helpful lemma that says conditional quantum entropy satisfies subadditivity.

▶ **Lemma 12.** *Let $X = (X_1, ..., X_N), Q$ be some quantum states, then*

$$\sum_{i=1}^{N} S(X_i|Q) \geq S(X|Q).$$

Proof. We will prove this for $N = 2$ and it is easy to extend this proof to any N using an inductive argument by showing that

$$\sum_{i=1}^{N-1} S(X_i|Q) + S(X_N|Q) \geq S(X_1...X_{N-1}|Q) + S(X_N|Q) \geq S(X|Q).$$

For $N = 2$, by the definition of conditional entropy, it is equivalent to prove $S(X_1Q) + S(X_2Q) \geq S(X_1X_2Q) + S(Q)$, which holds due to strong subadditivity of von Neumann entropy. ◀

▶ **Theorem 13.** *(Lower bound for QRAC-VL) For any QRAC-VL, let $X = (X_1, \ldots, X_N)$ be a random variable sampled uniformly random from the distribution (of truth tables) of functions F_N. Therefore, $S(X)$ is the (von Neumann) entropy of a uniformly random distribution of F_N and $S(X_J)$ is the average (or expected) entropy of a single element. We have that for all sufficiently large N,*

$$L \geq S(X) - N \cdot (H(\delta) + (1 - \delta) \cdot S(X_J)),$$

where $H(x) := -x \log_2 x - (1 - x) \log_2(1 - x)$ is the binary entropy function.

Proof. Sample R independently. Let $Q = \mathrm{Enc}(X; R)$ be the encoding. Using the fact in conditional mutual information that $I(Q, R; X) = I(Q; X|R) + I(X; R)$ and the fact that X and R are independent classical random variables,

$$I(Q, R; X) = I(Q; X|R) \leq S(Q|R). \tag{8}$$

Since R is classical, by Theorem 9,

$$S(Q|R) \leq S(Q) \leq L. \tag{9}$$

On the other hand, using Lemma 12,

$$\begin{aligned} I(Q, R; X) &= S(X) - S(X|Q, R) \\ &\geq S(X) - \sum_{i=1}^{N} S(X_i|Q, R) \\ &= S(X) - N \cdot S(X_J|Q, R, J), \end{aligned} \tag{10}$$

By data processing inequality, we know that

$$S(X_J|Q, R, J) \leq S(X_J|\mathrm{Dec}(Q, J; R)). \tag{11}$$

Note that $X_J, \mathrm{Dec}(Q, J; R)$ are both classical random variables. Let I be the indicator variable that indicates whether $X_J = \mathrm{Dec}(Q, J; R)$. By definition of success probability in quantum random access code, we can show that

$$\begin{aligned} S(X_J|\mathrm{Dec}(Q, J; R)) &= S(X_J, I|\mathrm{Dec}(Q, J; R)) - S(I|X_J, \mathrm{Dec}(Q, J; R)) \\ &= S(I|\mathrm{Dec}(Q, J; R)) + S(X_J|I, \mathrm{Dec}(Q, J; R)) - 0 \\ &\leq S(I) + \delta \cdot 0 + (1 - \delta) \cdot S(X_J) \\ &= H(\delta) + (1 - \delta)S(X_J). \end{aligned} \tag{12}$$

Combining (8), (9), (10), (11), and (12), we get the expected equation in the theorem. ◀

To see an immediate application of this theorem, we will demonstrate proving a bound for QRAC-VL for permutations. For permutations, $S(X) = \log N!$ and $S(X_J) = \log N$. Combining the theorem above with the following algebraic fact, we can prove a lower bound for QRAC-VL for permutations.

▶ **Fact 1.** $H(1-\delta) = H(\delta) \leq \delta \cdot \log(e/\delta)$.

▶ **Corollary 14.** *For any QRAC-VL for permutations S_N with $\delta = 1 - k/N$ for any $k = \Omega(1/N)$, we have*

$$L \geq \log N! - O(k \log N).$$

5 Proof of Theorem 3

Now we proceed to construct an encoding scheme given an inverter. Given a permutation inverter $(\alpha, \mathcal{A})$ that inverts an ε fraction of the input. Let $\varepsilon' = \varepsilon/2$. By how we defined success probability, we can show that there exists a large subset X of all the permutations S_N with size at least $\varepsilon' N!$, such that for any permutation $\pi \in X$, we have that

$$\Pr_y[(\alpha, \mathcal{A}) \text{ inverts } y \text{ for } \pi] \geq \varepsilon'.$$

Consider a permutation $\pi \in X$, and let I be the set of indices $x \in [N]$ such that $\mathcal{A}$ inverts $f(x)$. Recall that by the definition of X, we have $|I| \geq \varepsilon' N$. We use the shared randomness in the way such that we sample a subset $R \subseteq [N]$ with each element of $[N]$ independently chosen to be in R with probability γ/T^2, where $\gamma \in (0, 1)$ is some constant that we will decide later.

Let G be a subset of I, where an element $x \in G$ if it satisfies the following two conditions,

1. $x \in R$; (13)
2. The total query magnitude on $R \setminus \{x\}$ while running $\mathcal{A}^\pi(\alpha, \pi(x))$ is bounded by c/T for some constant c, that is,
$$\sum_{z \in R \setminus \{x\}} q_z(x) \leq \frac{c}{T}. \tag{14}$$

▷ Claim 15. With probability at least 0.8 over the choice of R, $|G| = \Omega(\varepsilon N/T^2)$.

Proof. Let $H = R \cap I$. Due to the definition of R, $|H|$ is distributed according to a binomial distribution. Therefore, the expected value of $|H|$ is $|I|\gamma/T^2$. By the multiplicative Chernoff bound and (2),

$$\Pr_R\left[|H| \geq \frac{|I|\gamma}{2T^2}\right] \geq 0.9 \tag{15}$$

for all sufficiently large N.

By definition, each query that $\mathcal{A}$ makes is of unit length. Since $\mathcal{A}$ makes at most T queries, by Definition 7,

$$\sum_{z \in [N]} q_z(x) \leq T.$$

By linearity of expectation,

$$\mathop{\mathbb{E}}_R\left[\sum_{z \in R \setminus \{x\}} q_z(x)\right] = \sum_{z \in [N] \setminus \{x\}} \frac{\gamma}{T^2} q_z(x) \leq \frac{\gamma}{T^2} T = \frac{\gamma}{T}.$$

Hence, by Markov's inequality,

$$\Pr_R\left[\sum_{z\in R\setminus\{x\}} q_z(x) \geq \frac{c}{T}\right] \leq \frac{T}{c}\cdot\frac{\gamma}{T} = \frac{\gamma}{c}. \tag{16}$$

Let J denote the subset of $x \in I$ that satisfy (13) but not (14). Note that (13) and (14) are independent for each $x \in I$, since (13) is whether $x \in R$ and (14) only concerns the intersection of R and $[N] \setminus \{x\}$. Therefore by (16), the probability that $x \in I$ satisfies $x \in J$ is at most $\gamma^2/(cT^2)$. Hence, by Markov's inequality,

$$\Pr_R\left[|J| \leq \frac{10|I|\gamma^2}{cT^2}\right] \geq 0.9. \tag{17}$$

From (15) and (17), we get that with probability at least 0.8 over the choice of R,

$$|G| = |H| - |J| \geq \frac{|I|\gamma}{2T^2} - \frac{10|I|\gamma^2}{cT^2} \geq \frac{\varepsilon'\gamma N}{2T^2}\left(1 - \frac{5\gamma^2}{c}\right) = \Omega\left(\frac{\varepsilon N}{T^2}\right),$$

given that γ is a small enough positive constant. ◀

We now proceed to describe the QRAC-VL scheme for encoding π^{-1}. If $\pi \notin X$ or $|G|$ is smaller than $O(\varepsilon N/T^2)$, the encoding simply sets a (classical) flag (which takes one bit) and stores the entire permutation table of π^{-1} (we will denote this as case A). In this case, it is straightforward to construct a decoder that succeeds with probability 1.

Otherwise assuming G is large enough, we clear the first flag, and proceed with our QRAC-VL that computes (if necessary) and outputs the following information β as our encoding: (which we will denote as case B)

- The size of G, encoded using $\log N$ bits;
- The set $G \subseteq R$, encoded using $\log\binom{|R|}{|G|}$ bits;
- The permutation π restricted to input outside of G, encoded using $\log(N!/|G|!)$ bits;
- Quantum advice used by the algorithm repeated ρ times $\alpha^{\bigotimes\rho}$, for some ρ that we will decide later. (We can compute this as the encoder can preprocess multiple copies of the same advice. Note that this is the only part of our encoding that is not classical.)

Upon given the encoding β, some image $y \in [N]$, and the algorithm's randomness R, the decoder first proceeds to recover set G and $\pi(x)$ for every $x \notin G$. If the given $y = \pi(x)$ for some $x \notin G$, the decoder outputs $x = \pi^{-1}(y)$. Otherwise, the decoder constructs π' to be

$$\pi'(x) = \begin{cases} y, & x \in G; \\ \pi(x), & x \notin G. \end{cases}$$

Then the decoder extracts $\alpha_1, \alpha_2, ..., \alpha_\rho$, and invokes $\mathcal{A}^{\pi'}(\alpha_i, y)$ for each $i \in [\rho]$ and outputs their majority vote. Let $|\phi_\pi\rangle$ and $|\phi_{\pi'}\rangle$ denote the final states of $\mathcal{A}$ when it is given the oracle π and π' respectively. Then by Lemma 8 and the definition of G,

$$\||\phi_\pi\rangle - |\phi_{\pi'}\rangle\| \leq \sqrt{T\sum_{z\in R\setminus\{x\}} q_z(x)} \leq \sqrt{T\cdot\frac{c}{T}} = \sqrt{c}.$$

As $x \in I$, by the definition of I, measuring $|\phi_\pi\rangle$ gives x with probability at least $2/3$. Given c is a small enough positive constant, measuring $|\phi_{\pi'}\rangle$ will also give x with probability at least 0.6.

We now examine the length of our encoding. With probability $1-\varepsilon'$, we have $\pi \notin X$; with probability $\varepsilon' \cdot (1-0.8)$, we have $\pi \in X$ but G is small. Therefore, over all, with probability $1-0.6\varepsilon$, our encoding will take case A, where the encoding consists of $1+\log N!$ classical bits and decoder succeeds with probability 1.

With probability 0.4ε, our encoding takes case B, and the size of the encoding will be

$$1+\log N+\log\binom{|R|}{|G|}+\log(N!/|G|!)+\rho S.$$

By (2), $\log\binom{|R|}{|G|}=O(|G|\log(|R|/|G|))=O(|G|\log 1/\varepsilon)=o(|G|\log|G|)$, and we can rewrite the size of the encoding as

$$\rho S-\log|G|!+\log N!+o(\log|G|!).$$

In this case, when the decoder is queried a point inside what she has remembered, that is $y \notin \pi(G)$ (which occurs with probability $1-|G|/N$), she recovers the correct pre-image with probability 1; otherwise, with one copy of the advice, she recovers the correct pre-image with probability 0.6, therefore with ρ copies, by Chernoff's bound, she recovers the correct pre-image using majority vote, with probability $1-\exp(-\Omega(\rho))$.

Overall, the average encoding length is at most $1/2\cdot(\varepsilon\rho S+|G|H(\varepsilon)-\varepsilon\log|G|!+\varepsilon\log N)+\log N!$, and the average success probability is $1-|G|/N\cdot\exp(-\Omega(\rho))$. By setting $\rho=\Omega(\log(N/\varepsilon))=\Omega(\log N)$, the average success probability[1] will be $1-O(\varepsilon/N)$.By (1) and Corollary 14, we have

$$\log N!+1/2\cdot(\varepsilon\log|G|!-\varepsilon\rho S-o(\varepsilon\log|G|!)-\varepsilon\log N)\geq\log N!-O(\log N).$$

Given (2), (3), i.e. S,T satisfy some non-trivial conditions, we can simplify the expression above and obtain

$$\log|G|!+o(\log|G|!)\geq\Omega(S\log N).$$

As we are conditioning on the event that G is large, plugging in the lower bound on $|G|$, we obtain that $ST^2\geq\tilde{\Omega}(\varepsilon N)$.

6 Proof of Theorem 4

Given a function inverter $(\alpha,\mathcal{A})$ that inverts an ε fraction of the input. For function $f:[M]\to[N]$, define $f^{-1}(y)=x$ if such x exists, else $\perp$. Using this notion, we can equivalently view sampling a function f from F_M as sampling an inverse function f^{-1} from all the possible partitions of $[M]$ into N bags, denoted as P_M. Let X sampled from P_M as in Theorem 13, then $S(X)=M\log N$ and

$$\begin{aligned}S(X_J)&=M\left(-\frac{1}{N}\log\frac{1}{N}-\left(1-\frac{1}{N}\right)\log\left(1-\frac{1}{N}\right)\right)\\&=\frac{M}{N}\left(N\log N-(N-1)\log(N-1)\right)\\&\leq\frac{M}{N}(\log N+\log e).\end{aligned}$$

[1] Technically, we proved that the average success probability will be at least this much. However, as the success probability is monotone in encoding length, it is not hard to see that we can still use Corollary 14.

▶ **Corollary 16.** *For any QRAC-VL for partitions P_M with $\delta = 1 - \beta$ for any β, we have*

$$L \geq M \log N - M\beta \left(\log(e/\beta) + \frac{M}{N}(\log N + \log e) \right).$$

Now we construct the encoding scheme given the inverter. Similarly as before, there is a subset $X_1 \subseteq F_M$ of size at least $0.5\varepsilon \cdot N^M$ such that for each function in X_1 the inverter is able to invert at least $\varepsilon/2$ fraction of the input. Let X_2 be functions where there exists an image in the function that has more than $K := \left(\frac{2M}{N} + 1\right) \cdot C \cdot \log(M/\varepsilon) = \tilde{O}(1)$ pre-images for some constant C. We claim that $|X_2| \leq 0.1\varepsilon N^M$ (for cases when $M \leq N$ and $M > N$, by using multiplicative form of Chernoff bound and union bound on the number of pre-images for each image. Let $X_3 = X_1 - X_2$ with size at least $0.4\varepsilon N^M$, that is the set of functions that both have a large amount of invertible points and each image does not have a lot of pre-images.

Consider a function $f \in X_3$, and let I be the set of indices $x \in [M]$ such that $\mathcal{A}$ when given input $f(x)$ returns exactly x (conditioned on f evaluating on the input is indeed $f(x)$) with the highest probability (ties are broken arbitrarily). It is not hard to prove that $|I| \geq \frac{\varepsilon M}{2K}$. We sample a subset $R \subseteq [M]$, with each element independently chosen with probability γ/T^2 for some constant γ that we will decide later.

Let $G \subseteq I$, where $x \in G$ if

1. $x \in R$; (18)
2. The total query magnitude on $R \setminus \{x\}$ while running $A^f(\alpha, f(x))$ is bounded by c/T, that is,
$$\sum_{z \in R \setminus \{x\}} q_z(x) \leq \frac{c}{T}. \tag{19}$$

▷ Claim 17. With probability at least 0.75 over the choice of R, $|G| = \Omega\left(\frac{\varepsilon M}{KT^2}\right)$.

Proof. The proof is almost exactly the same as in the case for permutations.

Let $H = R \cap I$. Due to the definition of R, $|H|$ is distributed according to a binomial distribution. Therefore, the expected value of $|H|$ is $|I|\gamma/T^2$. By the multiplicative Chernoff bound and (2),

$$\Pr_R \left[|H| \geq \frac{|I|\gamma}{2T^2} \right] \geq 0.95 \tag{20}$$

for all sufficiently large N.

By definition, each query that $\mathcal{A}$ makes is of unit length. Since $\mathcal{A}$ makes at most T queries, by Definition 7,

$$\sum_{z \in [N]} q_z(x) \leq T.$$

By linearity of expectation,

$$\mathop{\mathbb{E}}_R \left[\sum_{z \in R \setminus \{x\}} q_z(x) \right] = \sum_{z \in [N] \setminus \{x\}} \frac{\gamma}{T^2} q_z(x) \leq \frac{\gamma}{T^2} T = \frac{\gamma}{T}.$$

Hence, by Markov's inequality,

$$\Pr_R \left[\sum_{z \in R \setminus \{x\}} q_z(x) \geq \frac{c}{T} \right] \leq \frac{T}{c} \cdot \frac{\gamma}{T} = \frac{\gamma}{c}. \tag{21}$$

Let J denote the subset of $x \in I$ that satisfy (18) but not (19). Similarly, here (18) and (19) are also independent for each $x \in I$, since (18) is whether $f(x) \in R$ and (19) only concerns the intersection of R and $[N] \setminus \{f(x)\}$. Therefore by (21), the probability that $x \in I$ satisfies $x \in J$ is at most $\gamma^2/(cT^2)$. Hence, by Markov's inequality,

$$\Pr_R\left[|J| \leq \frac{10|I|\gamma^2}{cT^2}\right] \geq 0.9. \tag{22}$$

From (20) and (22), we get that with probability at least 0.75 over the choice of R,

$$|G| = |H| - |J| \geq \frac{|I|\gamma}{2T^2} - \frac{10|I|\gamma^2}{cT^2} \geq \frac{\varepsilon\gamma M}{4KT^2}\left(1 - \frac{5\gamma^2}{c}\right) = \Omega\left(\frac{\varepsilon M}{KT^2}\right),$$

given that γ is a small enough positive constant. ◀

We now proceed to describe the QRAC-VL scheme for encoding the partition f^{-1}. If $f \notin X_3$ or $|G|$ is not at least $\Omega(\varepsilon M/(KT^2))$, the encoding simply sets a (classical) flag (which takes one bit) and stores the entire table of f^{-1} (we will denote this as case A). In this case, it is straightforward to construct a decoder that succeed with probability 1.

Otherwise assuming $f \in X_3$ and G is large enough, we clear the first flag, and proceed with our QRAC-VL that computes (if necessary) and outputs the following information β as our encoding: (which we will denote as case B)

- The size of G, encoded using $\log(M+N)$ bits;
- The set $G \subseteq R$, encoded using $\log\binom{|R|}{|G|}$ bits;
- The set $f(G) \subseteq [N]$, encoded using $\log\binom{N}{|G|}$ bits;
- The function f restricted to input outside of G, encoded using $(M - |G|)\log N$ bits;
- Hash tags $h_1, ..., h_{|G|}$ for each $y \in f(G)$, each of length $\log(K \log N) = \log K + \log\log N$, encoded using $|G| \cdot (\log K + \log\log N)$;
- Quantum advice used by the algorithm repeated ρ times $\alpha^{\bigotimes \rho}$, for $\rho = \tilde{O}(K)$.

Upon given the encoding β, some image $y \in [N]$, and the algorithm's randomness R, the decoder first proceeds to recover set G, $f(G)$ and $f(x)$ for every $x \notin G$. If the given $y \notin f(G)$, the decoder outputs $x = f^{-1}(y)$. Otherwise, the decoder constructs

$$f'(x) = \begin{cases} y, & x \in G; \\ f(x), & x \notin G. \end{cases}$$

Then the decoder extracts $\alpha_1, \alpha_2, ..., \alpha_\rho$, and invokes $\mathcal{A}^{f'}(\alpha_i, y)$ to obtain ρ outputs. After measuring the outputs, the decoder hashes each output and compares with the hash h_y in the encoding. Finally, the decoder randomly chooses a output with the correct hash, combining other pre-images in the encoding as the output pre-image set.

Let $|\phi_f\rangle$ and $|\phi_{f'}\rangle$ denote the final states of $\mathcal{A}$ when it is given the oracle f and f' respectively. Then by Lemma 8 and the definition of a good element,

$$\||\phi_f\rangle - |\phi_{f'}\rangle\| \leq \sqrt{T \sum_{z \in R \setminus \{x\}} q_z(x)} \leq \sqrt{T \cdot \frac{c}{T}} = \sqrt{c}.$$

As $x \in I$, by the definition of I, measuring $|\phi_f\rangle$ gives some pre-image of y that is in G with probability at least $2/3 \cdot 1/K$. Given c is a small enough positive constant, measuring $|\phi_{f'}\rangle$ will also give x with probability at least $0.6/K$. Assuming the logarithmics in $\rho = \tilde{O}(K)$ is large enough, we can find at least one correct output in this process with probability at least

$1 - 1/\log N$. Due to the length of the hash tag and Theorem 10, all the incorrect outputs will be discarded with probability $1 - 1/\log N$. Overall, the success probability of our decoding procedure for a $y \in f(G)$ is at least $1 - 2/\log N$.

We now examine the length of our encoding. With probability $1 - 0.6\varepsilon$, we have $f \notin X_3$; with probability $\varepsilon \cdot 0.4 \cdot (1 - 0.75)$, we have $f \in X$ but G is small. Therefore, over all, with probability $1 - 0.7\varepsilon$, our encoding will take case A, where the encoding consists of $1 + \log N!$ classical bits and decoder succeeds with probability 1.

With probability 0.3ε, our encoding takes case B, and the size of the encoding will be

$$1 + \log(M+N) + \log \binom{|R|}{|G|} + \log \binom{N}{|G|} + (M - |G|)\log N \\ + |G| \log(K \log N) + \rho S,$$

which is at most

$$M \log N + |G| \log \frac{O(K^2 \log N)}{\varepsilon} - |G| \log |G| + \rho S,$$

for all sufficiently large N. In this case, when the decoder is queried a point inside what she has remembered, that is $y \notin \pi(G)$ (which occurs with probability $1 - |G|/N$), she recovers the correct pre-image with probability 1; otherwise, she recovers the correct pre-image with probability at least $1 - 2/\log N$.

Overall, the average success probability is at least $1 - 0.15\varepsilon|G|/(N \log N) \leq 1 - \Omega(1/N^{10})$. By Corollary 16 and $M/N + 1 = \Theta(1)$ by (4), we have

$$0.3\varepsilon \cdot \left(|G| \log \frac{O(K^2 \log N)}{\varepsilon} - |G| \log |G| + \rho S \right) \\ \geq -0.15 \frac{\varepsilon |G| M}{N \log N} \cdot O(\log N).$$

Using the fact that (5), (7), we can ignore the lower order terms and obtain

$$\tilde{O}(|G|) \geq \tilde{\Omega}(SK).$$

Thus, $ST^2 \geq \tilde{\Omega}(\varepsilon M)$.

7 Open Questions

Our work still does not answer whether there exists a tighter asymptotic lower bound like $ST + T^2 \geq \varepsilon N$, nor whether there exists an attack using quantum advice that achieves $ST^2 = \varepsilon N$.

On the other hand, it seems hard to generalize our techniques to handle random functions where $M \gg N$. Say $M = N^2$. It turns out that for whatever choice of $G \subseteq R$, remembering where G is, and f for points outside of G is already too much (requires number of bits greater than $M \log N$). Recall that $|R| \propto M/T^2$, but if we only remember one pre-image per image, $|G| \leq N$. Therefore under these parameters, $\log \binom{|R|}{|G|} \geq |G| \log N > |G| \log |G|$ and we will lose the non-trivial savings we get from the reduction. Therefore, a natural direction would be to prove any meaningful lower bound for random function inversion under the regime where $M \gg N$.

References

1 Gorjan Alagic, Stacey Jeffery, Maris Ozols, and Alexander Poremba. On non-adaptive quantum chosen-ciphertext attacks and learning with errors. *arXiv preprint arXiv:1808.09655*, 2018.

2 Andris Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002.

3 Andris Ambainis, Ashwin Nayak, Ammon Ta-Shma, and Umesh Vazirani. Dense quantum coding and a lower bound for 1-way quantum automata. In *Proceedings of the thirty-first annual ACM symposium on Theory of Computing*, pages 376–383. ACM, 1999.

4 Henry Corrigan-Gibbs and Dmitry Kogan. The function-inversion problem: Barriers and opportunities. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:182, 2018. URL: `https://eccc.weizmann.ac.il/report/2018/182`.

5 Anindya De, Luca Trevisan, and Madhur Tulsiani. Non-uniform attacks against one-way functions and prgs. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 16, page 113, 2009.

6 Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

7 Martin Hellman. A cryptanalytic time-memory trade-off. *IEEE transactions on Information Theory*, 26(4):401–406, 1980.

8 Minki Hhan, Keita Xagawa, and Takashi Yamakawa. Quantum random oracle model with auxiliary input. Cryptology ePrint Archive, Report 2019/1093, 2019. URL: `https://eprint.iacr.org/2019/1093`.

9 Aran Nayebi, Scott Aaronson, Aleksandrs Belovs, and Luca Trevisan. Quantum lower bound for inverting a permutation with advice. *Quantum Information & Computation*, 15(11-12):901–913, 2015.

10 Benjamin Schumacher and Michael D Westmoreland. Indeterminate-length quantum coding. *Physical Review A*, 64(4):042304, 2001.

11 Salil P Vadhan et al. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

12 Umesh Vazirani. On the power of quantum computation. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 356(1743):1759–1768, 1998.

13 William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802, 1982.

14 Andrew Chi-Chih Yao. Coherent functions and program checkers (extended abstract), stoc 1990, 1990.

Out-Of-Band Authenticated Group Key Exchange: From Strong Authentication to Immediate Key Delivery

Moni Naor
Department of Computer Science and Applied Mathematics,
Weizmann Institute of Science, Rehovot 76100, Israel
http://www.wisdom.weizmann.ac.il/~naor
moni.naor@weizmann.ac.il

Lior Rotem
School of Computer Science and Engineering,
Hebrew University of Jerusalem, Jerusalem 91904, Israel
lior.rotem@cs.huji.ac.il

Gil Segev
School of Computer Science and Engineering,
Hebrew University of Jerusalem, Jerusalem 91904, Israel
segev@cs.huji.ac.il

Abstract

Given the inherent ad-hoc nature of popular communication platforms, *out-of-band authenticated key-exchange protocols* are becoming widely deployed: Key exchange protocols that enable users to detect man-in-the-middle attacks by manually authenticating one short value. In this work we put forward the notion of *immediate key delivery* for such protocols, requiring that even if some users participate in the protocol but do not complete it (e.g., due to losing data connectivity or to other common synchronicity issues), then the remaining users should still agree on a shared secret. A property of a similar flavor was introduced by Alwen, Coretti and Dodis (EUROCRYPT '19) asking for immediate decryption of messages in user-to-user messaging *while assuming that a shared secret has already been established* – but the underlying issue is crucial already during the initial key exchange and goes far beyond the context of messaging.

Equipped with our immediate key delivery property, we formalize strong notions of security for out-of-band authenticated group key exchange, and demonstrate that the existing protocols either do not satisfy our notions of security or are impractical (these include, in particular, the protocols deployed by Telegram, Signal and WhatsApp). Then, based on the existence of any passively-secure key-exchange protocol (e.g., the Diffie-Hellman protocol), we construct an out-of-band authenticated group key-exchange protocol satisfying our notions of security. Our protocol is inspired by techniques that have been developed in the context of fair string sampling in order to minimize the effect of adversarial aborts, and offers the optimal tradeoff between the length of its out-of-band value and its security.

2012 ACM Subject Classification Security and privacy → Cryptography; Theory of computation → Cryptographic protocols; Theory of computation → Cryptographic primitives

Keywords and phrases End-to-end encryption, out-of-band authentication, key exchange

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.9

Related Version A full version of the paper is available at https://eprint.iacr.org/2019/1458.

Funding *Moni Naor*: Supported in part by a grant from the Israel Science Foundation (no. 950/16). Incumbent of the Judith Kleeman Professorial Chair.
Lior Rotem: Supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities and by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253).
Gil Segev: Supported by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253).

1 Introduction

A fundamental challenge in cryptography is that of generating shared secrets in communication networks that are susceptible to man-in-the-middle attacks. When a public-key infrastructure is available, this task has been thoroughly studied, and many protocols have been suggested (see Section 1.2). The question remains, however, of how to agree on an initial secret when connections are formed ad-hoc, and a public-key infrastructure is impractical to maintain. Such scenarios include, for example, communication platforms offering end-to-end encrypted messaging services, audio calls or video calls [50, 60, 63, 62, 5, 25, 10, 20, 35, 51, 30, 19, 23, 57, 2, 31], as well as secure pairing of IoT devices (e.g., [22, 37, 11]).

Out-of-band authenticated key exchange. Given that man-in-the-middle attacks are impossible to detect without any additional setup, one approach often taken is to provide users/devices with the ability to communicate "out-of-band", assuming that they have access to an external channel through which they can information-theoretically authenticate short values. Equipped with such an external channel, one can then rely on out-of-band authenticated key-exchange protocols: Protocols that are tailored to using both a standard insecure channel and a low-bandwidth out-of-band channel, and enable users to bootstrap the limited resource of information-theoretical authentication provided by the out-of-band channel in order to establish shared secrets while detecting man-in-the-middle attacks. Such an approach is taken by most communication platforms providing end-to-end encryption and by protocols for pairing of IoT devices (see the references above).

The out-of-band channel typically corresponds to having the users compare with each other a short value displayed by their devices (or having a single user compare a string displayed by all paired devices in the context of pairing of IoT devices), but can in fact be based on a variety of real-world assumptions (e.g., [43, 27, 56, 42, 32, 22, 37, 59, 63]). In most implementations the "manual" flavor of the out-of-band channel introduces a tradeoff between the effort invested by the users and the security guarantees: A longer out-of-band value may enable better security in principal, but also incurs a more intensive user effort, thus hurting usability and ultimately security.

Non-interactive vs. interactive protocols. As in standard key exchange, there are two main flavors of out-of-band authenticated key-exchange protocols: *Non-interactive* protocols in which each user sends at most one message and this message is sent independently of the other users' messages, and *interactive* protocols in which users may send more than one message and these messages may depend on other users' previously-sent messages.[1]

Non-interactive protocols are widely used by messaging platforms (e.g., WhatsApp and Signal [63]), since they do not require any two users to be online at any particular point in time. However, such protocols are inherently limited in the security they can provide – as we discuss in Section 1.3.

[1] These two flavors of protocols are sometimes referred to as "asynchronous" protocols vs. "synchronous" protocols (e.g., in the specific context of messaging protocols [52, 19, 2, 31] – to which we do not at all limit ourselves in this work). As discussed below we follow the more standard terminology of non-interactive protocols vs. interactive protocols since the standard model of synchronous computation in distributed computing is much more restrictive than the standard model required for interactive cryptographic protocols in general, and for the protocols considered in this paper in particular (e.g., a global clock synchronizing the entire execution of the protocol among the various parties is not required [41, 3]).

Our focus: Immediate key delivery in interactive protocols. In various popular scenarios, such as voice and video calls or pairing of IoT devices, the users or devices participating in the protocol are typically expected to remain online throughout its execution. In these scenarios, unlike in messaging applications, interactive protocols may be used in order to ensure stronger security guarantees (e.g., security which is independent of the adversary's concrete running time). This is the case, for example, in the out-of-band key-exchange protocol Telegram uses for its voice calls [59].

An additional approach to constructing interactive out-of-band authenticated key exchange protocols is to first run any passively-secure key-exchange protocol, and then use an *out-of-band message authentication protocol* in order to authenticate its transcript [61, 49, 54]. An out-of-band message authentication protocol allows for the authentication of long messages while using the out-of-band channel only to information-theoretically authenticate one short value. Although any solution to the general task of establishing shared keys must inherently rely on computational assumptions, out-of-band *message authentication* protocols may provide unconditional information-theoretical security. By now there is a sound theoretical understanding (i.e., protocols and matching lower bounds) of out-of-band message authentication protocols, in both the user-to-user and the group settings, as well as practically-relevant protocols in both settings [61, 48, 46, 54, 44] – these works were indeed motivated by the task out-of-band authenticated key exchange.

In contrast, out-of-band authenticated key exchange has been studied in the user-to-user setting (e.g., [49, 40]) but has been left without any rigorous treatment in the group setting. In particular, when considering the security of out-of-band authenticated key exchange in the group setting, a crucial requirement is that even if some users participate in the protocol but do not complete it, then the remaining users should still agree on a shared key that will enable them to start interacting in an end-to-end encrypted manner. We refer to this property as *immediate key delivery*. Alwen, Coretti and Dodis [2] have recently suggested a property of such flavor to which they referred to as "immediate decryption". Their work was in the context of messaging protocols *assuming that a shared secret key has already been established* – but the underlying issue is crucial already during the initial key exchange.

Providing immediate key delivery is a challenge that arises only in the interactive setting, as it is trivially guaranteed by any non-interactive protocol (but, as discussed above, such protocols provide somewhat weaker security guarantees). Although interactive protocols are suitable for scenarios in which users are typically expected to remain online, protocols still have to address cases where some of the users do not complete the protocol. Otherwise, for example, any user who loses connectivity prevents the successful completion of the protocol by the remaining users. Moreover, if a protocol does not offer immediate key delivery, then it becomes very easy for an attacker to prevent the users from agreeing on a shared secret, by simply blocking all outgoing communication from a single user in the group.

The significant and practical importance of immediate key delivery, together with various other security considerations for out-of-band protocols, motivate an in-depth examination of out-of-band authenticated key exchange, including formal definitions and protocols that satisfy them.

1.1 Our Contributions

Motivated by the above-described state of affairs, we present the following contributions:

- We suggest a framework for analyzing out-of-band authenticated group key-exchange protocols, capturing crucial security and functionality properties that arise in the group setting for out-of-band protocols.

- We observe that the existing approaches for constructing out-of-band authenticated key-exchange protocols either do not satisfy our (standard) notions of security or are impractical (already for rather small groups). This situation highlights the fact that it is highly non-trivial to satisfy our notions of security while keeping the out-of-band value short.
- Based on the existence of any passively-secure user-to-user key-exchange protocol (e.g., the Diffie-Hellman protocol), we construct an out-of-band authenticated group key-exchange protocol satisfying our notions of security, and offering the optimal tradeoff between the length of its out-of-band value and its security. Moreover, for some possible use-cases, instantiating our protocol in the random-oracle models leads to a concrete and efficient protocol.

In what follows we briefly discuss each of these contributions, and the reader is referred to Section 1.3 for a more elaborate and technical overview.

Modeling out-of-band authenticated group key exchange. We consider a group of users communicating over a completely-insecure channel that is susceptible to man-in-the-middle attacks, and in addition assume that *some* user of the group can information-theoretically authenticate one short value to all other users who have not yet aborted, over the out-of-band channel (note that we do not make any assumptions as to the particular identity of that user).[2]

Within this communication model (which we formally define in Section 2), we put forth a realistic framework and notions of security for out-of-band authenticated key-exchange in the group setting, considering the following three requirements:

- **Pseudorandomness:** If a man-in-the-middle adversary does not interfere with the communication, the resulting shared key should be computationally indistinguishable from an independent and uniformly-distributed key given the transcript of the protocol *which includes the out-of-band value*.
- **Man-in-the-middle detection:** If a man-in-the-middle adversary does interfere with the communication, this should be detected except with probability $\epsilon(\lambda) + \mathsf{negl}(\lambda)$, where ϵ is a pre-determined function of the security parameter $\lambda \in \mathbb{N}$, and negl is a negligible function which may depend on the adversary.
  Most importantly, ϵ must be fixed for all adversaries, and in particular it is not allowed to depend on the adversary's on-line or off-line running time or space usage – as the *effective length* of the out-of-band value might not always be sufficiently long (e.g., when executed by "lazy users" who may not consider the out-of-band value in its entirety [44]).
- **Immediate key delivery:** Even if a subset of the parties aborts the execution of the protocol before its completion, the remaining parties should still agree on a shared key (the abort decisions may be determined adversarially throughout the execution of the protocol). This requirement significantly strengthens the standard correctness requirement of key-exchange protocols, and achieving this requirement is the core technical contribution of our work.

[2] The way that the out-of-band value is propagated through the group might be different; e.g., if some users recognize the voice of one user in a voice group call, and the other users recognize the voice of another user, then informing all users of the out-of-band value requires the two recognized users to read it out loud. Our model, in which there is a single out-of-band value and a single user who sends it, can always be easily translated to such situations (e.g., by having an out-of-band channel from each of the said users to the users in the group who recognize her voice).

Note that the pseudorandomness and man-in-the-middle detection requirements are relevant already in the user-to-user setting (and we consider natural extensions of these requirements from passively-secure protocols to out-of-band protocols), and that the immediate key delivery is a new requirement that we introduce in the group setting.

Existing protocols do not meet our requirements. We show that even though the three requirements listed above seem fairly standard as far as cryptographic definitions go, they are not met by existing protocols. Namely, we observe that each of the out-of-band authenticated key-exchange protocols deployed by Signal, WhatsApp and Telegram, and that the protocol suggested by Rotem and Segev [54] does not satisfy at least one of the aforementioned requirements.

Already in the user-to-user setting, we show that the protocol deployed by Telegram does not satisfy our pseudorandomness requirement, and that the protocols deployed by Signal and WhatsApp do not satisfy our man-in-the-middle detection requirement. In the group setting, even though these protocols provide immediate key delivery, they are non-scalable in terms of the length of the out-of-band value, since they require running a user-to-user protocol with each member of the group separately, resulting in an out-of-band value whose length depends linearly on the size of the group. For example, in a group of size 32, in order to get 60 bits of security, the out-of-band value in these protocols has to be of length at least $31 \times 60 = 1860$ bits (i.e., the initiator of the key exchange has to compare at least 560 decimal digits with other users). In the group setting, the protocol of Rotem and Segev, which relies on the above-mentioned transcript-authentication approach [49] is more practical, and satisfies our pseudorandomness and man-in-the-middle detection properties, but does not provide immediate key delivery.

We stress that as mentioned above, some of these protocols have their advantages in particular use cases. However, the fact that none of them provide both optimal security guarantees per our security notion and also immediate key delivery in the group setting, exemplifies in our view the difficulty that lies in satisfying all of these requirements simultaneously and highlights the challenges that need to be overcome. Looking ahead, the main reason that immediate key delivery is challenging to obtain without substantially increasing the length of the out-of-band value, is that an adversary may choose a subset of aborting users out of an *exponential* number of such subsets – and this allows the adversary significant control over the execution of the protocol.

From strong(er) message authentication to out-of-band authenticated key exchange. We construct an out-of-band authenticated group key-exchange protocol which satisfies our notions of security, based on any passively-secure user-to-user key-exchange protocol. Moreover, we prove that our protocol enjoys the optimal tradeoff (within lower-order terms) between the length of its out-of-band value and the probability of an active attack going undetected.[3]

▶ **Theorem 1** (informal). *Assuming the existence of any passively-secure user-to-user key-exchange protocol, then for any functions $n = n(\lambda)$ and $\ell = \ell(\lambda)$ there exists an out-of-band authenticated key-exchange protocol for groups of $n(\lambda)$ users, with an out-of-band value of length $\ell(\lambda)$ bits such that any active man-in-the-middle attack is detected except with probability $\epsilon(\lambda) \leq 2(n(\lambda) - 1) \cdot (1/2 + o(1))^{\ell(\lambda)}$, where $\lambda \in \mathbb{N}$ in the security parameter.*

[3] Our protocol provides such an optimal tradeoff even when executed by "lazy users", who may not consider the out-of-band value in its entirety, as recently formalized by Naor et al. [44].

Our protocol is based on a general transformation that takes any passively-secure key-exchange protocol and produces an out-of-band authenticated key-exchange protocol. Concretely, we observe that although the above-mentioned transcript-authentication approach (i.e., using a group out-of-band message authentication protocol in order to authenticate the transcript of a passively-secure group key-exchange protocol) fails to guarantee immediate key delivery, this can be overcome if the underlying message authentication protocol provides a property we refer to as *immediate message delivery* (the precise transformation requires overcoming various additional challenges). We construct such a strengthened out-of-band message authentication protocol by starting from the basic structure of the group protocol of Rotem and Segev, and incorporating within it techniques from the realm of fair multi-party string-sampling protocols (i.e., protocols in which even if some parties abort then the remaining parties sample a "relatively unbiased" string [4, 18] – see Section 1.3 for more details). We view this as our main technical contribution.

A benefit of the fact that we present our protocol as a general transformation while relying on generic building blocks, is that this enables for a much greater modularity in its instantiation. In particular, this allows for the reliance on post-quantum secure assumptions as opposed to the currently deployed protocols by Telegram, Signal and WhatsApp that are based on the Decisional Diffie-Hellman assumption.

1.2 Related Work

The problem of detecting man-in-the-middle attacks in key exchange protocols has been studied extensively in various models (see, for example, [8, 6, 58, 7, 16, 36] for user-to-user protocols, and [9, 15, 34] for group protocols). Our setting and definitions bear some resemblance in particular to that of password-authenticated key exchange (PAKE; see [29, 12, 33, 26, 1] and the references therein), in that in both cases the security is inherently a function of the unpredictability of some short value (the out-of-band value in our case, and the shared password in the case of PAKE).

In particular, in the PAKE setting, Fiore, Vasco and Soriente [24] considered the problem of "partitioned group key exchange" which is conceptually somewhat similar to the problem we consider in this paper: Designing a PAKE protocol with the guarantee that even if some users provide a wrong password then all users who provided the correct password should still agree on a shared key. The main difference, however, between this problem and our work is the correctness requirement: Fiore et al. assume that all users are on-line and follow the instructions of the protocol, and require that all users who provide the same password output the same key, whereas we assume that some users may adversarially abort the protocol at any stage and require that all other users output the same key. This difference, together with the substantial differences of the two authentication models, lead to completely different technical challenges (and solutions).

More generally, although there are natural similarities between the various authentication models, there are several key differences between our work and the lines of works mentioned above. Namely, to provide immediate key delivery, our model and definitions accommodate users who abort prematurely, whereas most of the works on authenticated key-exchange are either in the user-to-user setting, or consider groups that remain static (i.e., no users are added or removed) *throughout the execution of the protocol*. Some works (e.g., [13, 14]) do consider dynamic groups that may change over time and their shared secret needs to be updated, but not the scenario that we are studying of users who abort *during the execution of the protocol itself*. In that respect, our work is focused on initial key exchange (and its authentication), and we do not explicitly consider the task of adding or removing users in

later stages. In any case, adding a user to the group while communicating with only a single existing member of the group, as is the case with the deployed protocols, can and must be authenticated using a user-to-user out-of-band protocol. This approach can also be used to add users who went offline during the initial setup, which again must require an additional out-of-band verification.

In the out-of-band model, most previous works concentrated on message authentication [53, 61, 46, 48, 54, 44], with the exception of Pasini and Vaudenay [49] and Lindell [40], who studied key exchange explicitly, but only in the user-to-user setting. Pasini and Vaudenay followed the transcript-authentication paradigm described above, while Lindell focused on analyzing the specific Bluetooth v2.1 comparison-based key-exchange protocol.

1.3 Overview of Our Security Notions and Construction

In this section we first discuss the motivation underlying our three security requirements (which were briefly mentioned in Section 1.1 and are formally defined in Section 3). Next, we overview the "transcript authentication" approach for constructing an out-of-band authenticated group key-exchange protocol (which serves as our starting point), and point out its current limitations. Then, we provide a high-level overview of our construction and of its proof of security.

Our notions of security. Our work puts forward extensions of the standard notions of pseudorandomness and man-in-the-middle detection that are tailored to out-of-band protocols, as well as introduces the notion of immediate key delivery, as discussed in Section 1.1.

Requirement 1: Pseudorandomness given the out-of-band value. The out-of-band channel is assumed to provide authenticity for one short value, but it is not assumed to provide any form of secrecy, and thus all communication over this channel may be completely visible to an adversary. Thus, the natural extension of the standard pseudorandomness requirement for key-exchange protocol must consider an adversary observing both the communication over the insecure channel and over the out-of-band channel. For such an adversary, the resulting shared key should be computationally indistinguishable from an independent and uniformly-distributed key.

Requirement 2: Adversary-independent man-in-the-middle detection. The probability of detecting an active man-in-the-middle attack depends (at least) on the bit-length ℓ of the out-of-band authenticated value (in Section 3 we provide a simple proof showing that any protocol can be undetectably attacked with probability essentially $\epsilon = n \cdot 2^{-\ell}$). We require that active attacks are detected with probability that depends on the protocol itself (e.g., $\epsilon = n \cdot 2^{-\ell}$), and do not scale in a meaningful manner with the adversary's on-line or off-line running time or space usage. For example, our requirement rules out protocols that out-of-band authenticate an 80-bit value, and an adversary that can execute 2^{40} computations of a certain hash function can break its security with probability $2^{40} \cdot 2^{-80}$. This property is even more crucial when considering the likely scenario of "lazy users", as formalized by Naor et al. [44], where users may consider only a short sub-string of the out-of-band authenticated value. This renders the "effective length" of the out-of-band value much shorter than its actual length ℓ. For example, if the security that a protocol provides is $T \cdot 2^{-\ell}$, where T is roughly the running time of the adversary and ℓ is the length of the "de-facto out-of-band value", then if the users consider, say 20 bits from the out-of-band value, an adversary running in reasonable time can break the security of the protocol quite easily (instead of having the protocol still guarantee the best-possible security of $\epsilon = 2^{-20}$).

Requirement 3: Immediate key delivery. We require that even if a subset of the parties aborts the execution of the protocol before its completion, the remaining parties should still agree on a shared key. This is a crucial requirement not only due to the above-described nature of mobile-based messaging, but even more in order to protect against devastating adversarial denial-of-service attacks that are undetected by other users. For example, in the recently-suggested protocol of Rotem and Segev [54], an adversary that can simply block the communication going out of just one user, can make sure that the other users will never agree on a shared key, leaving the group either completely vulnerable or utterly useless.

Although this property is a functionality-focused one, our main technical challenge in this work is to obtain it while retaining a good (and preferably optimal) level of security. As we discuss in length in the continuation of this section, simple attempts to add immediate key delivery to the protocol of Rotem and Segev make it completely insecure.

Interaction is essential. Satisfying all three requirements simultaneously requires an interactive protocol. The pseudorandomness requirement may be satisfied both by interactive and by non-interactive protocols (under suitable assumptions). The third requirement, immediate key delivery, is trivially satisfied by any non-interactive protocol, but as mentioned above, the second requirement – adversary-independent MitM detection – cannot be satisfied by such protocols. Concretely, in Section 3 we show that for any non-interactive protocol and for any running time T, there exists a successful man-in-the-middle attacker that runs in time essentially T and is undetected with probability $\min\{1/3, \Omega(T \cdot 2^{-\ell})\}$, where ℓ is the bit-length of the out-of-band value. In this light our goal is to come up with interactive protocols that simultaneously guarantee all three requirements, while retaining a short out-of-band value.

Our starting point: The "transcript authentication" approach. As mentioned in Section 1.1, the out-of-band group key-exchange protocols deployed by WhatsApp, Signal and Telegram provide immediate key delivery, but impose a heavy burden on the users: These protocols require running a user-to-user protocol with each member of the group separately, resulting in an out-of-band value whose length depends linearly on the size of the group. In addition, recall that these protocols do not satisfy our two additional security requirements, and thus they do not seem to be promising starting points for designing protocols satisfying our goals.

Our starting point is the transcript-authentication approach described above [49], while using the out-of-band group message authentication protocol of Rotem and Segev [54]. Roughly speaking, this approach suggests running any passively-secure group key-exchange protocol,[4] and afterwards to authenticate its transcript via the following out-of-band message authentication protocol:

1. P_1 chooses $r_S \leftarrow \{0,1\}^\ell$ and commits to $\mathsf{trans} \| r_S$ to all other users, where trans is the transcript of the key-exchange protocol from P_1's point of view.
2. $P_2, \ldots, P_n$ cooperatively choose a string r_R: Each P_i chooses $r_i \leftarrow \{0,1\}^\ell$ and commits to it to all other users. After all users have committed, each P_i decommits to reveal r_i, and sets $r_R = \bigoplus_{i \in \{2,\ldots,n\}} r_i$.

[4] Most naively, the initiator P_1 can execute a user-to-user protocol (such as the Diffie-Hellman protocol) with each other user P_i for obtaining a shared key k_i. Then, P_1 will sample a random key k and encrypt it to each other user P_i using the key k_i.

3. P_1 decommits to reveal r_S, and then out-of-band authenticates to $\sigma = r_S \oplus r_R$. Each of the other users accepts (and outputs the key agreed upon in the key exchange step) if and only if σ and trans are both consistent with her view.

This protocol falls short of satisfying our definition for out-of-band authenticated group key exchange in two respects. First, our definition requires that an active attack will be detected except with some pre-determined probability, but the only guarantee provided by the protocol of Rotem and Segev is that if trans is inconsistent with the view of some P_i, then with high probability this P_i will reject. It might still be the case though, that an active adversary modifies messages sent during the out-of-band message authentication phase described above.

This problem may be addressed in a simple manner (and in this specific protocol it is not that devastating to begin with): Instead of using the out-of-band message authentication protocol in order to authenticate the transcript trans of the group key exchange, P_1 samples a pair $(\mathsf{sk}, \mathsf{vk})$ of signing and verification keys for a *one-time strongly unforgeable signature scheme*; then uses the out-of-band message authentication protocol to authenticate vk to the other users; and finally uses sk to sign the transcripts of *both* the key-exchange protocol and the out-of-band message authentication protocol.

The second, more fundamental, problem is that the protocol of Rotem and Segev does not provide "immediate key delivery", even if the underlying passively-secure key-exchange protocol does provide it[5]. This is true since a user who identifies a deviation from the protocol (including a premature abort) terminates and rejects. In order for the out-of-band authenticated group key-exchange protocol to provide immediate key delivery, the out-of-band message authentication protocol needs to satisfy a similar property, to which we refer as *immediate message delivery*. This property essentially requires that even if a subset of the receivers in the protocol abort, but the execution is otherwise honest, the rest of the receivers should still accept the message.

Alas, the lacuna in the out-of-band message authentication protocol of Rotem and Segev, due to which it does not provide immediate message delivery, is far from being a mere technicality. To see why, consider what happens if we simply ignore aborting users, and take r_R to be the exclusive-or of only the r_i's of the users who opened their commitments. This might provide immediate key delivery, but gravely hurts the security of the protocol, by giving the man-in-the-middle adversary the ability to choose which commitments to open to each P_i *after observing* r_i. Concretely, in the full version of this paper [45], we present an attack showing that this change exponentially increases the forgery probability from roughly $n \cdot 2^{-\ell}$ to roughly $2^n \cdot 2^{-\ell}$, where n is the number of users in the group and ℓ is the length of the out-of-band value.

The underlying issue with the protocol of Rotem and Segev (explaining the exponential increase), is that a man-in-the-middle adversary interacting with, say P_i, can choose to abort any subset of $\{P_2, \ldots, P_n\} \setminus \{P_i\}$ towards P_i, before forwarding the decommitments of the users in this subset to P_i. Even if the interaction with P_i is otherwise honest, each possible aborting subset might induce a different value for r_R in the view of P_i. This enables a man-in-the-middle adversary to substantially "steer" the r_R that P_i computes, such that the attack will go undetected.

[5] The naive protocol described in Footnote 4 is a passively-secure protocol with immediate key delivery: Even if some user aborts then the remaining users still output the key k chosen by P_1.

Providing immediate message delivery: Attempt I. As a first attempt to limit the additional power provided to the adversary by allowing aborts, consider a "restart-after-abort" variant of the Rotem-Segev protocol, in which after an abort by any of the users, the remaining users start a fresh execution of the protocol. Intuitively, now the adversary has no incentive to abort more then a single user in each execution of the original Rotem-Segev protocol, and the identity of the particular user who aborts (if such a user exists) is of no consequence due to the symmetry of the protocol. Hence, instead of exponentially many choices of aborting subsets, in each execution of the original Rotem-Segev protocol the adversary effectively has only two (abort or not).

The problem with this approach however, is that now the adversary has up to $n-1$ attempts to break the security of the Rotem-Segev protocol, yielding a forgery probability of roughly $n^2 \cdot 2^{-\ell}$. This is much better than the $2^n \cdot 2^{-\ell}$ forgery probability of the "vanilla" Rotem-Segev protocol, but still quite far from optimal: The forgery probability grows quadratically with the number of users in the group, which may be significant in large groups, and as we show below, this can be avoided. Moreover, when the protocol is executed by lazy users as discussed above (who may not consider the out-of-band value in its entirety [44]), the effective value of ℓ might be relatively small, resulting in a substantial forgery probability. Instead, we are interested in a solution that provides security which is optimal with respect to the size of the group and to the length of the out-of-band value, so that it provides reasonable security even for lazy users (looking ahead, our protocol provides the *optimal* tradeoff within lower-order terms between the length of its out-of-band value and its security even when executed by lazy users).

Providing immediate message delivery: Attempt II. In light of the above, and inspired by techniques from protocols for fair string sampling, we construct a group out-of-band message authentication protocol that provides immediate message delivery – while retaining an optimal level of security (within lower order terms). The main idea behind our protocol is to replace the manner r_R is chosen in the protocol of Rotem and Segev, with a way which is more resilient to aborts. By that, intuitively speaking, we mean that even a man-in-the-middle adversary interacting with some P_i, and can simulate control over all users but P_i in that interaction, cannot force the r_R computed by P_i to hit the particular value that it needs in order for the attack to go unnoticed by P_i.

Instead of selecting r_R in "one shot" as done in the protocol of Rotem and Segev, in our protocol it is chosen in more gradual manner, which considerably limits the effect of adversarial aborts. Concretely, the users iteratively choose T ℓ-bit values $r_{R,1}, \ldots, r_{R,T}$ (where T is a parameter of the protocol) one after the other, in T consecutive iterations. In the tth iteration $r_{R,t}$ is chosen by the remaining users among $P_2, \ldots, P_n$ (i.e., the users who have not yet aborted) in the same manner as r_R is chosen in the protocol of Rotem and Segev. Finally, the value of r_R in our protocol is then taken to be the bit-wise majority of $r_{R,1}, \ldots, r_{R,T}$: The kth bit of r_R is the majority bit over the kth bits of $r_{R,1}, \ldots, r_{R,T}$. We refer the reader to Sections 4 and 5 for a complete and formal description of our protocol.

With this change, analyzing our new protocol proves to be technically involved, as a man-in-the-middle adversary has numerous more possible "synchronizations" (i.e., different orderings of messages) to impose on an execution of the protocol. Nevertheless, we manage to prove that when the commitment scheme used in our protocol is statistically-binding and concurrent non-malleable,[6] then the forgery probability is bounded roughly by $n \cdot (1/2 + n/\sqrt{T})^{\ell}$. Setting

[6] see the full version [45] as well as [21, 39, 28, 17] and the references therein for further details on such

the parameter T to be $n^2 \cdot \omega(1)$ (e.g., $n^2 \cdot \log^* \lambda$), we get that the forgery probability is $n \cdot (1/2 + o(1))^\ell$, matching our lower bound of $\min\{1/3, \Omega(n \cdot 2^{-\ell})\}$ (see Section 3) within lower order terms.

Overview of our proof of security. We provide a brief and high level overview of the proof of unforgeability of our out-of-band message authentication protocol, ignoring various technical difficulties and focusing on the main ideas. We prove that for every $i \in \{2, \ldots, n\}$, if the man-in-the-middle changes the verification key sent to P_i in the beginning of our out-of-band authenticated key-exchange protocol,[7] then the probability that P_i will not detect this interference (i.e., will not output $\perp$) is upper bounded by roughly $(1/2 + n/\sqrt{T})^\ell$. We do so by considering all possible synchronizations that a man-in-the-middle might impose on an execution of the protocol relative to P_i, and bound the probability of forgery in each of them relying on the statistical binding and on the concurrent non-malleability of the underlying commitment scheme. We manage to partition all possible such synchronizations into two families, and handle each one separately. For simplicity of presentation in this overview, we focus on the case where $\ell = 1$ (i.e., the initiator P_1 out-of-band authenticates a single bit), and the reader is referred to Section 5 for our formal proof of security.

Proof of security: Case I. In the first family of synchronizations, P_1 decommits to reveal r_S before P_i receives the first round of commitments from $P_2, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n$. In this case, by the statistical binding, the values of r_S and r_R according to the view of P_1 and the value of r_S according to the view of P_i, have all been determined by the time P_i receives the first round of commitments. Hence, in order for P_i to not reject, the man-in-the-middle must make sure that r_R according to the view of P_i hits the unique value $r_R^* \in \{0, 1\}$ which is the exclusive-or of the three aforesaid determined values. We bound the probability that $r_R = r_R^*$ using the concurrent non-malleability of the commitment scheme, where the heart of the proof lies in two parts.

The computational part, in which we show that no strategy of the man-in-the-middle can result in a noticeably-greater probability that $r_R = r_R^*$ then the following strategy, denoted M_{opt}: (1) In each iteration $t \in [T]$ and for every P_j that has not yet aborted according to the view of P_i, send P_i a commitment to the value 1 from P_j; (2) If the sampled value in this round $r_{R,t}$ is equal to r_R^* (when no user aborts), then open all commitments; (3) Otherwise, open all commitments except for that of the minimal-index user P_j that has not yet aborted (since P_j committed to the value 1, this is guaranteed to flip the bit $r_{R,t}$, so that it is equal to r_R^*).

We prove that this strategy is optimal in forcing $r_R = r_R^*$ (within a negligible additive factor) via a hybrid argument: We start with any other man-in-the-middle adversary M and gradually change its strategy to M_{opt}, iteration by iteration, proving that the probability that $r_R = r_R^*$ cannot decrease by too much in each change, or the concurrent non-malleability of the commitment scheme is violated. Concretely, we consider $T + 1$ hybrids, where the adversary in the tth hybrid, denoted by M_t, plays as M in the first $T - t$ iterations and as M_{opt} in the remaining t iterations. Observe, that in order for M_t to succeed with noticeably-greater probability than M_{t+1} (in forcing $r_R = r_R^*$), it must be the case that in the tth hybrid,

commitment schemes.

[7] Our protocol in Section 5 is a general-purpose out-of-band message authentication protocol. For concreteness in this overview, we focus on the case where the message to be authenticated is the verification key sampled by P_1, as is the case in our out-of-band authenticated key-exchange protocol.

$\Pr[r_{R,t} = r_R^*]$ is noticeably greater than $1/2$. This contradicts the concurrent non-malleability of the underlying commitment scheme: Intuitively, this is because in an ideal experiment in which the bit contributed by P_i in the tth iteration is sampled anew just before P_i decommits, it holds that $\Pr[r_{R,t} = r_R^*] = 1/2$.

The statistical part, in which we show that the optimal adversary described above, M_{opt}, succeeds in forcing $r_R = r_R^*$ with probability no greater than roughly $1/2 + n/\sqrt{T}$. To prove this, it is convenient to think of an equivalent experiment, in which $r_{R,1}, \ldots, r_{R,T}$ are first sampled uniformly from $\{0,1\}$, and then the adversary is given the option to flip $n-2$ of them.[8] In this experiment, the adversary can force $r_R = r_R^*$ if and only if $|\{t \in [T] : r_{R,t} = r_R^*\}| \geq T/2 - n + 2$. We observe that $|\{t \in [T] : r_{R,t} = r_R^*\}|$ is a random variable distributed according to the binomial distribution with parameters $1/2$ and T. We then use the symmetry of this distribution and the fact that every value in its support is obtained with probability no greater than roughly $1/\sqrt{T}$, in order to bound the probability that $|\{t \in [T] : r_{R,t} = r_R^*\}| \geq T/2 - n + 2$ by roughly $1/2 + n/\sqrt{T}$.

Proof of security: Case II. In the second family of possible synchronizations, P_1 decommits to reveal r_S after P_i has received at least one round (and possibly many) of commitments from the other users. Denote the last round of commitments received by P_i before P_1 decommits by $t^* \in [T]$. In this case the man-in-the-middle adversary's situation is worse than in Case 1: The hiding and the concurrent non-malleability of the commitment scheme, and in particular of the commitment by P_1 to the value r_S, imply that the adversary cannot hope to force any of $r_{R,1}, \ldots, r_{R,t^*}$ to be equal to r_R^* with probability noticeably greater than $1/2$. This is because in an ideal experiment in which r_S is sampled anew just before P_1 decommits, it holds that $\Pr[r_{R,t} = r_R^*] = 1/2$ for every $t \in [t^*]$ and independently of the other rounds, and irrespective of the identity of the aborted users in rounds $1, \ldots, t^*$. Intuitively speaking, it follows that the adversary is only more limited than in the previous case, as she can use her "abort quota" effectively only in rounds $t^* + 1, \ldots, T$, and hence the forgery probability in this case cannot be noticeably greater than that of the previous case.

1.4 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we review the out-of-band communication model, and in Section 3 we present our notions of security for out-of-band authenticated group key-exchange protocols. In Section 4 we show that any passively-secure user-to-user key-exchange protocol can be transformed into an out-of-band authenticated group key-exchange protocol that satisfies our notions of security, and in Section 5 we construct an out-of-band message authentication protocol with immediate message delivery, which is the main building block underlying our transformation.

2 The Out-of-Band Communication Model

In this section we review the out-of-band communication model as well as the notion of an out-of-band message authentication protocol [61, 48, 54].

[8] For the case $\ell = 1$, this is indeed equivalent. For the general case of $\ell \geq 1$, this may only add power to the adversary, and hence the probability that $r_R = r_R^*$ can only increase. Hence, in the general case as well, bounding the probability that $r_R = r_R^*$ in this experiment bounds the probability that the man-in-the-middle adversary can force $r_R = r_R^*$.

The out-of-band channel and man-in-the-middle attacks. As formalized by Vaudenay and by Naor et al. in the user-to-user setting [61, 47] and extended by Rotem and Segev to the group setting [54], interaction among users in the out-of-band communication model occurs over two types of channels: Insecure channels and a low-bandwidth authenticated channel (referred to as the "out-of-band channel"). It is assumed that a man-in-the-middle adversary has complete control over the insecure channels: The adversary can read, delay and remove messages sent by the parties over the insecure channels, as well as insert new messages at any point in time. One may consider various topologies for the network of insecure channels. For our protocols we assume the minimal such topology: An insecure channel between some user (e.g., the initiator of the protocol) and any other user in the group (i.e., a star network).

As for the out-of-band channel, it is assumed that there exists some user that can out-of-band authenticate one short value to all other users in the group. This value is assumed to be authenticated but not secret: The adversary may read or remove this message for some or all users, and may delay it for different periods of time for different users, but cannot modify it in an undetectable manner. We stress that our requirement of the out-of-band channel is a rather weak one: We only require that there exists *some* user that can out-of-band authenticate a short value to the rest of the group, and we do not apply any restrictions as to who that user is.

In addition, we do not make any synchronization assumption regarding the out-of-band channel: We do not assume that all users have to be on-line when the out-of-band value is transmitted. Specifically, any subset of the users may be off-line at that time, and any user that comes back on-line will be able to make her own decision regarding the authenticity of the execution if and when the out-of-band value reaches her (recall that the attacker can block the out-of-band value to all or to some of the users). See [55] for a more in-depth discussion of the group out-of-band communication model.

Out-of-band message authentication. An out-of-band message authentication protocol enables a sender S to authenticate a message m, which may be chosen by the adversary, to all other users $R_1, \ldots, R_n$ in the group ($n = 1$ is the user-to-user setting, whereas $n \geq 2$ is the group setting). Once the execution is completed, each receiver R_i outputs either some message $\widehat{m}_i$ or the unique symbol $\perp$ implying rejection. The following definition was introduced by Rotem and Segev [54], naturally extending those of Vaudenay and Naor et al. [61, 47]:

▶ **Definition 2.** *Let $\ell = \ell(\lambda), \epsilon = \epsilon(\lambda)$ and $n = n(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. A group (ℓ, ϵ)-out-of-band message authentication protocol for $n(\lambda)$ receivers and message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is an $(n(\lambda)+1)$-party protocol, in which S sends at most $\ell(\lambda)$ bits over the out-of-band channel, and the following requirements hold:*

1. ***Correctness:*** *For every $\lambda \in \mathbb{N}$, for every $m \in \mathcal{M}_\lambda$ and every $i \in [n(\lambda)]$ it holds that $\Pr[\widehat{m}_i = m] = 1$, where the probability is over the randomness of the parties in an honest execution of the protocol.*
2. ***Unforgeability:*** *For every probabilistic polynomial-time adversary M there exists a negligible function $\nu(\cdot)$ such that for every input message $m \in \mathcal{M}_\lambda$ chosen by the adversary for the sender S it holds that*

$$\Pr[\exists i \in [n(\lambda)] : \widehat{m}_i \notin \{m, \perp\}] \leq \epsilon(\lambda) + \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the probability is taken over the randomness of the parties and the randomness of M in an execution of the protocol with M as the man-in-the-middle adversary.

Existing out-of-band message authentication protocols. In the user-to-user setting, Vaudenay [61] constructed a protocol in which the forgery probability ϵ is upper bounded by $2^{-\ell}$, where ℓ is the bit-length of the out-of-band authenticated value, and Vaudenay and Pasini [48] proved a matching lower bound. In the group setting, considering a strengthened version of Definition 2, Rotem and Segev [54] constructed a protocol for groups of size n in which the forgery probability is bounded by $(n-1)\cdot 2^{-\ell}$, and proved a matching lower bound. Both protocols can be based on the existence of any one-way function [38, 54] via non-malleable commitments.

3 Out-of-Band Authenticated Group Key Exchange

In this section we first present our strengthened notion of security for out-of-band authenticated key-exchange protocols.[9]

Pseudorandomness, MitM Detection and Immediate Key Delivery. Our strengthened notion of security for out-of-band key-exchange protocols consists of three requirements: Pseudorandomness and man-in-the-middle detection that are relevant already in the user-to-user setting, and immediate key delivery that we introduce in the group setting (as discussed in Section 1.3). Our pseudorandomness and man-in-the-middle detection requirements are natural extensions of these requirements to the out-of-band model:

- If a man-in-the-middle adversary does not interfere with the communication, the resulting shared key should be computationally indistinguishable from an independent and uniformly-distributed key given the transcript of the protocol *which includes the out-of-band value*.
- If a man-in-the-middle adversary does interfere with the communication, this should be detected except with probability $\epsilon(\lambda)+\mathsf{negl}(\lambda)$, where ϵ is a pre-determined function of the security parameter $\lambda\in\mathbb{N}$, and negl is a negligible function which may depend on the adversary. Most importantly, ϵ must be fixed for all adversaries (e.g., it is not allowed to depend on the adversary's on-line or off-line running time or space usage).
  Our security definition requires that an active attack is detected by all users on the receiving end of the out-of-band channel, for whom communication to or from them has been actively modified by the attacker. The task of notifying all other users (who are still online at the end of the execution) of an active attack can be achieved, for example, by assuming that all users can send an "out-of-band feedback" signal to all other members, indicating an attack. Observe that such an assumption (or an assumption of the same nature) is essential in order for all users to detect an active attack, as without it (i.e., with only a single user that can send a message out-of-band and all other communication being subject to man-in-the-middle manipulation) an active attack in which some of the users do not identify the attack is always possible.

Our immediate key delivery requirement significantly strengthens the standard correctness requirement of key-exchange protocols: Even if a subset of the parties aborts the execution of the protocol before its completion, the remaining parties should still agree on a shared

[9] In the full version [45], we show that the protocols deployed by Signal, WhatsApp and Telegram do not satisfy it already in the user-to-user setting, and that the protocol obtained via the "out-of-band transcript authentication" approach does not satisfy it in the group setting. We also show that there is a simple and practically-relevant user-to-user protocol that does satisfy our notion of security (and offers the optimal trade-off between the length of its out-of-band authenticated value and its man-in-the-middle detection probability).

key. To capture this requirement, for an algorithm A and an n-party protocol π, we let $\mathsf{FailStopExec}(\pi, A, \lambda)$ denote the output of the following experiment:

1. Start an execution of π with joint input 1^λ.
2. For every $i \in [n] \setminus \{1\}$, before P_i sends a message v according to π, invoke $\mathsf{decision} \leftarrow A(1^\lambda, \mathsf{PartialTrans})$, where $\mathsf{decision} \in \{\mathsf{abort}, \mathsf{continue}\}$ and $\mathsf{PartialTrans}$ is the partial transcript of the execution up to this point. If $\mathsf{decision} = \mathsf{continue}$, P_i sends v, and the execution continues. If $\mathsf{decision} = \mathsf{abort}$, P_i aborts and the execution continues without P_i.
3. The output of $\mathsf{FailStopExec}(\pi, A, \lambda)$ is a $(n+1)$-tuple $(\mathsf{AbortSet}, \mathsf{v}_1, \ldots, \mathsf{v}_n)$, where $\mathsf{AbortSet}$ denotes the set of indices of aborted parties at the end of the execution and v_i is the output of P_i if $P_i \notin \mathsf{AbortSet}$ and $\mathsf{v}_i = \perp$ otherwise.

Note that in order for this experiment to be well defined, the protocol π has to be well defined for any possible pattern of aborts. In that case, this experiment is well defined both for group key-exchange protocols (including passively-secure ones) and for group authentication protocols (looking ahead, this experiment will enable us to formalize a notion of "immediate message delivery" for authentication protocols). When π is a key-exchange protocol, we use $\mathsf{k}_1, \ldots, \mathsf{k}_n$ instead of $\mathsf{v}_1, \ldots, \mathsf{v}_n$ to denote the output keys of the users.

Also note that we assume that P_1 does not prematurely abort throughout the execution. This is essential, as we will assume without loss of generality that P_1 is the user that can send a short message over the out-of-band channel. Hence, if P_1 aborts prior to sending the out-of-band value, no meaningful security can be guaranteed. Practically speaking, in the context of messaging platforms, P_1 who initiates the key-exchange protocol is typically the first party to send an encrypted message to the group. Hence, if P_1 aborts, the need for a shared key is postponed until another message is sent (at which point, the users will execute the out-of-band group key-exchange protocol when initiated by a potentially different user).

Our definition, provided below, relies on the following notation. We denote by $\mathsf{MitMExec}(\pi, M, \lambda)$ the distribution over $(n+1)$-tuples $(\mathsf{view}_M, \mathsf{k}_1, \ldots, \mathsf{k}_n)$ induced by an execution of the protocol with a man-in-the-middle M, where the adversary and all parties run on input 1^λ, and view_M is the view of M at the end of the protocol ($\mathsf{k}_1, \ldots, \mathsf{k}_n$ are defined as before). For every $i \in \{2, \ldots, n\}$, let Active_i be the event in which the adversary actively changes the communication from or to P_i; i.e., by either modifying or removing messages sent from or to P_i or by inserting new message to or from P_i.

We also define the event Active: Informally, Active is the event in which the man-in-the-middle adversary M changes the communication among the parties in any manner that goes beyond simulating an abort by a subset of the parties (by simulating an abort by a party, we mean blocking all messages sent by that party from some point onward). More formally, let $q = q(\lambda)$ be a bound on the number of rounds in an execution of π on joint input 1^λ. For an execution according to $\mathsf{MitMExec}(\pi, M, \lambda)$, we denote by $\mathsf{Msgs}_i = (m_{i,1}, \ldots, m_{i,q})$ the vector of messages sent (in order) by P_i, where if P_i has sent t messages for $t < q$, we denote $m_j = \perp$ for every $j \in \{t+1, \ldots, q\}$. Similarly, we denote by $\widehat{\mathsf{Msgs}}_i = (\widehat{m_{i,1}}, \ldots, \widehat{m_{i,q}})$ the vector of messages received (in order) by parties other than P_i, as messages from P_i. We denote by Active the event in which for some $i \in [n]$, there exits $t \in [p]$ such that $m_{i,t} \neq \widehat{m_{i,t}}$ and at least one of the following conditions hold: (1) $\widehat{m_{i,t}} \neq \perp$; or (2) There exists $t' > t$ such that $m_{i,t'} \neq \perp$.

▶ **Definition 3.** *Let $n = n(\lambda), \ell = \ell(\lambda)$ and $\epsilon = \epsilon(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. A group out-of-band (ℓ, ϵ)-key-exchange protocol over key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in \mathbb{N}}$ for a group of size $n = n(\lambda)$ is an n-party protocol $\pi = \langle P_1, \ldots, P_n \rangle$, in which P_1 sends at most $\ell(\lambda)$ bits over the out-of-band channel and the following requirements hold:*

- ***Immediate key delivery:*** *For every* $\lambda \in \mathbb{N}$ *and every probabilistic polynomial-time algorithm* A*, it holds that*

$$\Pr\left[\forall i \in [n(\lambda)] \setminus \mathsf{AbortSet} : \mathsf{k}_1 = \mathsf{k}_i \in \mathcal{K}_\lambda\right] = 1$$

  where $(\mathsf{AbortSet}, \mathsf{k}_1, \ldots, \mathsf{k}_n) \leftarrow \mathsf{FailStopExec}(\pi, A, \lambda)$*.*
- ***Man-in-the-middle detection:*** *For any probabilistic polynomial-time algorithm* M *there exists a negligible function* $\nu(\cdot)$ *such that*

$$\Pr\left[\exists i \in \{2, \ldots, n(\lambda)\} : \mathsf{Active}_i \wedge \mathsf{k}_i \neq \perp\right] \leq \epsilon(\lambda) + \nu(\lambda)$$

  for all sufficiently large $\lambda \in \mathbb{N}$*, where* $(\mathsf{view}_M, \mathsf{k}_1, \ldots, \mathsf{k}_n) \leftarrow \mathsf{MitMExec}(\pi, M, \lambda)$*.*
- ***Pseudorandomness:*** *For any probabilistic polynomial-time algorithms* M *and* D *there exists a negligible function* $\nu(\cdot)$ *such that*

$$\left|\Pr\left[\overline{\mathsf{Active}} \wedge D(1^\lambda, \mathsf{view}_M, \mathsf{k}_1) = 1\right] - \Pr\left[\overline{\mathsf{Active}} \wedge D(1^\lambda, \mathsf{view}_M, \mathsf{k}) = 1\right]\right| \leq \nu(\lambda)$$

  for all sufficiently large $\lambda \in \mathbb{N}$*, where* $(\mathsf{view}_M, \mathsf{k}_1, \ldots, \mathsf{k}_n) \leftarrow \mathsf{MitMExec}(\pi, M, \lambda)$ *and* $\mathsf{k} \leftarrow \mathcal{K}_\lambda$*.*

We note that when $n = 2$, Definition 3 captures the user-to-user setting. In this case, the immediate key delivery property simply reverts back to the standard correctness property of key-exchange protocols. In addition, note that the immediate key delivery property is defined with respect to an efficient algorithm A, but our construction provides immediate key delivery even in the case where A is unbounded and receives access to the random coins of the users.

Interaction is essential. As mentioned in Section 1.3, no non-interactive protocol can satisfy our man-in-the-middle detection requirement. To see why that is, let π be such a non-interactive protocol and let P_i be any user participating in the protocol (other than the one in charge of sending the out-of-band value). Consider the following man-in-the-middle attacker, that can compute the secret key outputted by P_i:

1. The attacker forwards all messages sent by the users to all users participating in the protocol, other than to P_i. Let σ be the true out-of-band value sent as a result.
2. Let m_i be the message sent by P_i. The attacker samples T independent tuples of messages $M_{-i}^{(1)}, \ldots, M_{-i}^{(T)}$ for the other users participating in the protocol, and computes the T resulting out-of-band values $\sigma^{(1)}, \ldots, \sigma^{(T)}$ (i.e., $\sigma^{(j)}$ is the out-of-band value in the execution in which the messages sent are m_i and the messages in $M_{-i}^{(j)}$).
3. If for any $j^* \in [T]$ it holds that $\sigma^{(j^*)} = \sigma$, then the attacker sends the messages in the tuple $M_{-i}^{(j^*)}$ to P_i (as the messages sent by the other users in the protocol). Otherwise, the attacker has failed and she terminates the attack.

Observe that if the attacker completes the attack, then: (1) She knows the randomness used to sample the messages in $M_{-i}^{(j^*)}$, so she can compute the key outputted by P_i; and (2) The view of P_i is the same as in an honest execution in which the messages are m_i and the messages in $M_{-i}^{(j^*)}$, so the attack is undetected by P_i.

Hence, in order to analyze the probability that this attack is successful, we need to look at the probability that there exists such an index j^*. It turns out that we can bound this probability for any choice of m_i, so let us fix m_i and look at σ and $\sigma^{(1)}, \ldots, \sigma^{(T)}$ when P_i sends m_i. These are $T + 1$ independent samples from the distribution over the out-of-band value in a random execution of π, conditioned on P_i sending the message m_i. One can verify that the probability that there exists an index $j^* \in [T]$ such that $\sigma^{(j^*)} = \sigma$ is minimized when

this conditional distribution is the uniform distribution over $\{0,1\}^\ell$. For this distribution, the probability that there exists such an index j^* – and that the attack is successful – is at least $\min\{1/3, \Omega(T \cdot 2^{-\ell})\}$. The complete analysis is in the full version of the paper [45].

The required length of the out-of-band value. Theorem 4 states that any out-of-band group key-exchange protocol for n users with an out-of-band value of length ℓ bits can be undetectably attacked by an efficient man-in-the-middle adversary with probability roughly $n \cdot 2^{-\ell}$. As discussed in Section 1.3, a key goal in the out-of-band model is to construct protocols offering the optimal trade-off between their security and the length of their out-of-band authenticated value, and our protocols in this paper offer this optimal trade-off both in the user-to-user setting and in the group setting (within lower order terms). The proof of Theorem 4 may be found in the full version of this paper [45].

▶ **Theorem 4.** *Let $\ell = \ell(\lambda), n = n(\lambda)$ and $\epsilon = \epsilon(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. For any out-of-band (ℓ, ϵ)-key-exchange protocol (over any key space $\mathcal{K}$) for a group of size $n(\lambda)$, there exists a negligible function $\nu(\cdot)$ such that*

$$\epsilon(\lambda) \geq \min\left\{\frac{1}{3}, \frac{n(\lambda)-1}{4} \cdot 2^{-\ell}\right\} - \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$.

Lazy users. Motivated by the recent work of Naor et al. [44], we consider in addition the security of out-of-band key-exchange protocols when executed by lazy users who may not consider the out-of-band value in its entirety (e.g., users who compare with each other only a subset of its positions). Given an out-of-band key-exchange protocol $\pi = \langle P_1, \ldots, P_n \rangle$ we define a collection of "lazy protocols", one per each possible subset of positions of the out-of-band authenticated value. Specifically, given a protocol π in which the out-of-band authenticated value consists of ℓ characters, for a subset $\mathcal{I} \subseteq [\ell]$ of indexes, we consider the "lazy protocol" $\pi_{\mathcal{I}}$ in which the parties execute π, with the exception that the party who sends the out-of-band value does not send the entire value, but rather sends only its substring that corresponds to the positions in the set $\mathcal{I}$ (we refer the reader to the work of Naor et al. [44] for an in-depth discussion of lazy protocols and of the motivation underlying them).

▶ **Definition 5.** *Let $n = n(\lambda), \ell = \ell(\lambda)$ and $\epsilon = \epsilon(\lambda, \cdot) : 2^{[\ell]} \to [0,1]$ be functions of the security parameter $\lambda \in \mathbb{N}$. A group out-of-band (ℓ, ϵ)-key-exchange protocol $\pi = \langle P_1, \ldots, P_n \rangle$ for a group of size $n(\lambda)$ is* secure for lazy users *if for every $\mathcal{I} = \mathcal{I}(\lambda) \subseteq [\ell]$ the lazy protocol $\pi_{\mathcal{I}}$ is a group out-of-band $(|\mathcal{I}|, \epsilon(\cdot, \mathcal{I}))$-key-exchange protocol for a group of size $n(\lambda)$.*

4 From Strong Authentication to Key Exchange

We show that any passively-secure key-exchange protocol can be transformed into an out-of-band authenticated key-exchange protocol that satisfies our strong notion of security (see Definition 3). Moreover, the resulting protocol offers the optimal trade-off between the length of its out-of-band value and its security within lower order terms (see Theorem 4). We prove the following theorem:

▶ **Theorem 6.** *Assuming the existence of any passively-secure key-exchange protocol, then for any functions $\ell = \ell(\lambda)$ and $n = n(\lambda)$ of the security parameter $\lambda \in \mathbb{N}$ there exists an (ℓ, ϵ)-out-of-band authenticated key-exchange protocol for a group of size $n(\lambda)$ over the same key space, where $\epsilon(\lambda) \leq 2 \cdot (n-1) \cdot (1/2 + o(1))^{\ell(\lambda)}$ for every $\lambda \in \mathbb{N}$.*

4.1 Immediate Message Delivery and Passively-Secure Immediate Key Delivery

Our construction relies on two main building blocks that satisfy a property similar to that of immediate key delivery, as defined in Section 3 via our experiment $\mathsf{FailStopExec}(\pi, A, \lambda)$ for modeling a fail-stop execution of a protocol (note that this experiment is well defined not only for key-exchange protocols, and can in fact be used to model aborting parties in a wide range of protocols).

Out-of-band message authentication with immediate message delivery. Our first building block is a strengthened form of an out-of-band group message authentication protocol, extending the notion introduced by Rotem and Segev [54] for such protocols (see Definition 2) by asking for *immediate message delivery*: Even if a subset of the parties aborts the execution of the authentication protocol before its completion, the remaining parties should still output the sender's input message m. Relying on the notion we introduced in Section 3, this property is formalized by strengthening Definition 2 as follows:

▶ **Definition 7.** *Let $\ell = \ell(\lambda), \epsilon = \epsilon(\lambda)$ and $n = n(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. We say that an (ℓ, ϵ)-out-of-band group message authentication protocol $\pi = \langle S, R_1, \ldots, R_{n-1} \rangle$ for groups of size n and message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ provides* immediate message delivery*, if for every $\lambda \in \mathbb{N}$, for every algorithm A, and for every input message $m \in \mathcal{M}_\lambda$ to S, it holds that*

$$\Pr_{(\mathit{AbortSet}, m_1, \ldots, m_n) \leftarrow \mathit{FailStopExec}(\pi, A, \lambda)} [\forall i \in [n] \setminus \mathit{AbortSet} : m_i = m] = 1.$$

In Section 5 we show that an out-of-band message authentication protocol with immediate message delivery can be constructed based on the existence of any a statistically-binding concurrent non-malleable commitment scheme (and thus based on any one-way function – see the full version [45] for specifics). Moreover, the protocol we construct offers the optimal tradeoff between the length of its out-of-band value and its security (i.e., the adversary's forgery probability).

Passively-secure key exchange with immediate key delivery. Our second building block is a passively-secure key-exchange protocol with immediate key delivery. This is naturally defined by replacing the standard correctness requirement of passively-secure key-exchange protocols with our immediate key delivery requirement stated in Definition 3. A passively-secure key exchange protocol $\langle P_1, \ldots, P_n \rangle$ with immediate key delivery can be easily obtained, for example, from any user-to-user passively-secure key-exchange protocol via the following simple transformation:

1. P_1 samples a random key $\mathsf{k} \leftarrow \mathcal{K}_\lambda$.
2. For every $i \in \{2, \ldots, n\}$, P_1 and P_i invoke the user-to-user key-exchange protocol and establish a shared key k_i.
3. For every $i \in \{2, \ldots, n\}$, P_1 uses a CPA-secure symmetric encryption scheme (whose existence is implied by that of any one-way function) to encrypt k using key k_i, and sends the resulting ciphertext to P_i.
4. Each P_i uses k_i from Step 2 to decrypt the received ciphertext, and then outputs the result of the decryption. P_1 outputs k.

It is straightforward to verify that this transformation indeed yields a passively-secure group key-exchange protocol with immediate key delivery: Even if a subset of the parties aborts the execution of the protocol before its completion, the remaining parties all output the key k chosen by P_1.

4.2 Our Construction

Our protocol relies on the following building blocks:

- A group (ℓ, ϵ)-out-of-band message authentication protocol $\langle S, R_1, \ldots, R_{n-1} \rangle$ with immediate message delivery, where $\ell = \ell(\lambda)$ and $\epsilon = \epsilon(\lambda)$ are functions of the security parameter $\lambda \in \mathbb{N}$.
- A passively-secure group key-exchange protocol $\langle P_{\mathsf{KE},1}, \ldots, P_{\mathsf{KE},n} \rangle$ with key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in \mathbb{N}}$ and immediate key delivery. We assume without loss of generality that each party in $\{P_2, \ldots, P_n\}$ sends messages to and receives messages from P_1 only.[10]
- A one-time strongly-unforgeable signature scheme $(\mathsf{KG}, \mathsf{Sign}, \mathsf{Vrfy})$.

Our protocol, which is denoted by $\langle P_1, \ldots, P_n \rangle$ and formally described below, starts by using the underlying out-of-band message authentication protocol for authenticating a verification key for the one-time signature scheme. This verification key is generated by the initiating party (denoted P_1), and its corresponding signing key is then used to sign the transcript of the out-of-band message authentication protocol, as well as the transcript of an execution of the passively-secure key-exchange protocol. The shared key resulting from executing the passively-secure key-exchange protocol is the output of each party, assuming that from this party's point of view the signature verifies correctly and the out-of-band message authentication protocol terminates successfully (i.e., no forgery was detected).

For describing the protocol, we assume for simplicity of presentation that all messages in the protocols $\langle S, R_1, \ldots, R_{n-1} \rangle$ and $\langle P_{\mathsf{KE},1}, \ldots, P_{\mathsf{KE},n} \rangle$ are sent to all participating users (and hence, the transcript of an honest execution of each of the protocols is the same according to the view of all users).

Out-of-Band Authenticated Group Key-Exchange Protocol $\langle P_1, \ldots, P_n \rangle$

Joint input: The security parameter 1^λ.

1. P_1 samples $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KG}(1^\lambda)$ and sends vk to all other users.
2. $P_1, \ldots, P_n$ execute the out-of-band message authentication protocol $\langle S, R_1, \ldots, R_{n-1} \rangle$, where P_1 runs S on input $(1^\lambda, \mathsf{vk})$, and P_i runs R_{i-1} on input 1^λ for every $i \in \{2, \ldots, n\}$. Denote by $\widehat{\mathsf{vk}}_i$ the output of R_{i-1} in this execution.
3. $P_1, \ldots, P_n$ execute the passively-secure key-exchange protocol $\langle P_{\mathsf{KE},1}, \ldots, P_{\mathsf{KE},n} \rangle$, where P_i runs $P_{\mathsf{KE},i}$ on input 1^λ for every $i \in [n]$. Denote by k_i the output of $P_{\mathsf{KE},i}$ in this execution.
4. Denote by trans_i the transcript of Steps 2 and 3 according to the view of P_i. P_1 computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{trans}_1)$ and sends σ to $P_2, \ldots, P_n$.
5. Denote by $\widehat{\sigma}_i$ the signature received by P_i for $i \in \{2, \ldots, n\}$. If $\widehat{\mathsf{vk}}_i \neq \perp$ and $\mathsf{Vrfy}(\widehat{\mathsf{vk}}_i, \mathsf{trans}_i, \widehat{\sigma}_i) = 1$ then P_i outputs k_i, and otherwise P_i outputs $\perp$.

The following theorem – which due to space limitations is proven in the full version of this paper [45] – establishes the correctness and security of our protocol according to Definition 3:

[10] Note that this is the case in the construction from any passively-secure (user-to-user) key-exchange protocol sketched in Section 4.1. Moreover, any passively-secure group key-exchange protocol can be easily compiled into one in which all parties communicate directly solely with P_1, by re-routing all messages through P_1 (i.e., if P_i wishes to send some message to P_j, it sends it to P_1 who then forwards it to P_j).

▶ **Theorem 8.** *Let $\ell = \ell(\lambda), \epsilon = \epsilon(\lambda)$ and $n = n(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. Then $\langle P_1, \ldots, P_n \rangle$ is an (ℓ, ϵ)-out-of-band authenticated group key-exchange protocol with key space $\mathcal{K}$ for groups of size n, assuming that:*

1. *$\langle P_{\mathsf{KE},1}, \ldots, P_{\mathsf{KE},n} \rangle$ is a passively-secure group key-exchange protocol with key space $\mathcal{K}$ and immediate key delivery.*
2. *$(\mathsf{KG}, \mathsf{Sign}, \mathsf{Vrfy})$ is a one-time strongly-unforgeable signature scheme.*
3. *$\langle S, R_1, \ldots, R_{n-1} \rangle$ is a group (ℓ, ϵ)-out-of-band message authentication protocol with immediate message delivery for $n-1$ receivers.*

If, in addition, $\langle S, R_1, \ldots, R_{n-1} \rangle$ is secure when executed by lazy users, then $\langle P_1, \ldots, P_n \rangle$ is secure when executed by lazy users.

Note that the existence of any user-to-user passively-secure key-exchange protocol implies the existence of a one-way function, which in turn implies the existence of a strongly-unforgeable signature scheme, and (as we show in Section 5) of a group (ℓ, ϵ)-out-of-band message authentication protocol with immediate message delivery and $\epsilon(\lambda) \leq 2 \cdot n(\lambda) \cdot (1/2 + o(1))^{\ell(\lambda)}$. In addition, as discussed in Section 4.1, any user-to-user passively-secure key-exchange protocol implies the existence of such a protocol with immediate key delivery. Theorem 6 thus immediately follows as a corollary of Theorem 8.

5 Out-of-Band Message Authentication with Immediate Message Delivery

In this section we construct a group out-of-band message authentication protocol with immediate message delivery based on the existence of any one-way function (instantiating the required building blocks in the random-oracle model leads to a concrete and efficient protocol). We prove the following theorem:

▶ **Theorem 9.** *Let $\ell = \ell(\lambda)$ and $n = n(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$ and assume the existence of one-way functions. Then, there exists a group (ℓ, ϵ)-out-of-band message authentication protocol for $n(\lambda)$ receivers with immediate message delivery, where $\epsilon(\lambda) = 2 \cdot n(\lambda) \cdot (1/2 + o(1))^{\ell(\lambda)}$.*

For a string $s \in \{0,1\}^*$ and an index $i \in [|s|]$, we let s_i (or $(s)_i$) denote the ith bit of s. Let $\mathsf{BitWiseMajority}$ be the operation that on input a set of strings $s^1, \ldots, s^q$ of length ℓ, returns a string s^* whose kth coordinate is the majority among the kth coordinates of $s^1, \ldots, s^q$; i.e., if $\mathsf{BitWiseMajority}(s^1, \ldots, s^q) = s^*$, then for every $k \in [\ell]$, $(s^*)_k = \mathsf{Majority}((s^1)_k, \ldots, (s^q)_k)$. Our protocol, denoted by π, is parameterized by the number of receivers $n = n(\lambda)$ and by a function $T = T(\lambda)$ of the security parameter $\lambda \in \mathbb{N}$. The protocol uses as a building block a statistically-binding concurrent non-malleable commitment scheme Com. As a commitment scheme may be interactive (unless one assumes the random-oracle model), when describing our protocol and referring to a commitment to a certain value, we mean the transcript of the interaction between the committer and the receiver during an execution of the commit phase of the commitment scheme (when the scheme is non-interactive, a commitment is simply a single string sent from the committer to the receiver).

Group Out-of-Band Message Authentication Protocol $\pi = \langle S, R_1 \ldots, R_n \rangle$

Joint input: The security parameter 1^λ.

Phase 0: Initialization

1. Each party initializes a set of aborted receivers, based on her view of the protocol. We denote by $\mathcal{A}_S$ the set initialized by S, and by $\mathcal{A}_i$ the set initialized by each R_i. At the beginning of the execution $\mathcal{A}_S = \mathcal{A}_1 = \cdots = \mathcal{A}_n = \emptyset$.

Phase 1: Commitments for string selection

2. The sender S, on input m, chooses a random string $r_s \leftarrow \{0,1\}^\ell$, and executes n (possibly parallel) executions of Com to commit to the message (m, r_s) to each receiver R_i. Denote the resulting commitments according to the view of S by $c_s^1, \ldots, c_s^n$, and denote the commitment received by each R_i by $\widehat{c_s^i}$. S also appends to the first message it sends each R_i the message m. Denote by $\widehat{m}_i$ the message received by each R_i.
3. Each receiver R_i chooses random ℓ-bit strings $r_{i,1}, \ldots, r_{i,T} \leftarrow \{0,1\}^\ell$, and commits to them to the sender S using T (parallel) executions of Com. For every $i \in [n]$ denote the resulting commitments according to the view of R_i by $c_{i,1}, \ldots, c_{i,T}$, and denote the commitments received by S by $\widehat{c_{i,1}}, \ldots, \widehat{c_{i,T}}$. If some receiver R_i aborts during the commitment protocol, then S updates $\mathcal{A}_S = \mathcal{A}_S \cup \{i\}$.
4. For every $i \in [n]$, S forwards to R_i the commitments $\left\{\left(\widehat{c_{j,1}}, \ldots, \widehat{c_{j,T}}\right)\right\}_{j \in [n] \setminus \{i\}}$ received by her in Step 3 of the protocol, as well as $\mathcal{A}_S$. We denote by $\left\{(\widehat{c_{j,1 \to i}}, \ldots, \widehat{c_{j,T \to i}})\right\}_{j \in [n] \setminus \{i\}}$ and $\widehat{\mathcal{A}_{Si}}$ the forwarded commitments and the aborted set, respectively, as received by R_i. In addition, R_i updates $\mathcal{A}_i = \widehat{\mathcal{A}_{Si}}$.

Phase 2: Gradual decommitments for string selection

5. For $t = 1, \ldots, T$:
   a. For every $i \in [n]$, R_i sends to S a decommitment $d_{i,t}$ of her commitment $c_{i,t}$ from Step 3. Let $\widehat{d_{i,t}}$ denote the decommitment received by S. For every $i \in [n]$ the sender S then checks whether $\widehat{d_{i,t}}$ is a valid decommitment to $\widehat{c_{i,t}}$. If so, let $\widehat{r_{i,t}}$ denote the committed value. If some receiver R_i either sends an invalid decommitment or aborts before sending $d_{i,t}$, then S updates $\mathcal{A}_S = \mathcal{A}_S \cup \{i\}$. For every $i \in \mathcal{A}_S$, S lets $\widehat{r_{i,t}} = 0^\ell$.
   b. For every $i \in [n]$, S forwards R_i the decommitments $(\widehat{d_{j,t}})_{j \in [n] \setminus \{i\}}$, as well as $\mathcal{A}_S$. We let $(\widehat{d_{j,t \to i}})_{j \in [n] \setminus \{i\}}$ and $\widehat{\mathcal{A}_{Si}}$ denote the decommitments and the set received by R_i, respectively. R_i updates $\mathcal{A}_i = \mathcal{A}_i \cup \widehat{\mathcal{A}_{Si}}$. If for some $j \in [n] \setminus (\mathcal{A}_i \cup \{i\})$ it holds that $\widehat{d_{j,t \to i}}$ is not a valid decommitment to $\widehat{c_{j,t \to i}}$ received by R_i is Step 4, then R_i updates $\mathcal{A}_i = \mathcal{A}_i \cup \{j\}$. Otherwise, denote by $(\widehat{r_{j,t \to i}})_{j \in [n] \setminus \{i\}}$ the values obtained by opening the commitments. For every $j \in \mathcal{A}_i$, R_i lets $\widehat{r_{j,t \to i}} = 0^\ell$.
   c. S computes $\sigma_t = \bigoplus_{i \in [n]} \widehat{r_{i,t}}$, and for every $i \in [n]$, R_i computes $\widehat{\sigma}_{i,t} = r_{i,t} \bigoplus_{j \in [n] \setminus \{i\}} \widehat{r_{j,t \to i}}$.
6. For every $i \in [n]$, the sender S sends receiver R_i a decommitment d_s^i to the corresponding commitment from Step 2. Denote by $\widehat{d_s^i}$ the decommitment received by R_i. For every $i \in [n]$ the receiver R_i checks if $\widehat{d_s^i}$ is a valid decommitment to $\widehat{c_s^i}$. If it is, denote the committed value by $(\widehat{m}_i, \widehat{r_s^i})$. If it is not a valid decommitment, then R_i outputs $\perp$ and terminates.

Phase 3: Out-of-band verification

7. S computes $\sigma_R = \mathsf{BitWiseMajority}(\sigma_1, \ldots, \sigma_T)$ and sends $\sigma = r_s \oplus \sigma_R$ over the out-of-band channel. For every $i \in [n]$, R_i computes $\widehat{\sigma_{Ri}} = \mathsf{BitWiseMajority}(\widehat{\sigma}_{i,1}, \ldots, \widehat{\sigma}_{i,T})$, and outputs $\widehat{m}_i$ if $\sigma = \widehat{r_s^i} \oplus \widehat{\sigma_{Ri}}$. Otherwise, R_i outputs $\perp$.

The following theorem captures the security of our protocol, and its proof can be found in the full version of this paper [45] (recall that in Section 1.3 we provided a high-level overview of the proof). Setting $T(\lambda) = (n(\lambda))^2 \cdot \omega(1)$ yields Theorem 9 as an immediate corollary.

▶ **Theorem 10.** *Let $\ell = \ell(\lambda), T = T(\lambda)$ and $n = n(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$, and let Com be a statistically-binding concurrent non-malleable commitment scheme. Then, the protocol π is a group (ℓ, ϵ)-out-of-band message authentication protocol for n receivers with immediate message delivery, where $\epsilon(\lambda) = 2 \cdot n(\lambda) \cdot \left(1/2 + O\left(n(\lambda)/\sqrt{T(\lambda)}\right)\right)^{\ell(\lambda)}$.*

References

1 Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In *Proceedings of the 8th International Conference on Practice and Theory in Public-Key Cryptography*, pages 65–84, 2005.

2 Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The double ratchet: Security notions, proofs, and modularization for the signal protocol. In *Advances in Cryptology – EUROCRYPT '19*, pages 129–158, 2019.

3 Hagit Attiya and Jennifer Welch. *Distributed computing: fundamentals, simulations, and advanced topics*. John Wiley & Sons, 2004.

4 Baruch Awerbuch, Manuel Blum, Benny Chor, Shafi Goldwasser, and Silvio Micali. How to implement Bracha's $O(\log n)$ byzantine agreement algorithm. Unpublished manuscript, 1985.

5 Richard Barnes, Jon Millican, Emad Omara, Katriel Cohn-Gordon, and Raphael Robert. The messaging layer security protocol, 2019. Available at `https://datatracker.ietf.org/doc/draft-ietf-mls-protocol/` (accessed 11-Dec-2019).

6 Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the 30th annual ACM Symposium on Theory of Computing*, pages 419–428, 1998.

7 Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology – EUROCRYPT '00*, pages 139–155, 2000.

8 Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – CRYPTO '93*, pages 232–249, 1993.

9 Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: the three party case. In *Proceedings of the 27th annual ACM Symposium on Theory of Computing*, pages 57–66, 1995.

10 Mihir Bellare, Asha Camper Singh, Joseph Jaeger, Maya Nyayapati, and Igors Stepanovs. Ratcheted encryption and key exchange: The security of messaging. In *Advances in Cryptology – CRYPTO '17*, pages 619–650, 2017.

11 Bluetooth Special Interest Group. Bluetooth core specification v. 5.1, 2019. Available at `https://www.bluetooth.com/specifications/bluetooth-core-specification/` (accessed 11-Dec-2019).

12 Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In *Advances in Cryptology – EUROCRYPT '00*, pages 156–171, 2000.

13 Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Provably authenticated group Diffie-Hellman key exchange - the dynamic case. In *Advances in Cryptology – ASIACRYPT '01*, pages 290–309, 2001.

14 Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In *Advances in Cryptology – EUROCRYPT '02*, pages 321–336, 2002.

15 Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean Jacques Quisquater. Provably authenticated group Diffie-Hellman key exchange. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 255–264, 2001.

16 Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology – EUROCRYPT '01*, pages 453–474, 2001.

17 Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Four-round concurrent non-malleable commitments from one-way functions. In *Advances in Cryptology – CRYPTO '17*, pages 127–157, 2017.

18 Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 364–369, 1986.

19 Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In *Proceedings of the 25th ACM conference on Computer and Communications Security*, pages 1802–1819, 2018.

20 Katriel Cohn-Gordon, Cas J.F̃. Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the Signal messaging protocol. In *Proceedings of the 2nd IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 451–466, 2017.

21 Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

22 Alexis Duque. Deep dive into Bluetooth LE security. *Medium*. Available at `https://medium.com/rtone-iot-security/deep-dive-into-bluetooth-le-security-d2301d640bfc`, 2018.

23 F. Betül Durak and Serge Vaudenay. Bidirectional asynchronous ratcheted key agreement without key-update primitives. In *Advances in Information and Computer Security – IWSEC '19*, pages 343–362, 2019.

24 Dario Fiore, Maria Isabel Gonzalez Vasco, and Claudio Soriente. Partitioned group password-based authenticated key exchange. *The Computer Journal*, 60(12):1912–1922, 2017.

25 Tilman Frosch, Christian Mainka, Christoph Bader, Florian Bergsma, Jörg Schwenk, and Thorsten Holz. How secure is TextSecure? In *Proceedings of the 1st IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 457–472, 2016.

26 Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In *Advances in Cryptology – EUROCRYPT '03*, pages 524–543, 2003.

27 Michael T. Goodrich, Michael Sirivianos, John Solis, Gene Tsudik, and Ersin Uzun. Loud and clear: Human-verifiable authentication based on audio. In *26th IEEE International Conference on Distributed Computing Systems*, page 10, 2006.

28 Vipul Goyal. Constant round non-malleable protocols using one way functions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, pages 695–704, 2011.

29 David Jablon. Strong password-only authenticated key exchange. *ACM SIGCOMM Computer Communication Review*, 26(5):5–26, 1996.

30 Joseph Jaeger and Igors Stepanovs. Optimal channel security against fine-grained state compromise: The safety of messaging. In *Advances in Cryptology – CRYPTO '18*, pages 33–62, 2018.

31 Daniel Jost, Ueli Maurer, and Marta Mularczyk. Efficient ratcheting: Almost-optimal guarantees for secure messaging. In *Advances in Cryptology – EUROCRYPT '19*, pages 159–188, 2019.

32 Ronald Kainda, Ivan Flechais, and AW Roscoe. Usability and security of out-of-band channels in secure device pairing protocols. In *Symposium on usable privacy and security (SOUPS)*, pages 11:1–11:12, 2009.

33 Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *Advances in Cryptology – EUROCRYPT '01*, pages 475–494, 2001.

34 Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. In *Advances in Cryptology – CRYPTO '03*, pages 110–125, 2003.

35 Nadim Kobeissi, Karthikeyan Bhargavan, and Bruno Blanchet. Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach. In *Proceedings of the 2nd IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 435–450, 2017.

36 Brian LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In *International Conference on Provable Security '07*, pages 1–16, 2007.

37 Sampsa Latvala, Mohit Sethi, and Tuomas Aura. Evaluation of out-of-band channels for IoT security. *SN Computer Science*, 1(1):1–18, 2019.

38 Sven Laur and Kaisa Nyberg. Efficient mutual data authentication using manually authenticated strings. In *International Conference on Cryptology and Network Security*, pages 90–107, 2006.

39 Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 705–714, 2011.

40 Yehuda Lindell. Comparison-based key exchange and the security of the numeric comparison mode in bluetooth v2.1. In *CT-RSA '09*, pages 66–83, 2009.

41 Nancy A. Lynch. *Distributed algorithms*. Elsevier, 1996.

42 Rene Mayrhofer and Hans Gellersen. Shake well before use: Authentication based on accelerometer data. In *International Conference on Pervasive Computing*, pages 144–161, 2007.

43 Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter. Seeing-is-believing: using camera phones for human-verifiable authentication. In *IEEE Symposium on Security and Privacy*, pages 110–124, 2005.

44 Moni Naor, Lior Rotem, and Gil Segev. The security of lazy users in out-of-band authentication. In *Proceedings of the 16th Theory of Cryptography Conference*, pages 575–599, 2018.

45 Moni Naor, Lior Rotem, and Gil Segev. Out-of-band authenticated group key exchange: From strong authentication to immediate key delivery. Cryptology ePrint Archive, Report 2019/1458, 2019.

46 Moni Naor, Gil Segev, and Adam Smith. Tight bounds for unconditional authentication protocols in the manual channel and shared key models. In *Advances in Cryptology – CRYPTO'06*, pages 214–231, 2006.

47 Moni Naor, Gil Segev, and Adam D. Smith. Tight bounds for unconditional authentication protocols in the manual channel and shared key models. *IEEE Transactions on Information Theory*, 54(6):2408–2425, 2008.

48 Sylvain Pasini and Serge Vaudenay. An optimal non-interactive message authentication protocol. In *CT-RSA '06*, pages 280–294, 2006.

49 Sylvain Pasini and Serge Vaudenay. SAS-based authenticated key agreement. In *Proceedings on the 9th International Conference on Theory and Practice of Public-Key Cryptography*, pages 395–409, 2006.

50 Trevor Perrin and Moxie Marlinspike. The double ratchet algorithm, 2016. Available at `https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf` (accessed 11-Dec-2019).

51 Bertram Poettering and Paul Rösler. Towards bidirectional ratcheted key exchange. In *Advances in Cryptology – CRYPTO '18*, pages 3–32, 2018.

52 Bertram Poettering and Paul Rösler. Asynchronous ratcheted key exchange. Cryptology ePrint Archive, Report 2018/296, 2018.

53 Ronald L. Rivest and Adi Shamir. How to expose an eavesdropper. *Communications of the ACM*, 27(4):393–395, 1984.

54 Lior Rotem and Gil Segev. Out-of-band authentication in group messaging: Computational, statistical, optimal. In *Advances in Cryptology – CRYPTO '18*, pages 63–89, 2018.

55 Lior Rotem and Gil Segev. Out-of-band authentication in group messaging: Computational, statistical, optimal. Cryptology ePrint Archive, Report 2018/493, 2018.

56 Nitesh Saxena, Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. Secure device pairing based on a visual channel. In *IEEE Symposium on Security and Privacy*, pages 306–313, 2006.

57 Michael Schliep and Nicholas Hopper. End-to-end secure mobile group messaging with conversation integrity and deniability. In *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, pages 55–73, 2019.

58 Victor Shoup. On formal models for secure key exchange. Theory of Cryptography Library (available at `www.shoup.net/papers/skey.pdf`), 1999.

59 Telegram. End-to-end encrypted voice calls – key verification. Available at `https://core.telegram.org/api/end-to-end/voice-calls#key-verification` (accessed 11-Dec-2019).

60 Telegram. End-to-end encryption. Available at `https://core.telegram.org/api/end-to-end` (accessed 11-Dec-2019).

61 Serge Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *Advances in Cryptology – CRYPTO '05*, pages 309–326, 2005.

62 Viber encryption overview. Available at `https://www.viber.com/app/uploads/Viber-Encryption-Overview.pdf`.

63 WhatsApp encryption overview. Available at `https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf`.

Hardness vs. (Very Little) Structure in Cryptography: A Multi-Prover Interactive Proofs Perspective

Gil Segev
School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem, Israel
segev@cs.huji.ac.il

Ido Shahaf
School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem, Israel
ido.shahaf@cs.huji.ac.il

Abstract

The hardness of highly-structured computational problems gives rise to a variety of public-key primitives. On one hand, the structure exhibited by such problems underlies the basic functionality of public-key primitives, but on the other hand it may endanger public-key cryptography in its entirety via potential algorithmic advances. This subtle interplay initiated a fundamental line of research on whether structure is inherently necessary for cryptography, starting with Rudich's early work (PhD Thesis '88) and recently leading to that of Bitansky, Degwekar and Vaikuntanathan (CRYPTO '17).

Identifying the structure of computational problems with their corresponding complexity classes, Bitansky et al. proved that a variety of public-key primitives (e.g., public-key encryption, oblivious transfer and even functional encryption) cannot be used in a black-box manner to construct either any hard language that has NP-verifiers both for the language itself and for its complement, or any hard language (and even promise problem) that has a statistical zero-knowledge proof system – corresponding to hardness in the structured classes $\mathsf{NP} \cap \mathsf{coNP}$ or SZK, respectively, from a black-box perspective.

In this work we prove that the same variety of public-key primitives do not inherently require *even very little structure* in a black-box manner: We prove that they do not imply any hard language that has *multi-prover interactive proof systems* both for the language and for its complement – corresponding to hardness in the class $\mathsf{MIP} \cap \mathsf{coMIP}$ from a black-box perspective. Conceptually, given that $\mathsf{MIP} = \mathsf{NEXP}$, our result rules out languages with very little structure.

Already the cases of languages that have IP or AM proof systems both for the language itself and for its complement, which we rule out as immediate corollaries, lead to intriguing insights. For the case of IP, where our result can be circumvented using non-black-box techniques, we reveal a gap between black-box and non-black-box techniques. For the case of AM, where circumventing our result via non-black-box techniques would be a major development, we both strengthen and unify the proofs of Bitansky et al. for languages that have NP-verifiers both for the language itself and for its complement and for languages that have a statistical zero-knowledge proof system.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases Hardness vs. Structure, Black-box Constructions, Interactive Proofs

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.10

Related Version A full version of the paper is available at `https://eprint.iacr.org/2020/330`.

Funding Supported by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253).
Ido Shahaf: Supported by the Clore Israel Foundation via the Clore Scholars Programme.

1 Introduction

Starting with the revolutionary invention of public-key cryptography [18, 32, 23], the hardness of highly-structured computational problems (e.g., factoring, discrete log, or various lattice-based problems) has given rise to a variety of public-key primitives. On one hand, the structure exhibited by such problems underlies the basic functionality of nearly all such primitives, but on the other hand it may also danger their conjectured hardness. As noted by Barak [5], this *"makes public-key cryptography somewhat of an endangered species that could wiped out by a surprising algorithmic advance"*.

This subtle interplay has led to the long-studied question of whether structure is inherently necessary for certain cryptographic primitives, and most notably for public-key primitives. While there may be different approaches for measuring or quantifying "structure", the main approach taken by the cryptography community over the years relies on computational complexity: Understanding which cryptographic primitives inherently require hardness in "structured" complexity classes such as $\mathsf{NP} \cap \mathsf{coNP}$, TFNP and SZK.

There are only a few known examples of cryptographic primitives that require hardness in such classes. Most notably, one-way permutations imply hardness in $\mathsf{NP} \cap \mathsf{coNP}$ [15], homomorphic encryption and non-interactive computational private-information retrieval imply hardness in SZK [14, 28], and indistinguishability obfuscation implies hardness in $\mathsf{PPAD} \subseteq \mathsf{TFNP}$ unless $\mathsf{NP} \subseteq \mathsf{ioBPP}$ [11, 21, 27].

Within the classic framework of black-box constructions, capturing "natural" cryptographic constructions [26, 31], Rudich [33] showed (based on [13, 25]) that a one-way function cannot be used in black-box manner to construct NP-verifiers for any hard language both for the language itself and for its complement – corresponding to hardness in $\mathsf{NP} \cap \mathsf{coNP}$ from a black-box perspective (we note that the known examples stated above all follow in such a black-box manner).

For several decades no progress has been made in extending Rudich's result to public-key primitives or to other complexity classes. This situation has recently changed dramatically with the work of Bitansky, Degwekar and Vaikuntanathan [10] (see also the refinements in the more recent work of Bitansky and Degwekar [9]): They showed that even indistinguishability obfuscation cannot be used in a black-box manner to construct any hard language that has NP verifiers both for the language itself and for its complement, or any hard language (and even a promise problem) that has a statistical zero-knowledge proof system – corresponding to hardness in $\mathsf{NP} \cap \mathsf{coNP}$ or SZK, respectively, from a black-box perspective. Proving their result within the framework of Asharov and Segev [2, 3] capturing indistinguishability obfuscation for oracle-aided computations, Bitansky et al. in fact proved their result for all primitives that can be based on indistinguishability obfuscation for circuits that access an injective one-way function in a black-box manner. These include, in particular, a variety of public-key primitives including public-key encryption, oblivious transfer and even functional encryption.

Focusing on the classes $\mathsf{NP} \cap \mathsf{coNP}$ and SZK , Bitansky et al. showed that, from a black-box perspective, public-key cryptography does not inherently require highly-structured hardness. However, going back to Barak's concern [5], even less stringent forms of structure may still endanger public-key cryptography in its entirety. This leads to the following fundamental question aiming at substantially refining our understanding of the interplay between hardness and structure:

> Does public-key cryptography inherently require hardness in complexity classes that are "less structured" than $\mathsf{NP} \cap \mathsf{coNP}$ or SZK?

1.1 Our Contributions

In this work we show that a wide variety of public-key primitives do not inherently require even very little structure in a black-box manner. Specifically, we prove that such primitives do not naturally imply hard languages that have *multi-prover interactive proof systems* (MIP) [8] both for the language and for its complement.

Conceptually, given that MIP = NEXP [4], our result considers languages with very little structure. Already the cases of languages that have IP or AM proof systems both for the language itself and for its complement, which we obtain as immediate corollaries, lead to intriguing insights. For the case of IP, where our result can be circumvented using non-black-box techniques, we reveal a gap between black-box and non-black-box techniques (as we discuss below). For the case of AM, where circumventing our result via non-black-box techniques would be a major development, we both strengthen and unify the proofs of Bitansky et al. for languages that have NP-verifiers both for the language itself and for its complement and for languages that have a statistical zero-knowledge proof system (since $\mathsf{NP} \subseteq \mathsf{AM}$ by definition, and since $\mathsf{SZK} \subseteq \mathsf{AM} \cap \mathsf{coAM}$ in a black-box manner [19, 1]).[1]

The following is an informal statement of our main result. We refer the reader to Section 1.2 for an overview of our result, and to Sections 3 and 4 for a formal definition of the class of constructions to which our result applies and for a formal theorem statement, respectively.

▶ **Theorem 1** (Informal). *There is no fully black-box construction of a pair of multi-prover interactive proof systems,* Π *and* $\overline{\Pi}$*, corresponding to a worst-case hard language* L *and to its complement* $\overline{L}$*, respectively, from an injective one-way function* f *and an indistinguishability obfuscator for the class of all oracle-aided circuits* C^f*.*

Note that as our result rules out constructions of languages that are *worst-case hard*, then it rules out in particular constructions of languages that are *average-case hard*.

Black-box vs. non-black-box constructions. Our result might seem too strong and somewhat contradicting to the fact that any one-way function implies a hard (even on average) language in $\mathsf{NP} \subseteq \mathsf{IP}$ in a *black-box manner*. Given that IP is closed under complement [30, 36], then

$$\mathsf{NP} \subseteq \mathsf{IP} \cap \mathsf{coIP} \subseteq \mathsf{MIP} \cap \mathsf{coMIP}.$$

In particular, any one-way function implies a hard language that has IP proof systems both for the language itself and for its complement, which seemingly contradicts our result. However, this sequence of containments cannot be established via relativizing reductions, and thus there is in fact no contradiction (note that any black-box reduction relativizes [31]), but rather a gap between black-box and non-black-box techniques. Specifically, Chang et al. [16] showed that there exists an oracle Γ relative to which $\mathsf{NP}^{\Gamma} \not\subseteq \mathsf{coIP}^{\Gamma}$, and in particular IP is *not* closed under complement with respect to relativizing reductions. Still, as mentioned above, our impossibility result already applies to $\mathsf{AM} \cap \mathsf{coAM}$, for which circumventing our result via non-black-box techniques would be a major development. We discuss this in much more detail in Section 1.2 in the context of black-box representations of complexity classes.

[1] We note that the result of Bitansky et al. for SZK holds not only for languages but in fact also for promise problems. This, however, cannot be covered by our result since already a hard promise problem that has NP verifiers both for its "YES" instances and for its "NO" instances can be constructed based on any one-way function in a black-box manner.

Implications to public-key cryptography. Similarly to Bitansky et al. [10] we prove our result within the framework of Asharov and Segev [2, 3], capturing indistinguishability obfuscation for oracle-aided circuits. Indistinguishability obfuscation for such circuits suffices for realizing a variety of public-key primitives (e.g., public-key encryption, oblivious transfer and even functional encryption) in a fully black-box manner [34, 37, 2], and therefore as a corollary we obtain that there is no construction of the above form based on any of these primitives.

We strongly emphasize that our result is unconditional, and in particular does not depend on whether or not indistinguishability obfuscation actually exists. Even if it does not exist in the actual world, then within the framework of Asharov and Segev it does exist information theoretically, and it implies the above variety of public-key primitives to which our result applies (once again, in an unconditional manner).

1.2 Overview of Our Approach

In this section we provide a high-level overview of the framework in which we prove our impossibility result, and then describe the main ideas and challenges underlying our proof.

Black-box constructions. Our goal is to prove a statement along the lines of "a cryptographic primitive $\mathcal{P}$ does not naturally imply a hard language in a complexity class $\mathcal{C}$". However, it is not clear how to prove such a statement in an unconditional manner, as it may be the case that the class $\mathcal{C}$ (e.g., $\mathsf{NP} \cap \mathsf{coNP}$ as discussed by Bitansky et al. [10]) does not contain hard languages. One possible approach is to prove a result that is conditioned on a specific assumption, but then it may be the case that the assumption itself already rules out the existence of hard languages in the class $\mathcal{C}$. Obtaining substantial insight using such an approach requires a deep understanding of the interplay between the primitive $\mathcal{P}$, the complexity class $\mathcal{C}$ and the additional assumption – which is somewhat rare when considering cryptographic primitives and assumptions.

Faced with such difficulties, the cryptography community has relied over the years on the framework of black-box constructions [26, 31] for proving impossibility results for "natural" construction techniques. In our context, a fully black-box construction of a hard language $L \in \mathcal{C}$ based on a cryptographic primitive $\mathcal{P}$ consists of two ingredients. The first ingredient is a "construction" of a language $L^{\mathcal{P}}$ that completely ignores the internal implementation of $\mathcal{P}$ and only requires black-box access to any given implementation of $\mathcal{P}$. Here, the notion of a "construction" depends on the specific complexity class $\mathcal{C}$. For example, in a natural black-box interpretation of $\mathsf{NP} \cap \mathsf{coNP}$, Rudich [33] and Bitansky et al. [10] considered as a construction a pair of oracle-aided NP-verifiers, V and $\overline{V}$, for the language itself and for its complement, respectively, where the verifiers have black-box access to the primitive $\mathcal{P}$. That is, for any oracle realizing $\mathcal{P}$, the two verifiers must be valid in the sense that for any instance $x \in \{0,1\}^*$ either there exists a "yes" witness for $V^{\mathcal{P}}$ and there do not exist any "no" witnesses for $\overline{V}^{\mathcal{P}}$ (i.e., $x \in L^{\mathcal{P}}$), or there exists a "no" witness for $\overline{V}^{\mathcal{P}}$ and there do not exist any "yes" witnesses for $V^{\mathcal{P}}$ (i.e., $x \notin L^{\mathcal{P}}$) – but never both. The second ingredient, is a black-box proof of hardness, showing that for any implementation of the primitive $\mathcal{P}$, any algorithm that decides the language $L^{\mathcal{P}}$ can be efficiently used in a black-box manner for breaking the security of the given implementation of $\mathcal{P}$.

At this point we would like to already emphasize that a "black-box representation" of a complexity class is in fact not unique, and that different representations are not always equivalent from a black-box perspective. For example, a natural black-box representation for the class $\mathsf{IP} \cap \mathsf{coIP}$ relative to a given primitive $\mathcal{P}$ is to consider all languages that have

interactive proof systems both for the language itself and for its complement, where the two proof systems access $\mathcal{P}$ in a black-box manner. However, since IP is closed under complement [30, 36] then $\mathsf{IP} \cap \mathsf{coIP} = \mathsf{IP}$ and therefore an additional representation is to consider all languages that have interactive proof systems for the language itself (without considering its complement) where the proof system accesses $\mathcal{P}$ in a black-box manner. As discussed in Section 1.1, these two representations are not equivalent from a black-box perspective since IP is not closed under complement with respect to relativizing reductions.

The structure of our proof. Following Bitansky et al. [10] we prove our result within the framework of Asharov and Segev [2] for capturing black-box constructions based on indistinguishability obfuscation, utilizing the latter as a "central hub" for deriving impossibility results for a variety of public-key primitives. As observed by Asharov and Segev, although constructions that are based on indistinguishability obfuscation are almost always *non-black-box*, most of their non-black-box techniques have essentially the same flavor: The obfuscator itself is used in a black-box manner and applied to circuits that can be constructed in a fully black-box manner from a low-level primitive, such as a one-way function. Thus, even though the obfuscator requires concrete implementations of such circuits, by introducing the stronger primitive of an indistinguishability obfuscator for *oracle-aided* circuits (see Section 2), Asharov and Segev showed that such non-black-box techniques in fact directly translate into black-box ones. These include, in particular, non-black-box techniques such as the punctured programming approach of Sahai and Waters [34] and Waters [37] leading to the construction of a variety of public-key primitive. Relying on the transitivity of black-box reductions, this enables to rule out black-box constructions based on all of these primitives by focusing only on indistinguishability obfuscation for oracle-aided circuits and one-way functions.

In order to prove our impossibility result within this framework, we present a distribution over oracles Γ relative to which we prove the following two properties:

- Relative to a random instance of Γ there exist an injective one-way function f and an indistinguishability obfuscator $i\mathcal{O}$ for the class of all oracle-aided circuits C^f.
- Relative to any instance of Γ, we can efficiently decide in the worst case any language that has multi-prover interactive proof systems, $\Pi^{f,i\mathcal{O}}$ and $\overline{\Pi}^{f,i\mathcal{O}}$, for the language itself and for its complement, respectively.[2]

Our oracle Γ is a pair of the form $(\Psi, \mathsf{Decide}^{\Psi})$, where Ψ is based on the oracle of Asharov and Segev that realizes a one-way function and an indistinguishability obfuscator, and Decide^{Ψ} is a generalization of the "decision oracle" introduced by Bitansky et al. for deciding languages that rely on Ψ in a black-box manner (more specifically, whose black-box representation as discussed above relies on Ψ). In the work of Bitansky et al. the decision oracle is defined in a manner that allows to easily decide any language L^{Ψ} that has NP-verifiers, V^{Ψ} and $\overline{V}^{\Psi}$, for the language itself and for its complement, and the main technical challenge underlying their work is proving that Ψ realizes a one-way function and an indistinguishability obfuscator relative to the decision oracle.

Our decision oracle is a natural generalization that allows to easily decide any language L^{Ψ} that has multi-prover proof systems, Π^{Ψ} and $\overline{\Pi}^{\Psi}$, for the language itself and for its complement. This decision oracle seems much more powerful than that of Bitansky et al.

[2] In fact, as discussed below we allow the honest provers to depend on the one-way function and the obfuscator in an arbitrary non-black-box manner, and only require that the verifiers are constructed in a black-box manner (this makes our result stronger).

as it decides a significantly larger class of languages, and our technical effort is devoted to proving that the oracle Ψ still realizes a one-way function and an indistinguishability obfuscator even relative to our generalized decision oracle.

In what follows we describe the decision oracle of Bitansky et al. (to which we refer as the BDV decision oracle) and discuss its key property that underlies their approach. Then, we describe our generalized oracle, relative to which this key property no longer seems to hold, and then describe our the main ideas underlying our proof.

The BDV decision oracle. For any oracle Ψ, taken from an appropriate family $\mathfrak{S}$ of oracles, the BDV decision oracle $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$ takes as input a triplet $(V, \overline{V}, x)$, where V and $\overline{V}$ are oracle-aided circuits. The oracle first checks whether or not the pair $(V, \overline{V})$ indeed consists of valid NP-verifiers for a language and for its complement in the standard black-box sense discussed above. That is, checks whether or not for any $\Psi' \in \mathfrak{S}$ and $x' \in \{0,1\}^n$ exactly one out of the following two cases holds: (1) There exists a "yes" witness w' such that $V^{\Psi'}(x', w') = 1$ and there do not exist any "no" witnesses w' such that $\overline{V}^{\Psi'}(x', w') = 1$; (2) there exists a "no" witness w' such that $\overline{V}^{\Psi'}(x', w') = 1$ and there do not exist any "yes" witnesses w' such that $V^{\Psi}(x', w') = 1$ (note that the witnesses are allowed to depend on Ψ'). If $(V, \overline{V})$ is not valid in this sense, then the oracle outputs $\perp$. If $(V, \overline{V})$ is valid, then the oracle outputs 1 if $x \in L^{\Psi}$ and 0 otherwise, where L^{Ψ} is the language defined by $(V^{\Psi}, \overline{V}^{\Psi})$.

Then, any language that has oracle-aided NP-verifiers both for the language itself and for its complement with respect to any $\Psi \in \mathfrak{S}$, can be easily decided in the worst case by an algorithm that issues a single query to the BDV decision oracle. The main challenge in the work of Bitansky et al. was in showing that a random instance of Ψ that is sampled from the family $\mathfrak{S}$ of oracles introduced by Asharov and Segev (or from any other appropriate family) realizes a one-way function and an indistinguishability obfuscator even relative to $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$.

The existence of small critical sets. The key property underlying the proof of Bitansky et al. is the following observation on the existence of "small critical sets". Fix an oracle $\Psi \in \mathfrak{S}$ and let $(V, \overline{V}, x)$ be a query to their decision oracle such that the pair $(V, \overline{V})$ is valid in the above sense, and V and $\overline{V}$ issue at most q oracle queries. Then, there exists a "critical set" of at most q queries, such that for any oracle $\Psi' \in \mathfrak{S}$ that agrees with Ψ on the outputs of all queries from the critical set it holds that $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}(V, \overline{V}, x) = \mathsf{Decide}^{\Psi'}_{\mathfrak{S}}(V, \overline{V}, x)$.

The existence of such a small critical set follows from the $\mathsf{NP} \cap \mathsf{coNP}$ structure of the pair $(V, \overline{V})$. Specifically, assume without loss of generality that $x \in L^{\Psi}$, and let w be a witness such that $V^{\Psi}(x, w) = 1$. Define the set of critical queries as all Ψ-queries that are issued in the computation $V^{\Psi}(x, w)$, and let Ψ' by any oracle that agrees with Ψ on this set. Then clearly $V^{\Psi'}(x, w) = V^{\Psi}(x, w) = 1$, and the validity of the pair $(V, \overline{V})$ guarantees that there is no witness $\tilde{w}$ such that $\overline{V}^{\Psi'}(x, \tilde{w}) = 1$. Thus, $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}(V, \overline{V}, x) = \mathsf{Decide}^{\Psi'}_{\mathfrak{S}}(V, \overline{V}, x) = 1$.

Relying on this key property, Bitansky et al. proved that Ψ realizes a one-way function and an indistinguishability obfuscator relative to their decision oracle via an elegant sequence of hybrids in each case. Specifically, in each sequence the first experiment is the actual security experiment of the one-way function or the indistinguishability obfuscator, the last experiment is one in which no algorithm can achieve any advantage, and the transition between each consecutive pair of experiment is enabled by this key property (or via standard arguments).

Representing $\mathsf{MIP} \cap \mathsf{coMIP}$ in a black-box manner. In order to describe our approach, we first need to describe our black-box representation of languages in the complexity class $\mathsf{MIP} \cap \mathsf{coMIP}$. Naturally generalizing the approach of Rudich and Bitansky et al. for $\mathsf{NP} \cap \mathsf{coNP}$,

we consider pairs of polynomial-time oracle-aided MIP-verifiers, V and $\overline{V}$, for the language itself and for its complement, respectively, subject to a similar validity requirement of their black-box flavor: For any oracle Ψ taken from an appropriate family $\mathfrak{S}$ of oracles, there should exist a language L^{Ψ} such that the following two conditions are satisfied[3]:

- For every $x \in L^{\Psi}$ there exist computationally-unbounded provers $P_1, \ldots, P_N$ such that[4]

$$\Pr_{r \leftarrow \{0,1\}^{\mathsf{poly}(|x|)}} \left[\left\langle V^{\Psi}(x;r), P_1, \ldots, P_N \right\rangle = 1 \right] \geq 2/3 \,,$$

  and for every computationally-unbounded provers $\overline{P}_1, \ldots, \overline{P}_N$ it holds that

$$\Pr_{r \leftarrow \{0,1\}^{\mathsf{poly}(|x|)}} \left[\left\langle \overline{V}^{\Psi}(x;r), \overline{P}_1, \ldots, \overline{P}_N \right\rangle = 1 \right] \leq 1/3 \,.$$

- For every $x \notin L^{\Gamma}$ there exist computationally-unbounded provers $\overline{P}_1, \ldots, \overline{P}_N$ such that

$$\Pr_{r \leftarrow \{0,1\}^{\mathsf{poly}(|x|)}} \left[\left\langle \overline{V}^{\Psi}(x;r), \overline{P}_1, \ldots, \overline{P}_N \right\rangle = 1 \right] \geq 2/3 \,,$$

  and for every computationally-unbounded provers $P_1, \ldots, P_N$ it holds that

$$\Pr_{r \leftarrow \{0,1\}^{\mathsf{poly}(|x|)}} \left[\left\langle V^{\Psi}(x;r), P_1, \ldots, P_N \right\rangle = 1 \right] \leq 1/3 \,.$$

Note that instead of considering oracle-aided MIP proof systems we consider oracle-aided MIP verifiers, and allow the honest provers to depend on any given oracle in an arbitrary non-black-box manner (thus our result rules out, in particular, oracle-aided proof systems). We refer the reader to Section 3 where we formally describe the proof systems we consider and the class of constructions to which our result applies.

Our generalized decision oracle. For any oracle $\Psi \in \mathfrak{S}$ our generalized decision oracle $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$ takes as input a triplet $(V, \overline{V}, x)$, where V and $\overline{V}$ are oracle-aided MIP-verifiers and $x \in \{0,1\}^n$. The oracle first checks whether or not the pair $(V, \overline{V})$ indeed consists of MIP-verifiers for a language and for its complement with respect to all oracles in $\mathfrak{S}$ as discussed above. If $(V, \overline{V})$ is not valid in this sense, then the oracle outputs $\perp$. If $(V, \overline{V})$ is valid, then the oracle outputs 1 if $x \in L^{\Psi}$ and 0 otherwise, where L^{Ψ} is the language defined by $(V^{\Psi}, \overline{V}^{\Psi})$.

At this point, we would ideally like to follow the approach of Bitansky et al. in proving that Ψ realizes a one-way function and an indistinguishability obfuscator relative to our generalized decision oracle. Recall that their proof consists of a sequence of hybrid experiments, where the transition between each consecutive pair of experiments is enabled by the existence of a small set of critical queries. Specifically, in each transition they modify Ψ on some set of queries into an oracle Ψ', and argue that unless these queries fall into the small critical set then the decision oracle behaves exactly the same.

---

[3] For an oracle Ψ, an instance x, a string r, a polynomial-time oracle-aided verifier V, and provers $P_1, \ldots, P_N$ we denote by $\left\langle V^{\Psi}(x;r), P_1, \ldots, P_N \right\rangle$ the output of V with oracle access to Ψ on input x and randomness r in the multi-prover execution with $P_1, \ldots, P_N$. Note that whenever the provers are computationally unbounded we can assume that they are deterministic.

[4] It is usually assumed that the same provers are used for every $x \in \{0,1\}^n$, and that they obtain x as input. However, since the provers are computationally unbounded, our definition is clearly equivalent and easier to work with for our purposes.

Are there small and useful critical query sets? Fix an oracle $\Psi \in \mathfrak{S}$, and fix a query $(V, \overline{V}, x)$ to our generalized decision oracle, where V and $\overline{V}$ are valid MIP-verifiers in the above sense. Unlike the case of NP-verifiers, when considering MIP-verifiers then at a first glance there does not seem to be a small set of queries that completely determines whether or not $x \in L^{\Psi}$. Specifically, assuming for the current discussion that $x \in L^{\Psi}$, in the case of NP-verifiers this is completely determined by the polynomial number of queries to the oracle Ψ in the execution $V^{\Psi}(x, w)$ where w is any specific witness (say, the lexicographically first such witness). However, in the case of MIP-verifiers, we are guaranteed that there exist provers $P_1, \ldots, P_N$ that lead the MIP-verifier $V^{\Psi}(x; r)$ to accept with probability at least $2/3$ over the randomness $r \leftarrow \{0,1\}^{\mathsf{poly}(|x|)}$ of the verifier – but this guarantee involves potentially exponentially-many executions and thus exponentially-many queries to the oracle Ψ. It may even be the case that any oracle Ψ' that agrees with Ψ on all of these queries, is in fact $\Psi' = \Psi$, and this is not very useful for the purpose of transitioning between two hybrid experiments.

Nevertheless, let us consider an oracle Ψ' that differs from Ψ on a *single* query z, and now suppose that suddenly $x \notin L^{\Psi'}$ although we started with $x \in L^{\Psi}$. Thus, no provers can now lead $V^{\Psi'}(x; r)$ to accept with probability larger than $1/3$ over the randomness $r \leftarrow \{0,1\}^{\mathsf{poly}(|x|)}$, and in particular this holds for the above provers $P_1, \ldots, P_N$ that led $V^{\Psi}(x; r)$ to accept with probability at least $2/3$. The only way that $V^{\Psi'}(x; r)$ can differ from $V^{\Psi}(x; r)$ in an execution with the same $P_1, \ldots, P_N$ is by having $V^{\Psi}(x; r)$ query Ψ on z – and we can deduce that with probability at least $1/3$ over the choice of $r \leftarrow \{0,1\}^{\mathsf{poly}(|x|)}$ it holds that $V^{\Psi}(x; r)$ queries Ψ on z when interacting with $P_1, \ldots, P_N$.

Therefore, it is quite tempting to fix a distance parameter $d \geq 1$, and then for an oracle $\Psi \in \mathfrak{S}$ and a query $(V, \overline{V}, x)$ such that $x \in L^{\Psi}$ to define the following "d-influential set" of queries: Let $P_1, \ldots, P_N$ be provers that lead $V^{\Psi}(x; r)$ to accept with probability at least $2/3$, then the d-influential set consists of all queries that $V^{\Psi}(x; r)$ issues to Ψ in at least a $1/(3d)$-fraction of these executions. Then, if V issues at most q queries in each execution, then this set consists of at most $3qd$ queries. Moreover, for any oracle Ψ' that differs from Ψ on at most d queries, and these queries are not in the d-influential set, then it must hold that $x \in L^{\Psi'}$ (the probability that $V(x; r)$ accepts cannot drop from $2/3$ to $1/3$ when switching from Ψ to Ψ' since they differ on at most d queries and each of these queries cannot affect more than a $1/(3d)$-fraction of the executions).

From influential queries to influential labels. Unfortunately, this observation is still insufficient for our purposes. In the proof of Bitansky et al. the number of differences between Ψ and Ψ' is irrelevant as long as these differences are not in the critical set. However, in our case more than d differences outside of the d-influential set may still cause the verifier's acceptance probability to drop from $2/3$ to below $1/3$.

Although our proof considers oracles Ψ and Ψ' that may differ on an exponential number of queries, we tailor the specific structure of our obfuscator in a way that enables us to "group together" related queries: We introduce labeling functions (depending on the specific structure of our oracles) that assign a label to each query to the oracle Ψ, where different queries may share the same label. We show that it now suffices to focus on the small number $d \leq 3$ of labels that result from the potentially-exponential number of differences between the oracles Ψ and Ψ'.

Specifically, we prove that for any Ψ and for any query $(V, \overline{V}, x)$ to our generalized decision oracle there exists a small set $\mathbf{I}$ of "d-influential labels" such that any changes to Ψ involving at most d labels outside of $\mathbf{I}$ do not change the answer to the query. That is,

let $\Psi' \in \mathfrak{S}$ be any oracle for which there exists a set $\mathcal{D} \subseteq \mathcal{X} \setminus \mathbf{I}$ of at most d labels such that if $\Psi'(\alpha) \neq \Psi(\alpha)$ then $\mathsf{lab}(\alpha) \in \mathcal{D}$, where $\mathcal{X}$ is the set of all possible labels and lab is a labeling function. Then, it holds that $\mathsf{Decide}_{\mathfrak{S}}^{\Psi'}(V, \overline{V}, x) = \mathsf{Decide}_{\mathfrak{S}}^{\Psi}(V, \overline{V}, x)$. This is a simplified description of the key property on which we rely in order to prove that a random instance of Ψ realizes a one-way function and an indistinguishability obfuscator relative to our generalized decision oracle, and we refer the reader to Section 4 for the proof of our impossibility result.

1.3 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we introduce some standard notation as well as the cryptographic primitives under consideration in this paper. In Section 3 we define the class of constructions to which our impossibility result applies, and in Section 4 we formally state and prove Theorem 1.

2 Preliminaries

In this section we present the notation and basic definitions that are used in this work. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. For every $n \in \mathbb{N}$ and $m \geq n$ we denote by $\mathsf{InjFunc}_n^m$ the set of all injective functions $f : \{0,1\}^n \to \{0,1\}^m$.

Oracle-aided languages and complexity classes. For a language $L \subseteq \{0,1\}^*$, we let $\chi_L : \{0,1\}^* \to \{0,1\}$ denote the characteristic function of L, that is, $\chi_L(x) = 1$ if and only if $x \in L$. A deterministic algorithm A decides a language L if for every $x \in \{0,1\}^*$ it holds that $A(x) = \chi_L(x)$.

We consider the standard notions of languages and complexity classes when naturally generalized to oracle-aided computations. In particular, an oracle-aided language L defines a set $L^\Gamma \subseteq \{0,1\}^*$ for any possible oracle $\Gamma : \{0,1\}^* \to \{0,1\}^*$. Our definitions throughout the paper follow the standard approach that was introduced in the classic complexity-theory literature for proving separations between complexity classes by considering type-2 languages and complexity classes (see, for example, [7, 17] and the references therein).

Indistinguishability obfuscation for oracle-aided circuits. We consider the standard notion of indistinguishability obfuscation [6, 20] when naturally generalized to oracle-aided circuits (i.e., circuits that may contain oracle gates in addition to standard gates) [2, 3]. We first define the notion of functional equivalence relative to a specific function (provided as an oracle), and then we define the notion of an indistinguishability obfuscation for a class of oracle-aided circuits. In what follows, when considering a class $\mathcal{C} = \{\mathcal{C}_n\}_{n \in \mathbb{N}}$ of oracle-aided circuits, we assume that each $\mathcal{C}_n$ consists of circuits of size at most n.

▶ **Definition 2.** *Let C_0 and C_1 be two oracle-aided circuits, and let f be a function. We say that C_0 and C_1 are functionally equivalent relative to f, denoted $C_0^f \equiv C_1^f$, if for any input x it holds that $C_0^f(x) = C_1^f(x)$.*

▶ **Definition 3.** *A probabilistic polynomial-time oracle-aided algorithm $i\mathcal{O}$ is an indistinguishability obfuscator relative to an oracle Γ for a class $\mathcal{C} = \{\mathcal{C}_n\}_{n \in \mathbb{N}}$ of oracle-aided circuits if the following conditions are satisfied:*

- ***Functionality.*** *For all $n \in \mathbb{N}$ and for all $C \in \mathcal{C}_n$ it holds that*

$$\Pr\left[C^\Gamma \equiv \widehat{C}^\Gamma : \widehat{C} \leftarrow i\mathcal{O}^\Gamma(1^n, C)\right] = 1.$$

- ***Indistinguishability.*** *For any probabilistic polynomial-time oracle-aided distinguisher $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\nu(\cdot)$ such that*

$$\mathsf{Adv}^{i\mathcal{O}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n) \stackrel{def}{=} \left|\Pr\left[\mathsf{Exp}^{i\mathcal{O}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n) = 1\right] - \frac{1}{2}\right| \leq \nu(n)$$

for all sufficiently large $n \in \mathbb{N}$, where the random variable $\mathsf{Exp}^{i\mathcal{O}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n)$ is defined via the following experiment:

1. $b \leftarrow \{0,1\}$.
2. $(C_0, C_1, \mathsf{state}) \leftarrow \mathcal{A}_1^\Gamma(1^n)$ *where* $C_0, C_1 \in \mathcal{C}_n$ *and* $C_0^\Gamma \equiv C_1^\Gamma$.
3. $\widehat{C} \leftarrow i\mathcal{O}^\Gamma(1^n, C_b)$.
4. $b' \leftarrow \mathcal{A}_2^\Gamma(\mathsf{state}, \widehat{C})$.
5. *If $b' = b$ then output 1, and otherwise output 0.*

For simplicity, note that whenever the algorithm $\mathcal{A}_1$ is deterministic there is in fact no need for $\mathcal{A}_1$ to transfer any state information state to $\mathcal{A}_2$ as the state can be reconstructed if needed by invoking $\mathcal{A}_1$. Looking ahead, in this paper we consider computationally-unbounded algorithms (i.e., we limit their query complexity but we do not limit their internal computation), and such algorithms can be assumed without loss of generality to be deterministic.

3 The Class of Constructions

The proof systems we consider in this paper can be formalized in a variety of seemingly equivalent manners, and here we choose a specific definition that we find to simplify the proof of our impossibility result:

▶ **Definition 4.** *For functions $V, P : \{0,1\}^* \to \{0,1\}^*$, an integer $k \geq 0$ and a string $s \in \{0,1\}^*$, we denote by $\langle V(s), P\rangle_k$ the output of the following computation:*

- *Let $m_0 = P(V(s, 0))$.*
- *For $1 \leq i < k$, let $m_i = P(V(s, i, m_0, \ldots, m_{i-1}))$.*
- *Output $V(s, k, m_0, \ldots, m_{k-1}) \in \{0,1\}$.*

That is, we consider a sequential process that is executed by two parties, a verifier V that is given as input a string s, and a prover P that is not given any input. The process consists of k rounds, where in each round the verifier sends the prover a message that is computed as a function of its input s, the index i of the current round, and the prover's previous responses $m_0, \ldots, m_{i-1}$. In turn, the prover replies with a response m_i, and following these k steps the verifier outputs a bit indicating acceptance or rejection.

A crucial property to notice is that the prover's response, m_i, in each step is a function of the verifier's ith message only, and not of the entire transcript which includes all of the verifier's previous messages as well (i.e., the prover is "memoryless"). A verifier may potentially include the entire transcript in each message, and then the definition would collapse to the class IP of languages that have an interactive proof system [24].

In general, however, a verifier need not send the entire transcript in each message, and this enables us to capture the class MIP of languages that have a multi-prover interactive proof system [8]. Specifically, any such proof system $\langle V, P_1, \ldots, P_N\rangle$ in which each prover sends at

most v messages can be transformed in a black-box manner into a proof system $\langle V, P\rangle_k$ of the above form with $k = v \cdot N$. This can be done, for example, by defining $P(i,\cdot) = P_i(\cdot)$ for every $i \in [T]$ (with P maintaining the local state of each prover if needed), and having the verifier include in each message the index of the prover to which this message is sent together with the entire transcript that this specific prover has seen so far. Although we have not yet defined the completeness and soundness properties for the above proof systems (these are defined as part of the following definition), we already note that this transformation naturally preserves them.

As discussed in Section 1.2, instead of considering oracle-aided MIP proof systems we consider oracle-aided MIP verifiers, and allow the honest provers to depend on any given oracle in an arbitrary (i.e., non-black-box) manner (thus our result rules out, in particular, oracle-aided proof systems). This is captured via the following definition:

▶ **Definition 5.** *A pair $\left(V, \overline{V}\right)$ of oracle-aided polynomial-time algorithms, together with polynomials $\ell_r(\cdot)$ and $k(\cdot)$, define a* (MIP, coMIP) protocol pair relative to an oracle $\Psi : \{0,1\}^* \to \{0,1\}^*$ *if there exists a language $L^\Psi \subseteq \{0,1\}^*$ and such that:*

- *For every $x \in L^\Psi$ there exists a function $P : \{0,1\}^* \to \{0,1\}$ such that*

$$\Pr_{r \leftarrow \{0,1\}^{\ell_r(|x|)}}\left[\left\langle V^\Psi(x,r), P\right\rangle_{k(|x|)} = 1\right] \geq 2/3 \ ,$$

  and for every function $\overline{P} : \{0,1\}^* \to \{0,1\}$ it holds that

$$\Pr_{r \leftarrow \{0,1\}^{\ell_r(|x|)}}\left[\left\langle \overline{V}^\Psi(x,r), \overline{P}\right\rangle_{k(|x|)} = 1\right] \leq 1/3 \ .$$

- *For every $x \notin L^\Psi$ there exists a function $\overline{P} : \{0,1\}^* \to \{0,1\}$ such that*

$$\Pr_{r \leftarrow \{0,1\}^{\ell_r(|x|)}}\left[\left\langle \overline{V}^\Psi(x,r), \overline{P}\right\rangle_{k(|x|)} = 1\right] \geq 2/3 \ ,$$

  and for every function $P : \{0,1\}^* \to \{0,1\}$ it holds that

$$\Pr_{r \leftarrow \{0,1\}^{\ell_r(|x|)}}\left[\left\langle V^\Psi(x,r), P\right\rangle_{k(|x|)} = 1\right] \leq 1/3 \ .$$

Note that the above definition considers provers that output only a single bit in each step. This is just for syntactical reasons, making sure that the verifier runs in polynomial-time with respect to the length of the input x. For example, if the prover was allowed to be a length-doubling function, then after $|x|$ rounds this would allow a polynomial-time verifier to run in time that is exponential in the length of $|x|$. There are naturally various ways in which this technical issue can be handled (e.g., providing the verifier with oracle access to the prover instead of direct communication), clearly without having any effect on our result.

The following definition is based on those of [2, 3, 10] (which, in turn, are motivated by [29, 22, 31]), and captures the class of construction that we consider in this paper. We remind the reader that two oracle-aided circuits, C_0 and C_1, are functionally equivalent relative to a function f, denoted $C_0^f \equiv C_1^f$, if for any input x it holds that $C_0^f(x) = C_1^f(x)$ (see Definition 2).

▶ **Definition 6.** *A fully black-box construction of a worst-case hard* (MIP, coMIP) *protocol pair from an injective one-way function f and an indistinguishability obfuscator for the class $\mathcal{C}$ of all oracle-aided circuits C^f, consists of a pair of oracle-aided polynomial-time algorithms $(V, \overline{V})$, polynomials $\ell_r(\cdot)$ and $k(\cdot)$, an oracle-aided polynomial-time algorithm M, and "security loss" functions $\epsilon_{M,1}(\cdot)$ and $\epsilon_{M,2}(\cdot)$, such that the following conditions hold:*

- ***Correctness:*** *For every ensemble* $f = \{f_n : \{0,1\}^n \to \{0,1\}^{n+1}\}_{n\in\mathbb{N}}$ *of injective functions, and for any function* $i\mathcal{O}$ *such that* $i\mathcal{O}(C;r)^f \equiv C^f$ *for any circuit* C *and* $r \in \{0,1\}^*$*, the pair* $(V,\overline{V})$*, together with the polynomials* $\ell_r(\cdot)$ *and* $k(\cdot)$*, define an* (MIP, coMIP) *protocol pair (with a corresponding language* $L^{f,i\mathcal{O}}$*) relative to the oracle* $(f, i\mathcal{O})$*.*
- ***Black-box proof of hardness:*** *For every ensemble* $f = \{f_n : \{0,1\}^n \to \{0,1\}^{n+1}\}_{n\in\mathbb{N}}$ *of injective functions, for any function* $i\mathcal{O}$ *such that* $i\mathcal{O}(C;r)^f \equiv C^f$ *for any circuit* C *and* $r \in \{0,1\}^*$*, and for any oracle-aided algorithm* $\mathcal{A}$ *that runs in time* $T_{\mathcal{A}}(\cdot)$*, if* $\mathcal{A}^{f,i\mathcal{O}}(x) = \chi_{L^{f,i\mathcal{O}}}(x)$ *for every* $x \in \{0,1\}^*$ *then either*

$$\Pr\left[M^{f,i\mathcal{O},\mathcal{A}}\left(f(x)\right) = x\right] \geq \epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right) \cdot \epsilon_{M,2}(n)$$

for infinitely many values of $n \in \mathbb{N}$*, where the probability is taken over the choice of* $x \leftarrow \{0,1\}^n$ *and over the internal randomness of* M*, or*

$$\left|\Pr\left[\mathit{Exp}^{i\mathcal{O}}_{(f,i\mathcal{O}),i\mathcal{O},M^{\mathcal{A}},\mathcal{C}}(n) = 1\right] - \frac{1}{2}\right| \geq \epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right) \cdot \epsilon_{M,2}(n)$$

for infinitely many values of $n \in \mathbb{N}$*.*

Intuitively, a black-box proof of hardness for $L^{f,i\mathcal{O}}$ means that any algorithm that decides $L^{f,i\mathcal{O}}$ can be used to construct an adversary that breaks either the one-wayness of f or the indistinguishability property of $i\mathcal{O}$ in a black-box way.

Note that restricting $\mathcal{A}$ to be deterministic and to decide the language in the worst case (i.e., on all inputs) only makes our result stronger. Also note that, following Asharov and Segev [2, 3], we split the security loss in the above definition to an adversary-dependent security loss (the function $\epsilon_{M,1}(\cdot)$) and an adversary-independent security loss (the function $\epsilon_{M,2}(\cdot)$), as this allows us to also rule out constructions in which one of these losses is super-polynomial while the other is polynomial.

4 Our Impossibility Result

Equipped with a formal definition of the class of constructions that we consider in this paper (recall Definition 6), in this section we prove the following theorem:

▶ **Theorem 7.** *Let* $((V,\overline{V}),\ell_r,k,M,T_M,\epsilon_{M,1},\epsilon_{M,2})$ *be a fully black-box construction of a worst-case hard* (MIP, coMIP) *protocol pair from an injective one-way function* f *and an indistinguishability obfuscator for all oracle-aided circuits* C^f*. Then, it holds that*

$$\epsilon_{M,1}(n) \cdot \epsilon_{M,2}(n) \leq 2^{-\Omega(n)}.$$

That is, at least one out of the adversary-dependent security loss $\epsilon_{M,1}(\cdot)$ *and the adversary-independent security loss* $\epsilon_{M,2}(\cdot)$ *is exponential.*

Theorem 7 rules out, in particular, standard "polynomial-time polynomial-loss" reductions. More generally, the theorem implies that if the adversary-dependent security loss $\epsilon_{M,1}(\cdot)$ is polynomial (as is typically the case in cryptographic reductions), then the adversary-independent security loss $\epsilon_{M,2}(\cdot)$ must be exponential. Thus, this also rules out constructions that are based on indistinguishability obfuscation with sub-exponential security (e.g., [11, 12]).

In what follows we first introduce our generalized decision oracle, and capture its main property on which we rely in our proof, as discussed in Section 1.2. Then, in Section 4.2 we introduce the additional oracles on which we rely, and in Sections 4.3 and 4.4 we prove that

relative to these oracles and to our decision oracle there exist an injective one-way function and an indistinguishability obfuscator, respectively. Finally, in Section 4.5 we derive the proof of Theorem 7.

4.1 Our Generalized Decision Oracle

For a family of oracles $\mathfrak{S}$ and for any specific oracle $\Psi \in \mathfrak{S}$, we define the oracle $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$ as the following function: Given as input tuple $(C_0, C_1, 1^{\ell_r}, 1^k)$, where C_0 and C_1 are oracle-aided circuits, and ℓ_r and k are non-negative integers, for every $\Phi \in \mathfrak{S}$ the oracle checks if exactly one of the following two cases holds:

- There exists a function $P_1 : \{0,1\}^* \to \{0,1\}$ such that

  $$\Pr_{r \leftarrow \{0,1\}^{\ell_r}} \left[\left\langle C_1^{\Phi}(r), P_1 \right\rangle_k = 1 \right] \geq 2/3 \,,$$

  and for every function $P_0 : \{0,1\}^* \to \{0,1\}$ it holds that

  $$\Pr_{r \leftarrow \{0,1\}^{\ell_r}} \left[\left\langle C_0^{\Phi}(r), P_0 \right\rangle_k = 1 \right] \leq 1/3 \,.$$

  In this case, we say that $(C_0^{\Phi}, C_1^{\Phi}, 1^{\ell_r}, 1^k)$ is a *yes-instance*.
- There exists a function $P_0 : \{0,1\}^* \to \{0,1\}$ such that

  $$\Pr_{r \leftarrow \{0,1\}^{\ell_r}} \left[\left\langle C_0^{\Phi}(r), P_0 \right\rangle_k = 1 \right] \geq 2/3 \,,$$

  and for every function $P_1 : \{0,1\}^* \to \{0,1\}$ it holds that

  $$\Pr_{r \leftarrow \{0,1\}^{\ell_r}} \left[\left\langle C_1^{\Phi}(r), P_1 \right\rangle_k = 1 \right] \leq 1/3 \,.$$

  In this case, we say that $(C_0^{\Phi}, C_1^{\Phi}, 1^{\ell_r}, 1^k)$ is a *no-instance*.

If there exists an oracle $\Phi \in \mathfrak{S}$ such that not exactly one of the above cases hold, then we say that the input $(C_0, C_1, 1^{\ell_r}, 1^k)$ is *invalid* and set $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$ to output $\perp$. Otherwise, $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$ outputs 1 or 0 according to whether $(C_0^{\Psi}, C_1^{\Psi}, 1^{\ell_r}, 1^k)$ is a yes-instance or a no-instance.[5]

The following simple lemma shows that the oracle $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$ can be easily used in order to decide any language that is defined via a $(\mathsf{MIP}, \mathsf{coMIP})$ protocol pair:

▶ **Lemma 8.** *Let $\mathfrak{S}$ be a family of oracles, and let $(V, \overline{V})$ be a pair of oracle-aided polynomial-time algorithms that is an $(\mathsf{MIP}, \mathsf{coMIP})$ protocol pair, with respect to polynomials $\ell_r(\cdot)$ and $k(\cdot)$, relative to every oracle $\Psi \in \mathfrak{S}$. Then, there exists a polynomial-time single-query algorithm $\mathcal{A}$ such that for every $\Psi \in \mathfrak{S}$, given oracle access to $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$ the algorithm $\mathcal{A}$ decides the language $L^{\Psi} \subseteq \{0,1\}^*$ defined by $(V, \overline{V}, \ell_r, k)$ relative to Ψ. That is, for every $\Psi \in \mathfrak{S}$ and $x \in \{0,1\}^*$ the algorithm $\mathcal{A}^{\mathsf{Decide}^{\Psi}_{\mathfrak{S}}}(x)$ outputs 1 if and only if $x \in L^{\Psi}$.*

Proof. Since V and $\overline{V}$ are polynomial time, there exists a polynomial $p(n)$ such that on input of size n their output is of size at most $p(n)$. Given $x \in \{0,1\}^*$ as input and oracle access to $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$, the algorithm $\mathcal{A}$ queries $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$ on $(C_0, C_1, 1^{\ell_r(|x|)}, 1^{k(|x|)})$, where C_0 and C_1 are the hardwired oracle-aided circuits $\overline{V}(x, \cdot)$ and $V(x, \cdot)$ respectively, the input size of both circuits is $\lceil \log(k(|x|)+1) \rceil + k(|x|)$ (where $\lceil \log(k(|x|)+1) \rceil$ bits are for the index of the communication round and $k(|x|)$ bits are for the messages of the prover) and the output size is $p(|x| + \lceil \log(k(|x|)+1) \rceil + k(|x|))$. Finally, the algorithm $\mathcal{A}$ outputs 1 if and only if the oracle's response to the query is 1. ◀

[5] Note that for an input $(C_0, C_1, 1^{\ell_r}, 1^k)$, either it is invalid and then $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$ outputs $\perp$ for every $\Psi \in \mathfrak{S}$, or it is valid and then $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$ outputs 0 or 1 depending on Ψ.

The following lemma captures the key property of our oracle, as discussed in Section 1.2:

▶ **Lemma 9.** *Let $\mathfrak{S}$ be a family of oracles, let $\mathcal{Q}$ be the set of all possible queries for every oracle in the family, let $\mathsf{lab} : \mathcal{Q} \to \mathcal{X}$ be a "labeling" of the possible queries, and let $d \in \mathbb{N}$ be a parameter.*

For any $\Psi \in \mathfrak{S}$ and for any $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}$-query $(C_0, C_1, 1^{\ell_r}, 1^k)$ such that each of the circuits C_0 and C_1 contains at most q oracle gates, there exists a set of labels $\mathbf{I} = \mathbf{I}(\mathfrak{S},\Psi,C_0,C_1,\ell_r,k,\mathsf{lab},d) \subseteq \mathcal{X}$, which we call the influential labels*, satisfying the following two properties:*

1. *The set is small: $|\mathbf{I}| \leq 3 \cdot q \cdot k \cdot d$.*
2. *Any changes to Ψ involving at most d labels outside of $\mathbf{I}$ do not change the answer of the query: Let $\Phi \in \mathfrak{S}$ be another oracle, such that there exists a set $\mathcal{D} \subseteq \mathcal{X} \setminus \mathbf{I}$ of labels with cardinality at most d such that if $\Phi(q) \neq \Psi(q)$ then $\mathsf{lab}(q) \in \mathcal{D}$. Then, it holds that*

$$\mathsf{Decide}^{\Phi}_{\mathfrak{S}}(C_0, C_1, 1^{\ell_r}, 1^k) = \mathsf{Decide}^{\Psi}_{\mathfrak{S}}(C_0, C_1, 1^{\ell_r}, 1^k) \ .$$

Proof. If $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}(C_0, C_1, 1^{\ell_r}, 1^k) = \perp$ this means that the input $(C_0, C_1, 1^{\ell_r}, 1^k)$ is invalid, and then $\mathsf{Decide}^{\Phi}_{\mathfrak{S}}(C_0, C_1, 1^{\ell_r}, 1^k) = \perp$ holds for every $\Phi \in \mathfrak{S}$ and the claim follows for $\mathbf{I} = \emptyset$. Otherwise, suppose without loss of generality that $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}(C_0, C_1, 1^{\ell_r}, 1^k) = 1$ and let $P_1 : \{0,1\}^* \to \{0,1\}$ such that

$$\Pr_{r \leftarrow \{0,1\}^{\ell_r}} \left[\left\langle C_1^{\Psi}(r), P_1 \right\rangle_k = 1 \right] \geq 2/3 \ .$$

Roughly speaking, we define $\mathbf{I} \subseteq \mathcal{X}$ to be the set of all labels for which a query with that label is performed during the execution of the protocol $\left\langle C_1^{\Psi}(\cdot), P_1 \right\rangle_k$ with high probability over the choice of r. More formally, we define

$$\mathbf{I} = \left\{ \mathsf{label} \in \mathcal{X} \middle| \Pr_{r \leftarrow \{0,1\}^{\ell_r}} \left[\begin{array}{c} \text{A query } q \in \mathcal{Q} \text{ such that } \mathsf{lab}(q) = \mathsf{label} \text{ is performed} \\ \text{during the execution of } \left\langle C_1^{\Psi}(r), P_1 \right\rangle_k \end{array} \right] \geq \frac{1}{3 \cdot d} \right\} \ .$$

First, for every $r \in \{0,1\}^{\ell_r}$ at most $q \cdot k$ queries are performed during the execution of $\left\langle C_1^{\Psi}(r), P_1 \right\rangle_k$. Therefore, for any $0 < \epsilon \leq 1$ there are at most $q \cdot k/\epsilon$ labels such that

$$\Pr_{r \leftarrow \{0,1\}^{\ell_r}} \left[\begin{array}{c} \text{A query } q \in \mathcal{Q} \text{ such that } \mathsf{lab}(q) = \mathsf{label} \text{ is performed} \\ \text{during the execution of } \left\langle C_1^{\Psi}(r), P_1 \right\rangle_k \end{array} \right] \geq \epsilon \ .$$

In our case, this means that $\mathbf{I} \leq q \cdot k \cdot 3 \cdot d$ as claimed.

Next, let $\Phi \in \mathfrak{S}$ such that there exists a set $\mathcal{D} \subseteq \mathcal{X} \setminus \mathbf{I}$ of labels with cardinality at most d such that if $\Phi(q) \neq \Psi(q)$ then $\mathsf{lab}(q) \in \mathcal{D}$. By a union bound it holds that

$$\Pr_{r \leftarrow \{0,1\}^{\ell_r}} \left[\begin{array}{c} \text{A query } q \in \mathcal{Q} \text{ such that } \mathsf{lab}(q) \in \mathcal{D} \text{ is performed} \\ \text{during the execution of } \left\langle C_1^{\Psi}(r), P_1 \right\rangle_k \end{array} \right] < \frac{|\mathcal{D}|}{3 \cdot d} \leq \frac{1}{3} \ .$$

If the above event does not occur then $\left\langle C_1^{\Phi}(r), P_1 \right\rangle_k = \left\langle C_1^{\Psi}(r), P_1 \right\rangle_k$. Hence,

$$\begin{aligned} &\Pr_{r \leftarrow \{0,1\}^{\ell_r}} \left[\left\langle C_1^{\Phi}(r), P_1 \right\rangle_k = 1 \right] \\ &\qquad \geq \Pr_{r \leftarrow \{0,1\}^{\ell_r}} \left[\left\langle C_1^{\Psi}(r), P_1 \right\rangle_k = 1 \right] - \Pr_{r \leftarrow \{0,1\}^{\ell_r}} \left[\left\langle C_1^{\Phi}(r), P_1 \right\rangle_k \neq \left\langle C_1^{\Psi}(r), P_1 \right\rangle_k \right] \\ &\qquad > \frac{2}{3} - \frac{1}{3} = \frac{1}{3} \ , \end{aligned}$$

so $(C_0^{\Phi}, C_1^{\Phi}, 1^{\ell_r}, 1^k)$ is not a no-instance. Since $\mathsf{Decide}^{\Psi}_{\mathfrak{S}}(C_0, C_1, 1^{\ell_r}, 1^k) \neq \perp$, $(C_0^{\Phi}, C_1^{\Phi}, 1^{\ell_r}, 1^k)$ must be a yes-instance and therefore $\mathsf{Decide}^{\Phi}_{\mathfrak{S}}(C_0, C_1, 1^{\ell_r}, 1^k) = 1 = \mathsf{Decide}^{\Psi}_{\mathfrak{S}}(C_0, C_1, 1^{\ell_r}, 1^k)$ as claimed. ◀

4.2 Our Indistinguishability Obfuscation Oracle

In what follows we define the family $\mathfrak{S}$ of oracles that realize injective functions and strongly-unambiguous obfuscations relative to our decision oracle, and define a distribution $\mathcal{D}(\mathfrak{S})$ over that family. The family $\mathfrak{S}$ consists of all triplets $(f, \mathcal{O}, E) = (\{f_n\}_{n\in\mathbb{N}}, \{\mathcal{O}_n\}_{n\in\mathbb{N}}, \{E_n\}_{n\in\mathbb{N}})$, satisfying the following three conditions for every $n \in \mathbb{N}$:

1. The function $f_n : \{0,1\}^n \to \{0,1\}^{n+1}$ is injective. Looking ahead, f will serve as an injective one-way function.
2. The function $\mathcal{O}_n : \{0,1\}^{2n} \to \{0,1\}^{10n}$ is injective. Looking ahead, for an oracle-aided circuit $C \in \{0,1\}^n$ with f-gates and randomness $r \in \{0,1\}^n$, the output $\mathcal{O}_n(C, r)$ will serve as an obfuscation of C, and the restriction that $\mathcal{O}_n$ is injective means that the obfuscation is *strongly-unambiguous* in the sense that any obfuscation $\widehat{C} \in \text{Image}(\mathcal{O}_n)$ only comes from a single circuit with a single randomness string.
3. The function $E_n : \{0,1\}^{11n} \to \{0,1\}^n$ satisfies the following condition: For every oracle-aided circuit $C \in \{0,1\}^n$ with f-gates, every randomness $r \in \{0,1\}^n$ and every input $\alpha \in \{0,1\}^n$, it holds that $E_n(\mathcal{O}_n(C, r), \alpha) = C^f(x)$. Namely, given an obfuscation $\widehat{C} = \mathcal{O}_n(C, r)$ and an input α, the function E_n evaluates C on input α with respect to the oracle f.
   We emphasize that for any $\widehat{C} \in \{0,1\}^{10n} \setminus \text{Image}(\mathcal{O}_n)$, there is no restriction on $E_n(\widehat{C}, \cdot)$, so there is no clear way to verify whether some $\widehat{C} \in \{0,1\}^{10n}$ is a valid obfuscation. As noted by Bitansky et al. [10], it is necessary for the obfuscation to not be verifiable since an unambiguous and verifiable indistinguishability obfuscator does imply hardness in $\mathsf{NP} \cap \mathsf{coNP}$.

Now, we define a distribution $\mathcal{D}(\mathfrak{S})$ over $\mathfrak{S}$, relative to which we prove that an oracle $\Psi \leftarrow \mathcal{D}(\mathfrak{S})$ realizes an injective one-way function and an indistinguishability obfuscator. The distribution $\mathcal{D}(\mathfrak{S})$ is obtained by sampling a triplet $(f, \mathcal{O}, \mathsf{Eval}^{f,\mathcal{O}})$ from $\mathfrak{S}$ as follows:

1. For every $n \in \mathbb{N}$ the function f_n is uniformly chosen from the set $\mathsf{InjFunc}_n^{n+1}$ of all injective functions $f_n : \{0,1\}^n \to \{0,1\}^{n+1}$.
2. For every $n \in \mathbb{N}$ the function $\mathcal{O}_n : \{0,1\}^{2n} \to \{0,1\}^{10n}$ is sampled as follows: A function h is uniformly chosen from the set $\mathsf{InjFunc}_n^{5n}$, and for every $r \in \{0,1\}^n$ a function g_r is uniformly chosen from the set $\mathsf{InjFunc}_n^{5n}$. Then, for every input $(C, r) \in \{0,1\}^n \times \{0,1\}^n$ we define $\mathcal{O}_n(C, r) = (h(r), g_r(C))$. Note that $\mathcal{O}_n$ is injective as required, and that this distribution of the function $\mathcal{O}$ differs from that of Asharov and Segev [2] and Bitansky et al. [10], where $\mathcal{O}_n$ was a uniformly chosen injective function.
3. For every $n \in \mathbb{N}$, the function $\mathsf{Eval}^{f,\mathcal{O}}$ on input $(\widehat{C}, \alpha) \in \{0,1\}^{10n} \times \{0,1\}^n$ is defined as follows: If there exists a pair $(C, r) \in \{0,1\}^n \times \{0,1\}^n$ such that $\widehat{C} = \mathcal{O}_n(C, r)$ then it outputs $C^f(\alpha)$, and otherwise it outputs $\perp$. Note that $\mathsf{Eval}^{f,\mathcal{O}}$ satisfies the above third condition for membership in $\mathfrak{S}$.

4.3 The Existence of an Injective One-Way Function

In this section we prove that the injective function f is one way relative to $(\Psi, \mathsf{Decide}_{\mathfrak{S}}^{\Psi})$, where $\Psi = (f, \mathcal{O}, \mathsf{Eval}^{f,\mathcal{O}})$ is sampled from the distribution $\mathcal{D}(\mathfrak{S})$ over $\mathfrak{S}$ (see Section 4.2 for the description of this distribution). Our proof follows the structure of that of Bitansky, Degwekar and Vaikuntanathan [10], while strengthened to deal with our generalized decision oracle as explained in Section 1.2.

In what follows we call an oracle-aided algorithm $\mathcal{A}$ a *q-query algorithm*, for a function $q = q(n)$, if when given any input $x \in \{0,1\}^n$ it issues at most $q(n)$ queries to the oracle

Γ, each of its queries to Eval and Decide consists of circuits with at most $q(n)$ oracle gates, and the number of communication rounds in the proof systems corresponding to each of its queries to Decide is at most $q(n)$.

▶ **Theorem 10.** *For any oracle-aided $2^{n/12}$-query algorithm $\mathcal{A}$ it holds that*

$$\Pr_{\substack{\Psi\leftarrow\mathcal{D}(\mathfrak{S})\\ x\leftarrow\{0,1\}^n}}\left[\mathcal{A}^{\Psi,\mathsf{Decide}^{\Psi}_{\mathfrak{S}}}(f(x))=x\right]\leq O(2^{-n/2})$$

for all sufficiently large $n\in\mathbb{N}$.

In what follows, we let $\mathfrak{F}$ denote the family of ensembles $f=\{f_n\}_{n\in\mathbb{N}}$ where $f_n\in\mathsf{InjFunc}_n^{n+1}$ for all $n\in\mathbb{N}$. As our first step, we prove that $f\leftarrow\mathfrak{F}$ is one way relative to the oracle $(f,\mathsf{Decide}^f_{\mathfrak{F}})$.

▶ **Lemma 11.** *For any oracle-aided $2^{n/6}$-query algorithm $\mathcal{A}$ it holds that*

$$\Pr_{\substack{f\leftarrow\mathfrak{F}\\ x\leftarrow\{0,1\}^n}}\left[\mathcal{A}^{f,\mathsf{Decide}^f_{\mathfrak{F}}}(f(x))=x\right]\leq O(2^{-n/2})\ .$$

Proof. We prove that the lemma holds when even fixing the oracles $f_{-n}=\{f_k\}_{k\neq n}$ and only sampling f_n. We introduce a sequence of three hybrid experiments such that the first hybrid experiment $\mathcal{H}_1$ is the real one-wayness experiment and the last hybrid experiment $\mathcal{H}_3$ is an experiment in which the probability of the adversary is of winning is $1/2^n$. Then, by upper bounding the difference in the winning probability between each pair of consecutive hybrid experiments we deduce our claim.

The hybrid $\mathcal{H}_1$. This is the real experiment in which we sample $x\leftarrow\{0,1\}^n$, give $f_n(x)\in\{0,1\}^{n+1}$ to $\mathcal{A}$ as input, and give $\mathcal{A}$ oracle access to $\Gamma=(f,\mathsf{Decide}^f_{\mathfrak{F}})$.

The hybrid $\mathcal{H}_2$. In this experiment, we sample $y\leftarrow\{0,1\}^{n+1}\setminus\mathrm{Image}(f_n)$, give y to $\mathcal{A}$ as input, and give $\mathcal{A}$ oracle access to $\Gamma'=(f_{x\mapsto y},\mathsf{Decide}^{f_{x\mapsto y}}_{\mathfrak{F}})$, where $f_{x\mapsto y}$ is defined as

$$f_{x\mapsto y}(z)=\begin{cases}y & \text{if } z=x\\ f(z) & \text{otherwise}\end{cases}\ .$$

That is, we "plant" y as the challenge and as the image of x.

The hybrid $\mathcal{H}_3$. This experiment is obtained from $\mathcal{H}_2$ by giving $\mathcal{A}$ oracle access to the original oracle Γ instead of the oracle Γ' with the planted y, while still giving $\mathcal{A}$ the planted y as input.

The following table summarizes our hybrid experiments:

Hybrid	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{H}_3$
Challenger Randomness	$x\leftarrow\{0,1\}^n$		
Injective Function	$f_n\leftarrow\mathsf{InjFunc}_n^{n+1}$		
Challenge	$f_n(x)$	$y\leftarrow\{0,1\}^{n+1}\setminus\mathrm{Image}(f_n)$	
Oracle	$\Gamma=(f,\mathsf{Decide}^f_{\mathfrak{F}})$	$\Gamma'=(f_{x\mapsto y},\mathsf{Decide}^{f_{x\mapsto y}}_{\mathfrak{F}})$	$\Gamma=(f,\mathsf{Decide}^f_{\mathfrak{F}})$
Winning Condition	$\mathcal{A}$ outputs x		

▷ Claim 12. $\Pr[\mathcal{A}\ \textit{wins in}\ \mathcal{H}_1] = \Pr[\mathcal{A}\ \textit{wins in}\ \mathcal{H}_2]$.

Proof. We couple the experiments $\mathcal{H}_1$ and $\mathcal{H}_2$ as follows.[6] First, we sample the same $x \leftarrow \{0,1\}^n$ for both experiments. Then, we a uniformly sample a random injective function $\widehat{f} : \{0,1\}^n \setminus \{x\} \to \{0,1\}^{n+1}$. Next, we sample two distinct $y, y' \leftarrow \{0,1\} \setminus \text{Image}(\widehat{f})$. Now, in $\mathcal{H}_1$ we let the injective function be

$$f_n(z) = \begin{cases} y & \text{if } z = x \\ \widehat{f}(z) & \text{otherwise} \end{cases} ,$$

whereas in $\mathcal{H}_2$ we let the injective function be

$$f'_n(z) = \begin{cases} y' & \text{if } z = x \\ \widehat{f}(z) & \text{otherwise} \end{cases} ,$$

and let y be the planted challenge. It is easy to see that the marginal distribution in both experiments is correct, and that both experiments are identical. That is, $\mathcal{A}$ gets the same challenge as input and gets access to the same oracle, thus the claim follows. ◁

▷ Claim 13. $|\Pr[\mathcal{A}\ \textit{wins in}\ \mathcal{H}_2] - \Pr[\mathcal{A}\ \textit{wins in}\ \mathcal{H}_3]| \leq 3 \cdot q^3/2^n$.

Proof. We observe that the view of $\mathcal{A}$ in $\mathcal{H}_3$ is independent of the choice of x. Therefore, if a query to f_n is made, then the probability of it to be x is at most $1/2^n$. In any other case, the answer to this query is the same in $\mathcal{H}_2$ and $\mathcal{H}_3$, and both executions proceed the same way.

Now, if a query $(C_0, C_1, 1^{\ell_r}, 1^k)$ to $\mathsf{Decide}^f_{\mathfrak{F}}$ is made, then we apply Lemma 9. We take the label function $\mathsf{lab} : \mathcal{Q} \to \mathcal{X}$ to be the identity function. The set $\mathbf{I} = \mathbf{I}(\mathfrak{F}, f, C_0, C_1, \ell_r, k, \mathsf{lab}, 1) \subseteq \mathcal{X}$ of influential labels is independent of the choice of x. Therefore, the probably that $\mathbf{I}$ contains x is at most $|I|/2^n \leq 3q^2/2^n$. In any other case, the oracle $f_{x \mapsto y}$ of $\mathcal{H}_2$ is obtained from f of $\mathcal{H}_3$ by changes involving one label outside of $\mathbf{I}$, and therefore by Lemma 9 it holds that

$$\mathsf{Decide}^{f_{x\mapsto y}}_{\mathfrak{F}}(C_0, C_1, 1^{\ell_k}, 1^k) = \mathsf{Decide}^{f}_{\mathfrak{F}}(C_0, C_1, 1^{\ell_k}, 1^k) ,$$

and both executions proceed the same way. Applying a union bound we deduce that

$$\begin{aligned} &|\Pr[\mathcal{A} \text{ wins in } \mathcal{H}_2] - \Pr[\mathcal{A} \text{ wins in } \mathcal{H}_3]| \\ &\qquad \leq \Pr\left[\text{A query was answered differently in } \mathcal{A}^{\Gamma}(y) \text{ and } \mathcal{A}^{\Gamma'}(y)\right] \\ &\qquad \leq \frac{3q^3}{2^n} , \end{aligned}$$

and the claim follows. ◁

▷ Claim 14. $\Pr[\mathcal{A}\ \textit{wins in}\ \mathcal{H}_3] = 1/2^n$.

Proof. In this experiment the view of $\mathcal{A}$ is independent of x. ◁

[6] To couple two probability distributions means to define a joint distribution whose marginals are exactly those two distributions.

Now we turn back to proving Lemma 11. It holds that

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins in } \mathcal{H}_1] &\leq |\Pr[\mathcal{A} \text{ wins in } \mathcal{H}_1] - \Pr[\mathcal{A} \text{ wins in } \mathcal{H}_2]| \\ &\quad + |\Pr[\mathcal{A} \text{ wins in } \mathcal{H}_2] - \Pr[\mathcal{A} \text{ wins in } \mathcal{H}_3]| + \Pr[\mathcal{A} \text{ wins in } \mathcal{H}_3] \\ &\leq 0 + \frac{3q^3}{2^n} + \frac{1}{2^n} = O\left(\frac{q^3}{2^n}\right) ,\end{aligned}$$

and by plugging $q = 2^{n/6}$ we obtain Lemma 11. ◀

In the full version of this paper [35], we show how to deduce Theorem 10 from Lemma 11.

4.4 The Existence of an Indistinguishability Obfuscator

In this section we prove that relative to the oracle $\Gamma = (\Psi, \mathsf{Decide}^{\Psi}_{\mathfrak{S}})$, where $\Psi = (f, \mathcal{O}, \mathsf{Eval}^{f,\mathcal{O}})$ is sampled from the distribution $\mathcal{D}(\mathfrak{S})$ defined in Section 4.2, there exists an indistinguishability obfuscator $i\mathcal{O}$ for all circuits with f-gates.

Our obfuscator is based on those of Asharov and Segev [2] and Bitansky et al. [10] but has a somewhat different structure. Similarly to their obfuscator, for every $n \in \mathbb{N}$, given an oracle-aided circuit $C \in \{0,1\}^n$, the obfuscator $i\mathcal{O}$ samples $r \leftarrow \{0,1\}^n$ and outputs the obfuscated circuit $\widehat{C} = \mathcal{O}_n(C, r) \in \{0,1\}^{10n}$. In turn, the oracle $\mathsf{Eval}^{f,\mathcal{O}}$ can be used for evaluating such an obfuscated circuit at any given point α: If there exists a pair $(C, r) \in \{0,1\}^n \times \{0,1\}^n$ such that $\widehat{C} = \mathcal{O}_n(C, r)$ then $\mathsf{Eval}^{f,\mathcal{O}}$ outputs $C^f(\alpha)$ and otherwise it outputs $\perp$.

However, unlike their obfuscator of Asharov and Segev [2] and Bitansky et al. [10], which was sampled uniformly at random among all injective functions (of the appropriate input and output lengths), recall that according to our definition of the distribution $\mathcal{D}(\mathfrak{S})$ it holds that $\mathcal{O}_n(C, r) = (h(r), g_r(C))$, where the function h is uniformly-chosen from the set $\mathsf{InjFunc}^{5n}_n$, and for every $r \in \{0,1\}^n$ a function g_r is uniformly-chosen from the set $\mathsf{InjFunc}^{5n}_n$.

Recall that we call an oracle-aided algorithm $\mathcal{A}$ a *q-query algorithm*, for a function $q = q(n)$, if when given any input $x \in \{0,1\}^n$ it issues at most $q(n)$ queries to the oracle Γ, each of its queries to Eval and Decide consists of circuits with at most $q(n)$ oracle gates, and the number of communication rounds in the proof systems corresponding to each of its queries to Decide is at most $q(n)$. Letting $\mathcal{C}$ denote the class of all oracle-aided circuit with f-gates, we prove the following theorem:

▶ **Theorem 15.** *For any oracle-aided $2^{n/6}$-query algorithm $\mathcal{A}$ it holds that*

$$\mathop{\mathbb{E}}_{\Gamma}\left|\Pr\left[\mathsf{Exp}^{i\mathcal{O}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n) = 1\right] - \frac{1}{2}\right| \leq O(2^{-n/4})$$

where the expectation is taken over the choice of $\Gamma = (\Psi, \mathsf{Decide}^{\Psi}_{\mathfrak{S}})$ where $\Psi \leftarrow \mathcal{D}(\mathfrak{S})$, and the inner probability is taken over the randomness of the experiment $\mathsf{Exp}^{i\mathcal{O}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n)$.

Toward proving Theorem 15, we first prove the following lemma.

▶ **Lemma 16.** *For any oracle-aided $4 \cdot 2^{n/6}$-query algorithm $\mathcal{A}$ it holds that*

$$\left|\Pr\left[\mathsf{Exp}^{i\mathcal{O}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n) = 1\right] - \frac{1}{2}\right| \leq O(2^{-n/2})$$

where the probability is taken both over the choice of $\Gamma = (\Psi, \mathsf{Decide}^{\Psi}_{\mathfrak{S}})$ where $\Psi \leftarrow \mathcal{D}(\mathfrak{S})$, and over the randomness of the experiment $\mathsf{Exp}^{i\mathcal{O}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n)$.

Proof. We prove that the lemma holds when even fixing the oracle f and $\mathcal{O}_{-n} = \{\mathcal{O}_k\}_{k\neq n}$, and only sampling $\mathcal{O}_n$. We introduce a sequence of 5 hybrid experiments such that the first hybrid experiment $\mathcal{H}_1$ is the real indistinguishability-obfuscation experiment $\mathsf{Exp}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},\mathcal{A},\mathcal{C}}(n)$ and the last hybrid experiment $\mathcal{H}_5$ is an experiment in which the advantage of the adversary is 0. Then, by upper bounding the difference in the advantage between each pair of consecutive hybrid experiments we deduce our lemma.

In what follows we first describe the hybrid experiments (see also the table below for a summary – where we omit the function f since it has been fixed), and then present a sequence of claims for bounding the differences in the advantages.

The hybrid $\mathcal{H}_1$. This is the real experiment in which we sample $\mathcal{O}_n$ by sampling $h \leftarrow \mathsf{InjFunc}_n^{5n}$, sampling $g_r \leftarrow \mathsf{InjFunc}_n^{5n}$ for every $r \in \{0,1\}^n$, and setting $\mathcal{O}_n(C,r) = (h(r), g_r(C))$.

The hybrid $\mathcal{H}_2$. In this experiment, instead of giving the pre-challenge adversary $\mathcal{A}_0$ access to the oracle $\Gamma = (\Psi, \mathsf{Decide}^{\Psi}_{\mathfrak{S}})$ where $\Psi = (f, \mathcal{O}, \mathsf{Eval}^{\mathcal{O}})$, we sample a string $\widehat{h} \leftarrow \{0,1\}^{5n} \setminus \mathrm{Image}(h)$ and a function $\widehat{g} \leftarrow \mathsf{InjFunc}_n^{5n}$, then we give $\mathcal{A}_0$ access to the oracle $\Gamma' = (\Psi', \mathsf{Decide}^{\Psi'}_{\mathfrak{S}})$ where $\Psi' = (f, \mathcal{O}_{(\cdot,r^*)\to(\widehat{h},\widehat{g}(\cdot))}, \mathsf{Eval}^{\mathcal{O}})$ and for every $C, r \in \{0,1\}^n$ we define

$$\mathcal{O}_{(\cdot,r^*)\to(\widehat{h},\widehat{g}(\cdot))}(C,r) = \begin{cases} (\widehat{h}, \widehat{g}(C)) & \text{if } r = r^* \\ \mathcal{O}(C,r) & \text{otherwise} \end{cases} .$$

That is, for the challenge randomness r^*, instead of obfuscating using $h(r^*)$ and $g_{r^*}(\cdot)$ we use our "planted obfuscation" $\widehat{h}$ and $\widehat{g}(\cdot)$. The rest of the experiment proceeds as before.

The hybrid $\mathcal{H}_3$. In this experiment, we return to giving the pre-challenge adversary $\mathcal{A}_0$ access to the real oracle Γ. However, we now give the post-challenge adversary $\mathcal{A}_1$ a "planted challenge" $(\widehat{h}, \widehat{g}(C_b))$, and we give $\mathcal{A}_1$ access to the oracle $\Gamma' = (\Psi', \mathsf{Decide}^{\Psi'}_{\mathfrak{S}})$ where $\Psi' = (f, \mathcal{O}_{(\cdot,r^*)\to(\widehat{h},\widehat{g}(\cdot))}, \mathsf{Eval}^{\mathcal{O}})$.

The hybrid $\mathcal{H}_4$. For an obfuscator function of the form $\mathcal{O}(C,r) = (h(r), g_r(C))$, $\widehat{h} \in \{0,1\}^{5n} \setminus \mathrm{Image}(h)$ and $\widehat{g} \in \mathsf{InjFunc}_n^{5n}$, we define the planted evaluation function $\mathsf{PEval}^{\mathcal{O}}_{(\widehat{h},\widehat{g})}$ as

$$\mathsf{PEval}^{\mathcal{O}}_{(\widehat{h},\widehat{g})}(\widetilde{C},\alpha) = \begin{cases} C^f(\alpha) & \text{if } \widetilde{C} = (\widehat{h}, \widehat{g}(C)) \text{ for a circuit } C \in \{0,1\}^n \\ \mathsf{Eval}^{\mathcal{O}}(\widetilde{C},\alpha) & \text{otherwise} \end{cases} .$$

Note that since $\widehat{h} \notin \mathrm{Image}(h)$, it holds that $\mathsf{PEval}^{\mathcal{O}}_{(\widehat{h},\widehat{g})}$ is a valid evaluation function and therefore $(f, \mathcal{O}, \mathsf{PEval}^{\mathcal{O}}_{(\widehat{h},\widehat{g})}) \in \mathfrak{S}$. Now, the experiment $\mathcal{H}_4$ is obtained from $\mathcal{H}_3$ by replacing the post-challenge oracle Γ' with the oracle $\Gamma'' = (\Psi'', \mathsf{Decide}^{\Psi''}_{\mathfrak{S}})$ where $\Psi'' = (f, \mathcal{O}, \mathsf{PEval}^{\mathcal{O}}_{(\widehat{h},\widehat{g})})$. Note that in this experiment, the randomness r^* has no role.

The hybrid $\mathcal{H}_5$. This experiment is obtained from $\mathcal{H}_4$ by replacing the challenge obfuscation $(\widehat{h}, \widehat{g}(C_b))$ with $(\widehat{h}, \widehat{g}(C_0))$. Note that in this experiment, the bit b has no role except for the winning condition, namely, $\mathcal{A}$ wins if $\mathcal{A}_1$ outputs b.

Hybrid	$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{H}_3$	$\mathcal{H}_4$	$\mathcal{H}_5$
Challenger Randomness	$b \leftarrow \{0,1\}$, $r^* \leftarrow \{0,1\}^n$			$b \leftarrow \{0,1\}$	
Obfuscator Fuction	$\mathcal{O}_n(C,r) = (h(r), g_r(C))$, where $h \leftarrow \mathsf{InjFunc}_n^{5n}$ and $g_r \leftarrow \mathsf{InjFunc}_n^{5n}$ for every $r \in \{0,1\}^n$				
Planted Obfuscation	N/A	$\widehat{h} \leftarrow \{0,1\}^{5n} \setminus \mathrm{Image}(h)$, $\widehat{g} \leftarrow \mathsf{InjFunc}_n^{5n}$			
Pre-challenge Oracle	$\Psi = (\mathcal{O}, \mathsf{Eval}^{\mathcal{O}})$ $\mathsf{Decide}_{\mathfrak{S}}^{\Psi}$	$\Psi' = (\mathcal{O}' = \mathcal{O}_{(\cdot,r^*)\to(\widehat{h},\widehat{g}(\cdot))}, \mathsf{Eval}^{\mathcal{O}'}), \mathsf{Decide}_{\mathfrak{S}}^{\Psi'}$	$\Psi = (\mathcal{O}, \mathsf{Eval}^{\mathcal{O}})$ $\mathsf{Decide}_{\mathfrak{S}}^{\Psi}$		
Challenge Obfuscation	$\mathcal{O}(C_b, r^*) = (h(r^*), g_{r^*}(C_b))$		$(\widehat{h}, \widehat{g}(C_b))$		$(\widehat{h}, \widehat{g}(C_0))$
Post-challenge Oracle	$\Psi = (\mathcal{O}, \mathsf{Eval}^{\mathcal{O}})$ $\mathsf{Decide}_{\mathfrak{S}}^{\Psi}$		$\Psi' = (\mathcal{O}' = \mathcal{O}_{(\cdot,r^*)\to(\widehat{h},\widehat{g}(\cdot))}, \mathsf{Eval}^{\mathcal{O}'}), \mathsf{Decide}_{\mathfrak{S}}^{\Psi'}$	$\Psi'' = (\mathcal{O}, \mathsf{PEval}^{\mathcal{O}}_{(\widehat{h},\widehat{g})})$ $\mathsf{Decide}_{\mathfrak{S}}^{\Psi''}$	
Winning Condition	$\mathcal{A}_1$ outputs b				

In the full version of this paper [35], we prove the following claims.

▷ Claim 17. $|\Pr[\mathcal{A} \textit{ wins in } \mathcal{H}_1] - \Pr[\mathcal{A} \textit{ wins in } \mathcal{H}_2]| \leq 27q^3/2^n$.

▷ Claim 18. $\Pr[\mathcal{A} \textit{ wins in } \mathcal{H}_2] = \Pr[\mathcal{A} \textit{ wins in } \mathcal{H}_3]$.

▷ Claim 19. $|\Pr[\mathcal{A} \textit{ wins in } \mathcal{H}_3] - \Pr[\mathcal{A} \textit{ wins in } \mathcal{H}_4]| \leq 12q^3/2^n$.

▷ Claim 20. $\Pr[\mathcal{A} \textit{ wins in } \mathcal{H}_4] = \Pr[\mathcal{A} \textit{ wins in } \mathcal{H}_5]$.

▷ Claim 21. $\Pr[\mathcal{A} \textit{ wins in } \mathcal{H}_5] = 1/2$.

Using the above claims, we can prove Lemma 16. It holds that

$$\begin{aligned}\left|\Pr[\mathcal{A} \text{ wins in } \mathcal{H}_1] - \tfrac{1}{2}\right| &= |\Pr[\mathcal{A} \text{ wins in } \mathcal{H}_1] - \Pr[\mathcal{A} \text{ wins in } \mathcal{H}_5]| \\ &\leq \sum_{i=1}^{4} |\Pr[\mathcal{A} \text{ wins in } \mathcal{H}_i] - \Pr[\mathcal{A} \text{ wins in } \mathcal{H}_{i+1}]| \leq \frac{40q^3}{2^n}\ ,\end{aligned}$$

and by plugging $q = 4 \cdot 2^{n/6}$ we obtain Lemma 16. ◀

In the full version of this paper [35], we show how to deduce Theorem 15 from Lemma 16.

4.5 Putting it All Together

Given Theorems 10 and 15 we can now derive Theorem 7.

Proof. Let $((V,\overline{V}), \ell_r, k, M, T_M, \epsilon_{M,1}, \epsilon_{M,2})$ be a fully black-box construction of a worst-case hard $(\mathsf{MIP}, \mathsf{coMIP})$ protocol pair from an injective one-way function f and an indistinguishability obfuscator for all oracle-aided circuits C^f. Lemma 8 guarantees the existence of a polynomial-time single-query algorithm $\mathcal{A}$ such that for every $\Psi = (f, \mathcal{O}, \mathsf{Eval}^{f,\mathcal{O}})$ in the support of the distribution $\mathcal{D}(\mathfrak{S})$, the algorithm $\mathcal{A}$ with oracle access to $\Gamma = (\Psi, \mathsf{Decide}_{\mathfrak{S}}^{\Psi})$ decides the language $L^{\Psi} \subseteq \{0,1\}^*$ defined by $(V, \overline{V}, \ell_r, k)$ relative to Ψ. That is, for every $x \in \{0,1\}^*$ it holds that $\mathcal{A}^{\Gamma}(x) = \chi_{L^{\Psi}}(x)$. For every $n \in \mathbb{N}$, denote by $T_{\mathcal{A}}(n)$ the polynomial running time of $\mathcal{A}$ on inputs of length n.

Definition 6 then states that there are two possible cases to consider: $\mathcal{A}$ can be used either for inverting the injective one-way function f, or for breaking the security of the indistinguishability obfuscator $i\mathcal{O}$. Specifically, in the first case we obtain from Definition 6 that for every $\Psi = (f, \mathcal{O}, \mathsf{Eval}^{f,\mathcal{O}})$ in the support of the distribution $\mathcal{D}(\mathfrak{S})$ it holds that

$$\Pr\left[M^{\Gamma,\mathcal{A}}(f(x)) = x\right] \geq \epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right) \cdot \epsilon_{M,2}(n)$$

for infinitely many values of $n \in \mathbb{N}$, where $\Gamma = (\Psi, \mathsf{Decide}^{\Psi}_{\mathfrak{S}})$ and the probability is taken over the choice of $x \leftarrow \{0,1\}^n$ and over the internal randomness of M. The algorithm M may invoke $\mathcal{A}$ on various input lengths (i.e., in general M is not restricted to invoking $\mathcal{A}$ only on input length n), and we denote by $\ell(n)$ the maximal input length on which M invokes $\mathcal{A}$ (when M itself is invoked on input $f(x)$ for $x \in \{0,1\}^n$). Thus, viewing $M^{\mathcal{A}}$ as a single oracle-aided algorithm that has access to Γ, its running time $T_{M^{\mathcal{A}}}(n)$ satisfies $T_{M^{\mathcal{A}}}(n) \leq T_M(n) \cdot T_{\mathcal{A}}(\ell(n))$ (this follows since M may invoke $\mathcal{A}$ at most $T_M(n)$ times, and the running time of $\mathcal{A}$ on each such invocation is at most $T_{\mathcal{A}}(\ell(n))$). In particular, viewing $M' = M^{\mathcal{A}}$ as a single oracle-aided algorithm that has oracle access to Γ, implies that M' is a q-query algorithm where $q(n) = T_{M^{\mathcal{A}}}(n)$. This holds for any Ψ in the support of the distribution $\mathcal{D}(\mathfrak{S})$, and given that $q(n)$ is polynomial in n then Theorem 10 guarantees that $\epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right) \cdot \epsilon_{M,2}(n) \leq O(2^{-n/2})$.

In the second case we obtain from Definition 6 that for every $\Psi = (f, \mathcal{O}, \mathsf{Eval}^{f,\mathcal{O}})$ in the support of the distribution $\mathcal{D}(\mathfrak{S})$ it holds that

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},M^{\mathcal{A}},\mathcal{C}}(n) = 1\right] - \frac{1}{2}\right| \geq \epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right) \cdot \epsilon_{M,2}(n)$$

for infinitely many values of $n \in \mathbb{N}$, where the probability is taken over the randomness of the experiment $\mathsf{Exp}^{\mathsf{iO}}_{\Gamma,i\mathcal{O},M^{\mathcal{A}},\mathcal{C}}(n)$. The same reasoning applied to the first case, together with Theorem 15 guarantee that $\epsilon_{M,1}\left(T_{\mathcal{A}}(n)\right) \cdot \epsilon_{M,2}(n) \leq O(2^{-n/4})$.

We conclude the proof noting that the algorithm $\mathcal{A}$ provided by Lemma 8 runs in fact in linear time. That is, $T_{\mathcal{A}}(n) = O(n)$, and thus from the above two cases we obtain $\epsilon_{M,1}\left(n\right) \cdot \epsilon_{M,2}(n) \leq 2^{-\Omega(n)}$. ◀

References

1. William Aiello and Johan Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *Journal of Computer and System Sciences*, 42(3):327–345, 1991.
2. Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, pages 191–209, 2015.
3. Gilad Asharov and Gil Segev. On constructing one-way permutations from indistinguishability obfuscation. In *Proceedings of the 13th Theory of Cryptography Conference*, pages 512–541, 2016.
4. László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.
5. Boaz Barak. Structure vs. combinatorics in computational complexity. Windows on Theory (available at `https://windowsontheory.org/2013/10/07/structure-vs-combinatorics-in-computational-complexity/`), 2013.
6. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6, 2012.
7. Paul Beame, Stephen A. Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. The relative complexity of NP search problems. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 303–314, 1995.
8. Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 113–131, 1988.
9. Nir Bitansky and Akshay Degwekar. On the complexity of collision resistant hash functions: New and old black-box separations. In *Proceedings of the 17th Theory of Cryptography Conference*, pages 422–450, 2019.

10 Nir Bitansky, Akshay Degwekar, and Vinod Vaikuntanathan. Structure vs. hardness through the obfuscation lens. In *Advances in Cryptology – CRYPTO '17*, pages 696–723, 2017.

11 Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a Nash equilibrium. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, pages 1480–1498, 2015.

12 Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos – trapdoor permutations from indistinguishability obfuscation. In *Proceedings of the 13th Theory of Cryptography Conference*, pages 474–502, 2016.

13 Manuel Blum and Russell Impagliazzo. Generic oracles and oracle classes. In *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*, pages 118–126, 1987.

14 Andrej Bogdanov and Chin Ho Lee. Limits of provable security for homomorphic encryption. In *Advances in Cryptology – CRYPTO '13*, pages 111–128, 2013.

15 Gilles Brassard. Relativized cryptography. In *Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science*, pages 383–391, 1979.

16 Richard Chang, Benny Chor, Oded Goldreich, Juris Hartmanis, Johan Håstad, Desh Ranjan, and Pankaj Rohatgi. The random oracle hypothesis is false. *Journal of Computer and System Sciences*, 49(1):24–39, 1994.

17 Stephen A. Cook, Russell Impagliazzo, and Tomoyuki Yamakami. A tight relationship between generic oracles and type-2 complexity theory. *Information and Computation*, 137(2):159–170, 1997.

18 Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theorey*, 22(6):644–654, 1976.

19 Lance Jeremy Fortnow. Complexity-theoretic aspects of interactive proof systems. Ph.D. Thesis, MIT, 1989.

20 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 54th Annual IEEE Symposium on Foundations of Computer Science*, pages 40–49, 2013.

21 Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a Nash equilibrium. In *Advances in Cryptology – CRYPTO '16*, pages 579–604, 2016.

22 Oded Goldreich. On security preserving reductions – revised terminology. Cryptology ePrint Archive, Report 2000/001, 2000.

23 Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

24 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

25 Juris Hartmanis and Lane A. Hemachandra. One-way functions, robustness, and the non-isomorphism of NP-complete sets. In *Proceedings of the 2nd Annual Conference on Structure in Complexity Theory*, 1987.

26 Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 44–61, 1989.

27 Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science*, pages 374–383, 2014.

28 Tianren Liu and Vinod Vaikuntanathan. On basing private information retrieval on NP-hardness. In *Proceedings on the 13th Theory of Cryptography Conference*, pages 372–386, 2016.

29 Michael Luby. *Pseudorandomness and Cryptographic Applications*. Princeton University Press, 1996.

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

31 Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In *Proceedings of the 1st Theory of Cryptography Conference, TCC 2004*, pages 1–20, 2004.

32 Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communication of the ACM*, 21(2):120–126, 1978.

33 Steven Rudich. *Limits on the Provable Consequences of One-way Functions*. PhD thesis, EECS Department, University of California, Berkeley, 1988.

34 Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 475–484, 2014.

35 Gil Segev and Ido Shahaf. Hardness vs. (very little) structure in cryptography: A multi-prover interactive proofs perspective. Cryptology ePrint Archive, Report 2020/330, 2020. URL: `https://eprint.iacr.org/2020/330`.

36 Adi Shamir. IP=PSPACE. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 11–15, 1990.

37 Brent Waters. A punctured programming approach to adaptively secure functional encryption. In *Advances in Cryptology – CRYPTO '15*, pages 678–697, 2015.

Oblivious Parallel Tight Compaction

Gilad Asharov
Bar-Ilan University, Ramat-Gan, Israel
Gilad.Asharov@biu.ac.il

Ilan Komargodski
NTT Research, East Palo Alto, CA, USA
ilan.komargodski@ntt-research.com

Wei-Kai Lin
Cornell University, Ithaca, NY, USA
wklin@cs.cornell.edu

Enoch Peserico
Università degli Studi di Padova, Italy
enoch@dei.unipd.it

Elaine Shi
Cornell University, Ithaca, NY, USA
runting@gmail.com

Abstract

In tight compaction one is given an array of balls some of which are marked 0 and the rest are marked 1. The output of the procedure is an array that contains all of the original balls except that now the 0-balls appear before the 1-balls. In other words, tight compaction is equivalent to sorting the array according to 1-bit keys (not necessarily maintaining order within same-key balls). Tight compaction is not only an important algorithmic task by itself, but its oblivious version has also played a key role in recent constructions of oblivious RAM compilers.

We present an oblivious deterministic algorithm for tight compaction such that for input arrays of n balls requires $O(n)$ total work and $O(\log n)$ depth. Our algorithm is in the Exclusive-Read-Exclusive-Write Parallel-RAM model (i.e., EREW PRAM, the most restrictive PRAM model), and importantly we achieve asymptotical optimality in both total work and depth. To the best of our knowledge no earlier work, even when allowing randomization, can achieve optimality in both total work and depth.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols

Keywords and phrases Oblivious tight compaction, parallel oblivious RAM, EREW PRAM

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.11

Related Version The full version is available on Cryptology ePrint Archive, Report 2020/125 [4], `https://eprint.iacr.org/2020/125`.

Funding *Wei-Kai Lin*: Wei-Kai Lin supported by NSF CNS-1453634, an ONR YIP award, a Packard Fellowship, and a DARPA Brandeis grant.
Elaine Shi: Elaine Shi supported by NSF CNS-1453634, an ONR YIP award, a Packard Fellowship, and a DARPA Brandeis grant.

Acknowledgements Wei-Kai thanks Jyun-Jie Liao for reminding the similarity between superconcentrators and oblivious tight compaction.

1 Introduction

Tight compaction aims to solve the following problem: given an array of elements each marked with either the label 0 or 1, move all the 0-elements to the front of the array and all the 1-elements to the end. In other words, we would like to sort an array of elements

each tagged with a 1-bit key. Moreover, due to its relationship to the classical sorting problem, tight compaction has been an abstraction of interest and has been studied in the core algorithms literature for several decades and in various models of computation. Notably, Pippenger's elegant self-routing superconcentrator construction [27] implied a deterministic tight compaction algorithm completing in $O(n)$ total work and $O(\log n)$ depth (see Section 1.2 for other classical algorithmic results on this important abstraction).

In this paper, we care about solving the tight compaction problem on a parallel RAM, but imposing an additional natural privacy requirement commonly referred to as *obliviousness*. Specifically, we require that the memory access patterns of the RAM be independent of the input array. In this way, an adversary (e.g., an untrusted cloud server) who observes the RAM's access patterns cannot gather any information about the secret input. Besides being an interesting question on its own, an important application of oblivious tight compaction is in the design of efficient Oblivious RAM (ORAM) algorithms, as shown repeatedly in a sequence of recent works [10, 8, 3], including a very recent work of Asharov et al. [3] which demonstrated an asymptotically optimal ORAM and closed a long-time open question in this line of work [16, 15].

Clearly a naïve method to solve oblivious tight compaction is to rely on a sorting circuit such as AKS [2] to sort the entire input of n elements, consuming $O(n \cdot \log n)$ work and $O(\log n)$ depth. However, since general-purpose oblivious sorting must consume $\Omega(n \cdot \log n)$ work (under the indivisibility assumption [7, 23] or assuming a well-known network coding conjecture [13]), a very natural question is whether we can accomplish oblivious tight compaction with asymptotically better overheads.

Pippenger's result [27], mentioned above, does not satisfy obliviousness. However, around the same time, another independent work by Leighton et al. [22] showed that there is an *almost* oblivious randomized algorithm that accomplishes tight compaction except with negligible probability in n in $O(n \cdot \log \log n)$ work and $O(\log n)$ depth – and the algorithm's access patterns leak only the total number of 0 elements in the input array, and nothing else. Subsequent works by Mitchell and Zimmerman [25] and Lin, Shi, and Xie [23] improve upon Leighton et al. [22] by showing how to achieve full obliviousness (i.e., also hiding the number of 0s) while retaining the same asymptotical overheads as Leighton et al. Like Leighton et al., Mitchell and Zimmerman and Lin et al.'s algorithms are also randomized and have a negligible probability of failure.

These results left open the question of devising a *deterministic* oblivious tight compaction algorithm with *linear* total work. For a very long time there was no progress in either fronts until the recent work by Asharov et al. [3] where they constructed a *deterministic* oblivious tight compaction algorithm that consumes linear total work.[1] While optimal in total work, Asharov et al.'s algorithm is sequential and turning it into one with logarithmic depth seems non-trivial.[2] We ask whether one could have a clean and optimal result statement, that is,

Is it possible to construct a deterministic oblivious tight compaction algorithm that consumes linear total work and logarithmic depth?

[1] The work of Asharov et al. [3] subsumes and contains the work of Peserico [26] so we neither mention the latter explicitly nor compare to it.

[2] More precisely, the algorithm of Asharov et al., as written, has linear depth. However, as we explain in Section 2, there are a couple of standard (yet non-trivial) tricks one can apply in order to make it consume $O(\log n \cdot \log \log n)$ depth. Modifying their scheme to consume only $O(\log n)$ depth seems to require new non-trivial ideas.

Essentially we ask whether one can match the best known non-oblivious deterministic result, i.e., Pippenger's algorithm [27], but additionally achieving obliviousness. Note that any tight compaction algorithm requires linear work and logarithmic depth on PRAMs with exclusive writes (which is the model we work in), even without obliviousness. The former is since any algorithms must at least read the input and the latter is due to an elegant and classical lower bound by Cook et al. [12].

1.1 Our Contributions

We close the gap in our understanding regarding this important algorithmic abstraction by answering the above question affirmatively, showing an algorithm that is optimal in total work as well as in depth.

▶ **Theorem 1** (Informal). *There exists a deterministic oblivious tight compaction algorithm such that for an input array of n elements, the algorithm completes in $O(n)$ total work and $O(\log n)$ total depth (assuming that each element fits into a single memory word and a memory word can hold at least $\log n$ bits).*

Our result holds on an Exclusive-Read-Exclusive-Write (EREW) PRAM,[3] i.e., the most restrictive PRAM model (which makes our result stronger). Furthermore, our algorithm is in the "indivisible model", i.e., while the algorithm can perform numeric computations on the 1-bit keys, the elements themselves are "indivisible" and can only be moved around in memory [7]. By asymptotically matching the best known non-oblivious algorithm [27], our result shows that obliviousness of tight compaction can be obtained "for free" (other examples of such tasks are sorting [2] and parallel merge sort [11]).

In a very high level, our algorithm combines ideas, in a non black-box way, from the non-oblivious tight compaction algorithm of Pippenger [27] together with the oblivious yet sequential algorithm of Asharov et al. [3]. See Section 2 for an overview of our technical highlights.

Towards an optimal OPRAM. Since oblivious tight compaction is an important building block in the optimal ORAM construction of [3], one may wonder if our result can be used to obtain a depth-efficient version of their work, i.e., an oblivious PRAM (OPRAM) [6, 10]. Unfortunately, not immediately: their ORAM construction is based on several other building blocks and algorithms which are sub-optimal in terms of depth (e.g., oblivious Cuckoo hashing [17, 21]). While our result does take us one step closer towards an optimal OPRAM, a full resolution is left for future work.

1.2 Related Work

As mentioned, the study of compaction algorithms is core to the classical algorithms literature due to its close relations to sorting. We thus discuss additional related work.

The tight compaction problem has been studied in the core algorithms literature under various models of computation. For PRAMs with exclusive-writes, which is the model we consider, a classical result by Cook et al. [12] shows that a logarithmic lower bound exists for *any* algorithm even without the obliviousness or indivisibility requirements. On a concurrent-read concurrent-write (CRCW) PRAM, there is a well-known $\Omega(\log n / \log \log n)$-depth lower

[3] In the EREW model, every memory cell can be read or written to by only one processor at a time.

bound for any algorithm even without the obliviousness or indivisibility assumptions. Moreover, there is a matching non-oblivious upper bound that achieves $O(\log n/\log\log n)$ depth and linear total work [28].

Another related abstraction, called *stable* tight compaction, aims to achieve the same task as tight compaction, but now additionally requiring stability, i.e., in the output array, elements with the same key must appear in the same order as the input. For oblivious algorithm subject to the indivisibility assumptions, there is a separation between stable tight compaction and non-stable ones. Specifically, a recent lower bound by Lin et al. [23] shows that any oblivious algorithm subject to the indivisibility assumption must incur $\Omega(n \cdot \log n)$ work to stably and tightly compact an arbitrary input array of n elements (while without stability one can achieve it in $O(n)$ work [3]). Therefore, in this paper, we allow our tight compaction to not have to respect stability.

Due to the close relationship of tight compaction and sorting, a natural question is whether one can design algorithms in the so-called *comparison-based* model where the algorithm is only allowed to perform comparisons on keys and move elements around. However, due to the well-known 0-1 principle for sorting, any comparison-based algorithm that can sort an array with 1-bit keys must be able to sort any array with arbitrary keys. Thus we cannot constrain ourselves to the comparison-based model since otherwise there would be an $\Omega(n \cdot \log n)$ lower bound [20].

A relaxed abstraction, called *loose* compaction, is also studied extensively in the algorithms literature. Loose compaction solves the following problem: given an input array containing n elements among which at most n/ℓ are real and the rest are dummy (for some constant $\ell > 0$), compress the input array to half of the original size while not losing any real element in this process. Pippenger's self-routing superconcentrator [27] implies a non-oblivious loose compaction algorithm with $O(n)$ total work and $O(\log n)$ depth. Asharov et al. [3], relying on Pippenger's work, showed how to obtain *oblivious* loose compaction without increasing the asymptotical overhead. Loose compaction has also received a lot of attention in the parallel (non-oblivious) algorithms literature. There is a separation between loose and tight compaction on a CRCW PRAM. Specifically, Bast and Hagerup [5] showed the existence of an $O(n)$ work and $O(\log^* n)$ depth parallel (non-oblivious) algorithm for performing loose compaction, while as mentioned $\Omega(\log n/\log\log n)$ depth is necessary for tight compaction.

We note that, in the current paper as well as in previous works (e.g., Asharov et al. [3]), we use loose compaction as an intermediate abstraction. Here, however, we are unable to use previous constructions directly. In fact, we introduce another relaxation of loose compaction which allows to "lose" a small fraction of real elements. While this allows us to implement this procedure very efficiently (in work and depth), it introduces a new challenge of correcting the mistakes in parallel afterwards. See Section 2 for details.

2 Technical Overview

In this section we give an overview of our construction with an emphasis on the main ideas used to get Theorem 1.

It has been known for a while that the tight compaction problem is very related to the notion of self-routing super-concentrator [27, 3]. Recall that a superconcentrator is a graph that consists of n source vertices and n target vertices such that for any $k \leq n$, any k-subset of sources is connected to any k-subset of targets by k vertex-disjoint paths [27, 1, 29]. Intuitively, one can imagine associating the input balls with the source vertices and then routing the 0 balls to the first target vertices and the 1 balls to the last target vertices.

However, this process (i.e., the way the superconcentrator decides how to route) is known to be non-oblivious and the naive way to make it oblivious is by sending "dummy messages" along unused edges – causing a logarithmic overhead in the total work. Actually, working out the details of the algorithm, one can see that the logarithmic overhead is independent of the size of the balls and so the actual total work is $O(n \cdot \log n + \lceil D/w \rceil \cdot n)$ [9], where D is the bit-size of each ball and w is the word size.

The recent work of Asharov et al. [3] managed to get this overhead down to $O(\lceil D/w \rceil \cdot n)$ using two ideas: (1) reducing to loose compaction and (2) packing and decomposition. We elaborate on these ideas next.

Reducing to loose compaction. In this step the task of tight compaction is reduced to the task of loose compaction. In loose compaction, one is given an array where balls are marked either real or dummy and it is guaranteed that there are at most $1/\ell$ fraction of reals. Henceforth, we arbitrarily assume that $\ell = 128$. The goal is to output an array of size $n/2$ which contains all the reals. First note that every real within the first $n/2$ locations is already in the "right" place and it should not be moved and so we only need to deal with the reals which are in the second half of the array. To this end, Asharov et al. imagine the elements as associated with the left nodes of a good bipartite expander and every misplaced real is swapped with a misplaced dummy (i.e. a real from the second half of the array with a dummy from the first half) if and only if they are neighbors of distance 2 on the expander. Using properties of the expander, one can show that this "handles" almost all of the misplaced reals. Now, one can use loose compaction to compress the array (since it now contains much fewer misplaced reals) and recurse on the the array of size $n/2$. The cost of this reduction is linear – the bipartite graph has constant degree and each recursive step halves the array size.

Decomposition and packing. These two ideas are used to implement loose compaction. The idea is to zoom-in on the input array in various resolutions. In the smallest instance case, one can pack lots of information into a single word and larger instances are decomposed to this smaller one. Concretely, Asharov et al. define three scenarios, depending on the relation between n (the number of balls in the input) and w (the word size):

1. Small instances – If $n \leq w/\log w$: In this case, one can basically "download" the input to the client and solve loose compaction. More precisely, one can download 1 bit per input element, saying whether it is real or not. This is enough to compute the disjoint routes which can then be used to perform the actual routing.
2. Medium instances – If $n \leq (w/\log w)^2$: The idea is to "zoom out" and view each block of $\sqrt{n} \leq w/\log w$ balls as one ball which is labeled dense if and only if it contains at least $\sqrt{n}/4$ real balls. Since the original array has at most $1/128$ reals, the "zoomed out" array has at most $\sqrt{n}/32$ denses. Notice that we can run our tight compaction procedure on this "zoomed out" array, moving the dense blocks to the front, since it is a "small instance". Next, we run tight compaction for small instances again but now within the non-dense blocks. This compresses $3/4$ of the space for $3/4$ of the blocks (which are non-dense by assumption).
3. Large instances – If $n > (w/\log w)^2$: Now that we have tight compaction for small and medium instances (by the above two items), and we get loose compaction for large instances in a very similar way. We "zoom out" and view the input array as $m = n/(w/\log w)^2$ blocks each consisting of $(w/\log w)^2$ elements. As before, we mark a block as dense if it contains more than $1/4$ or reals (as before, at most $1/32$ fraction of the blocks can be dense). To perform loose compaction on the "zoomed out" array we apply the naive

oblivious algorithm to compute routes. As mentioned, the naive algorithm has extra logarithmic overhead but this is okay since we apply it on an array that contains m elements (indeed, $O(m \cdot \log m + \lceil D/w \rceil \cdot m) \leq O(\lceil D/w \rceil \cdot n)$). Once the dense blocks are at the front, we can invoke tight compaction for medium instances and compact each block in linear time. As before, this compresses 3/4 of the space for 3/4 of the blocks.

This completes the high-level description of the algorithm of Asharov et al. We continue to explain what are the challenges & ideas used to make it in depth $O(\log n)$.

Challenge 1. The first component that is not optimal for depth is the reduction to loose compaction. There, we perform swaps over the edges of a bipartite expander and naively performing all of them in parallel does not work. Indeed, a single "misplaced" real node could be the distance-2 neighbor of two (or more) dummys so we have to be able to resolve these conflict somehow in low depth.

The solution for this is inspired by the solution for a similar problem from the superconcentrator and sorting networks literature. The idea is to use a particular property of the expander graph: there is a natural partitioning of the entire edge-set into a constant number of disjoint perfect matchings. Using this, we can perform the swaps in parallel between different copies and it is guaranteed that there will be no "collisions".

Challenge 2. Recall that the construction of Asharov et al. consists of first reducing tight compaction to loose compaction and then solving loose compaction (for small, medium and finally large instances). In terms of depth, naively the reduction from tight compaction to loose compaction resumes for $\log n$ steps until the instance becomes of constant size. A simple observation is that we can actually run the recursion only for $O(\log \log n)$ steps until the instance size becomes $O(n/\log n)$ size in which case we can just invoke full-fledged oblivious sort [2]. What about the depth of loose compaction? For small instances it is $O(n)$, for medium ones it is $O(\sqrt{n})$, and for large ones it is $O(\log n)$ (the latter is the dominant one). In total, the depth of Asharov et al.'s construction (after solving challenge 1 and the above observation about the depth of the recursion) is $O(\log n \cdot \log \log n)$. Getting rid of the extra $\log \log n$ factor is the most challenging part of our work.

We do not know how to get loose compaction in depth better than $O(\log n)$. Our main idea is to circumvent this by weakening the requirement from loose compaction by allowing it to err. Concretely, we consider a weak version of loose compaction that we call *weak compression* which takes an array of size n that has say $1/128$ fraction of reals and it outputs an array of size $n/2$ that contains *almost*, say ϵ-fraction of, all reals. The main observation is that if ϵ is set to be $1/\mathsf{polylog}(n)$, then weak compression can actually be realized in $O(\log \log n)$ depth (rather than $O(\log n)$ without errors). To see this, one has to recall the details of how the superconcentrator chooses its routes. Roughly, this is a process that proceeds in "rounds" over a bipartite graph, where at each round a constant fraction of nodes become satisfied (i.e., routes are found). After $O(\log n)$ rounds, all nodes are satisfied, but after $O(\log \log n)$ rounds all but $1/\mathsf{polylog}(n)$ fraction of nodes are satisfied.

Combining the above weak compression procedure with the reduction from tight compression to (this variant of) loose compaction gives an abstraction we call a *swapper* and it costs linear work and logarithmic depth. This abstraction can be viewed as (another) relaxation of tight compaction where any 1-ball that appears before a 0-ball is swapped except for a $1/\mathsf{polylog}(n)$ fraction of pairs which remain "in reverse order". All we are left to do is to correct these errors.

We correct the error by building (from scratch) a tight compaction procedure that works for very sparse inputs (of density $1/\mathsf{polylog}(n)$) in linear work and logarithmic depth. Indeed, using such a procedure we can easily swap the remaining misplaced elements. To get a tight compaction procedure for sparse inputs, the idea is to first compress the array into one that is of size $O(n/\log n)$. Then, we can run full-fledged oblivious sort which completes the task.[4]

The key technical contribution is the way we compress the sparse array to size $O(n/\log n)$ with only $O(\log n)$ depth. Towards this end, we stack $O(\log\log n)$ many instances of loose compaction, each compressing the size by factor $1/2$. Indeed, since the input is only of density $1/\mathsf{polylog}(n)$, $O(\log\log n)$ layers are sufficient in terms of functionality. But what about complexity? We said that the depth of loose compaction is $O(\log n)$ so stacking $O(\log\log n)$ many instances of them does not sound like a good idea. To see why this is okay, recall again that loose compaction is basically implemented by using a fixed bipartite expander graph and doing: (1) finding an appropriate matching (in rounds), and (2) performing the routing over the matching.

The basic observation is that step (1) can be parallelized among all layers and then using the computed matchings, the routing can be directly performed. Parallelizing step (1) is not straight-forward as in layer i for $i > 1$ we do not know who are the sources when the matching from layer $i-1$ has not been determined yet. But since the input is very sparse, we can compute *all possibilities* of sources in each layer i. Since the bipartite graph has constant degree, the number of possibilities only grows by a constant factor in every level and there are only $O(\log\log n)$ levels so choosing the parameters carefully, we can tolerate the extra $\mathsf{polylog}(n)$ factor in the number of possibilities. Let us remark that a similar issue came up in the non-oblivious self-routing superconcentrator of Pippenger [27] and the above idea is inspired by Pippenger's solution.

Simplified construction and reduced asymptotic constant. As a bonus, the fact that we introduce weak versions of compaction that permit various types of errors, allows us to simplify the construction of Asharov et al. Concretely, our new algorithm has only two cases "large instances" and "small instances" (i.e., we got rid of the "medium instances").

Recall that in Asharov et al. [3], for large instances (i.e., $n > (w/\log w)^2$) it takes $O(m \cdot \log m)$ work to compute the matching. Thus, they choose the large-medium cutoff to be of size $(w/\log w)^2$ so that in the large case $m = n/(w/\log w)^2$ implies that $O(m \cdot \log m) = O(n)$. Medium instances (i.e., $n \leq (w/\log w)^2$) are still too large to be solved directly, so they further divide each medium instance into $\sqrt{n}$ small instances. The latter can be solved directly by packing. We, on the other hand, have work overhead for large instances of only $O(\log\log m)$. This allows us to split large instances to $m = n/\log w$ blocks which implies overhead $O(m \cdot \log\log m) = O(n)$. This directly reduces us to the "small instance" case without going through the medium size instances case. While we are not motivated by optimizing asymptotic constants, compared to Asharov et al. (whose overhead is roughly 2^{30}), our simplified construction directly yields a better constant (roughly 2^{20}).

Sorting with more keys. Our tight compaction is an algorithm for sorting an array of n balls where each ball is marked with a 1-bit key. Our algorithm can be extended to sort n balls marked with K-bit keys where K is any constant. The idea is that our reduction to loose compaction can be modified such that every element that is not "misplaced" will remain in the same location throughout the execution of the algorithm. We elaborate on the

[4] Actually, this gives a *stable* tight compaction procedure for sparse input arrays.

details of this modification in Remark 15. With this feature, we can first compact the array, moving the elements tagged with the first key to the front, and then compact (recursively) the rest of the array, keeping those elements at the front.

In more detail, to sort an array of balls marked with keys from $[K]$, we first run the plain tight compaction algorithm so that 1-balls (by k-balls we mean all balls marked with key $k \in [K]$) are moved to the front. Then, we apply the variant of tight compaction, mentioned above, to compact 2-balls while keeping 1-balls unmoved. At the kth step for $k \in [K]$, to move the k-balls to their correct locations, we apply the variant of tight compaction to compact k-balls while keeping 1-balls through $(k-1)$-balls in place. The procedure finishes after K iteration. By inspection, this algorithm consumes $O(K \cdot n)$ total work and $O(K \cdot \log n)$ depth (which remain linear and logarithmic, respectively, as long as K is constant).

3 Preliminaries

3.1 Definitions

We use the standard exclusive-read and exclusive-write (EREW) parallel random-access machine (PRAM) model, parameterized by N memory *words*, unlimited number of CPUs, each word is $w = \Theta(\log N)$ bits, each CPU has an internal state that consists of a small constant number of words. For any PRAM algorithm, we characterize the efficiency by *work* and *depth*, where work is the total number of word-level operations performed by all CPUs, and depth is the number of parallel steps consumed by the algorithm. As the input size $n \leq N$, we always have $w = \Omega(\log n)$. The details are deferred to the full version.

In this work, we require algorithms being both *oblivious* and *deterministic*. Roughly speaking, given a PRAM algorithm M, we require the existence of a simulator Sim such that for all input $\mathbf{I}$, given only the input size $n = |\mathbf{I}|$ (without knowing $\mathbf{I}$), Sim outputs the identical sequence of accessed memory addresses during the computation $M(\mathbf{I})$. The formal definitions are deferred to the full version.

3.2 Tools

We will use several (standard) tools on which we elaborate next.

Oblivious sorting. Ajtai et al. [2] shows that there is a comparator-based circuit with $O(n \cdot \log n)$ comparators and $O(\log n)$ depth that can sort any array of length n.

▶ **Theorem 2** (Ajtai et al. [2]). *There is a deterministic oblivious sorting algorithm in the PRAM model with word size w that sorts n elements using $O(\lceil D/w \rceil \cdot n \cdot \log n)$ work and $O(\log n)$ depth, where D denotes the length of each element in bits.*

Expanders. Our construction relies on dense family (i.e., one per say every power of 2) of constant degree bipartite expander graphs that have several appealing properties: (1) their entire edge set can be computed in linear time in the number of nodes and (2) their entire edge set can be partitioned into a constant number of disjoint perfect matchings. For this, we use either of the well-known construction of expander graphs presented by Margulis [24], Gabber and Galil [14], or Jimbo and Maruoka [19]. It is well-known that these graph satisfies the above properties (e.g., it was used in the sorting network of Ajtai et al. [2] and the self-routing superconcentrators of Pippenger [27]). Below, we provide a precise statement for completeness. We note that more modern constructions of expanders, while giving better constants due to higher spectral gap, do not fit our purpose since they usually result with families which are neither dense enough nor satisfy property (1).

Let $G = (L, R, E)$ be a d-regular bipartite graph such that $|L| = |R|$. Let $P_1, \ldots, P_d$ be a partition of E into d disjoint perfect matchings. (Note that by Hall's theorem [18], such a partition always exists though it may not unique and may not be efficiently computable for an arbitrary d-regular bipartite.) We say the vertex u is the r-th neighbor of v, denoted as $\Gamma_r(v)$, if and only if (u, v) is an edge in P_r. The proof is deferred to the full version.

▶ **Theorem 3.** *For any constant $\lambda \in (0, 1)$, there exists a family of bipartite graphs $\{G_{\lambda,n}\}_{n\in\mathbb{N}}$ and a constant $d_\lambda \in \mathbb{N}$, such that for every $n \in \mathbb{N}$ being a power of 2, $G_{\lambda,n} = (L, R, E)$ has $|L| = |R| = n$ vertices on each side, it is d_λ-regular, and for every sets $S \subseteq L, T \subseteq R$, it holds that*

$$\left| e(S,T) - \frac{d_\lambda}{n} \cdot |S| \cdot |T| \right| \leq \lambda \cdot d_\lambda \cdot \sqrt{|S| \cdot |T|},$$

where $e(S, T)$ is the number of edges $(s, t) \in E$ such that $s \in S$ and $t \in T$. Additionally, in the word-RAM model with word size w such that $w \geq \Omega(\log n)$,

1. *there exists a (uniform) linear work algorithm that on input 1^n outputs the entire edge set of $G_{\lambda,n}$.*
2. *there exists a (uniform) constant work algorithm that on input $r \in [d_\lambda], v \in L \cup R$, computes $\Gamma_r(v)$, where $\Gamma_r(v)$ is defined with respect to a fixed partition of $G_{\lambda,n}$.*

4 Our Abstractions

We will realize tight compaction in Section 5 in linear work and logarithmic depth. On our path towards this goal, we implement a few abstractions that we define next. They will not only help us present the construction in a modular way, but we believe that some of them might be of independent interest. All the following abstractions take as input an array of balls and are parameterized by functions $\alpha(\star), \beta(\star), \epsilon(\star)$, or $\gamma(\star)$, where $\star$ is a placeholder for the number of balls in the input array.

4.1 Tight Compaction

In the tight compaction problem one is given an input array containing n balls each of which marked with a 1-bit label that is either 0 or 1. The output is a permutation of the input array such that all the 1-balls are moved to the front of the array.

▶ **Definition 4** (Tight compaction)**.** *Let $\boldsymbol{I}$ be an array of n balls such that each ball is labeled with 0 or 1. On input $\boldsymbol{I}$, tight compaction outputs an array $\boldsymbol{O}$ which is a permutation of the balls in $\boldsymbol{I}$ such that all the 0-balls appear before the 1-balls.*

4.2 Swapper and Imbalanced Swapper

An ϵ-swapper is parametrized by a function $\epsilon\colon \mathbb{N} \to [0, 1]$. This procedure gets as input an array $\mathbf{I}$ of n balls where each ball is marked with a color from $\{\mathsf{red}, \mathsf{blue}, \perp\}$. It is guaranteed that the number of red balls is equal to the number of blue balls. The output of the procedure is an array $\mathbf{O}$ of size n in which all but at most $\epsilon \cdot n$ red-blue ball pairs are swapped, where the swapped balls are marked as $\perp$ but the non-swapped balls keep the input colors. Looking forward, the output colors will be utilized to further swap balls later in Algorithm 1.

▶ **Definition 5** (ϵ-swapper)**.** *Let $\boldsymbol{I}$ be an array of n balls such that each ball is marked either red, blue, or $\perp$ and the number of red balls equals to the number of blue balls. On input $\boldsymbol{I}$, ϵ-swapper outputs an array $\boldsymbol{O}$ of size n which is a permutation of $\boldsymbol{I}$, the number of red balls*

equals to the number of blue *balls in* $\boldsymbol{O}$*, the total number of* red *and* blue *balls in* $\boldsymbol{O}$ *is at most* $\epsilon \cdot n$*, and for every* $i \in [n]$*:*

1. *If* $\boldsymbol{I}[i]$ *is* $\perp$*, then* $\boldsymbol{O}[i] = \boldsymbol{I}[i]$ *and it is marked* $\perp$*.*
2. *If* $\boldsymbol{I}[i]$ *is* red *(resp.* blue*), then either*
   a. $\boldsymbol{O}[i] = \boldsymbol{I}[i]$ *and it is marked* red *(resp.* blue*), or*
   b. $\boldsymbol{O}[i] = \boldsymbol{I}[j]$ *and marked* $\perp$ *for some* blue $\boldsymbol{I}[j]$ *(resp.* red $\boldsymbol{I}[j]$*).*[5]

An ϵ-imb-swapper (stands for *imbalanced swapper*) generalizes an ϵ-swapper. It also takes as input an array $\mathbf{I}$ of n balls, where each ball is marked with a color from $\{\mathsf{red}, \mathsf{blue}, \perp\}$, but the difference is that the number of red balls does not have to be equal to the number of blue balls. Let $q(\mathbf{I}) = |n_{\mathsf{red}} - n_{\mathsf{blue}}|$, where n_φ for $\varphi \in \{\mathsf{red}, \mathsf{blue}\}$ is the number of balls with color φ in $\mathbf{I}$, be the number of "extra" balls in $\mathbf{I}$ from either color. On input $\mathbf{I}$, an ϵ-imb-swapper outputs an array $\mathbf{O}$ that satisfies the same requirement as $\mathbf{O}$ in Definition 5 except that the total number of balls in $\mathbf{O}$ that are either red or blue is at most $\epsilon \cdot n + q(\mathbf{I})$.

We provide realizations for both primitives. We call the first realization Swapper and the second one ImWeakSwapper, where the difference is not only the imbalance of red and blue balls but also the value of ϵ. Specifically, for an input array with n balls, Swapper makes all necessary swaps except for $(1/\mathsf{poly}\log n)$-fraction in $O(\log n)$ depth, whereas ImWeakSwapper makes all the necessary swaps except for a constant fraction but in constant depth.

The following lemmas are proven in Sections 6.3 and 6.4, respectively:

▶ **Lemma 6** (Swapper). *For all constants* $c \in \mathbb{N}$*, letting* $\epsilon(\star) = 1/\log^c \star$*, there exists a deterministic oblivious procedure* Swapper *that implements* ϵ-swapper *in the PRAM model. Letting* w *be the word size,* n *be the number of balls in the input array, and* D *be the size of each ball in bits,* Swapper *consumes* $O(\lceil D/w \rceil \cdot n)$ *work and* $O(\log n)$ *depth.*

▶ **Lemma 7** (Imbalanced weak swapper). *For every constant* $\ell \in \mathbb{N}$*, there exists a deterministic oblivious procedure* ImWeakSwapper *that implements an* $(1/\ell)$-imb-swapper *in the PRAM model. Letting* w *be the word size,* n *be the number of balls in the input array, and* D *be the size of each ball in bits,* ImWeakSwapper *consumes* $O(\lceil D/w \rceil \cdot n)$ *work and* $O(1)$ *depth.*

4.3 Compression

The next abstraction is called (α, β)-compression and it is parametrized by $\alpha, \beta \colon \mathbb{N} \to [0, 1]$ such that $\forall \star \in \mathbb{N} \colon \alpha(\star) \leq \beta(\star)$. It gets as input an array $\mathbf{I}$ of n balls where each ball is either real or dummy. It is guaranteed that the number of real balls in $\mathbf{I}$ is at most $\alpha \cdot n$. The output of the procedure is an array $\mathbf{O}$ of size $\beta \cdot n$ that contains all the real balls from $\mathbf{I}$. The output may consist of "filler" balls that are not in the input (and note that even with fillers, we can still "reverse route" the real output balls back to the input).

▶ **Definition 8** ((α, β)-compression). *Let* $\boldsymbol{I}$ *be an array of* n *balls such that each balls is marked* real *or* dummy*, where the number of* real *balls is at most* $\alpha \cdot n$*. On input* $\boldsymbol{I}$*,* (α, β)-compression *outputs an array* $\boldsymbol{O}$ *of size* $\beta \cdot n$ *that consists of all the* real *balls in* $\boldsymbol{I}$*, where the* real *balls are still marked* real*, and the other balls are arbitrary and marked* dummy*.*

The following lemmas are proven in Sections 6.2 and 6.7, respectively.

[5] As the word "swap" hints, our algorithm will indeed swap a blue $\mathbf{I}[i]$ with a red $\mathbf{I}[j]$ and then output exactly $(\mathbf{O}[i], \mathbf{O}[j]) = (\mathbf{I}[j], \mathbf{I}[i])$ for the pair (i, j); the abstraction is relaxed (as no pairing required) yet sufficient later.

▶ **Lemma 9** (Compression). *For all large enough constants $c \in \mathbb{N}$, letting $\alpha(\star) = 1/\log^c \star$ and $\beta(\star) = 1/\log\star$, there exists a deterministic oblivious procedure* Compression *that implements an (α, β)-*compression *in the PRAM model. Letting w be the word size, n be the number of balls in the input array, and D be the size of each ball in bits,* Compression *consumes $O(\lceil D/w \rceil \cdot n)$ work and $O(\log n)$ depth.*

▶ **Lemma 10** (Fast compression for short inputs). *There exists a constant $\alpha \in (0, 1/2)$ for which there exists a deterministic oblivious procedure* FastCompression *that implements an $(\alpha, 1/2)$-*compression *in the PRAM model. Letting w be the word size, $n \leq w/\log w$ be the number of balls in the input array, and D be the size of each ball in bits,* FastCompression *consumes $O(\lceil D/w \rceil \cdot n)$-work and $O(n)$ depth.*

4.4 Weak Compression

Lastly, we define γ-approx-(α, β)-compression for $\alpha, \beta, \gamma \colon \mathbb{N} \to [0, 1]$ such that $\forall \star \in \mathbb{N} \colon \gamma(\star) \leq \alpha(\star) \leq \beta(\star)$. This algorithm is the same as (α, β)-compression except that there the is a "mistake" on $\gamma \cdot n$ inputs which are not in the output array $\mathbf{O}$. Those balls appear in another array $\mathbf{E}$ and they are in the same positions as in $\mathbf{I}$. That is, intuitively, the algorithm moves some real balls from $\mathbf{I}$ into the output array $\mathbf{O}$, while other real balls are not moved and still reside in $\mathbf{I}$. We call that array $\mathbf{E}$. Note that 0-approx-(α, β)-compression is equivalent to (α, β)-compression, as $\mathbf{E}$ will consist of only dummy balls.

▶ **Definition 11** (γ-approx-(α, β)-compression). *Let $\boldsymbol{I}$ be an array of n balls such that each ball is marked* real *or* dummy*, where the number of* real *balls is at most $\alpha \cdot n$. On input $\boldsymbol{I}$,* γ-approx-(α, β)-compression *is an algorithm that outputs two arrays $\boldsymbol{O}$ and $\boldsymbol{E}$, such that*

- *$\boldsymbol{O}$ is an array of $\beta \cdot n$ balls that consists of all* real *balls in $\boldsymbol{I}$ except γ fraction, where the* real *balls are still marked* real*, and the other balls are arbitrary and marked* dummy*.*
- *$\boldsymbol{E}$ is obtained by removing from $\boldsymbol{I}$ all the* real *balls that reside in $\boldsymbol{O}$ and replacing them with* dummy*s.*

The following lemmas are proven in Sections 6.5 and 6.6, respectively.

▶ **Lemma 12** (Weak compression). *There exists a constant $\alpha \in (0, 1/2)$, such that for all constants $c \in \mathbb{N}$, letting $\gamma(\star) = 1/\log^c \star$, there exists a deterministic oblivious procedure* WeakCompression *that implements a γ-*approx-$(\alpha, 1/2)$-compression *in the PRAM model. Letting w be the word size, n be the number of balls in the input array, and D be the size of each ball in bits,* WeakCompression *consumes $O(\lceil D/w \rceil \cdot n)$ work and $O(\log\log n)$ depth.*

▶ **Lemma 13** (Slow weak compression). *There exists a constant $\alpha \in (0, 1/2)$ such that for all constants $c \in \mathbb{N}$, letting $\gamma(\star) = 1/\log^c \star$, there exists a deterministic oblivious procedure* SlowWeakCompression *that implements a γ-*approx-$(\alpha, 1/2)$-compression *in the PRAM model. Letting w be the word size, n be the number of balls in the input array, and D be the size of each ball in bits,* SlowWeakCompression *consumes $O(n \cdot \log\log n + \lceil D/w \rceil \cdot n)$-work and $O(\log\log n)$ depth.*

We summarize our lemmas and their complexities in Figure 1, where we let n denote the number of balls in each input array. Figure 1 also depicts how our implementations correspond to each other and provides an overview of the roadmap towards our tight compaction algorithm.

Name	Reference	Abstraction	Work[a]	Depth
TightCompaction	Theorem 1 (§5)	Tight Compaction	$O(n)$	$O(\log n)$
Swapper	Lemma 6 (§6.3)	1/polylog-swapper	$O(n)$	$O(\log n)$
ImWeakSwapper	Lemma 7 (§6.4)	$O(1)$-imb-swapper	$O(n)$	$O(1)$
Compression	Lemma 9 (§6.2)	$(1/\text{polylog}, 1/\log)$-compression	$O(n)$	$O(\log n)$
FastCompression[b]	Lemma 10 (§6.7)	$(O(1), 1/2)$-compression	$O(n)$	$O(n)$
WeakCompression	Lemma 12 (§6.5)	1/polylog-approx-$(O(1), 1/2)$-compression	$O(n)$	$O(\log \log n)$
SlowWeakCompression	Lemma 13 (§6.6)	1/polylog-approx-$(O(1), 1/2)$-compression	$O(n \cdot \log \log n)$	$O(\log \log n)$

[a] In this table, we assume that D, each ball size in bits, is $O(w)$.
[b] Assuming $n \leq w/\log w$.

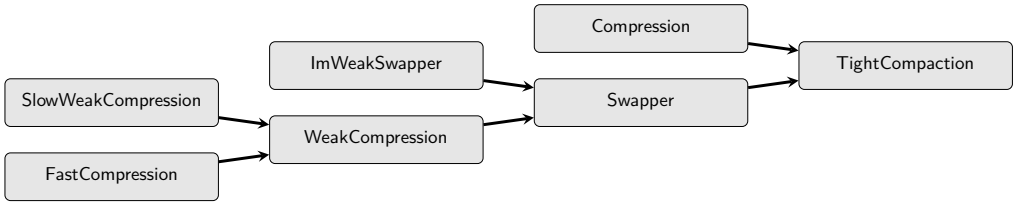


Figure 1 The diagram depicted the relationship between the implementations of our abstractions. TightCompaction is implemented using Compression and Swapper, where the latter is implemented using ImWeakSwapper and WeakCompression, and the latter is implemented using SlowWeakCompression and FastCompression.

5 Parallel Tight Compaction

In this section we present our tight compaction algorithm.

▶ **Theorem 14** (Restatement of Theorem 1)**.** *There exists a deterministic oblivious algorithm TightCompaction that implements tight compaction in the PRAM model. Letting w be the word size, n be the number of balls in the input array, and D be the size of each ball in bits, TightCompaction consumes $O(\lceil D/w \rceil \cdot n)$ work and $O(\log n)$ depth.*

We use the Swapper and Compression algorithms from Lemmas 6 and 9. Specifically, we use Compression to implement $(\frac{1}{\log^c \star}, \frac{1}{\log \star})$-compression for some constant c (see Lemma 9) and let Swapper implement $(\frac{1}{\log^c \star})$-swapper (for the same constant c).

Proof of Theorem 14. See the full version for detailed analysis. ◀

▶ Remark 15 (Compacting with more keys). In TightCompaction, the balls marked as $\perp$ in Step 1 are never moved throughout the execution and remain in the same place in the output array $\mathbf{O}$. Hence, whenever we want the first t balls to remain in the same location throughout the algorithm while compacting the last $n - t$ balls for some $t \leq n$, it suffices to modify TightCompaction as follows. First, we let TightCompaction get t as an extra input. Then, we modify Step 1 to always mark the first t balls as $\perp$, while counting 0-balls and marking blue and red on the remaining $n - t$ balls. This modification achieves the abstraction we mentioned in the end of Section 2 in the context of compacting balls tagged with more than 1-bit keys: compacting an array while keeping some elements in place.

6 Realizing the Abstractions

In this section we provide proofs of our abstractions, i.e., proofs for Lemmas 6, 7, 9, 10, 12, and 13. We start with a common procedure for obliviously finding a matching with a particular structure in a bipartite graph. We use this procedure to implement Compression (Lemma 9) and SlowWeakCompression (Lemma 13). As each of the building block (including

Algorithm 1 TightCompaction($\mathbf{I}$).

- **Input**: an array $\mathbf{I}$ of n balls, each ball is labeled by a single bit 0 or 1.
- **Procedure:**
  1. **Color the misplaced 1-balls by blue and the misplaced 0-balls by red (notice that there is the same amount of each).**
     - **a.** Count the number of 0-balls in $\mathbf{I}$, let d be this number.
     - **b.** For $i = 1, 2, \ldots, n$, in parallel, do the following:
       - **i.** If $\mathbf{I}[i]$ is a 1-ball and $i \leq d$, mark $\mathbf{I}[i]$ as blue.
       - **ii.** If $\mathbf{I}[i]$ is a 0-ball and $i > d$, mark $\mathbf{I}[i]$ as red.
       - **iii.** Otherwise, mark $\mathbf{I}[i]$ as $\perp$.
  2. **Swap red and blue balls guaranteeing that only $n/\mathsf{polylog}n$ misplaced balls remain.**
     - **a.** Run Swapper($\mathbf{I}$) and let $\mathbf{I}'$ be the resulting array.
  3. **Mark and compress the remaining misplaced balls into an array of size $n/\log n$.**
     - **a.** For each ball in $\mathbf{I}'$, in parallel, mark it as real if it is red or blue, and mark as dummy if $\perp$.
     - **b.** Run Compression($\mathbf{I}'$). Let C be the resulting array, and let Aux_1 be the array recording every move of balls during the compression.
  4. **Swap reds and blues in the compressed array by sorting the array (moving reds to the front and blues to the end).**
     - **a.** Using an oblivious sort (e.g. AKS; see Theorem 2), permute the array C so that red balls are at the front and blue balls are at the back. Let Aux_2 be the array recording every move of balls during the sorting.
     - **b.** For each $i \in [\lfloor n/2 \rfloor]$, in parallel, swap $C[i]$ and $C[n-i+1]$ if and only if $C[i]$ is red and $C[n-i+1]$ is blue. Let C' be the result.
  5. **Reverse route the swapped balls in the compressed array back into the original one.**
     - **a.** Using Aux_2 from Step 4a, perform the inversed permutation on C'. Then, using Aux_1 from Step 3b, perform the inversed compression on C' back to $\mathbf{I}'$. Let the result be $\mathbf{O}$.
- **Output:** The array $\mathbf{O}$.

bipartite expander graphs, counting, and oblivious sorting) are both deterministic and oblivious, a straightforward syntactic checking proves obliviousness and determinism for each above lemma, and hence we will focus only on proving the correctness and efficiency.

6.1 Find Matching

Let $G = (L, R, E)$ be a d-regular bipartite graph. For $r \in [d]$ and vertex u in $G_{\lambda,n}$, let $\Gamma_r(u)$ denote the r-th neighbor of u in $G_{\lambda,n}$. For a subset of edges $M \subseteq E$, and any node $u \in L \cup R$, let $\Gamma_M(u) = \{v \in L \cup R \mid (u, v) \in M\}$ be the set of neighboring vertices of u in M. We define an (a, b)-matching for a subset of nodes $S \subseteq L$ on the left, as a subset of edges for which every vertex from S is connected to at least a vertices on the right and that each vertex on the right is connected to at most b vertices from S.[6]

[6] The term "matching" follows previous works [3], and it is also known as "assignment" or "compactor" [9, 27].

▶ **Definition 16** ((a,b)-matching). *We say that $M \subseteq E$ is an (a,b)-matching of $S \subseteq L$ in G iff (1) for all $u \in S$, $|\Gamma_M(u)| \geq a$, and (2) for all $v \in R$, $|\Gamma_M(v)| \leq b$.*

We relax Definition 16 to allow for an error. Namely, we define γ-approx-(a,b)-matching as an (a,b)-matching except that condition (1) holds for all but a γ fraction of vertices from S. That is, there is a subset $S' \subseteq S$ such that $|S'| \geq (1-\gamma) \cdot |S|$ and for every $u \in S'$, $|\Gamma_M(u)| \geq a$.

▶ **Definition 17** (γ-approx-(a,b)-matching). *We say that $M \subseteq E$ is a γ-approx-(a,b)-matching of S in G iff condition (1) holds for all but γ faction of nodes in S, and condition (2) still holds.*

Let $G_{\lambda,n} = (L, R, E)$ with $\lambda := 1/64$ be the d_λ-regular expander from Theorem 3. Let $B := \lfloor d_\lambda/2 \rfloor$. In the rest of this subsection, we prove the following two claims.

▷ Claim 18 (SlowMatch). For any input $I \subseteq [n]$ such that $|I| \leq n/32$, the procedure SlowMatch (see Algorithm 2) outputs a $(B, B/4)$-matching of I in $G_{\lambda,n}$. It consumes $O(n \cdot \log n)$ work and $O(\log n)$ depth.

▷ Claim 19 (WeakSlowMatch). Let constant $c > 0$, and $\gamma(n) = 1/\log^c(n)$. For any input $I \subseteq [n]$ such that $|I| \leq n/32$, the procedure WeakSlowMatch (see Algorithm 3) outputs a γ-approx-$(B, B/4)$-matching of I in $G_{\lambda,n}$. It consumes $O(n \cdot \log\log n)$ work and $O(\log\log n)$ depth.

We also use the following claim from [3].

▷ Claim 20 (FastMatch, [3, Claim 5.16]). For any input $I \subseteq [n]$ such that $|I| \leq n/32$ and $n \leq w/\log w$, there exists a procedure FastMatch outputs a $(B, B/4)$-matching of I in $G_{\lambda,n}$. It consumes $O(n)$ work and $O(n)$ depth.

Overview. We start with a high-level overview of the non-oblivious matching algorithm, inspired by Pippenger [27], Chan et al. [9], and Asharov et al. [3]. Claims 18, 19, and 20 are all based on this algorithm with minor variations, as we explain below. Given a bipartite graph with vertices L, R and a set $S \subseteq L$ of m marked vertices, we first mark all vertices in S as "unsatisfied". Then, in each round:

- **Each unsatisfied vertex $u \in S$:** Send a request to each one of the neighbors of u.
- **Each vertex $v \in R$:** If v received more than $B/4$ requests in each round, it replies with "negative" to all requests it received in this round. Otherwise, it replies with "positive" to all requests it received. (If v did not receive any request, it replies no positive nor negative.)
- **Each unsatisfied vertex $u \in S$:** If u received more than B positive replies then take these edges to the matching and change the status to "satisfied".

The output is all the edges in the matching. Note that in each round there are $O(|S|) = O(m)$ transmitted messages, where each message is just a single bit. Using the expansion of the graph and the fact that $|S|$ is small enough, in each iteration the number of unsatisfied vertices is decreased by a factor $1/2$. This implies that within $O(\log m)$ iterations all unsatisfied vertices will become satisfied. Claim 18 is obtained by running this algorithm while always simulating dummy access to hide which node is transmitting messages and which is not. This causes a logarithmic blow-up in the total work. Claim 19 is obtained by observing that if the above process is executed for only $O(\log\log n)$ iterations, then all but $1/\mathsf{polylog}(n)$ fraction of vertices become satisfied. Lastly, Claim 20 is obtained by observing that if n is

small enough, the whole graph can fit into $O(1)$ words and so it can be read within $O(1)$ queries and so we can run the above algorithm without the logarithmic overhead incurred by simulating dummy accesses.

Algorithm 2 SlowMatch: $(B, B/4)$-matching.

- **Input:** An array $\mathbf{I}$ of n indicators representing a subset $I \subset [n]$ such that $|I| \leq \frac{n}{32}$ and $n > w/\log w$.
- **The procedure:**
  1. Let M be a $(d_\lambda \times n)$-array of indicators initialized to all 0s, where $M[r, i]$ indicates if the r-th edge of the i-th left vertex is in the $(B, B/4)$-matching.
  2. Let $I' = \mathbf{I}$. Repeat the following for $\lceil \log n \rceil$ iterations.
     a. Initialize two arrays *Request* and *Positive*, both containing n 0s.
     b. *For each vertex $u \in I'$, send a "request" to all neighbors of u:* For each $r \in [d_\lambda]$, perform the following sequentially.
     For all vertex $u \in L$ in parallel, if $u \in I'$, increment $Request[\Gamma_r(u)]$. **(If $u \notin I'$, perform fake accesses)**
     c. *For each vertex $v \in R$, if v received from 1 to $B/2$ requests, then reply "positive" to all neighbors of v:* For each $r \in [d_\lambda]$, perform the following sequentially.
     For all vertex $v \in R$ in parallel, if $1 \leq Request[v] \leq B/2$, then increment $Positive[\Gamma_r(v)]$. **(Otherwise, perform fake accesses)**
     d. *For each vertex $u \in I'$, if u received at least B positive replies, then u adds the edge (u, x) to M such that x replied positively:* For each $r \in [d_\lambda]$, perform the following sequentially:
     For all vertex $u \in L$ in parallel, if $u \in I'$ and $Positive[u] \geq B$ and $Request[\Gamma_r(u)] \leq B/2$, then set $M[r, u] := 1$. **(Otherwise, perform fake accesses)**
     e. *For each vertex $u \in I'$, if u received at least B positive replies, then u removes itself from I':* For all vertex $u \in L$ in parallel, if $Positive[u] \geq B$, set $I'[u] := 0$.
- **Output:** The array M.

Algorithm 3 WeakSlowMatch: $1/\log^c(\star)$-approx-$(B, B/4)$-matching.

- **Input:** An array $\mathbf{I}$ of n indicators representing a subset $I \subset [n]$ such that $|I| \leq \frac{n}{32}$.
- **The procedure:**
  1. Do everything exactly the same as in Algorithm 3, except that in step 2, perform only $c \cdot \log\log n$ iterations.
- **Output:** The array M.

Proof of Claim 18. See the full version for detailed analysis. ◁

Proof of Claim 19. See the full version for detailed analysis. ◁

6.2 Compression (Lemma 9)

▶ **Lemma 9** (Compression). *For all large enough constants $c \in \mathbb{N}$, letting $\alpha(\star) = 1/\log^c \star$ and $\beta(\star) = 1/\log \star$, there exists a deterministic oblivious procedure* Compression *that implements an (α, β)-*compression *in the PRAM model. Letting w be the word size, n be the number of balls in the input array, and D be the size of each ball in bits,* Compression *consumes $O(\lceil D/w \rceil \cdot n)$ work and $O(\log n)$ depth.*

To implement $(\frac{1}{\log^c \star}, \frac{1}{\log \star})$-compression for large enough constant $c \in \mathbb{N}$, the input is an array of n balls, where at most $\frac{n}{\log^c n}$ balls are real, and we want to compress the input down to $n/\log n$ balls. Previously, it is known how to implement $(\alpha', 1/2)$-compression for small enough constant fraction α' using $(B, B/4)$-matching in a bipartite expander [27, 3, 9]: Very roughly, every μ balls are interpreted as a *block*, and every B blocks are put on a left vertex of the bipartite expander, where μ and B are some parameters (which we will formalize later); Then, the $(B, B/4)$-matching is capable of routing all real blocks (i.e., a block contains any real ball) from left to right vertices while guaranteeing that every right vertex has at most $B/4$ real blocks; Hence, merging the real blocks on every two right vertices into one vertex yields an array of a half number of balls. A straw-man implementation of $(\frac{1}{\log^c \star}, \frac{1}{\log \star})$-compression is applying SlowMatch for $t := \log\log n$ rounds, but given that SlowMatch takes $O(\log n)$ depth (Claim 18) to compute $(B, B/4)$-matching, the straw man takes more than logarithmic depth.

To reduce the total depth, we separate every instance of $(B, B/4)$-matching into two phases. Notice that in the straw-man implementation, there is a bipartite expander in each round, and every real block is routed through t layers of bipartite expanders. Thus, we connect t expanders into a directed graph H, where each directed edge (u, v) represents a potential move of a real or dummy block from u to v. In the first phase, given the input array, our compression marks every vertex in the first-layer expander such that is associated with any real block, and then it marks all vertices on H that is reachable by the first-layer marked vertices (i.e., mark a vertex if there exists a path from the first-layer marked vertices). In the second phase, we compute all t instances of $(B, B/4)$-matching *in parallel* as the marked vertices are the only input to SlowMatch, which takes $O(\log n)$ depth as desired.

To ensure SlowMatch outputs correct $(B, B/4)$-matchings in the second phase, it suffices to ensure that the fraction of marked vertices is at most $1/32$ as required in Claim 18. Given that the expanders are d_λ-regular, after $t = \log\log n$ layers, the number of marked vertices grows by d_λ^t times, which is $\log^{c'} n$ for some constant c'; Choosing a sufficiently large constant $c > c'$ in the input satisfies the requirement. Finally, as the marked vertices consists of all the vertices that any real block will be routed through, the resulting matchings are capable of routing all real blocks. The algorithm is formalize in Algorithm 4.

Proof of Lemma 9. See the full version for detailed analysis. ◀

6.3 Swapper (Lemma 6)

▶ **Lemma 6** (Swapper). *For all constants $c \in \mathbb{N}$, letting $\epsilon(\star) = 1/\log^c \star$, there exists a deterministic oblivious procedure Swapper that implements ϵ-swapper in the PRAM model. Letting w be the word size, n be the number of balls in the input array, and D be the size of each ball in bits, Swapper consumes $O(\lceil D/w \rceil \cdot n)$ work and $O(\log n)$ depth.*

Let $c \in \mathbb{N}$ be the constant for which we wish to implement $(1/\log^c \star)$-swapper. We use ImWeakSwapper (from Lemma 7) and WeakCompression (from Lemma 12). Particularly, we use WeakCompression which implements $(1/\log^{c_1} \star)$-approx-$(\alpha, 1/2)$-compression for some constant $\alpha \in (0, 1/2)$ (as in Lemma 12) and $c_1 := \max\{2c+2, 4-\log\alpha\}$, and ImWeakSwapper which implements $(\alpha/2)$-imb-swapper.

In a high-level, we start by applying ImWeakSwapper to the input array. This swaps a constant fraction of balls in constant depth. Then, we compress (most of) the remaining balls into an array of size $n/2$ using WeakCompression. Then, we recursively on this smaller array. The end of the recursion is when the remaining array has size $O(n/\log n)$ (namely after $O(\log\log n)$ recursive steps), in which case we can afford to run a full oblivious sorting

Algorithm 4 Compression: $(\frac{1}{\log^c \star}, \frac{1}{\log \star})$-compression.

- **Input:** an array $\mathbf{I}$ of n balls such that at most $\frac{n}{\log^c n}$ are marked as real and all others are marked as dummy.
- **Procedure:**
  1. Let $\mu := \lfloor \log n \rfloor$ and $t := \lceil \log \log n \rceil$. Let $\lambda := 1/64$ and let $G_{\lambda,\star}$ be the family of d_λ-regular expander graphs from Theorem 3. Let $B := \lfloor d_\lambda/2 \rfloor$.
  2. *Interpret every μ balls as one block:* Interpret $\mathbf{I}$ as an array A_0 of n/μ blocks so that each block $A_0[i]$ consists of μ balls. For each block $A_0[i]$, in parallel, mark $A_0[i]$ real if $A_0[i]$ consists of at least one real ball (and mark dummy otherwise).
     Initialize the routing graph H: Let $m := n/(B \cdot \mu)$. For each $j \in \{0, \ldots, t-1\}$, let $G_{\lambda,m/2^j} = (L_j, R_j, E_j)$ be the expander defined in Theorem 3 so that $L_j = R_j = [m/2^j]$. Then, for each $j \in \{0, \ldots, t-2\}$, connect every two vertices from R_j to each vertex in L_{j+1}; That is, for each $i \in [m/2^{j+1}]$, add two edges from $2i-1, 2i \in R_j$ to $i \in L_{j+1}$. Let H be the resulting graph.
     Initialize an indicator bit to 0 for all vertices: For each $j = 0, \ldots, t-1$, initialize M_j and $\overline{M}_j$ as two arrays each consists of $m/2^j$ 0-bits.
  3. *Mark each vertex on H if the vertex is reachable by any real block:*
     For each $i \in [m]$, if any block in $A_0[(i-1) \cdot B + 1, \ldots, i \cdot B]$ is real, then set $M_0[i] := 1$.
     For j from 0 to $t-2$ sequentially, compute the array M_{j+1} indicators as follows:
     - **a.** For each $r \in [d_\lambda]$, perform the following sequentially:
       For all $v \in [m/2^j]$ in parallel, set $\overline{M}_j[\Gamma_r(v)] := 1$ if $M_j[v] = 1$, where $\Gamma_r(v)$ denotes the r-th neighbor of the vertex v in the expander $G_{\lambda,m/2^j}$ as defined in Theorem 3.
     - **b.** For all $i \in [m/2^{j+1}]$ in parallel, set $M_{j+1}[i] := 1$ if and only if $\overline{M}_j[2i-1] = 1$ or $\overline{M}_j[2i] = 1$.
  4. *Compute the $(B, B/4)$-*matching*:* For all j from 0 to $t-1$, *in parallel*, run $S_j \leftarrow \mathsf{SlowMatch}(M_j)$ (from Claim 18).
  5. *Route all real blocks via the computed $(B, B/4)$-*matching*s:* For each j from 1 to t, let A_j be an array of $\frac{n}{\mu \cdot 2^j}$ blocks initialized with dummy. Then, for each j from 0 to $t-1$ sequentially, route all real blocks from A_j to A_{j+1} using the matching S_j as follows:
     - For each $r \in [d_\lambda]$, perform the following sequentially:
       - **a.** For all $v \in [m/2^j]$ in parallel, do the following:
         - **i.** If $S_j[r, v] = 1$, proceed the following; Otherwise, perform fake accesses.
         - **ii.** Sequentially find a real block in $A_j[(v-1) \cdot B + 1, \ldots, v \cdot B]$, and move the real block to a scratch space $b[v]$.
         - **iii.** Let $u = \Gamma_r(v)$ be the vertex in the expander $G_{\lambda,m/2^j}$. Sequentially find an empty block in $A_{j+1}[(u-1) \cdot \frac{B}{2} + 1, \ldots, u \cdot \frac{B}{2}]$ and then overwrite this empty block with $b[v]$.
- **Output:** Interpret the array A_t as balls and output the interpreted array of balls.

algorithm (e.g., Theorem 2 which consumes $O(n)$ work and has $O(\log n)$ depth). The formal description is given next. For simplicity of notation in the recursive algorithm, we assume that n is a global fixed parameter.

Proof of Lemma 6. See the full version for detailed analysis. ◀

Algorithm 5 Swapper : ϵ-swapper for $\epsilon(\star) = 1/\log^c \star$.

- **Input:** An array $\mathbf{I}$ of size $\leq n$ in which all balls are marked red, blue or $\perp$. In the outermost recursion (i.e., $|\mathbf{I}| = n$), it is guaranteed that the number of red balls equals the number of blue balls.
- **The algorithm:**
  1. **Base case: array is short enough to run oblivious sort.**
     a. If $|\mathbf{I}| \leq n/\log n$: run oblivious sort (e.g., AKS; see Theorem 2) so that blue balls are in the front and red balls are in the back, swap (blue, red) balls which reside in symmetric locations (from the front and back), and reverse the previous oblivious sort (i.e., identical to Step 4 of Algorithm 1). Output the resulting array.
  2. **Swap all but an $O(1)$ fraction of balls.**
     a. Run $\mathbf{I}' \leftarrow \mathsf{ImWeakSwapper}(\mathbf{I})$.
  3. **Compress the array.**
     a. Consider all balls that are not marked red or blue in $\mathbf{I}'$ as dummies. Consider all the remaining red or blue balls as reals.
     b. Run $\mathsf{WeakCompression}(\mathbf{I}')$, and let $\mathbf{I}''$ and $\mathbf{E}$ be the results. Note that $\mathbf{I}''$ is of size $\left|\mathbf{I}'\right|/2$ and it contains all the reals from $\mathbf{I}'$ except for a γ-fraction, and $\mathbf{E}$ is of size $\left|\mathbf{I}'\right|$ and it contains the $\gamma = \frac{1}{\log^{c_1}|\mathbf{I}'|}$ fraction of reals that are not in $\mathbf{I}''$ (in the same positions). Record all ball movements during this step in Aux.
  4. **Continue recursively.**
     a. Run this algorithm Swapper recursively on $\mathbf{I}''$. Let $\mathbf{O}'$ be the result.
  5. **Reverse route.**
     a. Reverse route all real balls from $\mathbf{O}'$ and $\mathbf{E}$ back into $\mathbf{I}'$ using Aux, and let $\mathbf{O}$ be the resulting array (note that $|\mathbf{O}| = |\mathbf{I}|$).
- **Output:** The array $\mathbf{O}$.

6.4 Imbalanced Weak Swapper (Lemma 7)

▶ **Lemma 7** (Imbalanced weak swapper). *For every constant $\ell \in \mathbb{N}$, there exists a deterministic oblivious procedure ImWeakSwapper that implements an $(1/\ell)$-imb-swapper in the PRAM model. Letting w be the word size, n be the number of balls in the input array, and D be the size of each ball in bits, ImWeakSwapper consumes $O(\lceil D/w \rceil \cdot n)$ work and $O(1)$ depth.*

A procedure implementing $(1/\ell)$-swapper for all $\ell \in \mathbb{N}$ with total linear work was developed in Asharov et al. [3, Claim 5.7], and it actually implements $(1/\ell)$-imb-swapper for the same ℓ (we will prove this claim later in this subsection). However, the depth of their procedure is also linear. While this is insufficient for our purposes, we still use their ideas as a starting point. Let us recall the high-level details of their construction.

The procedure instantiates a d-regular bipartite expander (with sufficient expansion depending on ℓ) for $d \in O(1)$ that consists of n vertices on both sides. Every ball is associated with a vertex on the left, and for every two vertices that share the same neighbor, the two balls are swapped if and only if the labels are (red, blue). By the vertex expansion of the bipartite expander, only a $1/\ell$ fraction of misplaced balls may remain not swapped. The algorithm clearly requires linear work as the graph contains a linear number of edges, however parallelizing it is challenging. Concretely, every vertex on the bipartite expander has d neighbors, and so using a naive parallelization a node could be swapped with several other nodes simultaneously, and it is not clear how to resolve conflicts in low depth.

To get over this we use Property 2 in the expander of Theorem 3. Namely, we partition the edge set (of the bipartite expander) into disjoint perfect matchings (which can be computed efficiently), and then perform the swaps in within the matchings in parallel as below.

Given the disjoint perfect matchings $M_1, \ldots, M_d$ from Theorem 3, for every pair $i, j \in [d]$, we want to swap each pair of red and blue balls that are the 2-edge neighbors on the subgraph $M_i \cup M_j$. As the perfect matchings are parallel-friendly, one straw-man solution is to route all n balls via both M_i and M_j from left to right (so there are two copies for each ball), swap every pair of red and blue for every vertex on the right side, and then route balls backward via M_i and M_j, where the routing and swapping are performed in parallel. However, the straw-man solution doesn't work as every vertex has *two* neighbors on $M_i \cup M_j$, and a red ball may be swapped with both two blue balls in parallel, which still incurs a conflict. To this end, our second observation is that it suffices to copy only reds via M_i, copy only blues via M_j, and then swap the pairs if needed; Given a ball is either red or blue exclusively, every ball has at most one copy now. We formally describe the algorithm in the following.

Algorithm 6 ImWeakSwapper: ϵ-imb-swapper($\mathbf{I}$) for $\epsilon(\star) = 1/\ell$

- **Input:** An array $\mathbf{I}$ of n balls, each ball is labeled as red, blue or $\perp$.
- **Parameters:** A parameter $\ell \in \mathbb{N}$.
- **The algorithm:**
  1. Let $\lambda := \frac{1}{2\sqrt{\ell}}$, and let d_λ be the vertex degree given by Theorem 3.
  2. For each $(i_1, i_2) \in [d_\lambda]^2$, perform the following sequentially.
     a. Let M_{i_1} (resp. M_{i_2}) be the i_1-th (resp. i_2-th) perfect matching given in Theorem 3.
     b. For all edges $(k_1, j) \in M_{i_1}$ and $(k_2, j) \in M_{i_2}$, do the following: If $(\mathbf{I}[k_1], \mathbf{I}[k_2])$ are labeled as (red, blue), then swap between $\mathbf{I}[k_1]$ and $\mathbf{I}[k_2]$. Label both as $\perp$. Otherwise, perform dummy swap. That is realized as below, where all loops are performed in parallel.
        i. Initialize two arrays R_1, R_2, each consists of n empty balls labeled as $\perp$, For all $j \in [n]$, let $k_1 := \Gamma_{i_1}(j)$ and $k_2 := \Gamma_{i_2}(j)$ (so that (k_1, j) is an edge in M_{i_1} and (k_2, j) is an edge in M_{i_2} for each j by property 2 of Theorem 3).
        ii. For each edge (k_1, j) in M_{i_1}, let $R_1[j] := \mathbf{I}[k_1]$ if $\mathbf{I}[k_1]$ is red.
        For each edge (k_2, j) in M_{i_2}, let $R_2[j] := \mathbf{I}[k_2]$ if $\mathbf{I}[k_2]$ is blue.
        iii. For each $j \in [n]$, if the pair $(R_1[j], R_2[j])$ is labeled (red, blue), then swap between $R_1[j]$ and $R_2[j]$), label both as $\perp$. Otherwise, perform dummy swap.
        iv. For each edge (k_1, j) in M_{i_1}, let $\mathbf{I}[k_1] := R_1[j]$ if $\mathbf{I}[k_1]$ is red.
        For each edge (k_2, j) in M_{i_2}, let $\mathbf{I}[k_2] := R_2[j]$ if $\mathbf{I}[k_2]$ is blue.
- **Output:** The array $\mathbf{I}$.

Proof of Lemma 7. See the full version for detailed analysis. ◀

6.5 Weak Compression (Lemma 12)

▶ **Lemma 12** (Weak compression). *There exists a constant $\alpha \in (0, 1/2)$, such that for all constants $c \in \mathbb{N}$, letting $\gamma(\star) = 1/\log^c \star$, there exists a deterministic oblivious procedure WeakCompression that implements a γ-approx-$(\alpha, 1/2)$-compression in the PRAM model. Letting w be the word size, n be the number of balls in the input array, and D be the size of each ball in bits, WeakCompression consumes $O(\lceil D/w \rceil \cdot n)$ work and $O(\log \log n)$ depth.*

Let $c \in \mathbb{N}$ be the constant for which we wish to implement $(1/\log^c \star)$-approx-$(\alpha, 1/2)$-compression for some $\alpha \in (0, 1/2)$ to be determined shortly. We implement this procedure using FastCompression (from Lemma 10) and SlowWeakCompression (from Lemma 13). Particularly, we use SlowWeakCompression which implements $(1/\log^{c_1} \star)$-approx-$(\alpha_1, 1/2)$-compression, for $c_1 = 2c + 2$ and some $\alpha_1 \in (0, 1/2)$, using super linear work but doubly logarithmic depth, and FastCompression which implements $(\alpha_2, 1/2)$-compression for some $\alpha_2 \in (0, 1/2)$ using linear work and depth. We let $\alpha = \alpha_1 \cdot \alpha_2/4$.

Algorithm 7 WeakCompression: $1/\log^c \star$-approx-$(\alpha, 1/2)$-compression.

- **Public parameters:** Size of input array n,
- **Input:** An array $\mathbf{I}$ with n balls each of size D bits, where at most $\alpha \cdot n$ balls are real and the rest are dummy.
- **The procedure:**

1. Let $\mu := \min(\log w, \log\log n)$.
2. **Compress the array, keeping most of the dense blocks.**
   a. Represent $\mathbf{I}$ as another array A that consists of $m := n/\mu$ blocks each of size $\mu \cdot D$ bits: for each $i \in [m]$, let $A[i]$ be the block consists of all balls $\mathbf{I}[(i-1) \cdot \mu + 1], \ldots, \mathbf{I}[i \cdot \mu]$.
   b. For each $i \in [m]$, label $A[i]$ as *dense* if $A[i]$ consists of more than $\mu \cdot \alpha_2/2$ real balls.
   c. Run $(\mathbf{O}_1, \mathbf{E}_1) \leftarrow \mathsf{SlowWeakCompression}(A)$, where $|\mathbf{O}_1| = n/2$ and $|\mathbf{E}_1| = n$ (in number of balls). Record all moves in array Aux_1.
   d. Repeat the above process, this time on the array $\mathbf{O}_1$: interpret it as $m/2$ blocks each of size $\mu \cdot D$, mark dense blocks as before, and let $(\mathbf{O}_2, \mathbf{E}_2) \leftarrow \mathsf{SlowWeakCompression}(\mathbf{O}_1)$, where $|\mathbf{O}_2| = n/4$, $|\mathbf{E}_2| = n/2$ (in number of balls). Record all moves in array Aux_2.
   e. Using Aux_1, reverse route the real balls in $\mathbf{E}_2$ back into $\mathbf{O}_1$, and then using Aux_2, reverse route and merge real balls from $\mathbf{E}_1$ and $\mathbf{O}_1$ back into an error array $\mathbf{E}$ of size n (recall that $\mathbf{E}_2$ is in fact $\mathbf{O}_1$ where some elements were excluded into $\mathbf{O}_2$; reversing $\mathbf{E}_1$ and $\mathbf{O}_1$ to A is also possible using Aux_1).
3. **Compress the sparse blocks.**
   a. Replace all dense blocks in A with dummy blocks. For every $i \in [n/\mu]$, in parallel, run $\mathbf{O}_{3,i} \leftarrow \mathsf{FastCompression}(A[i])$, where $A[i]$ is interpreted as μ balls, and then again $\mathbf{O}'_{3,i} \leftarrow \mathsf{FastCompression}(\mathbf{O}_{3,i})$. Note that $|A[i]| = \mu$ and $\left|\mathbf{O}'_{3,i}\right| = \mu/4$.
4. Set $\mathbf{O} = \mathbf{O}_2 \| \mathbf{O}'_{3,1} \| \ldots \| \mathbf{O}'_{3,n/\mu}$ (which is of total size $n/2$, as $|\mathbf{O}_2| = n/4$ and $\sum_{i=1}^{n/\mu} \left|\mathbf{O}'_{3,i}\right| = n/4$.
5. **Output:** $\mathbf{O}$ and $\mathbf{E}$.

Proof of Lemma 12. See the full version for detailed analysis. ◀

6.6 Slow Weak Compression (Lemma 13)

▶ **Lemma 13** (Slow weak compression). *There exists a constant $\alpha \in (0, 1/2)$ such that for all constants $c \in \mathbb{N}$, letting $\gamma(\star) = 1/\log^c \star$, there exists a deterministic oblivious procedure SlowWeakCompression that implements a γ-approx-$(\alpha, 1/2)$-compression in the PRAM model. Letting w be the word size, n be the number of balls in the input array, and D be the size of each ball in bits, SlowWeakCompression consumes $O(n \cdot \log\log n + \lceil D/w \rceil \cdot n)$-work and $O(\log\log n)$ depth.*

In our implementation, $\alpha = 1/128$. Let $c \in \mathbb{N}$ be the constant for which we wish to implement $1/\log^c \star$-approx-$(\alpha, 1/2)$-compression. The algorithm SlowWeakCompression uses a sub-procedure WeakSlowMatch from Claim 19 (see Algorithm 3). Particularly, we use WeakSlowMatch that implements $(\log^{c/2} \star)$-approx-$(B, B/4)$-matching on the graph $G_{\lambda,n}$ (from Theorem 3) with $\lambda = 1/64$, regularity d_λ, and $B = d_\lambda/2$.

Algorithm 8 SlowWeakCompression: $1/\log^c \star$-approx-$(\alpha, 1/2)$-compression.

- **Input:** An array $\mathbf{I}$ of n balls, in which at most $\alpha \cdot n$ are real.
- **The Procedure:**
  1. Interpret the array $\mathbf{I}$ as $m := n/B$ bins, where each bin consists of B balls. Mark all bins in $\mathbf{I}$ as dense or sparse, where a bin is dense if it consists of more than $B/4$ real balls. Let S be an array of m indicators representing the set of indexes of the dense bins. Let $\mathbf{I}'$ be an array of m empty bins, where the capacity of a bin is B balls.
  2. Let $G_{\lambda,m} = (L, R, E)$ be the d_λ-regular bipartite graph guaranteed by Theorem 3, where $|L| = |R| = m$.
  3. Compute $M \leftarrow \mathsf{WeakSlowMatch}(S)$.
  4. **Distribute:** For each edge $(u, v) \in E$ (where $u \in L, v \in R$), if $(u, v) \in M$, move a real ball from bin $\mathbf{I}[u]$ to bin $\mathbf{I}'[v]$ and then mark bin $\mathbf{I}[u]$ as sparse. This step is achieved in the following parallel way. Recall that WeakSlowMatch outputs M as a $(d_\lambda \times m)$-array of indicators such that $M[r, u] = 1$ iff the r-th edge of vertex $u \in L$ is in the $(B, B/4)$-matching. For each $r \in [d_\lambda]$, perform the following sequentially:
     a. For all $u \in [m]$, in parallel, do the following:
        i. If $M[r, u] = 1$, proceed with the following (otherwise, perform fake accesses):
        ii. Sequentially read every ball in bin $\mathbf{I}[u]$ and fetch the first encountered real ball, then sequentially read every slot in bin $\mathbf{I}'[\Gamma_r(u)]$ and write the fetched real ball to the first encountered empty slot.
        iii. Mark bin $\mathbf{I}[u]$ as sparse.
  5. **Fold:** Let $\mathbf{O}$ be an array of size $m/2$ empty bins, each of capacity of B balls. For all $i \in [m/2]$, in parallel, move all real balls from the bins marked sparse in the four bins $\mathbf{I}[i], \mathbf{I}[m/2 + i], \mathbf{I}'[i], \mathbf{I}'[m/2B + i]$ into bin $\mathbf{O}[i]$, and pad $\mathbf{O}[i]$ with dummy balls if there are less than B real balls.
- **Output:** The array $\mathbf{O}$, as well as the input array $\mathbf{I}$.

Proof of Lemma 13. See the full version for detailed analysis. ◀

6.7 Fast Compression (Lemma 10)

▶ **Lemma 10** (Fast compression for short inputs)**.** *There exists a constant $\alpha \in (0, 1/2)$ for which there exists a deterministic oblivious procedure FastCompression that implements an $(\alpha, 1/2)$-compression in the PRAM model. Letting w be the word size, $n \leq w/\log w$ be the number of balls in the input array, and D be the size of each ball in bits, FastCompression consumes $O(\lceil D/w \rceil \cdot n)$-work and $O(n)$ depth.*

The algorithm is the same as Algorithm 8, while using FastMatch from Claim 20 instead of WeakSlowMatch at Step 3. Because FastMatch implements $(B, B/4)$-matching, the resulting matching M is capable of distributing all real balls in every dense bins at Step 4, correctness follows directly (so there is no need to calculate the number of real balls remains in $\mathbf{I}$). The work and depth follows also immediately from Claim 20.

References

1. Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1974.
2. Miklós Ajtai, János Komlós, and Endre Szemerédi. An $O(n \log n)$ sorting network. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, STOC*, pages 1–9, 1983.
3. Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, Enoch Peserico, and Elaine Shi. OptORAMa: optimal oblivious RAM. In *Advances in Cryptology - EUROCRYPT*, 2020.
4. Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Enoch Peserico, and Elaine Shi. Oblivious parallel tight compaction. Cryptology ePrint Archive, Report 2020/125, 2020. URL: `https://eprint.iacr.org/2020/125`.
5. Hannah Bast and Torben Hagerup. Fast parallel space allocation, estimation, and integer sorting. *Inf. Comput.*, 123(1):72–110, November 1995.
6. Elette Boyle, Kai-Min Chung, and Rafael Pass. Oblivious parallel RAM and applications. In *Theory of Cryptography - 13th International Conference, TCC*, pages 175–204, 2016.
7. Elette Boyle and Moni Naor. Is there an oblivious RAM lower bound? In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, ITCS*, pages 357–368, 2016.
8. Hubert Chan, Kai-Min Chung, and Elaine Shi. On the depth of oblivious parallel oram. In *Asiacrypt*, 2017.
9. T.-H. Hubert Chan, Kartik Nayak, and Elaine Shi. Perfectly secure oblivious parallel RAM. In *Theory of Cryptography - 16th International Conference, TCC 2018*, pages 636–668, 2018.
10. T.-H. Hubert Chan and Elaine Shi. Circuit OPRAM: unifying statistically and computationally secure orams and oprams. In *Theory of Cryptography - 15th International Conference, TCC*, pages 72–107, 2017.
11. Richard Cole. Parallel merge sort. *SIAM Journal on Computing*, 17(4):770–785, 1988.
12. S. Cook, C. Dwork, and R. Reischuk. Upper and Lower Time Bounds for Parallel Random Access Machines without Simultaneous Writes. *SIAM Journal on Computing*, 15(1):87–97, February 1986.
13. Alireza Farhadi, MohammadTaghi Hajiaghayi, Kasper Green Larsen, and Elaine Shi. Lower bounds for external memory integer sorting via network coding. In *STOC*, 2019.
14. Ofer Gabber and Zvi Galil. Explicit constructions of linear-sized superconcentrators. *J. Comput. Syst. Sci.*, 22(3):407–420, 1981.
15. Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, STOC*, pages 182–194, 1987.
16. Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 1996.
17. Michael T. Goodrich and Michael Mitzenmacher. Privacy-preserving access of outsourced data via oblivious RAM simulation. In *Automata, Languages and Programming - 38th International Colloquium, ICALP*, pages 576–587, 2011.
18. P. Hall. On Representatives of Subsets. *Journal of the London Mathematical Society*, s1-10(1):26–30, 1935.
19. Shuji Jimbo and Akira Maruoka. Expanders obtained from affine transformations. *Combinatorica*, 7(4):343–355, 1987.
20. Donald E. Knuth. *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., 1998.
21. Eyal Kushilevitz, Steve Lu, and Rafail Ostrovsky. On the (in)security of hash-based oblivious RAM and a new balancing scheme. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 143–156, 2012.

22 Tom Leighton, Yuan Ma, and Torsten Suel. On probabilistic networks for selection, merging, and sorting. In *Proceedings of the Seventh Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '95, pages 106–118. ACM, 1995.

23 Wei-Kai Lin, Elaine Shi, and Tiancheng Xie. Can we overcome the $n \log n$ barrier for oblivious sorting? In *SODA*, 2019.

24 Grigorii Aleksandrovich Margulis. Explicit constructions of concentrators. *Problemy Peredachi Informatsii*, 9(4):71–80, 1973.

25 John C. Mitchell and Joe Zimmerman. Data-oblivious data structures. In *31st International Symposium on Theoretical Aspects of Computer Science STACS*, pages 554–565, 2014.

26 Enoch Peserico. Deterministic oblivious distribution (and tight compaction) in linear time. *CoRR*, abs/1807.06719, 2018. URL: `http://arxiv.org/abs/1807.06719`.

27 Nicholas Pippenger. Self-routing superconcentrators. *J. Comput. Syst. Sci.*, 52(1):53–60, 1996.

28 P. Ragde. The parallel simplicity of compaction and chaining. In *Proceedings of the Seventeenth International Colloquium on Automata, Languages and Programming*, pages 744–751, Berlin, Heidelberg, 1990. Springer-Verlag.

29 Leslie G. Valiant. Graph-theoretic properties in computational complexity. *J. Comput. Syst. Sci.*, 13(3):278–285, December 1976.

On Polynomial Secret Sharing Schemes

Anat Paskin-Cherniavsky
Ariel University, Ariél, Israel
anatpc@ariel.ac.il

Radune Artiom
Ariel University, Ariél, Israel
The Open University, Raanana, Israel
tom.radune@gmail.com

Abstract

Nearly all secret sharing schemes studied so far are linear or multi-linear schemes. Although these schemes allow to implement any monotone access structure, the share complexity, SC, may be suboptimal – there are access structures for which the gap between the best known lower bounds and best known multi-linear schemes is exponential.

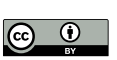
There is growing evidence in the literature, that non-linear schemes can improve share complexity for some access structures, with the work of Beimel and Ishai (CCC '01) being among the first to demonstrate it. This motivates further study of non linear schemes.

We initiate a systematic study of polynomial secret sharing schemes (PSSS), where shares are (multi-variate) polynomials of secret and randomness vectors $\vec{s}, \vec{r}$ respectively over some finite field $\mathbb{F}_q$. Our main hope is that the algebraic structure of polynomials would help obtain better lower bounds than those known for the general secret sharing. Some of the initial results we prove in this work are as follows.

On share complexity of polynomial schemes. First we study degree (at most) 1 in randomness variables $\vec{r}$ (where the degree of secret variables is unlimited). We have shown that for a large subclass of these schemes, there exist equivalent multi-linear schemes with $O(n)$ share complexity overhead. Namely, PSSS where every polynomial misses monomials of exact degree $c \geq 2$ in $\vec{s}$ and 0 in $\vec{r}$, and PSSS where all polynomials miss monomials of exact degree ≥ 1 in $\vec{s}$ and 1 in $\vec{r}$. This translates the known lower bound of $\Omega(n^{\log(n)})$ for multi linear schemes onto a class of schemes strictly larger than multi linear schemes, to contrast with the best $\Omega(n^2/\log(n))$ bound known for general schemes, with no progress since 94'. An observation in the positive direction we make refers to the share complexity (per bit) of multi linear schemes (polynomial schemes of total degree 1). We observe that the scheme by Liu et. al obtaining share complexity $O(2^{0.994n})$ can be transformed into a multi-linear scheme with similar share complexity per bit, for sufficiently long secrets. For the next natural degree to consider, 2 in $\vec{r}$, we have shown that PSSS where all share polynomials are of exact degree 2 in $\vec{r}$ (without exact degree 1 in $\vec{r}$ monomials) where $\mathbb{F}_q$ has odd characteristic, can implement only trivial access structures where the minterms consist of single parties.

Obtaining improved lower bounds for degree-2 in $\vec{r}$ PSSS, and even arbitrary degree-1 in $\vec{r}$ PSSS is left as an interesting open question.

On the randomness complexity of polynomial schemes. We prove that for every degree-2 polynomial secret sharing scheme, there exists an equivalent degree-2 scheme with identical share complexity with randomness complexity, RC, bounded by $2^{poly(SC)}$. For general PSSS, we obtain a similar bound on RC (preserving SC and $\mathbb{F}_q$ but not degree). So far, bounds on randomness complexity were known only for multi linear schemes, demonstrating that $RC \leq SC$ is always achievable. Our bounds are not nearly as practical as those for multi-linear schemes, and should be viewed as a proof of concept. If a much better bound for some degree bound $d = O(1)$ is obtained, it would lead directly to super-polynomial counting-based lower bounds for degree-d PSSS over constant-sized fields. Another application of low (say, polynomial) randomness complexity is transforming polynomial schemes with polynomial-sized (in n) algebraic formulas $C(\vec{s}, \vec{r})$ for each share, into a degree-3 scheme with only polynomial blowup in share complexity, using standard randomizing polynomials constructions.



1st Conference on Information-Theoretic Cryptography (ITC 2020).
Editors: Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs; Article No. 12; pp. 12:1–12:21

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

2012 ACM Subject Classification Theory of computation; Theory of computation → Cryptographic primitives

Keywords and phrases Secret sharing, polynomial, lower bounds, linear program

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.12

Related Version A full version of this article is available at `https://eprint.iacr.org/2019/361`.

Funding *Anat Paskin-Cherniavsky*: This work was supported by The Ariel Cyber Innovation Center in conjunction with the Israel National Cyber directorate in the Prime Minister's Office.

1 Introduction

Secret sharing is a primitive allowing a dealer to share a secret s among n players. The secret sharing scheme implements a (monotone) access structure $\mathcal{A} \subseteq 2^{[n]}$ if any $A \in \mathcal{A}$ can learn the secret from their joint share vector (A is called qualified set), and any set $B \notin \mathcal{A}$ learns nothing about the secret (B is called unqualified set). Secret sharing was introduced in '79 by Shamir [19] and Blakley [9] for threshold access structures, and was followed by thousands of works exploring the primitive itself, and its many applications found since. Quite early on [7,15] put forward a first construction realizing any monotone access structure. As a notable application, secret sharing is used as a key building block in various secure Multi-Party Computation (MPC) constructions [6,12].

Arguably, the most important complexity measure of a secret sharing scheme is its share complexity (SC). Share complexity is the maximum, over the parties' share length, received from the dealer by any of the parties. A somewhat relaxed measure is its information rate, which is the share complexity *per shared bit*. It can be viewed as "amortized" share complexity, which is a useful measure if secrets are allowed to be long.

Unfortunately, there is a huge gap in our understanding of this measure. Namely, the best known lower bound on share complexity for a general scheme is $\Omega(n/\log(n))$ [10], while the best known constructions for certain access structures have exponential complexity $O(2^{0.637n})$ [2]. In [10], techniques from information theory are used, characterizing the existence of a secret sharing scheme in terms of requirements on the entropy of various distributions. The lower bound in [10] is on information rate (making it stronger) and states an explicit access structure for which it holds. It is important to note that counting arguments do not work for general secret sharing schemes.[1]

In spite of extensive research attempting to improve [10]'s lower bound, the best known lower bound for general schemes has not improved since (even for implicit access structures). A major motivation for this work is the hope that departing from previous approaches relying mostly on information theoretic techniques, making use of algebraic techniques could potentially yield improved lower bounds for large classes of schemes, and hopefully eventually for general schemes. See [4] and references therein, for example, for a more thorough discussion of the many positive and negative results on share complexity of secret sharing schemes, as well as their numerous applications.

[1] In a nutshell, even if randomness domain is polynomially bounded in the share complexity, we still get a double-exponential number of secret sharing schemes of share complexity $O(n/\log(n))$, which is about the number of monotone access structures.

(Multi-)linear schemes

On the other hand, much more is known about the share complexity of the well studied family of linear secret sharing schemes, and more generally multi linear secret sharing schemes. In a nutshell, a linear scheme is a scheme, where each share is a linear combination of elements from a finite field $\mathbb{F}$, each of which is either the secret or a random variable, while a multi-linear scheme is a scheme where the secret can be vector of elements from $\mathbb{F}$ and the shares are a linear combination of these elements and the random variables. Linear schemes are relatively easy to design, often exploiting the insights and intuition we have into linear algebra. Perhaps a more important reason for their popularity is their "homomorphic" property. In MPC, for example, linear schemes are a useful building block, as they allow computing a sharing of the sum of shared secrets by locally adding the corresponding shares. Even more importantly, for (multi) linear schemes better lower bounds on share complexity are also known. In particular, counting arguments yield exponential lower bounds for non-explicit access structures, and recently, an exponential lower bound has been obtained on the share complexity of linear schemes for an explicit access structure. See next section for more details. For now, the observation important for discussion is that as well as upper bounds, lower bounds for (multi) linear secret sharing schemes heavily exploit the (linear-)algebraic structure of the sharing scheme.

Motivated by the hope to narrow the gap between upper and lower bounds for share complexity and information rate in secret sharing schemes, in this work, we continue the work of [5], which initiates a study of the power of non-linear secret sharing schemes. The main motivation in [5] for studying non-(multi) linear schemes is that most constructions of secret sharing schemes so far were either linear or multi linear, so new insights both on upper and lower bounds may be gained. Indeed [5] put forward several innovative secret sharing schemes for access structures for which linear schemes of comparable complexity are not known, or even do not exist under reasonable assumptions. In [5] the authors explore both arbitrary non-linear schemes, and a specific generalization of linear schemes, they refer to as *quasi-linear* schemes.

We have the additional motivation of obtaining new lower bounds for a broader class of schemes than linear and multi linear ones, making a step forward towards improved lower bounds for general schemes, which proved notoriously hard so far.

More specifically, we chose to explore the arguably natural extension of multi linear schemes, we call *polynomial schemes*, or PSSS. A PSSS is defined as multi linear scheme over a finite field $\mathbb{F}$, where each share is some polynomial over $\mathbb{F}$ in the secret and randomness elements, rather than necessarily a degree-1 polynomial (corresponding to a multi linear scheme). We hope that the rich algebraic structure of polynomials - especially of polynomials of low degree, say 2, would help develop techniques for lower bounds of more *algebraic* nature, as they proved useful for linear and multi linear schemes. A slightly more general notion of polynomial schemes is one where where the secret domain S is a subset of $\mathbb{F}^k$, rather than the entire set $\mathbb{F}^k$. We refer to such schemes as *generalized* polynomial schemes.

Besides the potential for useful analytic techniques, we believe PSSS is a useful set of schemes to study as it is very broad. In particular, as any function $f : \mathbb{F}^n \to \mathbb{F}$ can be represented by an n-variate polynomial over $\mathbb{F}$, it takes a moment to think why not every secret sharing scheme can be represented by a PSSS with the same share complexity. The reason is that a secret sharing scheme is a randomized mapping $Sh : S \times R \to S_1 \times \ldots \times S_n$, rather than a deterministic function. In Sh, the randomness is uniformly sampled from a finite set R. Now observe that in any PSSS scheme $Sh' : \mathbb{F}_p^s \times \mathbb{F}_p^r$ over a finite field $\mathbb{F}_p$, the probability of outputting any share vector is a multiple of p^{-r}. The straightforward way to

convert from Sh into an equivalent scheme Sh' as above is to embed S and R into $\mathbb{F}_p^s, \mathbb{F}_p^r$ for some s, r respectively, and evaluate the shares as polynomials corresponding to every share $Sh_i(s, r)$ (which are guaranteed to exist). More precisely, arbitrarily partition $\mathbb{F}_p^r$ into $|R|$ equal parts $R'_1, \ldots, R'_{|R|}$, the embedding labels every element of R'_j by r_j and sets Sh' accordingly. The problem with this approach in perfect secret sharing is that p^r may not be divisible by $|R|$ for any prime p and any r. For instance, for $|R| = 6$ in Sh there is no such embedding, as $1/6$ can not be written as $\frac{a}{p^r}$ for any prime p and $a \in \mathbb{N}$. We note that the above approach of transformation into PSSS (over any field $\mathbb{F}_p$) does work for statistical secret sharing, by choosing a sufficiently large r and R_j's of almost equal size, making the privacy "leakage" arbitrarily small, and keeping correctness perfect. In this work we focus on the standard notion of perfect secret sharing schemes, though.

1.1 Our Results

Feasibility and share complexity lens

On the negative side, we show that a large subclass of PSSS with r-degree 1 is equivalent to multi-linear schemes in the sense that for each such scheme, a multi-linear scheme for the same access structure with (almost) the same share complexity per secret bit and over the same field exists.

▶ **Theorem 1** (Informal). *Let $\mathcal{M}$ be a PSSS of degree 1 in $\vec{r}$, where all share polynomials are either missing monomials of (exact) degree $c \geq 2$ in $\vec{s}$ and 0 in $\vec{r}$, or all share polynomials miss monomials of exact degree ≥ 1 in $\vec{s}$ and 1 in $\vec{r}$. Then there exists an equivalent multi linear scheme $\mathcal{M}'$ with share complexity at most n times that of $\mathcal{M}$.*

We conjecture that all schemes with $\vec{r}$-degree 1 are as weak as multi-linear schemes, and leave it as an interesting open problem. See Theorem 9 and Theorem 11 for a formal statement and a proof of the above theorem. The proofs of both theorems are constructive, transforming the r-degree 1 schemes into multi linear schemes. The validity of the constructions is proved by rather simple linear algebraic techniques, but the constructions themselves, especially that of Theorem 9 are somewhat surprising, in our opinion.

Moving to the next natural class of $\vec{r}$-degree 2, we show that a certain natural subclass of such PSSS only allows to implement a small subset of access structures (regradless of share complexity).

▶ **Theorem 2** (Informal). *PSSS of degree exactly 2 in $\vec{r}$ over fields of odd characteristic capture only access structures where all minterms are singletons.*

That is, somewhat intuitively, linear terms are required in degree-2 schemes for implementing useful access structures. The proof here relies on facts regarding the number of solutions of equations of the form $p(x_1, \ldots, x_n) = b$, where b is a quadratic form.

To contrast with the bounds in [14] on functions representable by polynomial-sized randomizing polynomials with r-degree 2 and any constant degree in s (over small fields), indicating the corresponding functions are relatively simple, falling in NC_3. The reason why their bound does not directly imply that PSSS of r-degree 2 and polynomial share complexity works for relatively simple schemes, is that their bound holds for representations polynomial in input size. In particular, they assume the randomness vector's size is polynomially bounded in the input vector's size. For PSSS with $poly(n)$ randomness and share complexity we could indeed obtain a similar bound on the type of access structures for which such PSSS exists.

However, lacking bounds on the randomness complexity (see the following section), assuming only polynomial share complexity does not seem to suffice. [2]

On the positive side, we observe that a surprising recent result indicating all monotone access structures have a scheme construction share complexity $O(2^{0.994n})$ [18] can be replaced with a multi-linear construction (instead of a non-polynomial scheme).

We show that there exists (multi) linear secret sharing schemes based on the multi-linear CDS [1] with information rate O(1) for a certain class (not all) of access structures for a sufficiently large share domain.[3]

▶ **Observation 1.** *Let $n > 0$ be an integer. Then all monotone access structures on n parties admit a multi-linear scheme over $S = \mathbb{F}_2^{O(2^n)}$ with information rate $O(2^{0.994n})$ per party. (in our language, degree-1 polynomial scheme over $\mathbb{F}_2$).*

This observation demonstrates the power of amortization (increasing k) all else kept equal. Additionally, we can obtain a polynomial scheme of (possibly) high degree with the same share complexity.

▶ **Observation 2.** *Let $n > 0$ be an integer. Then all monotone access structures on n parties admit a polynomial scheme over $S = \mathbb{F}_{2^{O(2^n)}}$ with information rate of $O(2^{0.994n})$ per party.*

This is a direct corollary of Theorem 1. This holds due to the simple observation that any polynomial scheme over $\mathbb{F}_q^{k'}$, where q is a prime power (of any degree) can be replaced by a scheme where $S = \mathbb{F}_{q^{k'}}$, (that is, a scheme with $k = 1$) and the sharing polynomials are of possibly higher degree than the original ones. This is done by thinking of the vector of field elements in parties' shares and the vector of random field elements as vectors of elements over $\mathbb{F}_q^{k'}$, and the secret as an element of $\mathbb{F}^{q^{k'}}$. Then, the fact that any finite field $\mathbb{F}$ and function $\mathbb{F}^{1+r'} \to \mathbb{F}$ can be represented as a multi-variate polynomial over $\mathbb{F}$ implies that the original scheme can be implemented as a polynomial scheme with $k = 1$ over $\mathbb{F}_{q^{k'}}$. The overall share complexity overhead of this transformation is at most n, as the overall share complexity is at least $\log_2(|S|)$ to maintain perfect correctness. This general observation implies that there is certain redundancy regarding the usefulness of various parameters ($k, |F|$ and total degree) of polynomial schemes towards reducing share complexity. Namely, if we are free to adjust $\mathbb{F}$ and the degree arbitrarily, then without loss of generality k can be fixed to 1 without loss of generality.

Randomness complexity lens

An additional aspect that we have studied is the randomness complexity of PSSS. Here we study what is the best upper bound on the randomness complexity, as a function of the share complexity of a scheme – $\mathrm{RC}(SC)$. That is, for every scheme in the (sub) class of polynomial schemes with share complexity SC, there exists an equivalent scheme in the class with the same share complexity and randomness complexity at most $\mathrm{RC}(SC)$. For linear

[2] Still, if we had polynomial in share complexity upper bounds on randomness complexity, a modification of [14]'s result would yield bounds on this type of limited constant degree PSSS which are stronger than just counting-based bounds for constant-degree PSSS given suitable bounds on randomness complexity. Namely, not only do access structures that cannot be implemented efficiently exist, but there are candidates in relatively low complexity classes (under standard assumptions). See full version for details.

[3] The following pair of results are simple observations, which may be described and understood within the limits of the introduction, and we think they hope gain intuition on. The full proof of the first observation relies on particular details of [1]'s construction and is deferred to the full version. The proof of the second is simple and appears below.

and multi-linear schemes it is known that their randomness complexity is (without loss of generality) upper bounded by SC (the equivalent scheme is also over the same field). To the best of our knowledge, no such bounds appear in the literature for other broad classes of schemes. In particular, we have not found a bound for general (perfect) secret sharing schemes (we believe it was likely previously known).

In this work we put forward an upper bound for randomness complexity for general secret sharing schemes as well as various types of PSSS.

▶ **Theorem 3** (Informal). *Let $\mathcal{M}$ be a secret sharing scheme. Then, there exists an equivalent scheme $\mathcal{M}'$ with the same share complexity SC and randomness $RC = 2^{poly(SC)}$ such that if $\mathcal{M}'$ is a PSSS of degree 2, then so is $\mathcal{M}'$, and if $\mathcal{M}$ is a PSSS then so is $\mathcal{M}$. Also, in the two latter cases, $\mathcal{M}$ and $\mathcal{M}'$ are defined over the same field.*

The full proof of the theorem appears in the full version. To prove the bound for degree-2 PSSS, we restate the privacy requirements into sets of equality of distributions restrictions for single polynomials obtained using a variant of Vazirani's XOR lemma (already satisfied by $\mathcal{M}$). In particular, we prove there exists (via an explicit construction) a linear mapping from the vector space $span(r_1, \ldots, r_t)$ to a (much) smaller $span(r_1, \ldots, r_{t'})$ and every share polynomial $p(\vec{s}, \vec{r})$ is replaced by $p(\vec{s}, L(r_1), \ldots, L(r_n))$ so that privacy is still satisfied. The proof is based on a somewhat involved case analysis based on the theory on output distributions of quadratic forms. The bound for general secret sharing is proved using the following approach: given a PSSS scheme, we state the correctness and privacy requirements for any secret sharing scheme for the same access structure as an LP. Curiously, the LP formulation makes use of the scheme we already have at hand (with potentially high RC), rather than just a formulation of correctness and privacy. A solution to the LP determines the probabilities of mapping each secret s to each share vector $(\vec{\text{sh}}_1, \ldots, \vec{\text{sh}}_n)$, which easily extends into a PSSS over the same field and same share complexity. Briefly, the LP variables are probabilities $p_{i,k}$ where $\vec{s}_i$ is a secret and $\vec{sh}_k$ is a share vector. Privacy implies that for all maxterms A, and share vectors $\vec{\text{sh}}_A$ it must hold that

$$\sum_{\substack{all\ k\ for\ which\ the\ projection \\ of\ \vec{sh}_k\ on\ A\ is\ \vec{sh}_A}} p_{i,k} - \sum_{\substack{all\ k\ for\ which\ the\ projection \\ of\ \vec{sh}_k\ on\ A\ is\ \vec{sh}_A}} p_{j,k} = 0.$$

From correctness, it follows that for every minterm A, for every value $\vec{\text{sh}}_A$ all but at most $\vec{s}$, the projection value $\vec{\text{sh}}_A$ is seen with probability 0. This constraint would result in a degree-2 inequality in the $p_{\vec{s},\vec{\text{sh}}}$'s. To make it linear, the trick is to require that the 0 probabilities are exactly as in the scheme $\mathcal{M}$. That is, of every $(A, \vec{\text{sh}}_A)$ we require: $\sum_{\substack{all\ k\ for\ which\ the\ projection \\ of\ \vec{sh}_k\ on\ A\ is\ \vec{sh}_A \\ and\ j \notin I}} p_{j,k} = 0$, where I is either $\{i\}$ for some i, or empty, and is fixed according to $\mathcal{M}$. Finally, the requirement that $(p_{i,1}, \ldots, p_{i,l})$ is a probability vector is also expressed by linear inequalities. We look for solutions with small randomness vector length - as the LP has small integer entries, it easily follows that the probabilities are a multiple of some $1/L$, where L is not very large (exponential in LP dimensions). In particular, this implies a scheme with R of size L and same share complexity. This alone, already yields a bound on the randomness complexity ($\log(|R|)$) of general (perfect) secret sharing schemes. Given $\mathcal{M}$ is a PSSS, to obtain a PSSS with the required parameters it is necessary and sufficient that additionally the probabilities in the solution are powers of $q = |\mathbb{F}|$. In Theorem 14 we prove the former part, the proof of the stronger statement for $\mathcal{M}$ which is a PSSS is deferred to the full version.

All of the bounds above are exponential in SC and may serve as a proof of concept. A strong motivation here is that good upper bounds on randomness complexity $\mathrm{RC}(SC)$ for constant-degree PSSS would lead to good existential bounds on the share complexity of such PSSS which we do not currently have (over small enough $\mathbb{F}$). More concretely, for constant $\mathbb{F}$ and $poly(SC)$ randomness complexity there exist access structures with share complexity $2^{\Omega(n)}$ of PSSS over $\mathbb{F}$.

We stress that all our upper bounds on randomness complexity are for *perfect* secret sharing schemes, and are therefore require new techniques even in the general secret sharing and unbounded degree PSSS settings. For general non-PSSS (or PSSS) statistically secure schemes, partial derandomization techniques from the literature can be applied. In more detail, for ϵ-statistical secret sharing, bounds of $\ell(h) = O(SC + \log \epsilon)$ on randomness complexity can be easily obtained by replacing the randomness with the output of a non-boolean PRG (nb-PRG) [11] against the sharing algorithm, mapping from $\ell(h)$ random bits to h random bits as used by the sharing algorithm. By standard analysis similar to that in the proof of Claim 2 in [3]'s full version, a random function from ℓ to h bits is a suitable nb-PRG. Such results however are not useful for lower bounds, however. It is unclear whether nb-PRGs can be applied to constant-degree PSSS to yield even statistical secret sharing schemes, as the resulting sharing scheme does not necessarily remain low-degree (as the nb-PRG itself may be of high degree). Thus, good lower bounds for low-degree PSSS even in the statistical setting are left as an interesting open problem.

Roadmap

In Section 2 we provide the precise (standard) definition of secret sharing that we use, and introduce some new definitions and notations for PSSS. In Section 3, we present our results on feasibility and share complexity. Precise theorems and proofs of the randomness-related results and a broader survey of previous work from the perspective of PSSS implicit in it are deffered to the full version.

1.2 Open questions

In this work we have obtained some preliminary results on PSSS but many fundamental questions remain open.

▶ **Question 1** (Informal). *Do there exist access structures, that have non-polynomial schemes much more efficient than any PSSS?*

See a discussion on this question in the full version, with certain evidence in the positive direction. In a nutshell, it considers secret sharing constructions based on large matching vectors families such as [17], which are known to exist over rings $\mathbb{Z}_m$ of composite size but provably do not exist when m is a prime.

Other interesting questions concern understanding the effect of various parameters of PSSS on their power, in terms of achievable share complexity and information rate. There are various interesting parameters. One useful parameter is k - the length of the vector space $\mathbb{F}^k$ constituting the secret domain S. The distinction between $k = 1$ and arbitrary k is the difference between linear and multi-linear schemes, when considering PSSS of total degree $d = 1$. Generally, as we discuss below, the distinction between small secrets - $k = 1$ (or small k) appears meaningful in terms of achievable information rate - see further discussion in the full version. An Additional question to study is the effect of the particular field $\mathbb{F}_p$ on the power of the induced PSSS class.

A concrete natural question is obtaining lower bounds for low degree PSSS, say of degree $d = O(1)$. A simple approach for $k = 1$ would be to bound $|R|$ as a function of the share complexity, and then rely on the fact that there are few different degree-d polynomials in $R + 1$ variables (exponentially many in the share complexity) for a constant $\mathbb{F}_p$. The number of monotone access structures is double-exponential in n. For linear schemes, it is well known that wlog. $\log(|R|) \leq$ share complexity, leading to a $2^{\Omega(n)}$ lower bound on share complexity of linear schemes over any fixed $\mathbb{F}_p$. However, for any $d > 1$, there are no known explicit bounds on $|R|$ in terms of |share complexity|, so this approach does not currently work. In this work we make a first step in the direction of filling in the missing component, obtaining certain upper bounds on $|R|$ (as a function of share complexity). This leaves the following interesting question open.

▶ **Question 2** (informal). *Fix some finite field $\mathbb{F}_q$, and $d = O(1)$. Does there exist a polynomial bound $h(\cdot)$ on $|R|$ as a function of share complexity, such that any PSSS over $\mathbb{F}_q$ of degree d has an equivalent PSSS over $\mathbb{F}_q$ and degree q with the same share complexity, and $|R| \leq h(SC)$.*[4]

2 Preliminaries

General notation

In this work we consider finite fields $\mathbb{F}$. We write $\mathbb{F}_q$ to denote a field of size q (some prime power). For matrices M_1, M_2 (of the proper sizes) over some field $\mathbb{F}$, we denote by $(M_1|M_2)$ the matrix resulting from concatenating M_2 to the right of M_1, and $(M_1; M_2)$ results from concatenating M_2 below M_1. Vectors are denoted by $\vec{v}$ or just v when there is no risk of confusion (with scalars), and are by default column vectors. We let M_i denote the i'th row of M, and M^i its i'th column. We let M_I (M^I) denote a submatrix with rows (columns) restricted to I. For a matrix $M \in \mathbb{F}^{n\times n}$, we denote by $N \in \mathbb{F}^{m\times m}$ the matrix resulting from removing all row-column pairs such that $M^i = (M_i^T) = \vec{0}$.

Secret sharing

We use standard definitions of secret sharing schemes, following [4].

▶ **Definition 4** ([4]). *Access Structure: For a set of parties $\{p_1, ..p_n\}$ a subset $\mathcal{A} \subseteq 2^{\{p_1,..,p_n\}}$ is called monotone if $B \in \mathcal{A}$ and $B \subseteq C$ implies $C \in \mathcal{A}$. Sets in $\mathcal{A}$ are called authorized and sets not in $\mathcal{A}$ are called unauthorized.*

▶ **Definition 5** ([4]). *Distribution Scheme: Let S,$|S| \geq 2$ be a finite set of secrets. A secret sharing scheme with secrets domain S, is a tuple $\mathcal{M} =< Sh, \mu >$ where μ is a probability distribution over some finite set R (called the set of random strings) and Sh is a mapping from $S \times R$ to a set of n-tuples $S_1 \times S_2 \times \ldots \times S_n$, where S_j is called the domain of shares of p_j. For a set $A \subseteq \{p_1, \ldots, p_n\}$, we denote $Sh(s, r)_A$ as the restriction of $Sh(s, r)$ to its A-entries. Sh satisfies the following properties:*

***Perfect Correctness**. The secret $s \in S$ can be reconstructed by any authorized set of parties. That is, for any set $B \in \mathcal{A}$ (where $B = \{p_{i_1}, \ldots, p_{i_{|B|}}\}$), there exists a reconstruction function $Recon_B : S_{i_1} \times \ldots \times S_{i_{|B|}} \to S$ such that for every $s \in S$,*

$$Pr[Recon_B(Sh(s, r)_B) = s] = 1 \tag{1}$$

[4] A sufficiently small super-polynomial bound on $|R|$ would still imply non-trivial bounds on share complexity, say better than the best known bound of $\Omega(n/\log n)$ for general schemes.

We refer to sets in $\mathcal{A}$ as qualified, and to minimal qualified B in the sense that B is qualified and no $B' \subsetneq B$ is qualified as minterms of $\mathcal{A}$. We refer to maximal unqualified sets, in the sense that B is unqualified but for all $P_i \notin B$, $\{P_i\} \cup B$ is qualified as maxterms of $\mathcal{A}$.

***Perfect Privacy**. Every unauthorized set cannot learn anything about the secret (in the information theoretic sense) from their shares. Formally, for any set $T \notin \mathcal{A}$, for every two secrets $a, b \in S$, and for every possible vector of shares $< \vec{sh}_j >_{p_j \in T}$:*

$$Pr[Sh(a,r)_T =< \vec{sh}_j >_{p_j \in T}] = Pr[Sh(b,r)_T =< \vec{sh}_j >_{p_j \in T}] \tag{2}$$

Observe that wlog., each share polynomial $q_{i,j}$ has free coefficient 0 (as any constant may be locally added by *Recon*). We will assume this implicitly throughout the paper.

Sometimes, we will be interested in ϵ statistical secret sharing, where ϵ error in correctness is allowed, and the distributions $Sh(a,r)_T$ and $Sh(b,r)_T$ are for unqualified T may be at statistical distance up to ϵ. Our default notion throughout the paper is that of perfect secret sharing as in Definition 5.

(Multi)Linear secret sharing schemes

The most studied and most commonly used class of secret sharing schemes is the linear secret sharing schemes class. This class is subclass of multi-linear secret sharing schemes.

A secret sharing scheme is said to be multi-linear, if $S = \mathbb{F}^k, R = \mathbb{F}^m$ for some finite field $\mathbb{F}$, and each share $\vec{\mathrm{sh}}_i$ consists of g linear combinations $l_{i,1}(\mathrm{s}_1, \ldots, \mathrm{s}_k, r_1, \ldots, r_m) \ldots, l_{i,g}(\mathrm{s}_1, \ldots, \mathrm{s}_k, r_1, \ldots, r_m)$ over $\mathbb{F}$. The scheme is called linear if additionally $k = 1$.

Complexity measures of secret sharing schemes

The information rate, IR of a secret sharing scheme $\mathcal{M}$, is the ratio between the maximum length of the shares and the length of the secret. Formally, $IR(\mathcal{M}) = (\max_{i \in [n]} \log(|S_i|))/|\log S|$, where the maximum is taken over all dealer's random strings r.

The share complexity of secret sharing scheme, $\mathcal{M}$, is $SC(\mathcal{M}) = \max_{i \in [n]} \log(|S_i|)$.

We denote the randomness complexity of a secret sharing scheme $\mathcal{M}$ by $\mathrm{RC}(\mathcal{M})) = \lceil \log_2(|R|) \rceil$ - the number of bits required to represent an element of R.

2.1 Polynomials over finite fields

In this work we focus on the set $\mathbb{F}_q[y_1, \ldots, y_n]$ of multivariate polynomials over finite fields. We say a polynomial $p(y_1, \ldots, y_n)$ is of degree i if all monomials in the polynomials have a cumulative degree of at most i. We say p has degree exactly i if all monomials in p are of cumulative degree exactly i. Similarly, for a subset $I \subseteq [n]$, we say p is of degree i in $x_I = \{x_j | j \in I\}$ if every monomial of p has cumulative degree at most i in the variables from x_I (similarly, for exact degree in x_I). In a finite field $\mathbb{F} = \mathbb{F}_{p^\ell}$, where p is prime, let $Tr_{\mathbb{F}}(\alpha) = \sum_{i=0}^{\ell-1} \alpha^{p^i}$ is the *trace* mapping from $\mathbb{F}$ to itself.[5]

2.1.1 Output distributions of degree-2 polynomials

Some of our results require some theory on degree-2 polynomials over finite fields. In particular, we will reduce understanding the output distributions of (various subclasses of) degree-2 PSSS to understanding the output distribution of a *single* degree-2 multivariate

[5] In fact, the image of $Tr_{\mathbb{F}}$ is always contained in $\mathbb{F}_p$.

polynomial. For (any) polynomial in $p(x_1,\ldots,x_n) \in \mathbb{F}_q[x_1,\ldots,x_n]$, we let $N_{f,b}$ denote the number of solutions in $\mathbb{F}_q^n$ for the equation $f(x_1,\ldots,x_n) = b$. Polynomials in $\mathbb{F}[x_1,\ldots,x_n]_q$ where all monomials are of exactly degree 2, called *quadratic forms*. It is convenient to represent quadratic forms $f(x)$, by a matrix $A \in \mathbb{F}_q^{n\times n}$, where $f(x) = x^T Ax$. That is, $A_{i,j}$ is the coefficient of $x_i x_j$. We will need the following existing theory characterizing $N_{f,b}$ for f which are quadratic forms over a finite fields, and general degree-2 polynomials over fields of characteristic 2. All required theory and discussions appears in chapter 6 in [16], and is included here for self containment. Also, some of the theorems we state here are straightforward corollaries of [16], but were not explicitly stated there.

Fields of odd characteristic

Fix some finite field $\mathbb{F}$ of odd charactersitic. We let η denote the quadratic character on $\mathbb{F}^*$. That is, $\eta(x) = 1$ if x is a quadratic residue modulo q, and -1 otherwise. We extend its definition to 0 via $\eta(0) = 0$.

We also let $\nu : \mathbb{F} \to \mathbb{Z}$ be $\nu(b) = -1$ for $b \in \mathbb{F}^*$, and $\nu(0) = q-1$. Recall a quadratic form f over a characteristic field $\mathbb{F}$ in variables $x_1,\ldots,x_n$ is a polynomial where all monomials are of degree exactly 2. It is known that a quadratic form $f(x)$ in variables $x = (x_1,\ldots,x_n)$ has a representation of the form $f(x) = x^T C \cdot M_f \cdot C^T x$, where C is an invertible matrix in $\mathbb{F}^{n\times n}$, and $M_f \in \mathbb{F}_q^{n\times n}$ is diagonal, and all $rank(M_f)$ non-zero elements in the diagonal are at entries $M[i,i]$ for $i \le rank(M_f)$. Such a representation M_f is called canonical. Here, M_f represents a quadratic form $p'(v) = v^T M_f v$ in a new vector $\vec{v} = (v_1,\ldots,v_n)$ of variables, obtained from $\vec{x}$ via $\vec{v} = C^T x$. The number $m \le n$ of non-zero elements on M_f's diagonal is an invariant for all canonical representations of f. The function $\eta(det(M_f^-))$ is another invariant, independent of the concrete canonical representation M_f. (see Theorem 6.21 in [16] and discussion beforehand for more intuition). We denote the type of a quardatric form $f(x_1,\ldots,x_n)$ over $\mathbb{F}_q$ of odd characteristic as (n,m,η), where (m,η) are the corresponding values of the above invariants of equivalent canonical forms.

To understand the expression for $N_{f,b}$ for a quadratic form f, it suffices to understand $N_{g,b}$ for the quadratic form $g(v_1,\ldots,v_n)$ in a new vector of variables $v = (v_1,\ldots,v_n)$, where $g(v) = v^T M_f v$ where M_f is a canonical representation of a quadratic form, as $N_{f,b} = N_{g,b}$ for all $b \in \mathbb{F}_q$. We refer to such g as canonical forms. This holds as $v(x) = C^T x$ is a bijection between the domain of $f(x)$ and the domain of $g(v)$ satisfying $f(x) = g(v(x))$ for all $x \in \mathbb{F}_q^n$. We say that f is equivalent to a canonical form g as above. We define the *type* of a quadratic form $f(x_1,\ldots,x_n)$ of odd characteristic via the triple $(n,m,\eta(\det))$ (with $m,\eta(det)$ invariants of canonical forms equivalent to f).

By the above discussion, we may assume wlog. that $n = m$, and calculate the number of roots in that case. In the general case of f of type (n,m,η), compute the number of roots for an equivalent canonical g of type $(n = m,m,\eta)$, and multiply by q^{n-m}.

The following theorem now follows directly by combining theorems 6.26, 6.27 from [16]. For a quadratic form $f(x)$ we denote the number of solutions to the equation $f(x) = b$ by $N_{f,b}$,

▶ **Theorem 6.** *Let $p(x_1,\ldots,x_n)$ denote a quadratic form over a finite field $\mathbb{F}_q$ of odd characteristic of type (n,m,d). Consider a representation $f(x) = v^T M_f v$ as above, $x = (x_1,\ldots,x_n)$,and the v_i's are (independent, by choice of C) linear combinations of the x_j's. Then*

1. *If m is even, then for every $b \in \mathbb{F}$*

$$N_{f,b} = q^{n-m}(q^{m-1} + q^{(m-2)/2}\nu(b)\eta((-1)^{m/2})d).$$

2. *If m is odd, for every $b \in \mathbb{F}$*

$$N_{f,b} = q^{n-m}(q^{m-1} + q^{(m-1)/2}\eta(b(-1)^{m/2})d).$$

Following Theorem 6, we define the *type* of a quadratic form $f(x_1, \ldots, x_n)$ of odd characteristic via $(m, \det)$. Evidently, the type of f determines the distribution of $f(x)$ when x is picked uniformly from $\mathbb{F}^n$. Here we no longer assume $m = n$.

Fields of characteristic 2

Let $\mathbb{F}$ be a field of characteristic 2. Here we also have a canonical representation of quadratic forms, albeit somewhat less simple. Namely, for every quadratic form $f(x_1, \ldots, x_n)$, there exists a number $m \leq n$, and a non-signular matrix $C \in \mathbb{F}^{n \times n}$ such that $f(x) = x^T C M_f C^T x$, where M_f has one of the following forms:

1. (Type $T = 1$) m is even. M_f has 0's everywhere except for entries $M[2i-1, 2i]$ for $1 \leq i \leq m/2$ for some integer $m \leq n$, which are all 1.
2. (Type $T = 2$) m is even. M_f has 0's everywhere except for entries $M[2i-1, 2i]$ for all $1 \leq i \leq m/2$ for some integer $m \leq n$ which are 1, $M[m-1, m-1] = 1$, and $M[m, m] = a$, where $Tr_{\mathbb{F}}(a) = 1$.
3. (Type $T = 3$) m is odd. M_f has 0's everywhere except for entries $M[2i-1, 2i]$ for $1 \leq i \leq (m-1)/2$ which are all 1, and also $M[m, m] = 1$.

Similarly to the odd characteristic case, we refer to M_f as a canonical representation. By Theorem 6.30 in [16], the number m and T of the canonical M_f is and invariant depending only on f, and not on the particular representation f. Thus, we denote the type of a quadratic form $f(x_1, \ldots, x_n)$ as (n, m, T), according to n and the above invariants. For each type, and $b \in \mathbb{F}$, a characterization of $N_{f,b}$ for quadratic forms is known, as follows from Theorem 6.32 in [16].[6]

▶ **Theorem 7.** *Let $p(x_1, \ldots, x_n)$ denote a quadratic form of type (n, m, T) over a finite field $\mathbb{F}_q$ of characteristic 2. Then*

1. *If $T = 1$, for every $b \in \mathbb{F}_q$, $N_{f,b} = q^{n-m}(q^{m-1} + q^{(m-2)/2}\nu(b))$.*
2. *If $T = 2$, for every $b \in \mathbb{F}_q$, $N_{f,b} = q^{n-m}(q^{m-1} - q^{(m-1)/2}\nu(b))$.*
3. *If $T = 3$, for all $b \in \mathbb{F}_q$, $N_{f,b} = q^{n-1}$.*

2.2 Polynomial Secret Sharing Schemes (PSSS)

In this work, we put forward a natural generalization of (multi)-linear secret sharing schemes - where shares are allowed to be general polynomials of $\vec{s}, \vec{r}$, rather than just linear combinations. Namely:

▶ **Definition 8** (PSSS). *A polynomial secret sharing scheme (PSSS) $\mathcal{M} = (Sh, \mu)$ is a secret sharing scheme specified by $(\mathbb{F}, t, k, Sh)$ where $\mathbb{F}$ is a finite field, $S = \mathbb{F}^k$ is the domain of secrets, μ is uniform over $R = \mathbb{F}^t$, and $t, k \in \mathbb{N}^+$. The sharing function $Sh(\vec{s}; \vec{r})_i$ returns $(p_{i,1}(\vec{s}, \vec{r}), \ldots, p_{i,l_i}(\vec{s}, \vec{r}))$ as the i'th party's share, where each $p_{i,j}(\vec{s}, \vec{r})$ is a (multivariate) polynomial over $\mathbb{F}$.*

[6] The theorem applies to $m = n$, but reasoning similar to the odd characteristic case implies $N_{f,b}$ for general m, n as a simple corollary.

We will denote the corresponding classes of polynomial schemes over $\mathbb{F}$ via $PSSS_{regexp[s,r],\mathbb{F}}$, where $regexp$ is a (variant of) a regular expression in $r, s, 1$. The syntax and semantics of the expression set is defined recursively as follows: r encodes the set of polynomials $\{\sum_{j\in[k]} a_j r_j | a_j \in \mathbb{F}\}$, and s encodes $\{\sum_{j\in[k]} a_j \mathrm{s}_j | a_j \in \mathbb{F}\}$, 1 encodes $\{a | a \in \mathbb{F}\}$. For a pair of regular expressions g_1, g_2; g_1^* encodes the set $\{p_1 \cdot \ldots \cdot p_h | h \in \mathbb{N}, \forall i \in [h], p_i \in g_1\}$; $g_1 + g_2$ encodes $\{p_1 + p_2 | p_1 \in g_1, p_2 \in g_2\}$, and $g_1 \cdot g_2$ encodes the set $\{\sum_{j\in[h]} p_{1,j} \cdot p_{2,j} | h \in \mathbb{N}, \forall j p_{1,j} \in g_1, p_{2,j} \in g_2\}$. g_1^i is a shorthand for $g_1 \cdot \ldots \cdot g_1$ with i appearances of g_1. We also say that a scheme M has degree at most (exactly) d in r (s), if each monomial contains at most (exactly) d r_i's (s_i's).

For polynomial schemes $\mathcal{M}$, we measure share complexity in field elements, rather than in bits. Formally, these measures will be denoted by $SC_{\mathbb{F}}(\mathcal{M}$ etc. (it always the case $IR_{\mathbb{F}}(\mathcal{M}) = IR(\mathcal{M})$, as this measure is normalized by secret size).

Our definition is a generalization of the notion of multi linear secret sharing in a natural direction, which potentially adds power over multi-linear schemes. We try to keep it as close as possible to the definition of multi-linear schemes, and insist that the domain where secrets, randomness and computation are performed is a finite field.[7]

A slightly more general notion of polynomial schemes is one where $S \subseteq \mathbb{F}^k$, rather than the entire set $\mathbb{F}^k$.[8] We refer to such schemes as *generalized* polynomial schemes.

3 On Feasibility and Share Complexity of PSSS

In the next two sections, we present our negative results. Our positive result on the power of multilinear schemes is a rather simple observation based on existing work, and is deferred to the full version.

3.1 Bounds on efficiency of degree 1 in r PSSS

We show that a large sub-class of polynomial schemes of degree at most 1 in r ($PSSS_{s^*\cdot r+s^*}$) are not more powerful than multi-linear schemes, in the sense that they can not reduce share complexity super-polynomially over multi-linear schemes.

Our first result proves that $PSSS_{s^*\cdot r+s}$ can be replaced by a multi-linear scheme without any loss in parametres.

▶ **Theorem 9.** *For every scheme $\mathcal{M} = (\mathbb{F}, t, k, Sh)$ in $PSSS_{s^*\cdot r+s}$, there exists a $PSSS_{s+r}$ scheme $\mathcal{M}' = (\mathbb{F}, t, k, Sh')$ for the same access structure and $\mathcal{A}$ with $SC(\mathcal{M}') = SC(\mathcal{M})$.*

Proof. Somewhat surprisingly, for any scheme $PSSS_{s+r,\mathbb{F}}$ we build an equivalent multi-linear scheme by replacing the coefficient polynomials of the r_i's in the shares (which have the form $p(s)$) by constants resulting from substituting an arbitrary fixed vector $s' \in S$ into the coefficients.

To prove this theorem, let us restate the sharing algorithm Sh more conveniently. For such a scheme, $Sh(s, r)$ can be represented as $Vs + Mr$, where $V \in \mathbb{F}^{a\times k}, M \in \mathbb{F}[\mathrm{s}_1, \ldots, \mathrm{s}_n]^{a\times t}$. Here each entry of M is a formal polynomial $p_{i,j}$ in s, a the total number of polynomials in

[7] Note that some of the schemes appearing in [5] are quite close to "polynomial" schemes, but the domains employed there are rings R which are (crucially) not fields, and the secrets and randomness do not necessarily come from domains of the form R^t, R^k.

[8] If no restriction on the s-degree are made, we may replace the subset S with any other subset of the same size, without affecting the other parameters.

the share vector, and V a constant. M_s is a shorthand for $M(s)$ - substituting a concrete value s as the secret vector, into the matrix of polynomials.

A function $\rho : \{1, ..., a\} \longrightarrow \{p_1, ..., p_n\}$ labels each row by a party, so that party P_i receives the shares corresponding to rows $H_i = j|\rho(j) = i$. For a set A of parties, we abbreviate the submatrix pf M involved in generating A's shares on secret vector s (aka $\cup_{i\in A} H_i$) as $A_s = (V_A|M_{s,A})$.

▷ Claim 10. Let $\mathcal{M} = \{\mathbb{F}, t, k, (V|M)\}$, in $PSSS_{s^*r+s,\mathbb{F}}$, be a secret sharing scheme for an access structure $\mathcal{A}$. The scheme $\mathcal{M}'$ where M is substituted by a constant matrix $M_{\vec{s_1}}$ for some fixed secret $\vec{s_1}$ is a (multi-linear) secret sharing scheme for the same access structure.

Proof. Fix some secret vector $\vec{s_1}$ as in the statement of the claim. We prove the scheme remains valid.

Correctness: Consider any $\vec{s_0} \in \mathbb{F}^k$. Now we will look at authorized set A. Let us look at the two share distributions $(V_A|A_{\vec{s_1}}) \cdot (\vec{s_1}|\vec{r_1})$ and $(V_A|A_{\vec{s_0}}) \cdot (\vec{s_0}|\vec{r_0})$ of secrets $\vec{s_1}$ and $\vec{s_0}$, where $\vec{r_1}, \vec{r_0} \in \mathbb{F}^t$ are independent random vectors. The correctness of $\mathcal{M}$ is equivalent to stating that for all pairs $\vec{r_0}, \vec{r_1}$, we have:

$$\begin{aligned} &(V_A|A_{\vec{s_1}}) \cdot (\vec{s_1}|\vec{r_1}) \neq (V_A|A_{\vec{s_0}}) \cdot (\vec{s_0}|\vec{r_0}) \\ &\qquad\qquad \Downarrow \\ &V_A \cdot (\vec{s_0} - \vec{s_1}) \neq A_{\vec{s_1}} \cdot \vec{r_1} - A_{\vec{s_0}} \cdot \vec{r_0}. \end{aligned} \tag{3}$$

It is correct in particular for $\vec{r_0} = \vec{0}$. Which means that:

$$V_A \cdot (\vec{s_0} - \vec{s_1}) \neq A_{\vec{s_1}} \cdot \vec{r_1} \tag{4}$$

for all $\vec{r_1}$. Due to the fact that Equation 4 is correct for any $\vec{s_0} \in \mathbb{F}^k$ and by the structure of the secret domain, for any two distinct secret vectors $\vec{s_2}, \vec{s_3} \in \mathbb{F}^k$ there exists $\vec{s_0}$ for which $\vec{s_2} - \vec{s_3} = \vec{s_0} - \vec{s_1}$. From equation 4:

$$V_A \cdot (\vec{s_2} - \vec{s_3}) \neq A_{\vec{s_1}} \cdot r_1 \tag{5}$$

For all $\vec{r_1} \in \mathbb{F}^t$. Let $\vec{r_2}, \vec{r_3} \in \mathbb{F}^t$. Writing $\vec{r_1} = \vec{r_3} - \vec{r_2}$ we conclude that (as r_1 in Equation 5 is arbitrary),

$$\begin{aligned} &V_A \cdot (\vec{s_2} - \vec{s_3}) \neq A_{\vec{s_1}} \cdot \vec{r_1} \\ &\qquad\qquad \Downarrow \\ &(V_A|A_{\vec{s_1}}) \cdot (\vec{s_2}|\vec{r_2}) \neq (V_A|A_{\vec{s_1}}) \cdot (\vec{s_3}|\vec{r_3}) \end{aligned} \tag{6}$$

Which is precisely the definition of correctness for the new scheme (as $\vec{r_2}, \vec{r_3}, \vec{s_2} \neq \vec{s_3}$ are otherwise arbitrary).

Privacy: Consider some secret $\vec{s_0} \neq \vec{s_1} \in \mathbb{F}^k$. It follows directly from privacy that for each unauthorized set A, for any $\vec{r_0} \in \mathbb{F}^t$ there exists $\vec{r_1} \in \mathbb{F}^t$ for which:

$$\begin{aligned} &(V_A|A_{\vec{s_1}}) \cdot (\vec{s_1}|\vec{r_1}) = (V_A|A_{\vec{s_0}}) \cdot (\vec{s_0}|\vec{r_0}) \\ &\qquad\qquad \Downarrow \\ &V_A \cdot (\vec{s_0} - \vec{s_1}) = A_{\vec{s_1}} \cdot \vec{r_1} - A_{\vec{s_0}} \cdot \vec{r_0} \end{aligned} \tag{7}$$

In particular this is true for $\vec{r_0} = \vec{0}$. Then for any $\vec{s_0}$ there exists $\vec{r_1} \in \mathbb{F}^t$ for which:

$$V_A \cdot (\vec{s_0} - \vec{s_1}) = A_{\vec{s_1}} \cdot \vec{r_1} \tag{8}$$

Let $\vec{s_2}, \vec{s_3}$ denote a pair of secrets. Fix $\vec{s_0}$ for which $\vec{s_2} - \vec{s_3} = \vec{s_0} - \vec{s_1}$. From 8 it follows there exists $\vec{r_1}$ for which:

$$V_A \cdot (\vec{s_2} - \vec{s_3}) = A_{\vec{s_1}} \cdot \vec{r_1} \tag{9}$$

So for any vector $r_3 \in \mathbb{F}^t$ we get:

$$\begin{aligned} V_A \cdot (\vec{s_2} - \vec{s_3}) &= A_{\vec{s_1}} \cdot r_1 \\ &\Downarrow \\ V_A \cdot (\vec{s_2} - \vec{s_3}) &= A_{\vec{s_1}} \cdot (\vec{r_3} - (\vec{r_3} - \vec{r_1})) \\ &\Downarrow \\ (V_A | A_{\vec{s_1}}) \cdot (\vec{s_2} | \vec{r_3} - \vec{r_1}) &= (V_A | A_{\vec{s_1}}) \cdot (\vec{s_3} | \vec{r_3}) \end{aligned} \tag{10}$$

We prove that this implies privacy. Picking $\vec{r_3}$ at random, the vector $\vec{r_3} - \vec{r_1}$ is a random vector as well. Thus, the left hand size, where $\vec{r_3}$ is picked at random is distributed precisely as the shares seen by A when sharing $\vec{s_2}$ in $\mathcal{M}'$. This value is uniform over the affine subspace $V_A\vec{s_2} + \text{colSpan}(A_{\vec{s_1}})$. Similarly, the right hand side is also a random element of an affine subspace of the form $V_A\vec{s_3} + \text{colSpan}(A_{\vec{s_1}})$, and is distributed precisely as a share of $\vec{s_3}$ seen by A at M'. By Equation 10, these affine subspaces intersect, so they must be the same subspace, since both are cosets of $\text{colSpan}(A_{\vec{s_1}})$. This concludes the proof. ◁

◀

Next, we prove that a $PSSS_{s^*+r}$ scheme can be replaced by a multi-linear scheme up to a small loss in rate due to a small reduction in the dimension k of the secret space. Here, it will be convenient to specify $Sh(s, r)$ by a pair $(v(s), M)$, where $v(s) = (v_1(\mathrm{s}), \ldots, v_\ell(\mathrm{s}))$ is a vector of (multivariate) polynomials in s, and M is a constant matrix, and

$$Sh(s, r) = Mr + \sum_{i \in [k]} \mathrm{s}_i \frac{v^{(i)}}{\mathrm{s}_i}(s) = Mr + v(s) \tag{11}$$

Such an expression exists as we assume all share polynomials have a non-zero free coefficient. Here every $v^{(i)}(s)$ is a vector of formal polynomials, comprised of sums of all monomials in v in which s_i's degree is at least 1, and that were not included in $v^{(j)}$ for $j < i$ (we construct the $v^{(i)}$'s iteratively, starting from $i = 1$).[9] In this representation, s_i appears only in $v^{(j)}$ with $j \leq i$. We will sometimes denote Sh in $PSSS_{s^*+r}$ schemes as a pair (v, M) as above.

▶ **Theorem 11.** *For every scheme $\mathcal{M} = (\mathbb{F}, t, k, (v, M))$ in $PSSS_{s^*+r}$ there exists a multi-linear scheme $\mathcal{M}' = (\mathbb{F}, t, k - n, Sh)$ for the same access structure $\mathcal{A}$ with share complexity $SC(\mathcal{M}') \leq n \cdot SC(\mathcal{M})$.*

Proof. We construct a multi-linear scheme $\mathcal{M}' = (\mathbb{F}, t, k', (V'|M))$, by constructing a basis B for V''s column space, where $Sh(s, r) = (V'|M)(s, r)$ is the sharing algorithm of the multi-linear scheme (note V' here is constant). By Equation 11, for $s' = \vec{0}$, the distribution of $Sh(s', r)$ is therefor uniform over the zero coset of $Mr = colSpan(M)$. We conclude the following:

[9] Unlike in the previous section, it is more convenient to denote the formal polynomial vector by v, rather than v_s, in analog to M_s in the previous section, to simplify notation. We let $v(s)$ denote the evaluation of v on a specific vector s.

▷ Claim 12. For all $s' \in \mathbb{F}^k$ and every unqualified A, the vector $v_A(\mathrm{s}')$ is in $colSpan(M_A)$.

Proof. To see this, consider a representation of Sh as in Equation 11 of the form $Sh(\mathrm{s}', r) = Mr + v(\mathrm{s}')$ as above. Let v_A denote v restricted to entries held by A. We have $v_A(0, \mathrm{s}'_2, \ldots, \mathrm{s}'_k) = v_A(\mathrm{s}') - v_A^{(1)}(\mathrm{s}')$ (since only $v_A^{(1)}$ depends on s_1). Since by privacy of $\mathcal{M}$ both $v_A(s')$ and $v_A(0, s'_2, \ldots, s'_k)$ must belong to $colSpan(M_A)$ (as this holds for $s' = \vec{0}$), so does $v_A^{(1)}(s')$. Since s' is arbitrary, we conclude that $s''_1 v_A^{(1)}(s'')$ is in $colSpan(M_A)$ for all $s' = s''$. Now, comparing $Sh(s', r)$ and $s'' = (s'_1, 0, s'_3, \ldots, s'_k)$, by similar reasoning to the above, we conclude that $v_A^{(2)}(\mathrm{s}')$ is also $\vec{0}$ in $\mathbb{F}^{\#_{rows}(M_A)}/colSpan(M_A)$. This follows from the fact that $v_A^{(j)}$'s for $j > 2$ are independent of s_2, and the fact that $v_A^{(1)}(\mathrm{s}')$ and $v_A^{(1)}(\mathrm{s}'')$ are 0 in $\mathbb{F}^{\#_{rows}(M_A)}/colSpan(M_A)$ as we proved before, so it does not effect the coset. Similarly to the case of $j = 2$, by induction on j we can prove that $v_A^{(j)}(s')$ equals $\vec{0}$ in $\mathbb{F}^{\#_{rows}(M_A)}/colSpan(M_A)$. Now, as $v_A(s') = \sum_i v_A^{(i)}(s')$, it also equals $\vec{0}$, as required. ◁

From Claim 12, it follows that taking any V' with columns in $span(\{v(s')|s' \in \mathbb{F}^k\}$, $(V'|M)$ immediately satisfies privacy. We will indeed pick our basis B out of $span(\{v(s')|s' \in \mathbb{F}^k\}$, so we will only need to worry that the resulting scheme satisfies correctness. The construction is as follows.

1. Initialization: Initialize $B = \phi$ (recall $span(B)$ is $\{\vec{0}\}$).
2. Iteration $i > 0$: Find some $s' \in S$, so that for all minterms $A \subseteq [n]$, $v(s')$ belongs to a coset of $\mathbb{F}^{\#_{rows}(M_A)}/colSpan(M_A)$ that differs from coset(v) for all $v \in span(B)$. Halt if no such s exists. If it does, add one such V^s to B.

We prove by induction that at the end of every iteration $i \leq max(1, k - n)$, we B is a size-i independent set in $\mathbb{F}^{\#_{rows}(M)}$ such that $(B|M)(s, r)$ is correct for $\mathcal{A}$ with secret domain $S = \mathbb{F}^i$ (and private, which we observed before).

First, observe that the above procedure will yield at least a single vector. For every $s' \neq \vec{0}$, and every minterm A, $v_A(s')$ is non zero in $\mathbb{F}^{\#_{rows}(M_A)}/colSpan(M_A)$ by correctness of $\mathcal{M}$. Now, any product $\alpha\vec{s'}$ for $\alpha \in \mathbb{F}$ will yield a different coset in $\mathbb{F}^{\#_{rows}(M_A)}/colSpan(M_A)$, as v_A is non-zero. Thus, we can add $v_s(s')$ to our set. By the inductive hypothesis, at the end of iteration i, we have $|\mathbb{F}|^i$ vectors already in $span(B)$ - for clarity, denote B at the end of iteration i by $B^{(i)}$. We observe that for every minterm A all projections $v_A(s')$ are distinct for different values s' - which follows from correctness of $\mathcal{M}$. Therefor, going over all A's, at most

$$(\text{number of minterms})|\mathbb{F}|^i \leq 2^n|\mathbb{F}|^i \leq |\mathbb{F}|^{i+n}$$

vectors are excluded as candidates for the next $v_s(s')$ to join B. Finally, by the condition imposed on the new vector to join B, it follows that $B^{(i+1)}$ is a size-$i+1$ independent set, as satisfies that $(B|M)$ is correct for secret domain $S = \mathbb{F}^{i+1}$ (the formal argument is similar to the base case, observing that $v_A(s')$ is non-zero as a coset of $(M_A|B_A^{(i)})$). As there are $|\mathbb{F}|^k$ vectors in $\mathcal{M}$'s domain to begin with, we conclude (from the proof of the inductive step above) that at least $k - n$ iterations can be made before running out of vectors to add, which concludes the proof. ◀

3.2 $PSSS_{s^*+s^*r^2}$ is very weak

In this section we will show that if the shares are from the class $PSSS_{s^*+s^*r^2}$ (no r-degree 1 part) captures only the access structures consisting of a set of singletons as its minterms.[10]

▶ **Theorem 13.** *Let $\mathbb{F}$ be a finite field of odd characteristic. Then the class $PSSS_{s*+s*r^2,\mathbb{F}}$ can only implement a simple set of access structures where its minterms are all singletons.*

Indeed, observe that we can not expect a similar result for all fields, as for $\mathbb{F}_2$, for instance, we have $r_i^2 = r_i$, so one can represent any multi linear scheme over $\mathbb{F}_2$ as a $PSSS_{s*+s*r^2,\mathbb{F}}$ scheme, by replacing every variable r_i by r_i^2, which are equal over $\mathbb{F}_2$. However, linear schemes over $\mathbb{F}_2$ do capture all monotone access structures (e.g, via the formula-based construction of [8]). See 2 for required background and notation on quadratic forms.

Furthermore, we have

▶ **Observation 3.** *Let $f_1(x_1,\ldots,x_n), f_2(x_1,\ldots,x_n)$ be two quadratic forms over a field $\mathbb{F}_q$ of odd characteristic of (possibly same) types $(n_1,m_1,d_1),(n_2,m_2,d_2)$ respectively. Then for all $b \in \mathbb{F} - \{0\}$, $Pr_{x\leftarrow\mathbb{F}^n}(f_1(x)=0) \neq Pr_{x\leftarrow\mathbb{F}^n}(f_2(x)=b)$.*

The observation follows by simple case analysis. In some more detail, by Theorem 6, $N(f_1(x=0))$ is either a single q^x or of the form $q^{x_1} \pm q^{x_2} \pm q^{x_3}$ for $x_1 > x_2 > x_3$, while for $b \neq 0$, $N(f_2(x=b))$ is of the form $q^{x_1} \pm q^{x_2}$ for $x_1 > x_2$. So, the probabilities (after dividing both numbers by q^n) must differ. This is regardless of the values of m_1, m_2.

Now, consider a party P_h that receives a share of the form

$$f(\vec{s},\vec{r}) = p(\vec{s}) + \sum_{\substack{i,j\in\{1,..,n\}\\ i\leq j}} p_{i,j}(\vec{s})r_ir_j = p(\vec{s}) + q_{\vec{s}}(\vec{r}).$$

where each $q_{\vec{s}}(\vec{r})$ is a polynomial in $\vec{r}$ with coefficients in the ring $\mathbb{F}_q[s_1,\ldots,s_n]$, and $p(\vec{s})$ is non constant over $\mathbb{F}_q^n$. First consider the case when $p(\vec{s})$ is non-constant over $\mathbb{F}_q^n$. We prove that there exists a pair of secrets $\vec{s_1},\vec{s_2}$ that P_h can distinguish by itself. To see this, fix two vectors $\vec{s_1},\vec{s_2}$ such that $p(\vec{s_1}) \neq p(\vec{s_2})$. By observation 3, it directly follows that the unique probability (over the choice of $\vec{r}$) of $f(\vec{s_1},\vec{r})$ hitting $p(\vec{s_1})$ equals the probability of $q_{\vec{s_1}}(r)$ hitting 0, while the probability of hitting values $b \neq p(\vec{s_1})$, equals the probability of $q_{\vec{s_1}}(r)$ hitting corresponding non-zero values (indeed, adding a constant permutes the distribution). A similar situation occurs with $f(\vec{s_2},\vec{r})$ and the "spacial" point $p(\vec{s_2})$. Thus, the points with the "special 0-probability for the $q_{\vec{s_i}}$-part" for $\vec{s_1}$ and $\vec{s_2}$ differ for $f(\vec{s_1},\vec{r})$ and $f(\vec{s_2},\vec{r})$. We conclude that the two distributions $f(\vec{s_1},r), f(\vec{s_2},r)$ are distinct. To see this, note that the contribution of $b = p(\vec{s_1})$ to the statistical distance between $f(\vec{s_1},r)$ and $f(\vec{s_2},r)$ is $1/2|Pr[q_{\vec{s_1}}(\vec{r})=0] - Pr[q_{\vec{s_2}}(\vec{r}) = p(\vec{s_1}) - p(\vec{s_2})]|$, which is non-zero by Observation 3.

Finally, let us look at all the remaining parties with only shares where $p(\vec{s})$ is constant (zero, wlog. since the free coefficient is 0). Such parties receive only shares of the form $f(\vec{s},\vec{r}) = q_{\vec{s}}(\vec{r})$, where every $q_{\vec{s}}$ is a quadratic form. Therefore, for any $\vec{s} \in S$ we have $f_p(\vec{s},\vec{0}) = 0$. Thus, all these parties together can not reconstruct the secret with probability 1, implying that the singletons above are the only minterms of the access structure. ◀

[10] Note that our results only rule out perfect schemes.

References

1 Benny Applebaum and Barak Arkis. Conditional disclosure of secrets and d-uniform secret sharing with constant information rate. *IACR Cryptology ePrint Archive*, 2018:1, 2018. URL: http://eprint.iacr.org/2018/001.

2 Benny Applebaum, Amos Beimel, Oded Nir, and Naty Peter. Better secret-sharing via robust conditional disclosure of secrets. *Electronic Colloquium on Computational Complexity (ECCC)*, 27:8, 2020. URL: https://eccc.weizmann.ac.il/report/2020/008.

3 Benny Applebaum and Prashant Nalini Vasudevan. Placing conditional disclosure of secrets in the communication complexity universe. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 4:1–4:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.ITCS.2019.4.

4 Amos Beimel. Secret-sharing schemes: A survey. In Yeow Meng Chee, Zhenbo Guo, San Ling, Fengjing Shao, Yuansheng Tang, Huaxiong Wang, and Chaoping Xing, editors, *Coding and Cryptology*, pages 11–46, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

5 Amos Beimel and Yuval Ishai. On the power of nonlinear secret-sharing. *IACR Cryptology ePrint Archive*, 2001:30, 2001. URL: http://eprint.iacr.org/2001/030.

6 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988. doi:10.1145/62212.62213.

7 Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer, 1988. doi:10.1007/0-387-34799-2_3.

8 Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer, 1988. doi:10.1007/0-387-34799-2_3.

9 G. R. Blakley. One time pads are key safeguarding schemes, not cryptosystems fast key safeguarding schemes (threshold schemes) exist. In *Proceedings of the 1980 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 14-16, 1980*, pages 108–113. IEEE Computer Society, 1980. doi:10.1109/SP.1980.10016.

10 László Csirmaz. The size of a share must be large. 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 13–22. Springer, 1994. doi:10.1007/BFb0053420.

11 Bella Dubrov and Yuval Ishai. On the randomness complexity of efficient sampling. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 711–720. ACM, 2006. doi:10.1145/1132516.1132615.

12 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987. doi:10.1145/28395.28420.

13 R. K. Gupta. *Linear Programming*. Krishna Prakashan, 2009. URL: https://books.google.co.il/books?id=Ur2vi5kB5IoC.

14 Yuval Ishai, Eyal Kushilevitz, and Anat Paskin-Cherniavsky. From randomizing polynomials to parallel algorithms. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer*

Science 2012, Cambridge, MA, USA, January 8-10, 2012, pages 76–89. ACM, 2012. doi:10.1145/2090236.2090244.

15 Mitsuru Ito, Akira Saito, and Takao Nishizeki. Secret sharing scheme realizing general access structure. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72:56–64, 1989.

16 Rudolf Lidl and Harald Neiderreiter. *Introduction to finite fields and their applications*. Cambridge University Press, 1997.

17 Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Conditional disclosure of secrets via non-linear reconstruction. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 758–790. Springer, 2017. doi:10.1007/978-3-319-63688-7_25.

18 Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Towards breaking the exponential barrier for general secret sharing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 567–596. Springer, 2018. doi:10.1007/978-3-319-78381-9_21.

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

A Bounding the Number of Random Variables in (general) PSSS

In this section we will present a bound on the number of random variables in (perfectly correct and private) PSSS and general secret sharing schemes.

▶ **Theorem 14.** *Let $\mathcal{M}$ denote a secret sharing scheme implementing an access structure $\mathcal{A}$ (with perfect privacy and correctness). Then there exists an equivalent secret sharing scheme with $RC(\mathcal{M}) = 2^{\tilde{O}(SC(\mathcal{M}))}$. Furthermore, if $\mathcal{M} = (\mathbb{F}_{q^d}, t, k, Sh)$ is a PSSS, then there exists an equivalent PSSS scheme $\mathcal{M}' = (\mathbb{F}_{q^d}, t', k, Sh')$ with $SC(\mathcal{M}') = SC(\mathcal{M})$ and $RC(\mathcal{M}') = 2^{poly(SC(\mathcal{M}))}$.*

Notation and some facts on Linear programs

For a PSSS scheme $\mathcal{M} = (\mathbb{F}_{q^d}, t, k, Sh)$, let us denote by sc the number of polynomial evaluations (field elements) output by Sh. Thus, $sc \geq k$ (since the set of sharings must be at least as large as S). We will need some theory of linear programs (LP). Here we will only care about the feasible region of a linear program (LP), and will not have a target function to optimize. Without loss of generality we consider LP's comprised of systems of inequalities of the form $Ax = b, x \geq 0$, where A, b are over $\mathbb{R}$, all b's components are non-negative. We denote such LP's by (A, b). We may also assume without loss of generality that $A \in \mathbb{R}^{m \times n}$, where $m \leq n$, and A has full rank (m). We say that a solution to the system is a basic feasible solution (BFS) if x only has non zero coordinates corresponding to an invertible submatrix of A (taking a subset of columns). For a finite set $B \subseteq \mathbb{R}^m$ of vectors, a convex combination of B is a linear combination $\sum_{b \in B} \alpha_b b$, so that $\sum_{b \in B} \alpha_b = 1$, and $\forall b \in B, \alpha_b \geq 0$. The convex hall of a set $A \subseteq \mathbb{R}^m$ is the set of all linear combinations of finite subsets $B \subseteq A$. We denote it as $CH(A)$. We say a set $A \subseteq \mathbb{R}^m$ is convex if $CH(A) = A$. An extreme point of a convex set A is a point $y \in A$ such that if y is a convex combination of $\{x, z\} \subseteq A$, then either $x = y$ or $z = y$. It is well known that the set of solutions of an LP is convex. We say an LP has a *bounded* solution set X, if there exists an integer N, such that $\ell_\infty(x) \leq N$ for all $x \in X$.

For a set $A = \{a_1, \ldots, a_t\} \subseteq \mathcal{R}^m$, the affine dimension of A, aff(A), is the dimension of $\{a_2 - a_1, \ldots, a_t - a_1\}$. We say that a set A has *full affine dimension* if aff$(A) = |A| - 1$.

▶ **Theorem 15** ([13], chapter 2)**.** *The set of extreme points $\mathcal{B}$ of a bounded non-empty solution set X of an LP $(A, b) \in \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times 1}$ is non empty, and $X = CH(\mathcal{B})$. Furthermore, the set $\mathcal{B}$ is precisely the set of BFS's of (A, b). Furthermore, Any solution p of (A, b) is a convex combination of a subset $\{p_1, \ldots, p_\ell\} \subseteq \mathcal{B}$ of full affine dimension, where $\ell \leq m + 1$.*

▶ **Lemma 16** (Cramer's rule)**.** *Let $A \in \mathcal{R}^{m\times m}$ denote an invertible matrix. Then, $A^{-1}_{i,j} = |A_{i,j}|/|A|$. Here $A_{i,j}$ is the (i, j)'th cofactor of A, obtained from removing the i'th column and j'th row from A.*

▶ **Lemma 17.** *Let $A \in \mathbb{R}^{m\times m}$ denote a matrix whose entries $a_{i,j}$ all satisfy $|a_{i,j}| \in \{0\} \cup [\delta, 1]$ for $0 < \delta$. Then every entry $a'_{i,j}$ in A^{-1} satisfies*

$$|a'_{i,j}| \text{ or } |a'_{i,j}| \geq \delta^m/m^m.$$

Additionally, if the $a_{i,j}$'s are integers, then the $|a'_{i,j}|$'s are multiples of a constant $0 < L \leq m^m$.

The proof of the above lemma follows directly from Lemma 16. In the following we only prove the bound for general schemes. The full proof for PSSS appears in the full version.

Proof of Theorem 14. Let us consider the given polynomial scheme $\mathcal{M}$ as in the theorem statement. We denote $Q = q^d$, $SC = Q^{SC(\mathcal{M})}$, and $sc = \log_Q(SC)$.

We denote the share vector output by Sh for any $\vec{s} \in S$ by $\vec{sh} = (sh_1, ..., sh_n) \in \mathbb{F}_Q^{sc})$. For every secret $\vec{s_i} \in S$, and for every possible $\vec{sh_j} \in \mathbb{F}_Q^{sc}$ let us denote by $p_{i,j}$ the probability to receive $\vec{sh_j}$ as the share vector on input $\vec{s_i}$. (For each $\vec{s_i}$, there are Q^{sc} such probabilities.)

Now we will build a matrix that will hold all the constraints on the probabilities $p_{i,j}$ for a scheme $\mathcal{M}'$ with $S, S_1 \times \ldots \times S_n$ for $\mathcal{A}$. Let $p_{\mathcal{M}}$ denote the probabilities vector induced by $\mathcal{M}$. Our set of requirements will be stronger than stating that $\mathcal{M}'$ is a secret sharing scheme for $\mathcal{A}$, as it will additionally require that $\mathcal{M}'$ is "similar" to $\mathcal{M}$ in a certain way. A solution will be guaranteed to exist, as $p_{\mathcal{M}}$ is such a solution ($\mathcal{M}$ is "similar" to itself).

The constraints are divided into 3 sets:

privacy: For any max unqualified set A, for every two secrets $\mathrm{s}_i, \mathrm{s}_j \in S$ the probability of getting the same shares (for this specific set) should be equal. That is to say, for any two secrets $\mathrm{s}_i, \mathrm{s}_j \in S$ and projection of shares on A, $\vec{sh}'$ (some specific share that parties in A receive).

$$\sum_{\substack{\text{all } k \text{ for which the projection} \\ \text{of } \vec{sh}_k \text{ on } A \text{ is } \vec{sh}'}} p_{i,k} = \sum_{\substack{\text{all } k \text{ for which the projection} \\ \text{of } \vec{sh}_k \text{ on } A \text{ is } \vec{sh}'}} p_{j,k}$$

Reorganizing, we get.

$$\sum_{\substack{\text{all } k \text{ for which the projection} \\ \text{of } \vec{sh}_k \text{ on } A \text{ is } \vec{sh}'}} p_{i,k} - \sum_{\substack{\text{all } k \text{ for which the projection} \\ \text{of } \vec{sh}_k \text{ on } A \text{ is } \vec{sh}'}} p_{j,k} = 0 \tag{12}$$

correctness: For any minimal qualified set A, for every two secrets $\mathrm{s}_i, \mathrm{s}_j \in S$ there are no share $\vec{sh}_k$ for which both $p_{i,k}$ and $p_{j,k}$ are not zero. That is to say, for every two secret $\mathrm{s}_i \in S$ and projection of shares on A $\vec{sh}'$ (some specific share that parties in A receive), for each s_j

so that $Pr(Sh(\mathrm{s}_j, r)_A = \vec{sh}') = 0$

$$\sum_{\substack{\text{all } k \text{ for which the projection} \\ \text{of } \vec{sh}_k \text{ on } A \text{ is } \vec{sh}' \\ \text{and } j \neq i}} p_{j,k} = 0 \tag{13}$$

By correctness, for each $\vec{sh}'$, there are at least $|S| - 1$ such j's.

probability restrictions: For any secret $\vec{\mathrm{s}_i} \in S$

$$\sum_j p_{i,j} = 1 \tag{14}$$

That is to say, that for every secret the sum of all the probabilities to get any share is 1.Another constraint is for every i and j.

$$0 \leq p_{i,j} \tag{15}$$

We stress that the privacy and probability constraints follow from the requirements on any secret sharing scheme implementing $\mathcal{A}$. The correctness constraints are constructed based on the concrete scheme $\mathcal{M}$.

The matrix M_1 defining our LP will be built from these three sets of equations 12, 13, 14, where the variables are the the $p_{i,j}$-s. In addition we will remove all the rows that depend on other rows, so our matrix M_1 will have a full rank. Let us denote:

$$r = 2^n |\mathbb{F}_Q^k| SC \leq SC^3 \tag{16}$$

Here the inequality holds since $n, k \leq sc$. There are at most r columns in M_1 thus and at most r rows.[11]

This LP is solvable since $p_{\mathcal{M}}$ is a solution for it. The right hand side b is the vector obtained from Equations 12, 13, 14 $(0, 0, \ldots, 0, 1, \ldots, 1)$ (with $|S|$ 1's at the end).

▶ **Observation 4.** *In the LP (M_1, b) above, all the entries in M_1 and in b are 1, -1 or 0.*

Now, any solution $\vec{p}'$ to the LP specified by (M_1, b) defines a secret sharing scheme for the desired access structure. Namely, assuming all entries in a solution $\vec{p}$ are multiples of some $1/L$ for some integer, we can set R to be of size L, and an arbitrary mapping Sh from $(\vec{s}, \vec{r})$'s to share vectors in $\mathbb{F}_Q^{sc}$ that agree with the probabilities in $\vec{p}$.

The problem is that if the elements in $\vec{p}'$ will be not multiples of $Q^{-t'}$ for some t' it will be impossible to present this secret sharing scheme with polynomials over $\mathbb{F}_Q$. We know one solution $p_{\mathcal{M}}$ that has probabilities which are multiples of Q^{-t} for some, possibly very large, t (the one induced by $\mathcal{M}$). Now we want to show that there exists $t' = 2^{2^{poly(SC)}}$, for which there is solution p' to (M_1, b) where all probability $p_{i,j}$ are multiples of $Q^{-t'}$, which will prove the theorem. By theorem 15, there is a set of BFS's $G = \{p_1, ..., p_\ell\}$ for the system, so that there exists a solution (the one induced by $\mathcal{M}$) $p_{\mathcal{M}} \in CH(G)$.[12]. Next, we prove that the entries of all $p_i \in G$ are of "low" resolution.

▷ Claim 18. For every $g \in G$, there exists an integer $0 < L \leq r^{2r}$, every entry g_i of g is a multiple of $1/L$.

[11] The second inequality follows from correctness of the scheme.

[12] Note that (M_1, b)'s solution set is indeed bounded, as all coordinates of a solution p are in the range $[0, 1]$.

Proof. This follows from the fact that the BFS in G is of the form $M_{1,H}^{-1}b$, where $M_{1,H}$ is a subset of M_1's columns corresponding to an invertible (square) matrix so that the entries in b corresponding to the other columns are all 0's. As M_1, b have entries in $\{0, 1, -1\}$ by Observation 4, the claim follows from Lemma 17. ◁

For any G, if the resulting scheme $\mathcal{M}'$ is not required to be a PSSS, then we are also done, as we can take (e.g.) $p_1 \in G$ as a basis for the scheme, and set R of size $L \leq 2^{2r}$ as guaranteed by Lemma 18. The randomness complexity of the resulting scheme is $\log_2(L) = 2^{\tilde{O}(SC(\mathcal{M}))}$. Additionally, for the case of $\mathcal{M}$ is a PSSS, $\mathcal{M}'$ is as in the theorem if $|G| = 1$, then p_1 must be a single solution, and its entries are already multiples of q^d, and we are done, as $M \leq r^{2r} \leq 2^{2^{\tilde{O}(SC(\mathcal{M}))}}$. Therefor, the solution vector p_1 induces a PSSS where $t = \log_Q(M) = 2^{\tilde{O}(SC(\mathcal{M}))}$. This is also the case if some BFS $p_i \in G$ happens to have entries which are all multiples of $Q^{-t'}$ for some t'. Otherwise, we prove that $CH(G)$ contains some solution where all entries are multiples of $Q^{-t'}$ where $t' = 2^{poly(SC)}$. See the full version for a proof, which is somewhat technically involved, but uses only some basic number theory. ◀

One-One Constrained Pseudorandom Functions

Naty Peter
Ben-Gurion University of the Negev, Beer-Sheva, Israel
naty@post.bgu.ac.il

Rotem Tsabary
Weizmann Institute of Science, Rehovot, Israel
rotem.tsabary@weizmann.ac.il

Hoeteck Wee
CNRS, ENS, PSL, Paris, France
wee@di.ens.fr

Abstract

We define and study a new cryptographic primitive, named *One-One Constrained Pseudorandom Functions*. In this model there are two parties, Alice and Bob, that hold a common random string K, where Alice in addition holds a predicate $f : [N] \to \{0, 1\}$ and Bob in addition holds an input $x \in [N]$. We then let Alice generate a key K_f based on f and K, and let Bob evaluate a value K_x based on x and K. We consider a third party that sees the values (x, f, K_f) and the goal is to allow her to reconstruct K_x whenever $f(x) = 1$, while keeping K_x pseudorandom whenever $f(x) = 0$. This primitive can be viewed as a relaxation of constrained PRFs, such that there is only a single key query and a single evaluation query.

We focus on the information-theoretic setting, where the one-one cPRF has perfect correctness and perfect security. Our main results are as follows.

1. *A Lower Bound.* We show that in the information-theoretic setting, any one-one cPRF for punctured predicates is of exponential complexity (and thus the lower bound meets the upper bound that is given by a trivial construction). This stands in contrast with the well known GGM-based punctured PRF from OWF, which is in particular a one-one cPRF. This also implies a similar lower bound for all NC1.
2. *New Constructions.* On the positive side, we present efficient information-theoretic constructions of one-one cPRFs for a few other predicate families, such as equality predicates, inner-product predicates, and subset predicates. We also show a generic AND composition lemma that preserves complexity.
3. *An Amplification to standard cPRF.* We show that all of our one-one cPRF constructions can be amplified to a standard (single-key) cPRF via any key-homomorphic PRF that supports linear computations. More generally, we suggest a new framework that we call *the double-key model* which allows to construct constrained PRFs via key-homomorphic PRFs.
4. *Relation to CDS.* We show that one-one constrained PRFs imply conditional disclosure of secrets (CDS) protocols.

We believe that this simple model can be used to better understand constrained PRFs and related cryptographic primitives, and that further applications of one-one constrained PRFs and our double-key model will be found in the future, in addition to those we show in this paper.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques; Theory of computation → Cryptographic primitives

Keywords and phrases Constrained pseudorandom functions, function secret-sharing, conditional disclosure of secrets

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.13

Funding *Naty Peter*: Supported by ISF grant 152/17, by a grant from the Cyber Security Research Center at Ben-Gurion University of the Negev, and by the Frankel center for computer science.
Rotem Tsabary: Supported by the Israel Science Foundation (Grant No. 468/14), Binational Science

1st Conference on Information-Theoretic Cryptography (ITC 2020).
Editors: Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs; Article No. 13; pp. 13:1–13:22

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

Foundation (Grants No. 2016726, 2014276), by the European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482) and via Project PROMETHEUS (Grant 780701), and by the Azrieli Fellows Program.
Hoeteck Wee: Supported in part by ERC Project aSCEND (H2020 639554).

1 Introduction

In this paper we define and study a new cryptographic primitive, named *One-One Constrained Pseudorandom Functions*. In this model there are two parties, Alice and Bob, that hold a common random string K. In addition, Alice holds a predicate $f : [N] \to \{0,1\}$ and Bob holds an input $x \in [N]$. We then let Alice generate a key K_f based on f and K, and let Bob evaluate a value K_x based on x and K. We consider a third party that sees the values (x, f, K_f) and the goal is to allow her to reconstruct K_x whenever $f(x) = 1$, while keeping K_x pseudorandom whenever $f(x) = 0$.

This primitive can be viewed as a relaxation of constrained PRFs, such that there is only a single key query and a single evaluation query. In a constrained PRF (first defined in [12, 13, 23]), there is a master secret key msk with which it is possible to evaluate the PRF on all inputs x, and in addition there are constrained keys sk_f respective to predicates f, where sk_f is derived from msk, such that sk_f allows to evaluate the PRF only on inputs x where $f(x) = 1$, but on all points where $f(x) = 0$ the PRF value remains pseudorandom even given sk_f. Through this point of view, K is the master secret key of the PRF, K_x is the evaluation of the PRF on an input x and K_f is a constrained key for the predicate f.

We believe that the simplified model of one-one cPRF can be used to better understand constrained PRFs and related cryptographic primitives, and that further applications of one-one constrained PRFs will be found in the future, in addition to those we show in this paper.

Our Contributions. Our main focus is on the information-theoretic setting, where we require perfect correctness and perfect security. Our main results are as follows.

1. *A Lower Bound.* We show that in the information-theoretic setting, any one-one cPRF for punctured predicates is of exponential complexity (and thus the lower bound meets the upper bound that is given by a trivial construction). This stands in contrast with the well known GGM-based punctured PRF from OWF, which is in particular a one-one cPRF. This also implies a similar lower bound for all NC1.
2. *New Constructions.* On the positive side, we present efficient information-theoretic constructions of one-one cPRFs for a few other predicate families, such as equality predicates, inner-product predicates, and subset predicates. We also show a generic AND composition lemma that preserves complexity.
3. *An Amplification to cPRF.* We define a special *double-key* model and show that any one-one cPRF in this model, when combined with a key-homomorphic PRF, can support multiple evaluation queries. We then show that all of our constructions can be initialized in the double-key model, which implies that all of our constructions can be amplified to a standard (single-key) cPRF via any key-homomorphic PRF that supports linear computations. More generally, this approach reduces the task of constructing constrained PRFs to the possibly simpler task of constructing one-one constrained PRFs in the double-key model, and we believe that this framework will have more applications in the future.

4. *Relation to CDS.* We show that one-one constrained PRFs imply conditional disclosure of secrets (CDS) protocols, a cryptographic primitive that is used to construct secure protocols such as attribute based encryption, symmetrically-private information retrieval protocols, and secret-sharing schemes.
5. *Computational Constructions.* To complete the picture, we also go over existing computational cPRFs in the literature, which are in particular computational one-one cPRFs.

2 Technical Overview

2.1 A Lower Bound

We begin with describing our lower bound theorem and its proof. Consider the family of punctured predicates f_y over some field $\mathbb{F}$, such that for all $x, y \in \mathbb{F}$ it hold that $f_y(x) = 1$ if and only if $x \neq y$. We show that any perfect one-one cPRF for this predicate family must have keys K_{f_y} of size $\Omega(|\mathbb{F}|)$. To show that, we first argue that for every $x \in \mathbb{F}$ it must be that K_x has at least one bit of entropy, even given all of the values $\{K_{x'}\}_{x' \neq x}$. This is due to the correctness and security properties respective to the predicate f_x, which means that K_{f_x} allows to reconstruct all $\{K_{x'}\}_{x' \neq x}$ while keeping K_x random. Secondly, due to the fact that K_{f_y} allows to compute $\{K_x\}_{x \neq y}$ (by correctness), and since each such K_x has at least one independent bit of entropy (by the previous claim), it must be the case that K_{f_y} has at least $|\mathbb{F}| - 1 = \Omega(|\mathbb{F}|)$ bits of entropy.

2.2 New Constructions

A Generic Construction. We now describe a simple one-one constrained PRF for general functions over some field $\mathbb{F}$ with complexity $O(\mathbb{F})$. In this construction, we choose a random bit $k_y \xleftarrow{\$} \{0,1\}$ for every possible input y, and let the common random string be a concatenation of all of those values $K = \{k_y\}_{y \in \mathbb{F}}$. Alice, which holds a predicate f, returns the values K_y for all the inputs y such that $f(y) = 1$, and Bob, which holds an input x, simply returns $K_x = k_x$. Security and correctness follow immediately (in fact, this construction is also secure with multiple key queries and evaluation queries). The size of the common random string is therefore at most $|\mathbb{F}|$, and for a specific function f, the size of K_f is $|f^{-1}(1)|$.

Equality Testing. Our efficient construction for equality testing over a field $\mathbb{F}$ is as follows. The common random string K consists of two random field elements k_0, k_1. The functions that Alice and Bob compute over their inputs x and f_y respectively, where $f_y(x) = 1$ if and only if $x = y$, are identical: $K_x = k_1 x + k_0$ and $K_{f_y} = k_1 y + k_0$. This is essentially a degree-1 random polynomial computed over two elements, therefore K_x and K_{f_y} look independently random as long as $x \neq y$. This construction can be generalized to any constant number of evaluation / key queries by using a random polynomial of higher degree.

Subset Predicates. Subset predicates are defined with respect to a universe $[N] = \{1, \ldots, N\}$, where the input space is all subsets $X \subseteq [N]$ and the predicates f_Y are characterized by subsets $Y \subseteq [N]$ such that $f_Y(X) = 1$ if and only if $X \subseteq Y$. Our efficient construction for subset predicates is as follows. For every $i \in [N]$ there is a random bit k_i in the common string K. We then define $K_X = \oplus_{i \in X} k_i$ and $K_{f_Y} = \{k_i\}_{i \in Y}$. It is easy to see correctness. In addition, whenever $X \not\subseteq Y$, there exists an $k_i \notin K_{f_Y}$ that completely randomizes K_X.

Inner-Product Predicates. Inner-product predicates for vectors of length ℓ over a field $\mathbb{F}$ are defined such that for every $\mathbf{x}, \mathbf{y} \in \mathbb{F}^\ell$ it holds that $f_{\mathbf{y}}(\mathbf{x}) = 1$ if and only if $\langle \mathbf{y}, \mathbf{x} \rangle = 0$. We now describe our efficient construction. The common random string K consists of a random vector $\mathbf{v}$ and a random field element w. We then define $K_{f_{\mathbf{y}}} = w\mathbf{y} + \mathbf{v}$ and $K_{\mathbf{x}} = \langle \mathbf{v}, \mathbf{x} \rangle$. Correctness holds since whenever $\langle \mathbf{y}, \mathbf{x} \rangle = 0$ it holds that $\langle K_{f_{\mathbf{y}}}, \mathbf{x} \rangle = K_{\mathbf{x}}$. Security holds since $\mathbf{v}$ has one degree of freedom given $K_{f_{\mathbf{y}}}, \mathbf{y}$ and therefore it completely randomizes $K_{\mathbf{x}}$.

2.3 Amplification via Key-Homomorphic PRF

We consider one-one cPRFs that satisfy an additional property and show that such one-one cPRFs can be boosted to standard (single-key) cPRFs via key-homomorphic PRFs. We then show that all of our information-theoretic one-one cPRFs satisfy this property, thus receiving new cPRF constructions.

In more detail, we require an alternative algorithm for Alice, that on input K, f produces a *double-key* $(K_f, \hat{K}_f)$. Such double-key should have the property that $\hat{K}_f$ looks uniformly random even given K_f, but on the other hand, given both of the key parts $(K_f, \hat{K}_f)$ it should be possible to reconstruct K_x for all x (regardless of $f(x)$).

Recall that in a key-homomorphic PRF, given an evaluation $\mathsf{PRF}_k(x)$ of the PRF over some input x and a key k, it is possible to publicly evaluate a function g over the key which results in the value $\mathsf{PRF}_{g(k)}(x)$. Our construction uses a key-homomorphic PRF to homomorphically evaluate the function that takes a double-key $(K_f, \hat{K}_f)$ and outputs K_x. Security relies on the fact that $\hat{K}_f$ looks uniform even given K_f, which implies that multiple evaluations of the form $\mathsf{PRF}_{\hat{K}_f}(x)$ look uniform by the security of the PRF. In the construction we define the cPRF evaluation as $r_x := \mathsf{PRF}_{K_x}(x)$. In the security proof we use the homomorphic evaluation procedure to convert $\mathsf{PRF}_{\hat{K}_f}(x)$ into $r_x = \mathsf{PRF}_{K_x}(x)$.

Lastly, we show that all of our information-theoretic constructions can be instantiated in the double-key model, where the required homomorphic computation is linear. In more detail, in the equality testing construction we let $K_y = k_0 + yk_1$ and $\hat{K}_y = yk_1$, in the subset predicates construction we define $K_Y = \{k_i\}_{i \in Y}$ and $\hat{K}_Y = \{k_i\}_{i \in [N]/Y}$ and in the inner-product construction we define $K_{\mathbf{y}} = w\mathbf{y} + \mathbf{v}$ and $\hat{K}_{\mathbf{y}} = w$.

2.4 One-One Constrained PRFs and CDS Protocols

We study the connection between one-one constrained PRFs and CDS protocols. CDS protocols is a cryptographic primitive, introduced by Gertner et al. [20]. In a CDS protocol, each of two parties, Alice and Bob, holds a private input and sends one message to a referee, which knows the inputs of the parties, and should learn a secret held by the parties if and only if the inputs of the parties satisfy some condition (e.g., if the inputs are equal).

CDS protocols can be easily generalized into multi-party CDS protocols, and they are used to construct attribute based encryption [5, 18, 28], symmetrically-private information retrieval protocols [20], priced oblivious transfer [1], and secret-sharing schemes [2, 3, 8, 9, 24].

We present a transformation from one-one constrained PRFs to CDS protocols. In particular, we show that a one-one constrained PRF implies a CDS protocol for the index predicate. By the reduction of [18] from CDS protocols for general predicates to CDS protocols for the index predicate, we obtain a transformation from one-one constrained PRFs to CDS protocols for general predicates. This transformation preserve complexity, i.e., the message size of the resulting CDS protocol is the complexity of the one-one constrained PRF.

Private Simultaneous Messages Protocols. Another similar primitive is private simultaneous messages (PSM) protocols, presented by Feige et al. [16], which is a private case of MPC protocols. In a PSM protocol, each of the parties, Alice and Bob, holds a private input for a two-input function, and each of them sends only one message to a referee, which is based on its input and a common random string, such that the referee should be able to compute the function on the inputs of Alice and Bob using the messages it gets, without learn any additional information about the inputs of the parties. As CDS protocols, PSM protocols can be generalized into multi-party PSM protocols, and they imply some other cryptographic protocols, such as constant round MPC protocols [22], generalized oblivious transfer protocols [22], and zero-information Arthur-Merlin protocols [4].

The best known PSM protocol for general functions $f : [N] \times [N] \to \{0,1\}$ has message size $O(\sqrt{N})$ [7], so by our lower bound of $\Omega(N)$ on the complexity of one-one constrained PRFs, we cannot get a transformation that preserve complexity from PSM protocols to one-one constrained PRFs. There is a transformation that preserve message size from PSM protocols to CDS protocols [7, 20]; the other direction (an existence of transformation from CDS protocols to PSM protocols) is an open problem, and although there is no evidence that shows an equivalence or separation between CDS and PSM protocols, the best known CDS protocols [25] have better message size than the best known PSM protocols [7]. Studying the connections between one-one constrained PRFs and CDS protocols or PSM protocols may help understanding the connection between CDS protocols and PSM protocols and the bounds on the message size of CDS and PSM protocols.

3 Preliminaries

Notations. For any $n \in \mathbb{N}$ we use $[n]$ to denote the set $\{1, \ldots, n\}$. For any set S we use $s \xleftarrow{\$} S$ to denote a uniformly random sample s from S. For any distribution X we use $x \leftarrow X$ to denote a value x that is sampled according to the distribution X. For any $n \in \mathbb{N}$ we use $\mathcal{U}_n$ to denote the uniform distribution over the strings of length n, and for any set S we use $\mathcal{U}_S$ to denote the uniform distribution over the elements in S.

3.1 Entropy and Indistinguishability

▶ **Definition 1** (Shannon Entropy). *For a random variable X and $x \in \sup(X)$, the sample entropy of x with respect to X is $H_X(x) = \log(1/Pr[X = x])$. The Shannon entropy of X is then defined as*

$$H(X) = E_{x \leftarrow X}[H_X(x)] \ .$$

For random variables X, Y, the Shannon entropy of X conditioned on Y is

$$H(X|Y) = H(X,Y) - H(Y) \ .$$

▶ **Definition 2** (Statistical Distance). *The statistical distance between two random variables X_1, X_2 over a finite domain $\mathcal{X}$ is*

$$\Delta(X_1, X_2) = \frac{1}{2} \sum_{x \in \mathcal{X}} |Pr[X_1 = x] - Pr[X_2 = x]| \ .$$

We say that X_1, X_2 are δ-close if $\Delta(X_1, X_2) \leq \delta$.

X_1, X_2 are 0-close if and only if $Pr[X_1 = x] = Pr[X_2 = x]$ for all $x \in \mathcal{X}$, and in that case we say that they are identically distributed.

A function $\varepsilon(\cdot)$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large n's, it holds that $\varepsilon(n) < 1/p(n)$. We define three notions of indistinguishability as follows.

▶ **Definition 3** (Indistinguishability). *Let $X = \{X_n\}_{n\in\mathbb{N}}, Y = \{Y_n\}_{n\in\mathbb{N}}$ be two distribution ensembles.*

1. *X and Y are* perfectly *indistinguishable if for every $n \in \mathbb{N}$, the random variables X_n and Y_n are identically distributed.*
2. *X and Y are* statistically *indistinguishable if there exists a negligible function $\varepsilon(\cdot)$ such that for every $n \in \mathbb{N}$, X_n and Y_n are $\varepsilon(n)$-close.*
3. *X and Y are* computationally *indistinguishable if for every non-uniform PPT distinguisher D, there exists a negligible function $\varepsilon(\cdot)$ such that for every $n \in \mathbb{N}$,*

$$|[Pr[D(X_n) = 1] - [Pr[D(Y_n) = 1]| < \varepsilon(n) \ .$$

3.2 Notions of Pseudorandom Functions

▶ **Definition 4** (Constrained PRFs). *A* constrained *pseudorandom function (cPRF) for a predicate family $\mathcal{F}$ and an input space $\mathcal{X}$ is defined by the algorithms* $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Con}, \mathsf{ConEval})$ *where:*

- $\mathsf{KeyGen}(1^\lambda)$ *is a PPT algorithm that takes as input a security parameter λ and outputs a master key* msk.
- $\mathsf{Eval}(\mathsf{msk}, x)$ *is a deterministic algorithm that takes as input the master secret key* msk *and an input $x \in \mathcal{X}$, and outputs a string $r_x \in \{0,1\}^*$.*
- $\mathsf{Con}(\mathsf{msk}, f)$ *is a PPT algorithm that takes as input the master secret key* msk *and a predicate $f \in \mathcal{F}$, and outputs a constrained key sk_f.*
- $\mathsf{ConEval}(\mathsf{sk}_f, x)$ *is a deterministic algorithm that takes as input a constrained key sk_f and an input $x \in \mathcal{X}$, and outputs a string $r'_x \in \{0,1\}^*$.*

Correctness of Constrained Keys. *The scheme is* correct *if for all $x \in \mathcal{X}$ and $f \in \mathcal{F}$ for which $f(x) = 1$, and for all $\mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda)$, $\mathsf{sk}_f \leftarrow \mathsf{Con}(\mathsf{msk}, f)$, it holds that*

$$\mathsf{Eval}(\mathsf{msk}, x) = \mathsf{ConEval}(\mathsf{sk}_f, x) \ .$$

(Single-Key) Pseudorandomness. *The security requirement is captured via a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as follows.*

- *$\mathcal{C}$ computes $\mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda)$.*
- *In an order of her choice, $\mathcal{A}$ makes an arbitrary number of* evaluation queries*, a single* challenge query *and a single* key query*:*
  - Evaluation Query: *$\mathcal{A}$ selects $x \in \mathcal{X}$ and sends it to $\mathcal{C}$. In return, $\mathcal{C}$ computes $r_x \leftarrow \mathsf{Eval}(\mathsf{msk}, x)$ and sends it to $\mathcal{A}$.*
  - Challenge Query: *$\mathcal{A}$ selects $x^* \in \mathcal{X}$ and sends it to $\mathcal{C}$. In return, $\mathcal{C}$ computes $r_{x^*} \leftarrow \mathsf{Eval}(\mathsf{msk}, x^*)$ and samples a bit $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, it sends r_{x^*} to $\mathcal{A}$, otherwise it sends to $\mathcal{A}$ a random value $u \leftarrow \mathcal{U}_{|r_{x^*}|}$.*
  - Key Query: *$\mathcal{A}$ selects $f^* \in \mathcal{F}$ and sends it to $\mathcal{C}$. In return, $\mathcal{C}$ computes $\mathsf{sk}_{f^*} \leftarrow \mathsf{Con}(\mathsf{msk}, f^*)$ and sends sk_{f^*} to $\mathcal{A}$.*
- *$\mathcal{C}$ sends to $\mathcal{A}$ a bit b'.*

The advantage *of $\mathcal{A}$ in the game is $Pr[b = b'] - \frac{1}{2}$ where the probability is over the coins of $\mathcal{C}$ and $\mathcal{A}$. The adversary $\mathcal{A}$ is* admissible *if $f^*(x^*) = 0$ and x^* does not appear in any of the evaluation queries. The scheme is computationally secure if for any PPT admissible adversary $\mathcal{A}$, her advantage in the game is* $\mathrm{negl}(\lambda)$.

We define key-selective *pseudorandomness identically to the definition above, except that $\mathcal{A}$ is forced to make the key query first.*

▶ **Definition 5** (Key-Homomorphic PRFs). *A* key-homomorphic *pseudorandom function (khPRF) for an input space $\mathcal{X}$, a key space $\mathcal{K}$, and a function family $\mathcal{G} = \{\mathcal{G}_n : \mathcal{K}^n \to \mathcal{K}\}_{n\in\mathbb{N}}$, is defined by the algorithms* $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{HomKeyEval})$ *where:*

- $\mathsf{KeyGen}(1^\lambda)$ *is a PPT algorithm that takes as input a security parameter λ and outputs a key $\mathsf{sk} \in \mathcal{K}$ and possibly public parameters* pp.
- $\mathsf{Eval}(\mathsf{sk}, x)$ *is a deterministic algorithm that takes as input a key $\mathsf{sk} \in \mathcal{K}$ and an input $x \in \mathcal{X}$, and outputs $r_x \in \{0,1\}^*$.*
- $\mathsf{HomKeyEval}(g, r_x)$ *is a PPT algorithm that takes as input a function $g \in \mathcal{G}$ and a value $r_x \in \{0,1\}^*$, and outputs a value $\hat{r}_x \in \{0,1\}^*$.*

Correctness of Homomorphic Key Evaluation. *The scheme is* correct *if for all $x \in \mathcal{X}$, all $g \in \mathcal{G}$ where $g : \mathcal{K}^n \to \mathcal{K}$, and all $\{\mathsf{sk}_i\}_{i\in[n]} \in \mathcal{K}$, it holds that*

$$\mathsf{HomKeyEval}\left(g, \mathsf{Eval}(\mathsf{sk}_1, x), \ldots, \mathsf{Eval}(\mathsf{sk}_n, x)\right) = \mathsf{Eval}\left(g(\mathsf{sk}_1, \ldots, \mathsf{sk}_n), x\right) \ .$$

Pseudorandomness. *The security requirement is captured via a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as follows.*

- *$\mathcal{C}$ computes $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$.*
- *In an order of her choice, $\mathcal{A}$ makes an arbitrary number of* evaluation queries *and a single* challenge query*:*
  - Evaluation Query: *$\mathcal{A}$ selects $x \in \mathcal{X}$ and sends it to $\mathcal{C}$. In return, $\mathcal{C}$ computes $r_x \leftarrow \mathsf{Eval}(\mathsf{sk}, x)$ and sends it to $\mathcal{A}$.*
  - Challenge Query: *$\mathcal{A}$ selects $x^* \in \mathcal{X}$ and sends it to $\mathcal{C}$. In return, $\mathcal{C}$ computes $r_{x^*} \leftarrow \mathsf{Eval}(\mathsf{sk}, x^*)$ and samples a bit $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, it sends r_{x^*} to $\mathcal{A}$, otherwise it sends to $\mathcal{A}$ a random value $u \leftarrow \mathcal{U}_{|r_{x^*}|}$.*
- *$\mathcal{C}$ sends to $\mathcal{A}$ a bit b'.*

The advantage *of $\mathcal{A}$ in the game is $Pr[b = b'] - \frac{1}{2}$ where the probability is over the coins of $\mathcal{C}$ and $\mathcal{A}$. The adversary $\mathcal{A}$ is* admissible *if x^* does not appear in any of the evaluation queries. The scheme is computationally secure if for any PPT admissible adversary $\mathcal{A}$, her advantage in the game is* $\mathrm{negl}(\lambda)$.

4 Definition of One-One Constrained PRFs

▶ **Definition 6** (One-One Constrained PRFs). *A* One-One Constrained Pseudorandom Function *for a predicate family $\mathcal{F}$ and an input space $\mathcal{X}$ is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{A}, \mathsf{B}, \mathsf{Rec})$ *with the following syntax.*

- $\mathsf{Setup}(1^\lambda) \to K$ *is a PPT algorithm that (possibly) takes a security parameter λ and outputs a common random string K.*
- $\mathsf{A}(K, f) \to K_f$ *is a PPT algorithm that takes a common random string K and a predicate $f \in \mathcal{F}$. It outputs a key K_f.*
- $\mathsf{B}(K, x) \to K_x$ *is a deterministic algorithm that takes a common random string K and an input $x \in \mathcal{X}$. It outputs a value K_x.*
- $\mathsf{Rec}(x, f, K_f) \to K'_x$ *is a deterministic algorithm that takes an input $x \in \mathcal{X}$, a predicate $f \in \mathcal{F}$, and a key K_f. It outputs a value K'_x.*

Complexity. *We say that the scheme is of complexity $p(\cdot,\cdot,\cdot)$ if for every $(\lambda, \mathcal{X}, \mathcal{F})$, for every $x \in \mathcal{X}$ and $f \in \mathcal{F}$, for every K, K_f, K_x where $K \leftarrow setup(1^\lambda)$, $K_f \leftarrow \mathsf{A}(K, f)$, and $K_x \leftarrow \mathsf{B}(K, x)$, the size of (K, K_f, K_x) is $O(p(\lambda, |\mathcal{X}|, |\mathcal{F}|))$.*

Correctness. *A one-one constrained PRF is* correct *if for all $f \in \mathcal{F}$ and $x \in \mathcal{X}$ for which $f(x) = 1$, for all $K \leftarrow \mathsf{Setup}(1^\lambda)$, and for all coins of A, it holds that*

$$\mathsf{Rec}(x, f, \mathsf{A}(K, f)) = \mathsf{B}(K, x) \ .$$

Security. *The security requirement is captured via a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ as follows.*

- *$\mathcal{C}$ computes $K \leftarrow \mathsf{Setup}(1^\lambda)$.*
- *In an order of her choice, $\mathcal{A}$ makes two queries:*
  - Challenge Query: *$\mathcal{A}$ selects $x^* \in \mathcal{X}$ and sends it to $\mathcal{C}$. In return, $\mathcal{C}$ computes $K_{x^*} \leftarrow \mathsf{B}(K, x^*)$ and samples a bit $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, it sends K_{x^*} to $\mathcal{A}$, otherwise it sends to $\mathcal{A}$ a random value $u \leftarrow \mathcal{U}_{|K_{x^*}|}$.*
  - Key Query: *$\mathcal{A}$ selects $f^* \in \mathcal{F}$ and sends it to $\mathcal{C}$. In return, $\mathcal{C}$ computes $K_{f^*} \leftarrow \mathsf{A}(K, f^*)$ and sends K_{f^*} to $\mathcal{A}$.*
- *$\mathcal{A}$ sends to $\mathcal{C}$ a bit b'.*

The advantage *of $\mathcal{A}$ in the game is $Pr[b = b'] - \frac{1}{2}$ where the probability is over the coins of $\mathcal{C}$ and $\mathcal{A}$. The scheme is* perfectly (/statistically /computationally) secure *if for any unbounded (/unbounded /PPT) adversary $\mathcal{A}$ that selects x^*, f^* for which $f^*(x^*) = 0$, her advantage in the game is 0 (/negl(λ) /negl(λ)).*

We define key-selective *(resp.* input-selective*) security identically to the definition above, except that $\mathcal{A}$ is forced to make the key query (resp. challenge query) first.*

Indistinguishability Based Perfect Security. *In the* prefect *setting, we can omit the security parameter and selective security implies full security, therefore the following definition is equivalent to the one above:*

A one-one constrained PRF is perfectly secure *if for all $f^* \in \mathcal{F}$ and $x^* \in \mathcal{X}$ for which $f^*(x^*) = 0$, the following distributions are identical:*

$$(f^*, x^*, K_{f^*}, K_{x^*}) \equiv (f^*, x^*, K_{f^*}, u)$$

where $K \leftarrow \mathsf{Setup}()$, $K_{f^*} \leftarrow \mathsf{A}(K, f^*)$, $K_{x^*} \leftarrow \mathsf{B}(K, x^*)$, $u \leftarrow \mathcal{U}_{|K_{x^*}|}$.

Note. This is essentially a constrained PRF secure against one constrained key and one evaluation query. Related primitives: constrained PRFs, conditional disclosure of secrets (CDS), function secret-sharing.

5 A Lower Bound

We now show that for inequality predicates, the upper bound that we show in Theorem 10 is tight, i.e., it is not possible to do better than the trivial construction. This means we cannot hope to achieve efficient information-theoretic constructions for NC1.

▶ **Theorem 7.** *Let $\mathcal{F} = \{f_{\neq y}\}_{y \in \mathbb{F}}$ be the family of punctured predicates over some field $\mathbb{F}$, i.e., for any fixed $y \in \mathbb{F}$, $f_{\neq y} : \mathbb{F} \to \{0,1\}$ where $f_{\neq y}(x) = 1$ if and only if $x \neq y$. Then, for any field $\mathbb{F}$, any one-one constrained PRF for $\mathcal{F}$ with perfect correctness and perfect security is of complexity $\Omega(|\mathbb{F}|)$.*

We now prove Theorem 7. Denote $N = |\mathbb{F}|$ and consider all of the field elements $x_1, \ldots, x_N$ according to some fixed order. Consider the random variable $K \leftarrow \mathsf{Setup}()$ and for all $i \in [N]$ consider the random variables K_i and $K_{\neq i}$ defined as:

$$K_i \leftarrow \mathsf{B}(K, x_i), \qquad K_{\neq i} \leftarrow \mathsf{A}(K, f_{\neq x_i}) \ .$$

▷ Claim 8. For all $i \in [N]$, it holds that $H(K_i \mid K_1, \ldots, K_{i-1}) \geq 1$.

Proof. Fix an $i \in [N]$. By the perfect security it holds that $(K_{\neq i}, K_i) \equiv (K_{\neq i}, u)$ where $u \leftarrow \mathcal{U}_{|K_x|}$. Since $H(u \mid K_{\neq i}) \geq 1$, it follows that

$$H(K_i \mid K_{\neq i}) \geq 1 \ . \tag{1}$$

By the perfect correctness, $K_{\neq i}$ uniquely determines all of the values $\{K_j\}_{j\neq i}$. That is,

$$H(\{K_j\}_{j\neq i} \mid K_{\neq i}) = 0 \ ,$$

i.e., (by Definition 1)

$$H(\{K_j\}_{j\neq i}, K_{\neq i}) = H(K_{\neq i}) \ ,$$

and therefore

$$H(K_{\neq i}) \geq H(\{K_j\}_{j\neq i}) \ .$$

Since $\{K_1, \ldots, K_{i-1}\} \subseteq \{K_j\}_{j\neq i}$, it follows that

$$H(K_{\neq i}) \geq H(K_1, \ldots, K_{i-1}) \ . \tag{2}$$

Lastly, equations (1) and (2) imply that

$$H(K_i \mid K_1, \ldots, K_{i-1}) \geq H(K_i \mid K_{\neq i}) \geq 1 \ . \qquad \triangleleft$$

▷ Claim 9. For any subset $I \subseteq [N]$, it holds that $H(\{K_i\}_{i\in I}) \geq |I|$.

Proof. Fix a subset $I \subseteq [N]$ and denote $I = \{i_1, \ldots, i_{|I|}\}$ such that $i_1 < i_2 < \cdots < i_{|I|}$. By Claim 8, for all $j = 1, \ldots, |I|$ it holds that

$$H(K_{i_j} \mid K_{i_1}, \ldots, K_{i_{j-1}}) \geq H(K_{i_j} \mid K_1, \ldots, K_{i_j-1}) \geq 1 \ ,$$

i.e., (by Definition 1)

$$H(K_{i_j}, K_{i_1}, \ldots, K_{i_{j-1}}) - H(K_{i_1}, \ldots, K_{i_{j-1}}) \geq 1 \ ,$$

i.e.,

$$H(K_{i_j}, K_{i_1}, \ldots, K_{i_{j-1}}) \geq 1 + H(K_{i_1}, \ldots, K_{i_{j-1}}) \ .$$

Since this is true for all $j = 1, \ldots, |I|$, it follows that

$$\begin{aligned} H(\{K_i\}_{i\in I}) &= H(K_{i_{|I|}}, K_{i_1}, \ldots, K_{i_{|I|-1}}) \\ &\geq 1 + H(K_{i_1}, \ldots, K_{i_{|I|-1}}) \\ &\geq 1 + 1 + H(K_{i_1}, \ldots, K_{i_{|I|-2}}) \\ &\ldots \\ &\geq |I| \ . \end{aligned}$$ $\triangleleft$

Consider now an arbitrary $i \in [N]$. From Claim 9 it holds that $H(\{K_j\}_{j\neq i}) \geq N - 1$. Since $H(K_{\neq i}) \geq H(\{K_j\}_{j\neq i})$ (see the proof of Claim 8), it follows that $H(K_{\neq i}) \geq N - 1$. Hence, for any $i \in [N]$ the size of $K_{\neq i}$ is at least $N - 1$, which completes the proof. ◀

6 Information-Theoretic Constructions

In this section we consider one-one constrained PRFs with perfect correctness and security. We begin in Section 6.1 with a generic construction for all predicates over a field $\mathbb{F}$, which is of complexity $O(|\mathbb{F}|)$. We then present more efficient constructions for specific predicate families; We begin in Section 6.2 with a complexity-preserving composition lemma for the AND operator. That is, we show that given a one-one cPRF for predicates families $\mathcal{F}^1$ and $\mathcal{F}^2$, there is a one-one cPRF for the predicate family $\mathcal{F}^1 \wedge \mathcal{F}^2$ of proportional complexity. In Section 6.3 we show a construction for the equality predicate over a field $\mathbb{F}$ with complexity $O(\log|\mathbb{F}|)$, which extends to a ℓ-vector-equality construction of complexity $O(\ell \cdot \log|\mathbb{F}|)$ via the generic AND composition. In Section 6.4 we show a construction for the subset relation with complexity $O(N)$, where N is the maximal size of sets, which extends to t-CNFs with complexity $O(\binom{\ell}{t} \cdot |\mathbb{F}|^t \cdot \log|\mathbb{F}|)$ as pointed out by [15,27]. In Section 6.5 we construct a one-one cPRF for inner-product predicates for vectors in $\mathbb{F}^\ell$ with complexity $O(\ell \cdot \log|\mathbb{F}|)$, which can be extended to polynomials via embedding of polynomial zero testing as inner-product.

6.1 Generic Predicates

▶ **Theorem 10.** *Let $\mathcal{F}$ be the family of all predicates over some field $\mathbb{F}$, i.e., any $f \in \mathcal{F}$ is of the form $f : \mathbb{F} \to \{0,1\}$. Then, there is a one-one constrained PRF for $\mathcal{F}$ with perfect correctness, perfect security, and complexity $O(|\mathbb{F}|)$.*

Proof. The construction is as follows.

- $\mathsf{Setup}()$: For any $y \in \mathbb{F}$ sample $k_y \xleftarrow{\$} \{0,1\}$. Output $K = \{k_y\}_{y\in\mathbb{F}}$.
- $\mathsf{A}(K, f)$: Parse $K = \{k_y\}_{y\in\mathbb{F}}$ and output $K_f = \{k_y : f(y) = 1\}_{y\in\mathbb{F}}$.
- $\mathsf{B}(K, x)$: Parse $K = \{k_y\}_{y\in\mathbb{F}}$ and output $K_x = k_x$.
- $\mathsf{Rec}(x, f, K_f)$: Parse $K_f = \{k_y : f(y) = 1\}_{y\in\mathbb{F}}$. If $f(x) = 1$ then output k_x, otherwise output $\perp$.

Correctness. If $f(x) = 1$ then $k_x \in K_f$.

Security. Consider $K \leftarrow \mathsf{Setup}()$, $K_f \leftarrow \mathsf{A}(K, f)$, $K_x \leftarrow \mathsf{B}(K, x)$, $u \xleftarrow{\$} \{0,1\}$. If $f(x) = 0$ then $k_x \notin K_f$. Since k_x is a uniformly sampled bit and $H(k_x|K_f) = H(k_x)$, it holds that

$$(f, x, K_f, K_x) \equiv (f, x, K_f, u) \ .$$ ◀

6.2 AND Composition

▶ **Lemma 11.** *Let $(\mathsf{Setup}^1, \mathsf{A}^1, \mathsf{B}^1, \mathsf{Rec}^1)$ and $(\mathsf{Setup}^2, \mathsf{A}^2, \mathsf{B}^2, \mathsf{Rec}^2)$ be perfect one-one constrained PRFs for some predicate families $\mathcal{F}^1 = \{f^1 : \mathbb{F} \to \{0,1\}\}$ and $\mathcal{F}^2 = \{f^2 : \mathbb{F} \to \{0,1\}\}$ respectively for some field $\mathbb{F}$, with complexity bounded by some functions P^1 and P^2 respectively. Then, there is a perfect one-one constrained PRF for the predicate family $\mathcal{F}^1 \wedge \mathcal{F}^2 = \{f^1 \wedge f^2 : \mathbb{F} \to \{0,1\}\}_{f^1\in\mathcal{F}^1, f^2\in\mathcal{F}^2}$ with complexity bounded by $P^1 + P^2$.*

Proof. For simplicity we assume that for all x it holds that $\left|K^1_x\right| = \left|K^2_x\right|$. This can always be enforced because it is possible to ignore some of the bits of K_x without compromising correctness and security. The construction is as follows.

- $\mathsf{Setup}()$: Compute $K^1 \leftarrow \mathsf{Setup}^1()$ and $K^2 \leftarrow \mathsf{Setup}^2()$, output $K = (K^1, K^2)$.
- $\mathsf{A}(K, f)$: Parse $K = (K^1, K^2)$ and $f = f^1 \wedge f^2$. Compute $K^1_{f^1} \leftarrow \mathsf{A}^1(K^1, f^1)$ and $K^2_{f^2} \leftarrow \mathsf{A}^2(K^2, f^2)$. Output $K_f = (K^1_{f^1}, K^2_{f^2})$.
- $\mathsf{B}(K, x)$: Parse $K = (K^1, K^2)$. Compute $K^1_x \leftarrow \mathsf{B}^1(K^1, x)$ and $K^2_x \leftarrow \mathsf{B}^2(K^2, x)$. Output $K_x = K^1_x + K^2_x$.

- $\mathsf{Rec}(x, f, K_f)$: Parse $K_f = (K^1_{f^1}, K^2_{f^2})$. Compute $R^1_x \leftarrow \mathsf{Rec}^1(x, f^1, K^1_{f^1})$ and $R^2_x \leftarrow \mathsf{Rec}^2(x, f^2, K^2_{f^2})$, output $R_x = R^1_x + R^2_x$.

Correctness. If $f(x) = 1$ then $f^1(x) = 1 \wedge f^2(x) = 1$, therefore by the correctness of the underlying constructions $R^1_x = K^1_x$ and $R^2_x = K^2_x$, thus $R_x = K_x$.

Security. Consider $K \leftarrow \mathsf{Setup}()$, $K_f \leftarrow \mathsf{A}(K, f)$, $K_x \leftarrow \mathsf{B}(K, x)$, $u \leftarrow \mathcal{U}_{|K_x|}$ where $K = (K^1, K^2)$, $f = f^1 \wedge f^2$, $K_f = (K^1_{f^1}, K^2_{f^2})$ and $K_x = K^1_x + K^2_x$. If $f(x) = 0$ then there exists $i \in \{1, 2\}$ for which $f^i(x) = 0$. By the security of the underlying constructions it holds that

$$(f^i, x, K^i_{f^i}, K^i_x) \equiv (f^i, x, K^i_{f^i}, u^i) \ .$$

where $K^i \leftarrow \mathsf{Setup}^i()$, $K^i_{f^i} \leftarrow \mathsf{A}^i(K^i, f^i)$, $K^i_x \leftarrow \mathsf{B}^i(K^i, x)$, $u^i \leftarrow \mathcal{U}_{|K^i_x|}$. Since the two instances of the underlying constructions are independent (i.e., $H(K^1|K^2) = H(K^1)$ and $H(K^2|K^1) = H(K^2)$), and in particular K^i_x is independent of $(K^{3-i}_{f^{3-i}}, K^{3-i}_x)$, it follows that

$$\begin{aligned}(f, x, K_f, K_x) &= (f, x, (K^1_{f^1}, K^2_{f^2}), K^1_x + K^2_x) \\ &\equiv (f, x, (K^1_{f^1}, K^2_{f^2}), K^{3-i}_x + u^i) \\ &\equiv (f, x, (K^1_{f^1}, K^2_{f^2}), u) \\ &= (f, x, K_f, u) \ .\end{aligned}$$

◀

6.3 Equality Testing

▶ **Theorem 12.** *Let $\mathcal{F} = \{f_y\}_{y \in \mathbb{F}}$ be the family of point predicates over some field $\mathbb{F}$, i.e., for any fixed $y \in \mathbb{F}$, $f_y : \mathbb{F} \to \{0, 1\}$ where $f_y(x) = 1$ if and only if $x = y$. Then, there is a one-one constrained PRF for $\mathcal{F}$ with perfect correctness, perfect security, and complexity $O(\log |\mathbb{F}|)$.*

Proof. The construction is as follows.

- $\mathsf{Setup}()$: Sample $k_0, k_1 \xleftarrow{\$} \mathbb{F}$ and output $K = (k_0, k_1)$.
- $\mathsf{A}(K, f_y)$: Parse $K = (k_0, k_1)$ and output $K_{f_y} = k_0 + yk_1$.
- $\mathsf{B}(K, x)$: Parse $K = (k_0, k_1)$ and output $K_x = k_0 + xk_1$.
- $\mathsf{Rec}(x, f_y, K_{f_y})$: If $x = y$ then output K_{f_y}, otherwise output $\perp$.

Correctness. If $f_y(x) = 1$, i.e., $x = y$, then $K_{f_y} = K_x$.

Security. Consider $K \leftarrow \mathsf{Setup}()$, $K_{f_y} \leftarrow \mathsf{A}(K, f_y)$, $K_x \leftarrow \mathsf{B}(K, x)$, $u \xleftarrow{\$} \mathbb{F}$. If $f_y(x) = 0$, i.e., $x \neq y$, then $(K_{f_y}, K_x) = (k_0 + yk_1,\ k_0 + xk_1)$ are two distinct points on a random linear function defined by (k_0, k_1). Since for every possible value of the uniformly sampled $u \in \mathbb{F}$ there is a unique $(k'_0, k'_1) \in \mathbb{F}^2$ such that $k'_0 + yk'_1 = k_0 + yk_1 = K_{f_y}$ and $k'_0 + xk'_1 = u$, it holds that (f_y, x, K_{f_y}, K_x) and (f_y, x, K_{f_y}, u) are identically distributed. ◀

Extensions. The construction above can be extended to other notions as follows.

- The predicates family of equality testing of *vectors* over some field $\mathbb{F}$, i.e., the family $\mathcal{F} = \{f_{\mathbf{y}}\}_{\mathbf{y} \in \mathbb{F}^\ell}$ where for any fixed $\mathbf{y} \in \mathbb{F}^\ell$, $f_{\mathbf{y}} : \mathbb{F}^\ell \to \{0, 1\}$ such that $f_{\mathbf{y}}(\mathbf{x}) = 1$ if and only if $\mathbf{x} = \mathbf{y}$. The construction is derived via the AND composition lemma (see Lemma 11) and is of complexity $O(\ell \cdot \log |\mathbb{F}|)$.

- A 1-key t-queries variant, i.e., a construction where for any $f_y \in \mathcal{F}$ and any $\mathcal{X} \subseteq \mathbb{F}/y$ of size $|\mathcal{X}| < t$, it holds that

$$(f_y, x, K_{f_y}, \{K_x\}_{x\in\mathcal{X}}) \equiv (f_y, x, K_{f_y}, \{u_x\}_{x\in\mathcal{X}})$$

where $K \leftarrow \mathsf{Setup}(),\ K_{f_y} \leftarrow \mathsf{A}(K, f_y),\ K_x \leftarrow \mathsf{B}(K, x),\ u_x \leftarrow \mathcal{U}_{|K_x|}$. In the construction, sample a random polynomial p of degree t as the common random string K and compute $K_x = p(x),\ K_{f_y} = p(y)$. The complexity is $O(t \cdot \log|\mathbb{F}|)$.

6.4 Subset Predicates

▶ **Theorem 13.** *Let $\mathcal{F} = \{f_Y\}_{Y\subseteq[N]}$ be the family of subset predicate over the set $[N]$, i.e., for any fixed subset $Y \subseteq [N]$, $f_Y : 2^{[N]} \to \{0,1\}$ and $f_Y(X) = 1$ if and only if $X \subseteq Y$. Then, there is a one-one constrained PRF for $\mathcal{F}$ with perfect correctness, perfect security, and complexity $O(N)$.*

Proof. The construction is as follows.

- $\mathsf{Setup}()$: For any $i \in [N]$ sample $k_i \xleftarrow{\$} \{0,1\}$. Output $K = \{k_i\}_{i\in[N]}$.
- $\mathsf{A}(K, f_Y)$: Parse $K = \{k_i\}_{i\in[N]}$ and output $K_{f_Y} = \{k_i\}_{i\in Y}$.
- $\mathsf{B}(K, X)$: Parse $K = \{k_i\}_{i\in[N]}$ and output $K_X = \bigoplus_{i\in X} k_i$.
- $\mathsf{Rec}(X, f_Y, K_{f_Y})$: Parse $K_{f_Y} = \{k_i\}_{i\in Y}$. If $X \subseteq Y$ then compute and output $K'_X = \bigoplus_{i\in X} k_i$, otherwise output $\perp$.

Correctness. If $f_Y(X) = 1$, i.e., $X \subseteq Y$, then $K'_X = K_X$.

Security. If $f_Y(X) = 0$, i.e., $X \not\subseteq Y$, then there exists an index $i^* \in X$ such that $i^* \notin Y$, therefore

$$(f_Y, X, \{k_i\}_{i\in Y}, k_{i^*}) \equiv (f, x, \{k_i\}_{i\in Y}, u)$$

and thus

$$(f_Y, X, \{k_i\}_{i\in Y}, \bigoplus_{i\in X} k_i) \equiv (f_Y, X, \{k_i\}_{i\in Y}, u)$$

and thus

$$(f_Y, X, K_{f_Y}, K_X) \equiv (f_Y, X, K_{f_Y}, u)$$

where $\{k_i\}_{i\in[N]} = K \leftarrow \mathsf{Setup}(),\ K_{f_Y} \leftarrow \mathsf{A}(K, f_Y),\ K_X \leftarrow \mathsf{B}(K, X),\ u \xleftarrow{\$} \{0,1\}$. ◀

Extensions. Using techniques similar to [15, 27], the construction above implies a one-one constrained PRF for the class of t-CNF predicates with inputs of length ℓ over some field $\mathbb{F}$, where the construction is of complexity $O(\binom{\ell}{t} \cdot |\mathbb{F}|^t \cdot \log|\mathbb{F}|)$.

6.5 Inner-Product Predicates

▶ **Theorem 14.** *Let $\mathcal{F} = \{f_{\mathbf{y}}\}_{\mathbf{y}\in\mathbb{F}^\ell}$ be the family of inner-product predicates of length-ℓ vectors over some field $\mathbb{F}$, i.e., for any fixed $\mathbf{y} \in \mathbb{F}^\ell$, $f_{\mathbf{y}} : \mathbb{F}^\ell \to \{0,1\}$ where $f_{\mathbf{y}}(\mathbf{x}) = 1$ if and only if $\langle \mathbf{y}, \mathbf{x}\rangle = 0$. Then, there is a one-one constrained PRF for $\mathcal{F}$ with perfect correctness, perfect security, and complexity $O(\ell \cdot \log|\mathbb{F}|)$.*

Proof. The construction is as follows.

- $\mathsf{Setup}()$: Sample $\mathbf{v} \xleftarrow{\$} \mathbb{F}^\ell$ and $w \xleftarrow{\$} \mathbb{F}$. Output $K = (\mathbf{v}, w)$.
- $\mathsf{A}(K, f_{\mathbf{y}})$: Parse $K = (\mathbf{v}, w)$ and output $K_{f_{\mathbf{y}}} = w\mathbf{y} + \mathbf{v}$.
- $\mathsf{B}(K, \mathbf{x})$: Parse $K = (\mathbf{v}, w)$ and output $K_{\mathbf{x}} = \langle \mathbf{v}, \mathbf{x} \rangle$.
- $\mathsf{Rec}(\mathbf{x}, f_{\mathbf{y}}, K_{f_{\mathbf{y}}})$: Output $\langle K_{f_{\mathbf{y}}}, \mathbf{x} \rangle$.

Correctness. If $f_{\mathbf{y}}(\mathbf{x}) = 1$, i.e., $\langle \mathbf{y}, \mathbf{x} \rangle = 0$, then

$$\langle K_{f_{\mathbf{y}}}, \mathbf{x} \rangle = \langle w\mathbf{y} + \mathbf{v}, \mathbf{x} \rangle = \underbrace{w\langle \mathbf{y}, \mathbf{x} \rangle}_{0} + \underbrace{\langle \mathbf{v}, \mathbf{x} \rangle}_{K_x}.$$

Security. If $f_{\mathbf{y}}(\mathbf{x}) = 0$, i.e., $\langle \mathbf{y}, \mathbf{x} \rangle \neq 0$, then

$$(f_{\mathbf{y}}, \mathbf{x}, K_{f_{\mathbf{y}}}, K_{\mathbf{x}}) = (f_{\mathbf{y}}, \mathbf{x}, w\mathbf{y} + \mathbf{v}, \langle \mathbf{v}, \mathbf{x} \rangle) \equiv (f_{\mathbf{y}}, \mathbf{x}, w\mathbf{y} + \mathbf{v}, u) = (f_{\mathbf{y}}, \mathbf{x}, K_{f_{\mathbf{y}}}, u)$$

where $K \leftarrow \mathsf{Setup}()$, $K_{f_{\mathbf{y}}} \leftarrow \mathsf{A}(K, f_{\mathbf{y}})$, $K_{\mathbf{x}} \leftarrow \mathsf{B}(K, \mathbf{x})$, $u \xleftarrow{\$} \mathbb{F}$.
To see that, fix some values $(\mathbf{y}, \mathbf{x}, u)$. Sample a random $\mathbf{v}$ under the constraint that $\langle \mathbf{v}, \mathbf{x} \rangle = u$, then sample w and output $(f_{\mathbf{y}}, \mathbf{x}, w\mathbf{y} + \mathbf{v}, u)$. ◀

7 Removing the One-One Restriction via Key-Homomorphic PRF

In this section we consider one-one cPRFs that satisfy an additional property and show that such one-one cPRFs can be boosted to standard cPRFs via key-homomorphic PRFs. We then show that all of our information-theoretic one-one cPRFs satisfy this property, thus receiving new cPRF constructions.

In more detail, we require an alternative algorithm for Alice, $\mathsf{dkA}(K, f)$, that produces a *double-key* $(K_f, \hat{K}_f)$. Such double-key should have the property that $\hat{K}_f$ looks uniformly random even given K_f, but on the other hand, given both of the key parts $(K_f, \hat{K}_f)$ it should be possible to compute K_x for all x (regardless of $f(x)$). We now formally define this additional property.

▶ **Definition 15.** *A one-one constrained pseudorandom function for a predicate family $\mathcal{F}$ and an input space $\mathcal{X}$ is in the* double-key *model, if in addition to the algorithms* $(\mathsf{Setup}, \mathsf{B}, \mathsf{Rec})$ *as in Definition 6, there exists algorithms* dkA *and* dkRec *with the following syntax.*

- $\mathsf{dkA}(K, f) \to (K_f, \hat{K}_f)$ *is a PPT algorithm that takes a common random string K and a predicate $f \in \mathcal{F}$. It outputs a pair of keys $(K_f, \hat{K}_f)$.*
- $\mathsf{dkRec}(x, f, K_f, \hat{K}_f) \to K'_x$ *is a deterministic algorithm that takes an input x, a predicate $f \in \mathcal{F}$, and a pair of keys $(K_f, \hat{K}_f)$. It outputs a value K'_x.*

(Standard) Correctness. *A one-one constrained PRF is* correct *if it satisfies standard correctness (as in Definition 6) with respect to the keys K_f. That is, for all $f \in \mathcal{F}$ and $x \in \mathcal{X}$ for which $f(x) = 1$, for all $K \leftarrow \mathsf{Setup}(1^\lambda)$, and for all $(K_f, \hat{K}_f) \leftarrow \mathsf{dkA}(K, f)$, it holds that $\mathsf{Rec}(x, f, K_f) = \mathsf{B}(K, x)$.*

Correctness of Double-Keys. *A one-one constrained PRF has* correct double-key *if for all $f \in \mathcal{F}$ and $x \in \mathcal{X}$ for which $f(x) = 0$, for all $K \leftarrow \mathsf{Setup}(1^\lambda)$, and for all $(K_f, \hat{K}_f) \leftarrow \mathsf{dkA}(K, f)$, it holds that $\mathsf{dkRec}(x, f, K_f, \hat{K}_f) = \mathsf{B}(K, x)$.*

Security of Double-Keys. *The scheme has perfect (/statistical /computational) double-key security if for any $f \in \mathcal{F}$ and any unbounded (/unbounded /PPT) distinguisher, the following distributions are identical (/statistically close /computationally indistinguishable).*

$$(f, K_f, \hat{K}_f) \equiv (f, K_f, u)$$

where $K \leftarrow \mathsf{Setup}(1^\lambda)$, $(K_f, \hat{K}_f) \leftarrow \mathsf{dkA}(K, f)$, $u \leftarrow \mathcal{U}_{|\hat{K}_f|}$.

Given the new definition in hand, we now state the main theorem.

▶ **Theorem 16.** *Let* $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{HomKeyEval})$ *be a key-homomorphic PRF for an input space* $\mathcal{X}_\alpha$*, a key space* $\mathcal{K}$*, and a function family* $\mathcal{G} = \{\mathcal{G}_n : \mathcal{K}^n \to \mathcal{K}\}_{n\in\mathbb{N}}$*, and let* $(\mathsf{Setup}, \mathsf{dkA}, \mathsf{B}, \mathsf{Rec}, \mathsf{dkRec})$ *be a one-one cPRF in the double-key model (as in Definition 15) for an input space* $\mathcal{X}_\beta$ *and a predicate family* $\mathcal{F}$*, such that:*

- *The key space* $\mathcal{K}$ *is some field* $\mathbb{F}$ *and the algorithm* $\mathsf{KeyGen}(1^\lambda)$ *outputs a random value* $\mathsf{sk} \xleftarrow{\$} \mathbb{F}$*.*
- *For all* $x \in \mathcal{X}_\beta$ *and all* $K \leftarrow \mathsf{Setup}(1^\lambda)$ *it holds that* $K_x \in \mathbb{F}$ *where* $K_x \leftarrow \mathsf{B}(K, x)$ *and* $\mathbb{F}$ *is the same field as above.*
- *For all* $f \in \mathcal{F}$ *there exists* $n \in \mathbb{N}$ *such that for all* $K \leftarrow \mathsf{Setup}(1^\lambda)$ *it holds that* $\hat{K}_f \in \mathbb{F}^n$*, where* $(K_f, \hat{K}_f) \leftarrow \mathsf{dkA}(K, f)$ *and* $\mathbb{F}$ *is the same field as above.*
- *For all* $x \in \mathcal{X}_\beta$ *and* $f \in \mathcal{F}$ *such that* $f(x) = 0$*, and all* $(K_f, \hat{K}_f) \leftarrow \mathsf{dkA}(K, f)$ *where* $K \leftarrow \mathsf{Setup}(1^\lambda)$*, let* $\mathsf{dkRec}_{x,f,K_f} : \mathbb{F}^n \to \mathbb{F}$ *denote the algorithm* dkRec *with the hard-coded inputs* (x, f, K_f)*, i.e.,* $\mathsf{dkRec}_{x,f,K_f}(\cdot) = \mathsf{dkRec}(x, f, K_f, \cdot)$*. Then:*
  - $\mathsf{dkRec}_{x,f,K_f}(\cdot)$ *can be homomorphically evaluated over keys of the key-homomorphic scheme, i.e.,* $\mathsf{dkRec}_{x,f,K_f}(\cdot) \in \mathcal{G}$*.*
  - $\mathsf{dkRec}_{x,f,K_f}(\cdot)$ *preserves uniformity, i.e., the distributions* $\mathsf{dkRec}_{x,f,K_f}(\mathcal{U}_{\mathbb{F}^n})$ *and* $\mathcal{U}_{\mathbb{F}}$ *are identical.*

Then, there exists a key-selective secure single-key constrained PRF for the predicate family $\mathcal{F}$ *and the input space* $\mathcal{X} = \mathcal{X}_\alpha \cap \mathcal{X}_\beta$*.*

7.1 The Reduction

We proceed to the proof of Theorem 16. Assuming a key-homomorphic PRF and a one-one constrained PRF as described in the theorem, the constrained PRF is defined as follows.

▶ **Construction 17.** *Define:*

- $\mathsf{CPRF.KeyGen}(1^\lambda)$*: Compute* $K \leftarrow \mathsf{Setup}(1^\lambda)$ *and output* $\mathsf{msk} = K$*.*
- $\mathsf{CPRF.Eval}(\mathsf{msk}, x)$*: Parse* $\mathsf{msk} = K$*. Compute* $K_x := \mathsf{B}(K, x)$ *and output* $r_x := \mathsf{Eval}_{K_x}(x)$*.*
- $\mathsf{CPRF.Con}(\mathsf{msk}, f)$*: Parse* $\mathsf{msk} = K$*. Compute* $(K_f, \hat{K}_f) \leftarrow \mathsf{dkA}(K, f)$ *and output* $\mathsf{sk}_f = (f, K_f)$*.*
- $\mathsf{CPRF.ConEval}(\mathsf{sk}_f, x)$*: Parse* $\mathsf{sk}_f = (f, K_f)$*. Compute* $K'_x := \mathsf{Rec}(x, f, K_f)$ *and output* $r'_x := \mathsf{Eval}_{K'_x}(x)$*.*

Correctness of Constrained Keys. Fix $x \in \mathcal{X}$ and $f \in \mathcal{F}$ for which $f(x) = 1$, and $\mathsf{msk} \leftarrow \mathsf{CPRF.KeyGen}(1^\lambda)$, $\mathsf{sk}_f \leftarrow \mathsf{CPRF.Con}(\mathsf{msk}, f)$. Then $\mathsf{msk} = K$ and $\mathsf{sk}_f = (f, K_f)$ where $K \leftarrow \mathsf{Setup}(1^\lambda)$ and $(K_f, \hat{K}_f) \leftarrow \mathsf{dkA}(K, f)$. Consider $\mathsf{CPRF.ConEval}(\mathsf{sk}_f, x)$ and in particular $K'_x = \mathsf{Rec}(x, f, K_f)$, then by the standard correctness of the one-one cPRF, since $f(x) = 1$ it holds that $K'_x = K_x$ and therefore $\mathsf{Eval}_{K'_x}(x) = \mathsf{Eval}_{K_x}(x)$.

Pseudorandomness. We now prove that Construction 17 is a single-key key-selective secure constrained PRF as in Definition 4. The proof goes via a sequence of hybrids.

Proof. Hybrid H_0. This is the real security game as in Definition 4. Note that this is the selective-key game, i.e., the key query for f happens before any other queries. Explicitly, $\mathcal{C}$ computes $K \leftarrow \mathsf{Setup}(1^\lambda)$ and then immediately answers the key query:

- *Key Query:* Upon receiving $f \in \mathcal{F}$, $\mathcal{C}$ computes $(K_f, \hat{K}_f) \leftarrow \mathsf{dkA}(K, f)$ and sends K_f to $\mathcal{A}$.

In the rest of the game $\mathcal{C}$ answers queries made by $\mathcal{A}$ as follows.

- *Evaluation Query:* Upon receiving $x \in \mathcal{X}$, $\mathcal{C}$ computes $K_x := \mathsf{B}(K, x)$ and $r_x := \mathsf{Eval}_{K_x}(x)$.
  It sends r_x to $\mathcal{A}$.
- *Challenge Query:* Upon receiving $x^* \in \mathcal{X}$, $\mathcal{C}$ computes $K_{x^*} := \mathsf{B}(K, x^*)$ and $r_{x^*} := \mathsf{Eval}_{K_{x^*}}(x^*)$.
  It samples a bit $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, it sends r_{x^*} to $\mathcal{A}$, otherwise it sends to $\mathcal{A}$ a random value $u \leftarrow \mathcal{U}_{|r_{x^*}|}$.

Hybrid H_1. In this hybrid we change the way that $\mathcal{C}$ evaluates K_x when it answers evaluation queries and the challenge query. Instead of using the common random string K, it will use the double-key $(K_f, \hat{K}_f)$. That is, upon receiving $x \in \mathcal{X}$, instead of computing $K_x := \mathsf{B}(K, x)$ it computes

$$K_x := \begin{cases} \mathsf{Rec}(x, f, K_f) & f(x) = 1 \\ \mathsf{dkRec}(x, f, K_f, \hat{K}_f) & f(x) = 0 \end{cases}$$

and then proceeds as in the previous hybrid. Due to the prefect standard correctness and correctness of double-keys, the distributions that $\mathcal{C}$ outputs in Hybrids H_0 and H_1 are identical.

Hybrid H_2. Note that $\mathcal{C}$ does not use K anymore except of when it first generates $(K_f, \hat{K}_f)$. In this hybrid we change $\hat{K}_f$ to uniform. That is, after computing $(K_f, \hat{K}_f) \leftarrow \mathsf{dkA}(K, f)$, $\mathcal{C}$ samples a random value $u \leftarrow \mathcal{U}_{|\hat{K}_f|}$ and overrides the value of $\hat{K}_f$ such that $\hat{K}_f := u$. By the perfect (/statistical /computational) security of double-keys, the distributions that $\mathcal{C}$ outputs in hybrids H_1 and H_2 are identical (/statistically close /computationally indistinguishable).

Hybrid H_3. In this hybrid we change the way that $\mathcal{C}$ evaluates r_x when it answers evaluation queries and the challenge query, whenever $f(x) = 0$. Recall that in the previous hybrid it computes $r_x := \mathsf{Eval}_{K_x}(x)$ where $K_x := \mathsf{dkRec}(x, f, K_f, \hat{K}_f)$, and recall that by assumption, $\mathsf{dkRec}_{x,f,K_f}(\cdot) = \mathsf{dkRec}(x, f, K_f, \cdot)$ can be homomorphically evaluated over the PRF key-space using $\mathsf{HomKeyEval}$. Moreover, by our assumption it holds that $\hat{K}_f \in \mathbb{F}^n$ for some $n \in \mathbb{N}$. Therefore,

$$\begin{aligned} r_x &= \mathsf{Eval}_{K_x}(x) \\ &= \mathsf{Eval}_{\mathsf{dkRec}(x,f,K_f,\hat{K}_f)}(x) \\ &= \mathsf{Eval}_{\mathsf{dkRec}_{x,f,K_f}(\hat{K}_f)}(x) \\ &= \mathsf{HomKeyEval}\left(\mathsf{dkRec}_{x,f,K_f},\ \mathsf{Eval}_{\hat{K}_f^1}(x), \ldots, \mathsf{Eval}_{\hat{K}_f^n}(x)\right) \end{aligned}$$

where $\hat{K}_f = (\hat{K}_f^1, \ldots, \hat{K}_f^n)$. In this hybrid, $\mathcal{C}$ first computes $s_x^i := \mathsf{Eval}_{\hat{K}_f^i}(x)$ for all $i \in [n]$, and then

$$r_x := \mathsf{HomKeyEval}\left(\mathsf{dkRec}_{x,f,K_f},\ s_x^1, \ldots, s_x^n\right).$$

By the perfect correctness of homomorphic key evaluation, the distributions that $\mathcal{C}$ outputs in hybrids H_2 and H_3 are identical.

Hybrid H_4. In this hybrid we change the way that $\mathcal{C}$ evaluates $s_x^1, \ldots, s_x^n$ whenever $f(x) = 0$. Note that in Hybrid H_3 the value $\hat{K}_f$, which is uniformly random in $\mathbb{F}^n$, is only used when computing $s_x^i := \mathsf{Eval}_{\hat{K}_f^i}(x)$ for all $i \in [n]$. Thus, we can replace the values $s_x^1, \ldots, s_x^n$ with uniformly random strings. That is, whenever a query is made for an x such that $f(x) = 0$,

$\mathcal{C}$ samples $s_x^1, \ldots, s_x^n \xleftarrow{\$} \{0,1\}^*$ of appropriate size and then proceeds as in the previous hybrid. By the pseudorandomness of the key-homomorphic PRF respective to the keys $\hat{K}_f^1, \ldots, \hat{K}_f^n$, the distributions that $\mathcal{C}$ outputs in hybrids H_3 and H_4 are computationally indistinguishable.

Hybrid H_5. We change again the way that $\mathcal{C}$ evaluates r_x whenever $f(x) = 0$. In this hybrid the values r_x are replaced with uniformly sampled strings. Indistinguishability will follow immediately from the next lemma, since $s_x^1, \ldots, s_x^n$ are random values and $\mathsf{dkRec}_{x,f,K_f}(\cdot)$ preserves uniformity. ◀

▶ **Lemma 18.** *Let* $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{HomKeyEval})$ *be a secure key-homomorphic PRF for a key space* $\mathcal{K}$ *and a function family* $\mathcal{G} = \{\mathcal{G}_n : \mathcal{K}^n \to \mathcal{K}\}_{n\in\mathbb{N}}$*, where valid keys are samples from* $\mathcal{U}_\mathcal{K}$ *and the output space is* $\mathcal{O}$*. Let* $g \in \mathcal{G}_n$ *be a function such that* $g(\mathcal{U}_{\mathcal{K}^n})$ *and* $\mathcal{U}_\mathcal{K}$ *are computationally indistinguishable. Then, the distribution* $\mathsf{HomKeyEval}(g, \mathcal{U}_{\mathcal{O}^n})$ *is computationally indistinguishable from* $\mathcal{U}_\mathcal{O}$*.*

Proof. Via hybrids:

1. The distribution $\mathsf{HomKeyEval}(g, \mathcal{U}_{\mathcal{O}^n})$.
2. The distribution $\mathsf{HomKeyEval}\left(g, \mathsf{Eval}_{\mathcal{U}_\mathcal{K}}(0), \ldots, \mathsf{Eval}_{\mathcal{U}_\mathcal{K}}(0)\right)$ (Ind. by the pseudorandomness). Note that the distribution $\mathsf{Eval}_{\mathcal{U}_\mathcal{K}}(0)$ is concatenated n times and that the PRF input 0 was chosen arbitrarily.
3. The distribution $\mathsf{Eval}_{g(\mathcal{U}_{\mathcal{K}^n})}(0)$ (Ind. by the correctness of homomorphic key evaluation).
4. The distribution $\mathsf{Eval}_{\mathcal{U}_\mathcal{K}}(0)$ (Ind. by the assumption about g).
5. The distribution $\mathcal{U}_\mathcal{O}$ (Ind. by the pseudorandomness). ◀

7.2 Constructions of One-One cPRFs in the Double-Key Model

We now show that all of our information-theoretic constructions (see Section 6) have a *double-key* variant (see Definition 15) of the same complexity. Moreover, in all of those double-key constructions, the corresponding function dkRec_{x,f,K_f} (see Theorem 16 for definition) is linear and preserves uniformity. Due to the similarity to the constructions in Section 6, we provide here an overview.

Generic Predicates. For an input space $\mathbb{F}$ and any predicate family $\mathcal{F}$, the common random string $K = \{k_y\}_{y\in\mathbb{F}}$ consists of random bits $k_y \xleftarrow{\$} \{0,1\}$ for every element in the field. The value K_x is then simply k_x. The double-key for a predicate f splits K into a set of authorized inputs $K_f = \{k_y : f(y) = 1\}_{y\in\mathbb{F}}$ and a set of unauthorized inputs $\hat{K}_f = \{k_y : f(y) = 0\}_{y\in\mathbb{F}}$. Clearly, $\hat{K}_f$ is uniformly distributed even given K_f, and recovering a value k_x from $\hat{K}_f$ is a linear function.

AND Composition. Consider two perfect one-one cPRFs in the double-key model for some predicate families $\mathcal{F}^1 = \{f^1\}$ and $\mathcal{F}^2 = \{f^2\}$. In the construction for the predicate family $\mathcal{F}^1 \wedge \mathcal{F}^2 = \{f^1 \wedge f^2\}_{f^1\in\mathcal{F}^1, f^2\in\mathcal{F}^2}$, there is one instance of each of the underlying constructions. The common random string $K = (K^1, K^2)$ and double-keys $K_f = (K_{f^1}^1, K_{f^2}^2)$ and $\hat{K}_f = (\hat{K}_{f^1}^1, \hat{K}_{f^2}^2)$ where $f = f^1 \wedge f^2$ are a concatenation of the corresponding values in the underlying schemes. The values $K_x = K_x^1 + K_x^2$ are the *sum* of the corresponding values in the underlying schemes. Since the two instances are secure and generated independently, uniformity of $\hat{K}_f$ given K_f follows immediately. Moreover, the algorithm $\mathsf{dkRec}(x, f, K_f, \hat{K}_f)$ that computes and outputs $\mathsf{dkRec}^1(x, f^1, K_{f^1}^1, \hat{K}_{f^1}^1) + \mathsf{dkRec}^2(x, f^2, K_{f^2}^2, \hat{K}_{f^2}^2)$ is clearly correct, linear in $\hat{K}_f$, and preserves uniformity if the underlying schemes are correct, linear in $\hat{K}_{f^1}^1$ and $\hat{K}_{f^2}^2$, and preserve uniformity.

Equality Testing. Consider equality testing over some field $\mathbb{F}$. The common random string $K = (k_0, k_1)$ consists of two random elements in the field $k_0, k_1 \xleftarrow{\$} \mathbb{F}$, and we define $K_x = k_0 + xk_1$. The double-key for some value y is defined as $K_{f_y} = k_0 + yk_1$ and $\hat{K}_{f_y} = yk_1$. Note that by the uniformity of k_0 and k_1, for all $y \neq 0$ the key part $\hat{K}_{f_y}$ is distributed uniformly in $\mathbb{F}$ even given K_{f_y}. To reconstruct K_x for some $x \neq y$, compute

$$\mathsf{dkRec}(x, y, K_{f_y}, \hat{K}_{f_y}) = \frac{x-y}{y}\hat{K}_{f_y} + K_{f_y} = (x-y)k_1 + k_0 + yk_1 = k_0 + xk_1 = K_x \ ,$$

which is correct and linear in $\hat{K}_{f_y}$.

Subset Predicates. Recall that we consider all subsets $X \subseteq [N]$ as the input space and all subsets $Y \subseteq [N]$ as the predicate family, such that $f_Y(X) = 1$ if and only if $X \subseteq Y$. The common random string $K = \{k_i\}_{i\in[N]}$ consists of random bits $k_i \xleftarrow{\$} \{0,1\}$ for every element in the set $[N]$. The value K_X is then set to $K_X = \bigoplus_{i\in X} k_i$. The double-key for a predicate f_Y splits K into a set of authorized elements $K_{f_Y} = \{k_i\}_{i\in Y}$ and a set of unauthorized elements $\hat{K}_{f_Y} = \{k_i\}_{i\in[N]/Y}$. Clearly, $\hat{K}_{f_Y}$ is uniformly distributed even given K_{f_Y}, and recovering a value k_i from $(K_{f_Y}, \hat{K}_{f_Y})$ is a linear function that preserves uniformity.

Inner-Product Predicates. Consider inner-product testing of vectors of length ℓ over some field $\mathbb{F}$. The common random string $K = (\mathbf{v}, w)$ consists of a random vector $\mathbf{v} \xleftarrow{\$} \mathbb{F}^\ell$ and a random field element $w \xleftarrow{\$} \mathbb{F}$. We define $K_{\mathbf{x}} = \langle \mathbf{v}, \mathbf{x} \rangle$. The double-key for a vector predicate $\mathbf{y}$ is defined as $K_{f_\mathbf{y}} = w\mathbf{y} + \mathbf{v}$ and $\hat{K}_{f_\mathbf{y}} = w$. Note that by the uniformity of $\mathbf{v}$ and w, the key part $\hat{K}_{f_\mathbf{y}}$ is distributed uniformly in $\mathbb{F}$ even given $K_{f_\mathbf{y}}$. To reconstruct $K_\mathbf{x}$ for some $\mathbf{x}$, compute

$$\mathsf{dkRec}(\mathbf{x}, f_\mathbf{y}, K_{f_\mathbf{y}}, \hat{K}_{f_\mathbf{y}}) = \langle K_{f_\mathbf{y}}, \mathbf{x} \rangle - \hat{K}_{f_\mathbf{y}} \langle \mathbf{y}, \mathbf{x} \rangle = \langle w\mathbf{y} + \mathbf{v}, \mathbf{x} \rangle - w\langle \mathbf{y}, \mathbf{x} \rangle = \langle \mathbf{v}, \mathbf{x} \rangle = K_\mathbf{x} \ ,$$

which is correct, linear in $\hat{K}_{f_\mathbf{y}}$, and preserves uniformity.

8 One-One Constrained PRF and CDS Protocols

In this section we study the connection between one-one constrained PRFs and conditional disclosure of secrets (CDS) protocols, a cryptographic primitive used to construct many secure protocols (see discussion in the introduction). In CDS protocols, Alice and Bob hold a secret and a common random string, and each of them holds a private input for some two-input predicate. Then, each of the parties sends one message to a referee, which is based on its private input, the secret, and the common random string. The referee, knowing the inputs of the parties, should learn the secret if and only if the inputs of the parties satisfy the predicate. We next provide a formal definition of CDS protocols, originally presented in [20].

▶ **Definition 19** (Conditional Disclosure of Secrets Protocols). *Let $f : X \times Y \to \{0,1\}$ be a predicate. A* conditional disclosure of secrets (CDS) protocol *$\mathcal{P}$ for f with domain of secrets S, domain of common random strings R, and finite message domains M_A, M_B consists of two deterministic message computation functions $\mathsf{Enc}_A, \mathsf{Enc}_B$, where $\mathsf{Enc}_A : X \times S \times R \to M_A$ and $\mathsf{Enc}_B : Y \times S \times R \to M_B$, and a deterministic reconstruction function $\mathsf{Dec} : X \times Y \times M_A \times M_B \to S$. We say that $\mathcal{P}$ is a CDS protocol for f if it satisfies the following requirements.*

Correctness. *For every inputs $x \in X$ and $y \in Y$ for which $f(x,y) = 1$, every secret $s \in S$, and every common random string $r \in R$,*

$$\mathsf{Dec}(x, y, \mathsf{Enc}_A(x, s, r), \mathsf{Enc}_B(y, s, r)) = s \ .$$

Security. *For every inputs $x \in X$ and $y \in Y$ for which $f(x,y) = 0$ and every secret $s \in S$,*

$$(x, y, \mathsf{Enc}_A(x, s, r), \mathsf{Enc}_B(y, s, r), s) \quad \equiv \quad (x, y, \mathsf{Enc}_A(x, s, r), \mathsf{Enc}_B(y, s, r), u)$$

where $r \xleftarrow{\$} R$ and $u \xleftarrow{\$} S$.

Message Size. *The* message size *of a CDS protocol $\mathcal{P}$ is defined as the sizes of the messages sent by the parties, i.e.,* $\log |M_A| + \log |M_B|$.

We consider the index predicate, which gets as inputs an N-bit string (or a database) D and an index $i \in [N]$, and returns the ith bit in D, denoted by D_i.

▶ **Definition 20** (The Index Function)**.** *The index predicate is the predicate $f_{\text{index}} : \{0,1\}^N \times [N] \to \{0,1\}$, where $f_{\text{index}}(D, i) = D_i$.*

We next show a transformation that preservers complexity from one-one constrained PRFs to CDS protocols for the index predicate.

▶ **Theorem 21.** *Let $f_{\text{index}} : \{0,1\}^N \times [N] \to \{0,1\}$ be the index predicate, and assume that for every predicate $f : [N] \to \{0,1\}$ there is a one-one constrained PRF for f with complexity $c(N)$. Then, there is a CDS protocol for f_{index} with message size $c(N)$.*

Proof. We consider a one-one constrained PRF scheme, when Alice holds a predicate $f : [N] \to \{0,1\}$, Bob holds an input $x \in [N]$, and both hold a common random string $K \leftarrow \mathsf{Setup}(1^\lambda)$, where $K_f \leftarrow \mathsf{A}(K, f)$ and $K_x \leftarrow \mathsf{B}(K, x)$. By the correctness requirement of the one-one constrained PRF, if $f(x) = 1$ then there exist a deterministic function Rec such that $\mathsf{Rec}(x, f, K_f) = K_x$, and by security requirement, if $f(x) = 0$ then $(f, x, K_f, K_x) \equiv (f, x, K_f, u)$, where $u \leftarrow \mathcal{U}_{|K_x|}$.

We show a construction of a CDS protocol for the index predicate f_{index}, which is based on the above one-one constrained PRF. The construction is as follows.

- Inputs: Alice holds $D \in \{0,1\}^N$ and Bob holds $i \in [N]$. We represent D as the predicate $f_D : [N] \to \{0,1\}$, where $f_D(j) = f_{\text{index}}(D, j) = D_j$.
- The secret: A string s of size at most $|\mathsf{B}(K, i)|$.
- The common random string: An element $K \leftarrow \mathsf{Setup}(1^\lambda)$.
- $\mathsf{Enc}_A(D, K)$: Alice computes and sends the message $\mathsf{Enc}_A(D, K) = \mathsf{A}(K, f_D) = K_{f_D}$.
- $\mathsf{Enc}_B(i, K)$: Bob computes and sends the message $\mathsf{Enc}_B(i, K) = \mathsf{B}(K, i) \oplus s' = K_i \oplus s'$, where $s' = 0^t \circ s$ such that $|s'| = |\mathsf{B}(K, i)|$.
- $\mathsf{Dec}(D, i, \mathsf{Enc}_A(D, K), \mathsf{Enc}_B(i, K))$: If $f_{\text{index}}(D, i) = 1$, the referee computes

$$\mathsf{Rec}(i, f_D, \mathsf{Enc}_A(D, K)) \oplus \mathsf{Enc}_B(i, K).$$

Correctness. If $f_{\text{index}}(D, i) = 1$, i.e., $f_D(i) = 1$, then the referee computes

$$\mathsf{Rec}(i, f_D, \mathsf{Enc}_A(D, K)) \oplus \mathsf{Enc}_B(i, K) = \mathsf{Rec}(i, f_D, K_{f_D}) \oplus K_i \oplus s' = K_i \oplus K_i \oplus s' = s'$$

where the second equality follows from the correctness of the one-one constrained PRF. Hence, the referee learns s' and so the secret s.

Security. If $f_{\text{index}}(D, i) = 0$, i.e., $f_D(i) = 0$, then by the security of the one-one constrained PRF,

$$(f_D, i, \mathsf{A}(K, f_D), \mathsf{B}(K, i)) \equiv (f_D, i, \mathsf{A}(K, f_D,), u)$$

where $K \leftarrow \mathsf{Setup}(1^\lambda)$ and $u \leftarrow \mathcal{U}_{|\mathsf{B}(K,i)|}$. Then, since $s' = \mathsf{Enc}_B(i, K) \oplus \mathsf{B}(K, i)$ we get that

$$(D, i, \mathsf{Enc}_A(D, K), \mathsf{Enc}_B(i, K), s') \equiv (D, i, \mathsf{Enc}_A(D, K), \mathsf{Enc}_B(i, K), u)$$

where $K \leftarrow \mathsf{Setup}(1^\lambda)$ and $u \leftarrow \mathcal{U}_{|s'|}$.

Message size. The message size of this CDS protocol is equal to the complexity of the one-one constrained PRF, which is $c(N)$. ◀

Using the above result and the reduction that appears in [18], from CDS protocols for general predicates to CDS protocols for the index predicate, we get a transformation from one-one constrained PRF to CDS protocols for general predicates.

▶ **Corollary 22.** *Let $g : [N] \times [N] \to \{0,1\}$ be a predicate, and assume that for every predicate $f : [N] \to \{0,1\}$ there is a one-one constrained PRF for f with complexity $c(N)$. Then, there is a CDS protocol for g with message size $c(N)$.*

Note the best known CDS protocol for general predicates $g : [N] \times [N] \to \{0,1\}$ has message size $2^{O(\sqrt{\log N \log\log N})}$ [25]. Thus, by the above lower bound of $\Omega(N)$ on the complexity of one-one constrained PRFs of Theorem 7, we cannot get a similar transformation that preserve complexity in the other direction (i.e., a transformation from CDS protocols to one-one constrained PRFs).

9 Computational Constructions

Every single-key constrained PRF is in particular a one-one constrained PRF under the same security notion (which is either adaptive, key selective or challenge selective, see Definition 6). For completeness, we now go over some of the known computational constructions in the literature of single-key constrained PRFs.

9.1 Constructions from OWFs

Punctured Predicates. As was shown in [12, 13, 23], the OWF-based PRF of [21] is in fact puncturable. Using our previous terminology, it supports punctured predicates over the input space $\{0,1\}^n$ as defined in Theorem 7. The complexity of the construction is $O(\lambda \cdot n)$ where λ is the security parameter. Furthermore, [17] showed that the aforementioned construction satisfies *adaptive* security with a security loss exponential in the number of queries. Since we focus on the single-query scenario, we can use their proof strategy to claim adaptive security on the implied one-one cPRF.

▶ **Theorem 23.** *Let $\mathcal{F} = \{f_y\}_{y \in \{0,1\}^n}$ be the family of punctured predicates over the set of n-bit strings, i.e., for any fixed $y \in \{0,1\}^n$, $f_{\neq y} : \{0,1\}^n \to \{0,1\}$ where $f_{\neq y}(x) = 1$ if and only if $x \neq y$. Then, for every security parameter λ, there is a one-one constrained PRF for $\mathcal{F}$ with perfect correctness, computational adaptive security, and complexity $O(\lambda \cdot n)$.*

One-Dimensional Interval Predicates. [23] showed how to further generalized the [21] approach in order to support (one-dimensional) interval predicates without compromising the complexity. Such predicates allow to compute the PRF on all inputs x that are within some range $[a, b]$, i.e., all x such that $a \leq x \leq b$, where the key size is $O(\lambda \cdot \log|b - a|)$. Due to the similarity to the construction for punctured predicates, the adaptive security proof strategy of [17] can also be applied here.

▶ **Theorem 24.** *Let $\mathcal{F} = \{f_{[a,b]}\}_{a,b \in \{0,1\}^n}$ be the family of one-dimensional interval predicates over the set of n-bit strings, i.e., for any fixed $a, b \in \{0,1\}^n$, $f_{[a,b]} : \{0,1\}^n \to \{0,1\}$ where $f_{[a,b]}(x) = 1$ if and only if $a \leq x \leq b$. Then, for every security parameter λ, there is a one-one constrained PRF for $\mathcal{F}$ with perfect correctness, computational adaptive security, and complexity $O(\lambda \cdot n)$.*

Multi-Dimensional Interval Predicates. We now consider multi-dimensional interval predicates (that were previously studied in [11,19,26]). Such predicates are characterized by d intervals $\{[a_i, b_i]\}_{i\in[d]}$, and the input space is $(\{0,1\}^n)^d$. An input $X = (x_1, \ldots, x_d) \in (\{0,1\}^n)^d$ is authorized by the multi-dimensional predicate $f_{[a_i,b_i]_{i\in[d]}}$ if and only if $a_i \leq x_i \leq b_i$ for all $i \in [d]$. Each interval $[a_i, b_i]$ can be verified by checking the n bits of the input corresponding to x_i. In particular, in order to verify that the ith dimension of X (i.e., x_i) is within the ith range $[a_i, b_i]$, we have to check whether the ith block of n bits of X is within the range $[a_i, b_i]$. To do so, we will use the AND composition lemma (that is, Lemma 11) sequentially $d-1$ times on d independent instances of one-one cPRFs for one-dimensional interval predicates over inputs of length n (as in Theorem 24), where each instance i handles the ith block of X.

▶ **Theorem 25.** *Let $\mathcal{F} = \{f_{[a_i,b_i]_{i\in[d]}}\}_{a_i,b_i\in\{0,1\}^n, i\in[d]}$ be the family of d-dimensional interval predicates over the set of n-bit strings, i.e., for any fixed $a_i, b_i \in \{0,1\}^n$ for all $i \in [d]$, $f_{[a_i,b_i]_{i\in[d]}} : (\{0,1\}^n)^d \to \{0,1\}$ where $f_{[a_i,b_i]_{i\in[d]}}(x_1, \ldots, x_d) = 1$ if and only if $a_i \leq x_i \leq b_i$ for all $i \in [d]$. Then, for every security parameter λ, there is a one-one constrained PRF for $\mathcal{F}_n$ with perfect correctness, computational adaptive security, and complexity $O(d \cdot \lambda \cdot n)$.*

9.2 Additional Constructions

Lattice Assumptions. [10] construct a selectively-secure single-key cPRF for all circuits from LWE. [14] construct an adaptively-secure single-key cPRF for NC1 from LWE.

Group Assumptions. [6] construct a selectively-secure single-key bit-fixing cPRF from DDH.

References

1 William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135. Springer, 2001. doi:10.1007/3-540-44987-6_8.

2 Benny Applebaum, Amos Beimel, Oriol Farràs, Oded Nir, and Naty Peter. Secret-sharing schemes for general and uniform access structures. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 441–471. Springer, 2019. doi:10.1007/978-3-030-17659-4_15.

3 Benny Applebaum, Amos Beimel, Oded Nir, and Naty Peter. Better secret-sharing via robust conditional disclosure of secrets. *Electronic Colloquium on Computational Complexity (ECCC)*, 27:8, 2020. To be published in *STOC 2020*. URL: https://eccc.weizmann.ac.il/report/2020/008.

4 Benny Applebaum and Pavel Raykov. From private simultaneous messages to zero-information arthur-merlin protocols and back. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 65–82. Springer, 2016. doi:10.1007/978-3-662-49099-0_3.

5 Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 557–577. Springer, 2014. doi:10.1007/978-3-642-55220-5_31.

6 Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained PRFs for NC^1 in traditional groups. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 543–574. Springer, 2018. doi:10.1007/978-3-319-96881-0_19.

7 Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In Yehuda Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 317–342. Springer, 2014. doi:10.1007/978-3-642-54242-8_14.

8 Amos Beimel and Naty Peter. Optimal linear multiparty conditional disclosure of secrets protocols. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 332–362. Springer, 2018. doi:10.1007/978-3-030-03332-3_13.

9 Amos Beimel and Naty Peter. Secret-sharing from robust conditional disclosure of secrets. *IACR Cryptology ePrint Archive*, 2019:522, 2019. URL: https://eprint.iacr.org/2019/522.

10 Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 171–190. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.20.

11 Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, 2007. doi:10.1007/978-3-540-70936-7_29.

12 Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300. Springer, 2013. doi:10.1007/978-3-642-42045-0_15.

13 Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519. Springer, 2014. doi:10.1007/978-3-642-54631-0_29.

14 Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC^1 from LWE. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 446–476, 2017. doi:10.1007/978-3-319-56620-7_16.

15 Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, and Shota Yamada. Constrained PRFs for bit-fixing from OWFs with constant collusion resistance. *IACR Cryptology ePrint Archive*, 2018:982, 2018. URL: https://eprint.iacr.org/2018/982.

16 Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 554–563. ACM, 1994. doi:10.1145/195058.195408.

17 Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained PRFs. In Palash Sarkar and Tetsu Iwata, editors, *Advances in*

Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II, volume 8874 of *Lecture Notes in Computer Science*, pages 82–101. Springer, 2014. doi:10.1007/978-3-662-45608-8_5.

18 Romain Gay, Iordanis Kerenidis, and Hoeteck Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 485–502. Springer, 2015. doi:10.1007/978-3-662-48000-7_24.

19 Romain Gay, Pierrick Méaux, and Hoeteck Wee. Predicate encryption for multi-dimensional range queries from lattices. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 752–776. Springer, 2015. doi:10.1007/978-3-662-46447-2_34.

20 Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *J. Comput. Syst. Sci.*, 60(3):592–629, 2000. doi:10.1006/jcss.1999.1689.

21 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 464–479. IEEE Computer Society, 1984. doi:10.1109/SFCS.1984.715949.

22 Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *Fifth Israel Symposium on Theory of Computing and Systems, ISTCS 1997, Ramat-Gan, Israel, June 17-19, 1997, Proceedings*, pages 174–184. IEEE Computer Society, 1997. doi:10.1109/ISTCS.1997.595170.

23 Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 669–684. ACM, 2013. doi:10.1145/2508859.2516668.

24 Tianren Liu and Vinod Vaikuntanathan. Breaking the circuit-size barrier in secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 699–708. ACM, 2018. doi:10.1145/3188745.3188936.

25 Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Conditional disclosure of secrets via non-linear reconstruction. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 758–790. Springer, 2017. doi:10.1007/978-3-319-63688-7_25.

26 Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 350–364. IEEE Computer Society, 2007. doi:10.1109/SP.2007.29.

27 Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 62–85. Springer, 2019. doi:10.1007/978-3-030-26948-7_3.

28 Hoeteck Wee. Dual system encryption via predicate encodings. In Yehuda Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 616–637. Springer, 2014. doi:10.1007/978-3-642-54242-8_26.

The Power of Synergy in Differential Privacy: Combining a Small Curator with Local Randomizers

Amos Beimel
Dept. of Computer Science, Ben-Gurion University, Beer-Sheva, Israel
https://www.cs.bgu.ac.il/~beimel
amos.beimel@gmail.com

Aleksandra Korolova
Dept. of Computer Science, University of Southern California, Los Angeles, CA, USA
https://www.korolova.com
korolova@usc.edu

Kobbi Nissim
Dept. of Computer Science, Georgetown University, Washington, DC, USA
http://people.cs.georgetown.edu/~kobbi/
kobbi.nissim@georgetown.edu

Or Sheffet
Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel
http://www.eng.biu.ac.il/sheffeo1/
or.sheffet@biu.ac.il

Uri Stemmer
Dept. of Computer Science, Ben-Gurion University, Beer-Sheva, Israel
Google Research
https://www.uri.co.il
u@uri.co.il

Abstract

Motivated by the desire to bridge the utility gap between local and trusted curator models of differential privacy for practical applications, we initiate the theoretical study of a hybrid model introduced by "Blender" [Avent et al., USENIX Security '17], in which differentially private protocols of n agents that work in the local-model are assisted by a differentially private curator that has access to the data of m additional users. We focus on the regime where $m \ll n$ and study the new capabilities of this (m, n)-*hybrid* model. We show that, despite the fact that the hybrid model adds no significant new capabilities for the basic task of simple hypothesis-testing, there are many other tasks (under a wide range of parameters) that can be solved in the hybrid model yet cannot be solved either by the curator or by the local-users separately. Moreover, we exhibit additional tasks where at least one round of *interaction* between the curator and the local-users is necessary – namely, no hybrid model protocol without such interaction can solve these tasks. Taken together, our results show that the combination of the local model with a small curator can become part of a promising toolkit for designing and implementing differential privacy.

2012 ACM Subject Classification Security and privacy → Privacy-preserving protocols

Keywords and phrases differential privacy, hybrid model, private learning, local model

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.14

Related Version A full version of the paper is available at https://arxiv.org/abs/1912.08951.

Funding The work was partially done while the authors were at the "Data Privacy: Foundations and Applications" program held in spring 2019 at the Simons Institute for the Theory of Computing, UC Berkeley.

Work of A. B. and K. N. was supported by NSF grant No. 1565387 TWC: Large: Collaborative: Computing Over Distributed Sensitive Data. This work was done when A. B. was hosted by Georgetown University. Work of A. B. was also supported by ISF grant no. 152/17, a grant from the Cyber Security Research Center at Ben-Gurion University, and ERC grant 742754 (project NTSC).
Work of A. K. was supported by NSF grant No. 1755992 CRII: SaTC: Democratizing Differential Privacy via Algorithms for Hybrid Models, a VMWare fellowship, and a gift from Google.
Work of O. S. was supported by grant #2017–06701 of the Natural Sciences and Engineering Research Council of Canada (NSERC). The bulk of this work was done when O. S. was affiliated with the University of Alberta, Canada.
Work of U. S. was supported in part by the Israel Science Foundation (grant No. 1871/19).

Acknowledgements We thank Adam Smith for suggesting the select-then-estimate task discussed in the introduction.

1 Introduction

Data has become one of the main drivers of innovation in applications as varied as technology, medicine [33], and city planning [34, 52]. However, the collection and storage of personal data in the service of innovation by companies, researchers, and governments poses significant risks for privacy and personal freedom. Personal data collected by companies can be breached [22]; subpoenaed by law enforcement in broad requests [44]; mis-used by companies' employees [37, 21]; or used for purposes different from those announced at collection time [63]. These concerns alongside data-hungry company and government practices have propelled privacy to the frontier of individuals' concerns, societal and policy debates, and academic research.

The *local model of differential privacy* [65, 43] has recently emerged as one of the promising approaches for achieving the goals of enabling data-driven innovation while preserving a rigorous notion of privacy for individuals that also addresses the above challenges. The *differential privacy* aspect provides each participating individual (almost) with the same protection she would have, had her information not been included [26], a guarantee that holds even with respect to all powerful adversaries with access to multiple analyses and rich auxiliary information. The *local* aspect of the model means that this guarantee will continue to hold even if the curator's data store is fully breached. From the utility perspective, the deployment of local differential privacy protocols by Google, Apple, and Microsoft demonstrate that the local differential privacy model is a viable approach in certain applications, without requiring trust in the companies or incurring risks from hackers and intelligence agencies [28, 54, 35, 1, 23].

The adoption of the local model by major industry players has motivated a line of research in the theory of local differential privacy (e.g., [9, 8, 17, 58, 40, 42, 41]). Alongside algorithmic improvements, this body of work highlighted the wide theoretical and practical gap between utility achievable in the more traditional trusted curator model of differential privacy (where the curator ensures the privacy of its output but can perform computations on raw individual data) and that achievable in the local differential privacy model. In particular, the number of data points necessary to achieve a particular level of accuracy in the local model is significantly larger than what is sufficient for the same accuracy in the curator model (see, e.g., [11, 43, 55, 9]). This has negative consequences. First, data analysis with local differential privacy becomes the privilege of the data-rich, handicapping smaller companies and helping to cement monopolies. Second, in an effort to maintain accuracy the entities deploying local differential privacy are tempted to use large privacy loss parameters [59], ultimately putting into question the privacy guarantee [36].

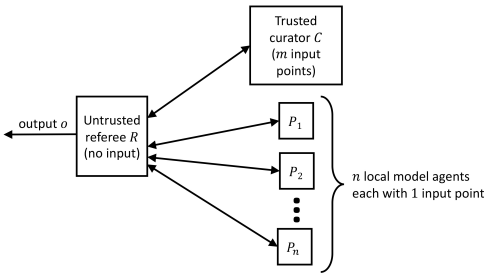


Figure 1 The hybrid model.

New models for differentially private computation have recently emerged to alleviate the (inevitable) low accuracy of the local model, of which we will discuss the *shuffle model* [38, 15, 20, 6, 5, 32] and the *hybrid model* [2].[1] In the shuffle model, it is assumed that the curator receives data in a manner disassociated from a user identifier (e.g., after the raw data has been stripped of identifiers and randomly shuffled). Recent work has proved that protocols employing shuffling can provide better accuracy than local protocols and sometimes match the accuracy of the curator model [20, 6, 5, 32].[2] Although the shuffle model is a promising approach for bridging the gap between the local model and the trusted curator model, it suffers from two weaknesses: it requires individuals' trust in the shuffler (which itself may be subject to breaches, subpoenas, etc., and the infrastructure for which may not exist), and, as highlighted by [6], its privacy guarantees to an individual rely on the assumption that sufficiently many other individuals do not deviate from the protocol.

The focus of this work is on a generalization of the hybrid model introduced by [2], where a majority of individuals that participate in a local differential privacy computation is augmented with a small number of individuals who contribute their information via a trusted curator. From a practical point of view, this separation is aligned with current industry practices, and the small number of individuals willing to contribute via a curator can be employees, technology enthusiasts, or individuals recruited as alpha- or beta-testers of products in exchange for early access to its features or decreased cost [47, 48, 50].

Furthermore, unlike Blender [2], in an effort to explore a rich trust and communication model, and anticipate development of future technologies and practices, we do not assume that the curator trusted by the opt-in users and the eventual data recipient (whom we call the referee) are the same entity (see Figure 1). The detailed discussion of the benefits of this modeling assumption appears after the formal model definition in Section 2.2.

1.1 This work: The power of the hybrid model

We initiate a theoretical study of the extent to which the hybrid model improves on the utility of the local model by addition of a small curator (and vice versa, improves on the utility of a small curator by addition of parties that participate in a local model computation). We ask whether there are tasks that cannot be computed privately by a small curator, or in the local model (even with many participants), but are feasible in a model that combines both. We answer this question in the affirmative.

[1] Approaches which weaken differential privacy itself or justify the use of large privacy loss parameters are outside our scope and deserve a separate discussion.

[2] Furthermore, the shuffle model provides new "privacy amplification" tools that can be used in the design of differentially private algorithms [27, 6].

A concatenation of problems (direct-sum)

Our first example is composed of two independent learning tasks, each task can be privately learned in one model, however, cannot be privately learned in the other model. Each data point is parsed as two points (x, y) that are labeled $(\mathsf{Par}_k(x), \mathsf{Thr}_t(y))$ where the former is a parity function $\mathsf{Par}_k(x) = \langle k, x \rangle$ and the latter is a threshold function $\mathsf{Thr}_t(y) = \mathbb{1}_{\{y \geq t\}}$ (see Section 2.4 for complete definitions). For a choice of parameters, known sample complexity bounds imply that the parity learning part of the task can be performed by the curator but cannot be privately computed in the local model with sub-exponential number of messages [43], and, conversely, that the threshold learning part cannot be performed by the curator [12, 30] but can be performed by the local model parties. It follows that for our choice of parameters the combined task cannot be performed neither by the curator nor by the local model with sub-exponential number of rounds (as the number of local agents is small, each agent in the local model must send many messages), but is feasible in the hybrid model without interaction (see the full version of this paper for a detailed analysis [10]).

Select-then-estimate

In our second example, each input point x is sampled i.i.d. from an unknown distribution over $\{-1, 1\}^d$. Letting $\mu = E[x]$, the goal is to output a pair $(i, \hat{\mu}_i)$ where i approximately maximizes μ_i (i.e., $\mu_i \geq \max_j \mu_j - \alpha$) and $\hat{\mu}_i$ approximates μ_i (i.e., $|\hat{\mu}_i - \mu_i| \leq \alpha'$), where $\alpha' < \alpha$. That is, once we found a good coordinate i, we want to approximate its quality with smaller error. A number of sample complexity bounds apply here (in particular, [60]; see also the full version of this paper [10]), yielding a wide range of parameters where the task cannot be performed by the curator alone, nor by the local model parties alone. The hybrid model, however, allows for a solution where the curator first identifies i such that $\mu_i \geq \max_j \mu_j - \alpha$ and the estimation of μ_i within accuracy α' is then performed by the local model parties (see the full version of this paper for a detailed analysis [10]).

The select-then-estimate problem is a sub-component of many statistical and data-mining applications [4, 57, 2]. It is also of core interest in genome-wide association studies (GWAS), where the inputs are genomes of patients affected by a particular disease and the goal is to (approximately) identify disease-related genome locations and estimate their significance [3, 66, 16]. Solving the problem while ensuring privacy is particularly challenging when the feature dimension is large compared to the number of available samples, which is the case for GWAS [14, 3, 39, 56]. As genomic data is particularly sensitive, the hybrid model of differential privacy appears appropriate for it from the privacy perspective – the majority of the data would be analyzed with the guarantee of local differential privacy, and only a small number of data points would be entrusted to a curator, whose analysis's output should also be differentially private [64]. As our example suggests, the hybrid model may be useful also from the utility perspective.

We next present and study tasks that require protocols with different interaction patterns involving both the curator and the local agents; that is, the referee needs to relay messages from the curator to the local model agents in one problem or vice versa in a second problem.

Learning parity XOR threshold

This task, which is a twist on the above concatenation of problems, combines two independent learning tasks. Rather than merely concatenating, in the learning parity XOR threshold problem points are labeled by $\mathsf{Par}_k(x) \oplus \mathsf{Thr}_t(y)$. A simple argument shows that (for specifically

chosen parameters) the task cannot be performed by the curator alone or by the local model agents with sub-exponential number of rounds. The task is, however, feasible in the hybrid model without interaction. Observe that the local model agents can complete the task once the parity part is done, and that $\mathsf{Thr}_t(y) = 0$ for the lower half of the points y or $\mathsf{Thr}_t(y) = 1$ for the upper half of the points y (or both). These observations suggest a protocol where the curator first performs two parity learning tasks (splitting points according to y values), and the task is then completed by the local model agents. This requires communication (as the referee needs to relay a message from the curator to the local model agents), and it may seem that this interaction is necessary for the task. However, in Section 3 we show that this is not the case, by demonstrating a non-interactive protocol where all parties send a message to the referee at the same round.

1-out-of-2^d-parity

Our next task can be solved in the hybrid model (but neither by a small curator model nor in the local model with sub-exponential number of rounds). This task requires interaction, first with the local model agents and then with the curator. The task consists of a multitude of parity problems, only one of which – determined by the input – needs to be solved. The curator is capable of solving the parity task privately, however, the curator's limited number of samples does not allow solving all parity problems privately, nor does it allow distinguishing which problem to solve. The local model agents cannot solve parity (with sub-exponential number of rounds) [43] but can recover which problem needs to be solved (via a heavy hitters computation [17]). Thus, the referee needs to first interact with the local agents and then the curator. See Section 4.

Parity-chooses-secret

The third task in this part can be solved in the hybrid model (but neither with a small curator, nor in the local model with sub-exponential number of rounds). The task requires interaction in the reverse order from the previous task: first with the curator, then with the local model agents. The input to this task contains shares of a large collection of secrets and the goal is to recover one of these secrets. The secret that should be recovered is determined by the input as the solution to a parity problem. The curator can solve the parity problem but does not have enough information to recover any of the secrets. The local model agents receive enough information to recover all secrets, but doing so would breach privacy (as implied by [46]). They cannot solve the parity problem with sub-exponential number of rounds. In the hybrid model protocol, the curator first solves the parity problem privately, and relates the identity of the secret to be recovered through the referee to the local model agents who then can send enough information to recover the required secret. See Section 5.

The latter two tasks highlight information and private-computation gaps between the curator and the local model agents. The local model agents receive enough information to solve the task, but lack the ability to privately solve an essential sub-task (when they are not allowed to use exponentially many rounds). The curator does not receive enough information to solve the task (even non-privately), but can solve the hard sub-task.

When the hybrid model does not help much

Although most of the results in this work are on the positive side, demonstrating that cleverly utilizing a curator in synergy with local agents can allow for new capabilities, we also show that for one of the most basic tasks – namely, basic hypothesis testing – the hybrid model has

no significant advantage over what can be done separately in either the local model with m agents or in the curator model with database of size n. We show that if for two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ there is a private protocol in the hybrid model that given i.i.d. samples from $\mathcal{D}_j$ correctly identifies j, then there is a private protocol with the same sample size either in the local model or in the curator model that correctly identifies j (with some loss in the success probability). We then consider two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ over the the domain $\{0,1\}$, where in $\mathcal{D}_0$ the probability of 1 is strictly less than $1/2$ and in $\mathcal{D}_1$ the probability of 1 is strictly greater than $1/2$ and identify values of m and n such that in the hybrid model, where the curator has m samples and there are n agents (each holding one sample), no protocol exists that can differentiate whether the $m+n$ inputs were sampled i.i.d. from $\mathcal{D}_0$ or from $\mathcal{D}_1$. Since computing the sum of i.i.d. sampled bits from $\mathcal{D}_0$ or $\mathcal{D}_1$ can distinguish between these distributions, the above results imply that for computing the sum, the hybrid model is no better than each model separately. See Section 6.

A new lower bound for selection in the local model

As mentioned above, our analysis for the select-then-estimate task relies on lower bounds on the sample complexity of selection in the local model. In the selection problem, there is some distribution over vectors in $\{-1,1\}^d$ and each agents gets an i.i.d. sample from this distribution; the goal is to select a coordinate such that the expected value of this coordinate is as large as possible. Ullman [60] gives a (tight) lower bound of $\Omega(\frac{d\log d}{\alpha^2\epsilon^2})$ samples for the non-interactive case. In the full version of this paper [10], we show that for *interactive* local model protocols, the number of messages in such protocol is $\Omega(d^{1/3})$. For example, if the curator interacts with the local model parties so that each party sends t messages, then the number of parties must be at least $\Omega(d^{1/3}/t)$. The proof is by a generic reduction from any private PAC learning task to selection, which preserves sample complexity. The bound is obtained by applying the reduction from privately learning parity and applying bounds on the sample complexity of privately learning parity from [43].

1.2 Discussion and future work

Our results show that the combination of the local model with a small curator can become part of a promising toolkit for designing and implementing differential privacy. More work is needed to develop the theory of this model (and possibly introduce variants), and, in particular, characterize which tasks can benefit from it. From an algorithms design perspective, now that we know that the hybrid model can lead to significant improvements over both the curator and local models, an exciting open question is understanding what other non-trivial algorithms can be designed that take advantage of the synergy.

Selection bias

In this work we assume that the inputs for the curator and for the local model agents come from the same distribution. However, the recruitment of individuals for participating via a curator can create unintended differences between the input distributions seen by the curator and the entire population, and hence lead to biases, an issue which is outside the scope of the current work. Selection bias remains an important issue that needs to be addressed.

Approximate differential privacy

Our separations between the hybrid model and the curator and local models hold for pure differential privacy (i.e., ϵ-differential privacy). Specifically, we use lower bounds for ϵ-differential private learning of the threshold functions in the curator model [12, 30]; these lower bounds do not hold for (ϵ, δ)-differential private learning of the threshold functions [13, 18]. We also use lower bounds for ϵ-differential private learning of parity in the local model [43]; it is open whether these lower bounds hold for fully interactive (ϵ, δ)-differential private learning protocols of parity. Possible separations for (ϵ, δ)-differential privacy are left for future research.

2 Preliminaries

2.1 Protocols for differentially private computations

Let X be a data domain. We consider a model where the inputs and the computation are distributed among parties $P_1, \ldots, P_n$. Each party is an interactive randomized functionality: it can receive messages from the other parties, perform a randomized computation, and send messages to the other parties. At the beginning of a computation, each party P_i receives its input $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,\ell_i}) \in X^{\ell_i}$. I.e., the input of party P_i consists of a sequence of $\ell_i \geq 0$ entries taken from the data domain X, and the entire joint input to the protocol is $(x_{1,1} \ldots, x_{1,\ell_1}, x_{2,1}, \ldots, x_{2,\ell_2}, \ldots, x_{n,1}, \ldots, x_{n,\ell_n})$. The parties engage in a randomized interactive protocol $\Pi = (\Pi_{P_1}, \ldots, \Pi_{P_n})$, where a message sent by a party P_i in some round is computed according to Π_{P_i} and depends on its input $\mathbf{x}_i$, its random coins, and the sequence of messages it has seen in previous rounds. When a party P_i halts, it writes its output to a local output register $\mathbf{o}_{P_i}$. The number of messages in a protocol is the number of rounds multiplied by the number of parties.

▶ **Definition 2.1.** *We say that* $\mathbf{x} = (x_1, \ldots, x_\ell) \in X^\ell$ *and* $\mathbf{x}' = (x'_1, \ldots, x'_\ell) \in X^\ell$ *are* neighboring *if they differ on at most one entry, i.e., there exists* $i^* \in [\ell]$ *such that* $x_i = x'_i$ *for* $i \in [\ell] \setminus \{i^*\}$.

▶ **Definition 2.2.** *We say that two probability distributions* $\mathcal{D}_0, \mathcal{D}_1 \in \Delta(\Omega)$ *are* (ϵ, δ)*-close and write* $\mathcal{D}_0 \approx_{\epsilon,\delta} \mathcal{D}_1$ *if* $\Pr_{t\sim\mathcal{D}_b}[t \in T] \leq e^\epsilon \cdot \Pr_{t\sim\mathcal{D}_{1-b}}[t \in T] + \delta$ *for all measurable events* $T \subset \Omega$ *and* $b \in \{0, 1\}$.

We are now ready to define what it means for a protocol to be differentially private in a fully malicious setting, i.e., in a setting where an arbitrary adversary controls the behavior of all but one party. Intuitively, a protocol is differentially private in a fully malicious setting if there do not exist a party P_i and an adversary A controlling $P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n$ such that A can "trick" P_i to act non-privately. More formally, we model the adversary as an interactive randomized functionality. For a party P_i, define A_{P_i} to be a randomized functionality as follows.

1. An input of A_{P_i} consists of a sequence of ℓ_i entries taken from the data domain, $\mathbf{x} \in X^{\ell_i}$.
2. A_{P_i} simulates an interaction between party P_i with $\mathbf{x}$ as its input, and A. The simulated P_i interacts with A following the instructions of its part in the protocol, Π_{P_i}. The adversary A interacts with P_i sending messages for parties $P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n$. However, A does not necessarily adhere to the protocol Π.
3. The simulation continues until A halts with an output $\mathbf{o}_A$, at which time A_{P_i} halts and outputs $\mathbf{o}_A$.

▶ **Definition 2.3** (Multiparty differential privacy [26, 43, 11, 61])**.** *A protocol* Π *is* (ϵ, δ)*-differentially private if for all* $i \in [n]$*, for all interactive randomized functionalities* A*, and all neighboring* $\mathbf{x}, \mathbf{x}' \in X^{\ell_i}$*,* $A_{P_i}(\mathbf{x}) \approx_{\epsilon,\delta} A_{P_i}(\mathbf{x}')$*. When* $\ell_1 = \ell_2 = \cdots = \ell_n = 1$ *we say that the protocol operates in the* local model*, and when* $n = 1$ *we say that the protocol (or the algorithm) operates in the* curator model*. We say that a protocol* Π *is* ϵ*-differentially private if it is* $(\epsilon, 0)$*-differentially private.*

Comparison to previous definitions

In contrast to [11, 61], our definition applies also to a malicious adversary that can send arbitrary messages. The definition of [43] also applies to a malicious adversary, however it requires that each answer of an agent preserves ϵ-differential privacy (i.e., if there are d rounds, the protocol is $d\epsilon$-differentially private). In contrast, the definitions of [11, 61] and our definition measures the privacy of the entire transcript of an agent.

Note that (i) Restricting the outcome $\mathbf{o}_A$ to binary results in a definition that is equivalent to Definition 2.3. (ii) It is possible to consider a relaxed version of Definition 2.3 where the adversary A is "semi-honest" by requiring A to follow the protocol Π. (iii) Definition 2.3 differs from definitions of security in the setting of secure multiparty computation as the latter also state correctness requirements with respect to the outcome of the protocol. The difference between the setting is that secure multiparty computation implements a specified functionality[3] whereas differential privacy limits the functionality to hide information specific to individuals, but does not specify it.

2.2 The hybrid model

A computation in the (m, n)-hybrid model is defined as the execution of a randomized interactive protocol $\Pi = (\Pi_C, \Pi_{P_1}, \ldots, \Pi_{P_n}, \Pi_R)$ between three types of parties: a (single) curator C, n "local model" agents $P_1, \ldots, P_n$, and a referee R. The referee has no input, the curator C receives m input points $\mathbf{x} = (x_1, \ldots, x_m) \in X^m$ and the n "local model" agents $P_1, \ldots, P_n$ each receive a single input point $y_i \in X$. We use the notation $\mathbf{D}$ to denote the joint input to the computation, i.e., $\mathbf{D} = (x_1, \ldots, x_m, y_1, \ldots, y_n)$.

The communication in a hybrid model protocol is restricted to messages exchanged between the referee R and the other parties $C, P_1, \ldots, P_n$ (i.e., parties $C, P_1, \ldots, P_n$ cannot directly communicate among themselves). Parties $C, P_1, \ldots, P_n$ have no output, whereas when the execution halts the referee R writes to a special output register $\mathbf{o}$. See Figure 1. We require the protocol $\Pi = (\Pi_C, \Pi_{P_1}, \ldots, \Pi_{P_n}, \Pi_R)$ to satisfy differential privacy as in Definition 2.3.

The hybrid model is a natural extension of well-studied models in differential privacy. Setting $n = 0$ we get the trusted curator model (as C can perform any differentially private computation), and setting $m = 0$ we get the local model. In this work, we are interested in the case $0 < m \ll n$, because in this regime, the hybrid model is closest in nature to the local model. Furthermore, in many applications, once m is comparable to n it is possible to drop parties $P_1, \ldots, P_n$ from the protocol without a significant loss in utility.

Comparing with Blender [2], where the curator C and the referee R are the same party, we observe that the models are equivalent in their computation power – every differentially

[3] Furthermore, secure multiparty computation is silent with respect to the chosen functionality, regardless whether it is "privacy preserving" or "secure".

private computation in one model is possible in the other (however, the models may differ in the number of interaction rounds needed). Nevertheless, the separation between the curator and the referee has merits as we now discuss.

On the separation between the curator and referee

From a theory point of view, it is useful to separate these two parties as this allows to examine effects related to the order of interaction between the parties (e.g., whether the referee communicates first with the curator C or with the local model parties $P_1, \ldots, P_n$).

Moreover, by separating the curator and referee, the hybrid model encapsulates a richer class of trust models than [2], and, in particular, includes a trust model where data sent to the curator is not revealed to the referee. In an implementation this may correspond to a curator instantiated by a publicly trusted party, or by using technologies such as secure multiparty computation, or secure cryptographic hardware which protects data and computation from external processes [49].

The curator-referee separation also makes sense from a practical point of view within a company. It is reasonable that only a small fraction of a company's employees, with appropriate clearance and training, should be able to access the raw data of those who contribute their data to the trusted curator model, whereas the majority of employees should only see the privacy-preserving version of it [53].

▶ Remark 2.4 (A note on public randomness). Some of our protocols assume the existence of a shared random string. In an implementation, shared randomness can be either set up offline or be chosen and broadcast by the referee. We stress that the privacy of our protocols does not depend on the shared random string actually being random. Furthermore, all our lower bounds hold even when the local agents hold a shared (public) random string.

2.3 Parity and threshold functions

A *concept* $c : X \to \{0,1\}$ is a predicate that labels *examples* taken from the domain X by either 0 or 1. A *concept class* C over X is a set of concepts (predicates) mapping X to $\{0,1\}$. Let $b, c \in \mathbb{N}$ be parameters. The following two concept classes will appear throughout the paper:

- $\mathsf{Threshold}_b = \left\{\mathsf{Thr}_t : t \in \{0,1\}^b\right\}$ where $\mathsf{Thr}_t : \{0,1\}^b \to \{0,1\}$ is defined as $\mathsf{Thr}_t(x) = \mathbb{1}_{\{x \geq t\}}$, where we treat strings from $\{0,1\}^b$ as integers in $\left\{0, \ldots, 2^b - 1\right\}$.
- $\mathsf{Parity}_c = \{\mathsf{Par}_k : k \in \{0,1\}^c\}$ where $\mathsf{Par}_k : \{0,1\}^c \to \{0,1\}$ is defined as $\mathsf{Par}_k(x) = \langle k, x\rangle = \oplus_{j=1}^{c} k_j \cdot x_j$.

2.4 Preliminaries from learning theory and private learning

We recall the probably approximately correct (PAC) model of [62]. Given a collection of labeled examples, the goal of a learning algorithm (or protocol) is to *generalize* the given data into a concept (called a "hypothesis") that accurately predicts the labels of fresh examples from the underlying distribution. (See [62], or the full version of this paper [10], for formal definitions.) When the learner is a protocol, its *sample complexity* is the total number of labeled examples it operates on. That is, if there are n parties where party P_i gets as input ℓ_i labeled examples, then the sample complexity of the protocol is $\ell_1 + \cdots + \ell_n$.

▶ **Definition 2.5.** *The* generalization error *of a hypothesis* $h : X \to \{0,1\}$ *w.r.t. a target concept* c *and a distribution* $\mathcal{D}$ *is defined as* $\mathrm{error}_{\mathcal{D}}(c,h) = \Pr_{x\sim\mathcal{D}}[h(x) \neq c(x)]$. *The* empirical error *of* h *w.r.t. a labeled sample* $\mathbf{D} = (x_i, y_i)_{i=1}^m$ *is* $\mathrm{error}_{\mathbf{D}}(h) = \frac{1}{m}|\{i : h(x_i) \neq y_i\}|$. *The*

empirical error *of h w.r.t. an unlabeled sample $\mathbf{D} = (x_i)_{i=1}^m$ and a concept c is* $\mathrm{error}_{\mathbf{D}}(h, c) = \frac{1}{m}|\{i : h(x_i) \neq c(x_i)\}|$. *Let $c : X \to \{0,1\}$ be a concept and let $\mathbf{D} \in (X \times \{0,1\})^n$ be a labeled database. We say that $\mathbf{D}$ is* consistent *with c if for every $(x, y) \in \mathbf{D}$ it holds that $y = c(x)$.*

▶ **Definition 2.6.** *Let C be a concept class over a domain X, and let Π be a protocol in which the input of every party is a collection of (1 or more) labeled examples from X. The protocol Π is called an* (α, β)-empirical learner *for C if for every concept $c \in C$ and for every joint input to the protocol $\mathbf{D}$ that is consistent with c, with probability at least $1 - \beta$, the outcome of the protocol is a hypothesis $h : X \to \{0,1\}$ satisfying $\mathrm{error}_{\mathbf{D}}(h) \leq \alpha$.*

We will be interested in PAC-learning protocols that are also differentially private.

▶ **Definition 2.7** ([43])**.** *A* private learner *is a PAC learner and that satisfies Definition 2.3. Similarly, a* private empirical learner *is a protocol Π that satisfies both Definitions 2.3 and 2.6.*

Dwork et al. [25] and Bassily et al. [7] showed that if a hypothesis h is the result of a differentially private computation on a random sample, then the empirical error of h and its generalization error are guaranteed to be close. We will use the following multiplicative variant of their result [51], whose proof is a variant of the original proof of [7].

▶ **Theorem 2.8** ([25, 7, 51, 29])**.** *Let $\mathcal{A} : X^n \to 2^X$ be an (ε, δ)-differentially private algorithm that operates on a database of size n and outputs a predicate $h : X \to \{0,1\}$. Let $\mathcal{D}$ be a distribution over X, let $S = (x_1, \ldots, x_n)$ be a database containing n i.i.d. elements from $\mathcal{D}$, and let $h \leftarrow \mathcal{A}(S)$. Then,*

$$\Pr_{\substack{S \sim \mathcal{D}^n \\ h \leftarrow \mathcal{A}(S)}}\left[e^{-2\varepsilon} \cdot \mathop{\mathrm{E}}_{x \sim \mathcal{D}}[h(x)] - \frac{1}{n}\sum_{i=1}^{n} h(x_i) > \frac{10}{\varepsilon n}\log\left(\frac{1}{\varepsilon\delta n}\right)\right] < O\left(\varepsilon\delta n\right).$$

We next state known impossibility results for privately learning threshold and parity.

▶ **Fact 2.9** ([12, 30])**.** *Let $b \in \mathbb{N}$. Any ϵ-differentially private (α, β)-PAC learner for $\mathsf{Threshold}_b$ requires $\Omega(\frac{b}{\epsilon\alpha})$ many samples.*

▶ **Fact 2.10** ([43])**.** *Let $c \in \mathbb{N}$. In any ϵ-differentially private (α, β)-PAC learning protocol for Parity_c* in the local model *the number of messages is $\Omega(2^{c/3})$. This holds even when the underlying distribution is restricted to be the uniform distribution.*

Fact 2.10 implies, for example, that when there are $\mathrm{poly}(c)$ agents the number of rounds is $2^{\Omega(c)}$. It is open whether there exists an ϵ-private protocol (or an (ϵ, δ)-private protocol) for learning Parity_c in the local model with $\mathrm{poly}(c)$ agents and any number of rounds.

▶ Remark 2.11. The proof of Fact 2.10 in [43] is stated in a weaker model, where in each round the referee sends an ϵ_i-differentially private local randomizer to an agent and the agent sends the output of this randomizer on its input to the referee, such that $\epsilon_1 + \cdots + \epsilon_\ell \leq \epsilon$. However, in their proof they only use the fact that $\epsilon_i \leq \epsilon$ in every round, thus, their lower bound proof also applies to our model.

Our protocols use the private learner of [43] for parity functions, a protocol of [8] for answering all threshold queries, a protocol of [17] for heavy hitters, and a protocol of [31] for approximating a quantile. These are specified in the following theorems.

▶ **Theorem 2.12** ([8])**.** *Let $\alpha, \beta, \varepsilon \leq 1$, and let $b \in \mathbb{N}$. There exists a non-interactive ϵ-differentially private protocol in the local model with $n = O\left(\frac{b^3}{\alpha^2\varepsilon^2} \cdot \log\left(\frac{b}{\alpha\beta\epsilon}\right)\right)$ agents in which the input of every agent is a single element from $\{0,1\}^b$ and the outcome is a function $q : \{0,1\}^b \to [0,1]$ such that for every joint input to the protocol $\mathbf{D} \in (\{0,1\}^b)^n$, with probability at least $1 - \beta$, the outcome q is such that $\forall w \in \{0,1\}^b$ we have $q(w) \in |\{x \in \mathbf{D} : x \leq w\}|/|\mathbf{D}| \pm \alpha$.*

Theorem 2.12 does not appear explicitly in [8], but it is implicit in their analysis.

▶ **Theorem 2.13** ([43]). *Let $\alpha, \beta, \varepsilon \leq 1$, and let $c \in \mathbb{N}$. There exists an ϵ-differentially private algorithm in the curator model that (α, β)-PAC learns and (α, β)-empirically learns Parity_c with sample complexity $O\left(\frac{c}{\alpha\varepsilon}\log(\frac{1}{\beta})\right)$.*

▶ **Theorem 2.14** (Heavy hitters protocol [17]). *There exist constants $\lambda_1, \lambda_2 > 0$ such that the following holds. Let $\beta, \epsilon \leq 1$ and X be some finite domain. There exists a non-interactive ϵ-differentially private protocol in the local model with n agents in which the input of each agent is a single element from X and the outcome is a list* Est *of elements from X such that for every joint input to the protocol $\mathbf{D} \in X^n$, with probability at least $1-\beta$, every x that is an input of at least $\frac{\lambda_1}{\epsilon}\sqrt{n\log\left(\frac{|X|}{\beta}\right)}$ agents appears in* Est*, and vice versa, every element x in* Est *is an input of at least $\frac{\lambda_2}{\epsilon}\sqrt{n\log\left(\frac{|X|}{\beta}\right)}$ agents.*

3 Learning parity XOR threshold

In this section we present a learning task that cannot be solved privately in the curator model or in the local model, but can be solved in the hybrid model (without interaction). The task we consider in this section – parity XOR threshold – is similar to the simpler task of the direct product of parity and threshold discussed in Section 1.1. In this section we design a non-interactive protocol in the hybrid model for the parity XOR threshold task, which is more involved than the trivial protocol for the parity and threshold task. This demonstrates that non-interactive protocols in the hybrid model may have more power than one might initially suspect.

Fix $b, c > 0$, and let $k \in \{0,1\}^c$ and $t \in \{0,1\}^b$ be parameters. Define the function $f^{k,t}_{b,c} : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}$ as follows: $f^{k,t}_{b,c}(x,y) = \mathsf{Par}_k(x) \oplus \mathsf{Thr}_t(y) = \langle k, x\rangle \oplus \mathbb{1}_{\{y \geq t\}}$ (recall that we treat strings in $\{0,1\}^b$ as integers in $\{0,1,\ldots,2^b-1\}$). Define the concept class ParityThresh as follows: $\mathsf{ParityThresh}_{b,c} = \left\{ f^{k,t}_{b,c} \ : \ k \in \{0,1\}^c \text{ and } t \in \{0,1\}^b \right\}$.

We show that every differentially private algorithm (even in the curator model) for learning ParityThresh must have sample complexity $\Omega(b)$. See the full version of this paper for details [10].

▶ **Lemma 3.1.** *Every ϵ-differentially private algorithm for $(\frac{1}{4}, \frac{1}{4})$-PAC learning $\mathsf{ParityThresh}_{b,c}$ must have sample complexity $\Omega(b)$.*

We next show that no protocol in the local model can learn ParityThresh, unless the number of exchanged messages is very large.

▶ **Lemma 3.2.** *In every ϵ-differentially private protocol in the local model for $(\frac{1}{4}, \frac{1}{4})$-PAC learning $\mathsf{ParityThresh}_{b,c}$ the number of messages is $\Omega(2^{c/3})$.*

The proof of Lemma 3.2 is analogous to the proof of Lemma 3.1 (using Fact 2.10 instead of Fact 2.9).

So, privately learning $\mathsf{ParityThresh}_{b,c}$ in the curator model requires $\Omega(b)$ labeled examples, and privately learning it in the local model requires $\Omega(2^{c/3})$ messages. We now show that $\mathsf{ParityThresh}_{b,c}$ can be learned privately by a non-interactive protocol in the hybrid model with roughly $O(c)$ examples for the curator and with roughly $O(b^3)$ local agents. We will focus on the case where $c \ll b$. Recall that a function $f^{k,t}_{b,c}(x,y) \in \mathsf{ParityThresh}_{b,c}$ is defined

as $f^{k,t}_{b,c}(x,y) = \mathsf{Par}_k(x) \oplus \mathsf{Thr}_t(y)$. The difficulty in learning $\mathsf{ParityThresh}$ in the hybrid model is that we could only learn the threshold part of the target function using the local agents (since if $c \ll b$ then the curator does not have enough data to learn it), but the threshold label is "hidden" from the local agents (because it is "masked" by the parity bit that the local agents cannot learn). This false intuition might lead to the design of an *interactive* protocol, in which the referee first obtains some information from the curator and then passes this information to the local agents, which would allow them to learn the threshold part of the target function. We now show that such an interaction is not needed, and design a *non-interactive* protocol in which the local agents and the curator communicate with the referee only once, simultaneously.

▶ **Lemma 3.3.** *There exists a non-interactive ϵ-differentially private protocol in the (m,n)-hybrid model for (α,β)-PAC learning $\mathsf{ParityThresh}_{b,c}$ where $m = O\left(\frac{c}{\alpha^4\varepsilon}\log(\frac{1}{\alpha\beta})\right)$ and $n = O\left(\frac{b^3}{\alpha^4\varepsilon^2}\cdot\log\left(\frac{b}{\alpha\beta\epsilon}\right)\right)$.*

Proof. We begin by describing a non-interactive protocol Π. The (joint) input to the protocol is a database $\mathbf{D}$ where every point in $\mathbf{D}$ is of the form $(x_i,y_i,\sigma_i) \in \{0,1\}^c \times \{0,1\}^b \times \{0,1\}$. At a high level, the protocol works by using the local agents to obtain an approximation to the CDF of the (marginal) distribution on the y_i's (this approximation is given to the referee). In addition, the trusted curator solves $1/\alpha$ parity leaning problems. In more detail, the trusted curator sorts its database according to the y_i's, divides its database into $1/\alpha$ chunks, and then applies a private learner for parity functions on each of the chunks. The trusted curator sends the referee the resulting $1/\alpha$ parity functions. The referee then defines the final hypothesis h that, given a point (x,y), first uses the approximation to the CDF (obtained from the local agents) to match this input point to one of the chunks, and then uses the parity function obtained for that chunk from the trusted curator to predict the label of the point.

The key observation here is that the threshold part of the target function is *constant* on all but at most one of the chunks defined by the trusted curator. As we show, applying a learner for parity on such a "consistent chunk" results in a good predictor for the labels of elements of that chunk. Hence, provided that the approximation for the CDF of the y_i's is accurate enough, this results in an accurate learner for $\mathsf{ParityThresh}$. We now formally present the protocol Π.

- Local agents on a (distributed) input $D = (x_i,y_i,\sigma_i)_{i=1}^n \in \left(\{0,1\}^c \times \{0,1\}^b \times \{0,1\}\right)^n$: Run the protocol from Theorem 2.12 on the (distributed) database $\hat{D} = (y_1,y_2,\ldots,y_n)$ with privacy parameter ε and utility parameters α^2,β; thereafter, the referee obtains a function $q : \{0,1\}^b \to [0,1]$ that approximates all threshold queries w.r.t. $\hat{D}$.
- The curator on input $S = (x_i,y_i,\sigma_i)_{i=1}^m \in \left(\{0,1\}^c \times \{0,1\}^b \times \{0,1\}\right)^m$:
  - Sort S according to the y_i's in non-decreasing order.
  - Divide S into blocks of size αm: $S_1,S_2,\ldots,S_{1/\alpha}$. For $\ell \in [1/\alpha]$ we denote $S_\ell = (x_{\ell,i},y_{\ell,i},\sigma_{\ell,i})_{i=1}^{\alpha m}$.
  - For every $\ell \in [1/\alpha]$, apply an $\alpha\varepsilon$-differentially private $(\alpha^2,\alpha\beta)$-PAC learner for Parity on the database $\hat{S}_\ell = (x_{\ell,i} \circ 1, \sigma_{\ell,i})_{i=1}^{\alpha m} \in \left(\{0,1\}^{c+1} \times \{0,1\}\right)^{\alpha m}$ to obtain a vector $k_\ell \in \{0,1\}^{c+1}$ (using Theorem 2.13), and end $k_1,\ldots,k_{1/\alpha}$ to the referee.
- The referee:
  - Obtain the function q and the vectors $k_1,\ldots,k_{1/\alpha}$.
  - Define a hypothesis $h : \{0,1\}^c \times \{0,1\}^b \to \{0,1\}$ as $h(x,y) = \langle x \circ 1, k_{I(y)}\rangle$, where $I(y) = \left\lceil \frac{q(y)}{\alpha} \right\rceil$ and output h.

The privacy properties of the protocol Π are straightforward, as both the local agents and the curator apply ε-differentially private computations: the local agents apply the algorithm from Theorem 2.12, and the curator applies an $\alpha\varepsilon$-differentially private computation on each of the blocks $S_1, \ldots, S_{1/\alpha}$ (note that changing one element of S can change at most one element of each of these blocks).

We now proceed with the utility analysis. Fix a target function $f_{b,c}^{k^*,t^*} \in \mathsf{ParityThresh}_{b,c}$ and fix a target distribution $\mathcal{D}$ on $\{0,1\}^c \times \{0,1\}^b$. We use $\mathcal{D}_c$ and $\mathcal{D}_b$ to denote the marginal distributions on $\{0,1\}^c$ and $\{0,1\}^b$, respectively. We will make the simplifying assumption that $\mathcal{D}_b$ does not give too much weight on any single point in $\{0,1\}^b$, specifically, $\Pr_{w\sim\mathcal{D}_b}[w = y] \leq \beta/m^2$ for every $y \in \{0,1\}^b$. This assumption can be enforced by padding every example with $O(\log(m/\beta))$ uniformly random bits.

Let S and D (the inputs to the curator and the local agents) be sampled i.i.d. from $\mathcal{D}$ and labeled by $f_{b,c}^{k^*,t^*}$. We next show that w.h.p. the resulting hypothesis h has low empirical error on S. By standard generalization arguments, such an h also has low generalization error.

First observe that there is at most one index $\ell^* \in [1/\alpha]$ such that $\mathsf{Thr}_{t^*}(y_{\ell^*,1}) \neq \mathsf{Thr}_{t^*}(y_{\ell^*,\alpha m})$. In all other blocks S_ℓ we have that $\mathsf{Thr}_{t^*}(\cdot)$ is constant on all the $y_{\ell,i}$'s of that block. We will show that w.h.p. the hypothesis h has small empirical error on every such block. Fix $\ell \neq \ell^*$, and let $\nu \in \{0,1\}$ be the value of $\mathsf{Thr}_{t^*}(\cdot)$ on the $y_{\ell,i}$'s of the ℓth block (that is, for every $i \in [\alpha m]$ we have $\mathsf{Thr}_{t^*}(y_{\ell,i}) = \nu$). Recall that since the elements of S are labeled by $f_{b,c}^{k^*,t^*}$, for every $i \in [\alpha m]$ we have that

$$\sigma_{\ell,i} = f_{b,c}^{k^*,t^*}(x_{\ell,i}, y_{\ell,i}) = \langle k^*, x_{\ell,i}\rangle \oplus \mathsf{Thr}_{t^*}(y_{\ell,i}) = \langle k^*, x_{\ell,i}\rangle \oplus \nu = \langle k^* \circ \nu,\ x_{\ell,i} \circ 1\rangle.$$

Hence, the elements of $\hat{S}_\ell$ are all labeled by the parity function defined by $k^* \circ \nu$. Therefore, as k_ℓ is the outcome of the learner from Theorem 2.13 on $\hat{S}_\ell$, for $m \geq O\left(\frac{c}{\alpha^2\varepsilon}\log(\frac{1}{\alpha\beta})\right)$, with probability at least $1 - \alpha\beta$ we have that $\mathrm{error}_{\hat{S}_\ell}(\mathsf{Par}_{k_\ell}) \leq \alpha^2$, that is, $\langle k_\ell,\ x \circ 1\rangle$ is a good predictor for the label of the elements in block S_ℓ.

Recall that the hypothesis h matches inputs (x, y) to the vectors $k_1, \ldots, k_{1/\alpha}$ using the function q obtained from the local agents, that is, on input (x, y), the hypothesis uses $k_{\lceil q(y)/\alpha\rceil}$. Therefore, to complete the proof we need to show that most of the elements from block S_ℓ are matched by the hypothesis h to the vector k_ℓ. To that end, let $\#_S(w) = |\{(x, y, \sigma) \in S : y \leq w\}|$, and consider the following event $\boldsymbol{E}_1 : \forall w \in \{0,1\}^b$ it holds that $\left|q(w) - \frac{1}{m} \cdot \#_S(w)\right| \leq 4\alpha^2$.

We first conclude the proof assuming that Event E_1 occurs. Fix $\ell \neq \ell^*$, and recall that the elements of S (and in particular the elements of S_ℓ) are sorted in a non-decreasing order according to their y_i's. Now fix $8\alpha^2 m \leq i \leq \alpha m - 8\alpha^2 m$. By our simplifying assumption (that the distribution $\mathcal{D}_b$ does not put a lot of mass on any single point), we may assume that all the y_i's in S are distinct, which happens with probability at least $1 - \beta$. In that case, we have that $\#_S(y_{\ell,i}) = \max\{0, \ell - 1\} \cdot \alpha m + i$, and hence, $\max\{0, \ell - 1\} \cdot \alpha + 8\alpha^2 \leq \frac{1}{m}\#_S(y_{\ell,i}) \leq \max\{0, \ell - 1\} \cdot \alpha + \alpha - 8\alpha^2$. By Event E_1 we get that

$$\max\{0, \ell - 1\} \cdot \alpha + 4\alpha^2 \leq q(y_{\ell,i}) \leq \max\{0, \ell - 1\} \cdot \alpha + \alpha - 4\alpha^2,$$

and so, $\left\lceil \frac{q(y_{\ell,i})}{\alpha} \right\rceil = \ell$. That is, for all but at most $16\alpha^2 m$ elements of the block S_ℓ we get that $h(x_{\ell,i}, y_{\ell,i}) = \langle x_{\ell,i} \circ 1,\ k_\ell\rangle = \mathsf{Par}_{k_\ell(x_{\ell,i}, y_{\ell,i})}$. Recall that Par_{k_ℓ} errs on at most $\alpha^2 m$ elements of S_ℓ, and so the hypothesis h errs on at most $17\alpha^2 m$ elements of the block S_ℓ. That is, h errs on at most $17\alpha^2 m$ elements of every block S_ℓ for $\ell \neq \ell^*$, and might err on all of S_{ℓ^*} which is of size αm. So, h errs on at most $\frac{1}{\alpha} \cdot 17\alpha^2 m + \alpha m = 18\alpha m$ elements of S.

Standard generalization bounds now state that, except with probability at most β, we get that $\text{error}_{\mathcal{D}}(h, f_{b,c}^{k^*,t^*}) \leq O(\alpha)$ (in particular, this follows from the generalization properties of differential privacy; see Section 2.4 for more details). Overall, with probability at least $1 - O(\beta)$ the resulting hypothesis has generalization error at most $O(\alpha)$.

It remains to show that Event E_1 occurs with high probability. First, Theorem 2.12 ensures that for $n \geq O\left(\frac{b^3}{\alpha^4\varepsilon^2} \cdot \log\left(\frac{b}{\alpha\beta\epsilon}\right)\right)$ with probability at least $1-\beta$ the function q is such that $\forall w \in \{0,1\}^b$ it holds that $\left|q(w) - \frac{1}{n}\#_{\hat{D}}(w)\right| \leq \alpha^2$, where $\#_{\hat{D}}(w) = |\{y \in \hat{D} : y \leq w\}|$. Second, by standard generalization arguments, assuming that n and m are big enough, we would also have that $\frac{1}{n}\#_{\hat{D}}(w)$ and $\frac{1}{m}\#_S(w)$ are both within α^2 from $\Pr_{y\sim\mathcal{D}_b}[y \leq w]$. Specifically, by the Dvoretzky-Kiefer-Wolfowitz inequality [24, 45], assuming that n and m are at least $\Omega\left(\frac{1}{\alpha^4}\log(\frac{1}{\beta})\right)$, this happens with probability at least $1-\beta$. Assuming that this is the case, by the triangle inequality we have that Event E_1 holds. This shows that Event E_1 happens with probability at least $1-3\beta$, and completes the proof. ◀

We remark that it is possible to design a more efficient learner for ParityThresh (in terms of sample complexity) by constructing a protocol in which there are multiple rounds of communication between the referee and the local agents (but this communication is still independent from the message that the curator sends to the referee). See the full version of this paper for more details [10]. We summarize our possibility and impossibility results w.r.t. learning ParityThresh in the next theorem (which follows from Lemma 3.1 and Lemma 3.2 and from Lemma 3.3).

▶ **Theorem 3.4.** *Let $c \in \mathbb{N}$ and $b = c^2$. Then there is a non-interactive $\frac{1}{4}$-differentially private $(\frac{1}{4}, \frac{1}{4})$-PAC learner for $\text{ParityThresh}_{b,c}$ in the (m,n)-hybrid model with $m = O(c)$ samples for the curator and $n = O(c^6 \log c)$ local agents. However, every such learner in the local model with $o(2^{(n/\log n)^{1/6}})$ local agents requires $2^{\Omega((n/\log n)^{1/6})}$ rounds, and every such learner in the curator model requires $\Omega(m^2)$ samples.*

4 The 1-out-of-2^d-parity task

In this section we describe a task that cannot be privately solved neither in the curator model nor in the local model with sub-exponential number of rounds. In the hybrid model, this task can only be solved with interaction, first with the local agents and then with the curator. In this task there are many instances of the parity problem and the referee needs to solve only one instance, which is determined by the inputs. The local agents can determine this instance (using a heavy hitters protocol) and the curator can now solve this instance. The curator cannot solve all instances since this will exceed its privacy budget, and by the definition of the task the curator will not have enough information to determine the instance; thus interaction with both the local agents and the curator is required.

▶ **Definition 4.1** (The 1-out-of-2^d-parity task)**.** *The inputs in the 1-out-of-2^d-parity task are generated as follows:*

1. ***Input:*** *2^d strings $(r_j)_{j\in\{0,1\}^d}$, where $r_j \in \{0,1\}^c$ for every $j \in \{0,1\}^d$, and $m+1$ elements $s_1, \ldots, s_{m+1} \in \{0,1\}^d$.*
2. *Set $s = s_1 \oplus \cdots \oplus s_{m+1}$.*[4]

[4] The strings $s_1, \ldots, s_{m+1}$ are an $m+1$-out-of-$m+1$ secret sharing of s, that is, together they determine s, but every subset of them gives no information on s.

3. *Each sample $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$ is generated independently as follows:*
   - *with probability half choose $x \in_R \{0,1\}^c$ with uniform distribution and output $(x, (\langle x, r_j \rangle)_{j \in \{0,1\}^d})$ (that is, every point contains a string x of length c and 2^d bits which are the inner products of x and each of the r_j's).*
   - *with probability half choose $t \in_R [m+1]$ with uniform distribution and output (t, s_t) (that is, every point contains a number t and the t-th string s_t).*

The goal of the referee in the 1-out-of-2^d-parity task is for every $(r_j)_{j \in \{0,1\}^d}$ and $s_1, \ldots, s_{m+1}$ to recover r_s with probability at least $1 - \beta$, where the probability is over the generation of the inputs in Step 3 and the randomness of the parties in the protocol.

We design a protocol for this task and obtain the following lemma (see the full version of this paper for details [10]).

▶ **Lemma 4.2.** *Let $\beta > 1/m$ and assume that $m = \Omega\left(\frac{c \log(1/\beta)}{\epsilon}\right)$ and $n = \Omega\left(\frac{m^2}{\epsilon^2} \log(\frac{m2^d}{\beta})\right)$. The 1-out-of-$2^d$-parity task can be solved in the (m,n)-hybrid model by an ϵ-differentially private protocol with three rounds, where in the first round each local agent sends one message to the referee (without seeing any other messages), in the second round the referee sends one message to the curator, and in the third round the curator sends one message to the referee.*

We next prove that, unless the database is big, the 1-out-of-2^d-parity task requires interaction. To prove this result, we first convert a protocol for the 1-out-of-2^d-parity task to a private algorithm in the trusted curator model that recovers all strings $(r_j)_{j \in \{0,1\}^d}$. We then prove, using a simple packing argument, that, unless the database is "big", such algorithm cannot exist. For our proof, we define the all-2^d-parity task as the task in which all inputs are of the form $(x, (\langle x, r_j \rangle)_{j \in \{0,1\}^d})$ and the goal of the referee is to reconstruct all strings $(r_j)_{j \in \{0,1\}^d}$.

▷ Claim 4.3. Let $m < n$. If there is an ϵ-differentially private protocol for the 1-out-of-2^d-parity problem in the (m,n)-hybrid model in which the curator and the referee can exchange many messages and then the referee simultaneously sends one message to each local agent and gets one answer from each agent, then there is an ϵ-differentially private algorithm in the trusted curator model for the all-2^d-parity problem for a database of size $O(nd)$.

Proof. Let Π be an ϵ-differentially private protocol with the above interaction pattern for the 1-out-of-2^d-parity task in the (m,n)-hybrid model in which the referee reconstructs r_s with probability at least $1 - \beta$. We construct, in three steps, an algorithm $\mathcal{A}$ for the all-2^d-parity task in the trusted curator.

First, we construct from Π a protocol Π' in the $(O(md), O(dn))$-hybrid model that reconstructs r_s with error probability at most $\beta/2^d$ (e.g., execute Π with disjoint inputs $O(d)$ times and take the value r_s that is returned in the majority of the executions).

Next, we construct from protocol Π' a protocol Π'' for the the all-2^d-parity task in the $(O(md), O(nd))$-hybrid model (with error probability $\leq \beta$). In Π'', the parties holding inputs of the all-2^d-parity problem simulate Π' on inputs for the 1-out-of-2^d-parity task as follows:

- The curator on input $(x_i, (\langle x_i, r_j \rangle)_{j \in \{0,1\}^d})_{i=1}^m$:
  - Chooses random $s_1, \ldots, s_{m+1} \in_R \{0,1\}^d$.
  - For each $i \in [m]$, with probability $1/2$ replaces its i-th input by (t_i, s_{t_i}) for a uniformly distributed $t_i \in_R [m+1]$.
  - Exchanges messages with the referee as specified by Π' on the new input. In addition it sends to the referee $s_1, \ldots, s_{m+1}$ and an index ℓ such that (ℓ, s_ℓ) does not appear in its new input.

- The referee after getting the message from the curator:
  - Chooses a set $A \subseteq [n]$ with uniform distribution.
  - For every $i \notin A$, sends its message in Π' to the i-th agent and gets an answer M_i from the agent.
  - For every $i \in A$, chooses a random $q_i \in_{\mathrm{R}} [m+1]$. Let $B = \{i \in A : q_i = \ell\}$.
  - For every $i \in A \setminus B$, computes (without any interaction) its message in Π' to agent P_i and the answer M_i of agent P_i with input (q_i, s_{q_i}).
  - For every $i \in B$ and $s \in \{0,1\}^d$, computes (without any interaction) its message in Π' to agent P_i and the answer $M_{i,s}$ of agent P_i with input $(\ell, s \oplus \bigoplus_{k \neq \ell} s_k)$.
  - For every $s \in \{0,1\}^d$, reconstructs r_s from the messages of the curator in Π', $(M_i)_{i \notin B}$, and $(M_{i,s})_{i \in B}$.

As the curator holds m samples and there are $m+1$ values $s_1, \ldots, s_{m+1}$, there exists an index ℓ such that (ℓ, s_ℓ) does not appear in the new input of the curator. Thus, the referee for every $s \in \{0,1\}^d$ can choose a value s'_ℓ such that it is consistent with the messages of the curator in Π' and $s = s'_\ell \oplus \bigoplus_{k \neq \ell} s_k$. Furthermore, each of $x_1, \ldots, x_m, y_1, \ldots, y_n$ is replaced with probability half with a value (t, s_t) for a uniformly distributed t, thus, these inputs are distributed as required for the 1-out-of-2^d-parity task. This implies that for every $s \in \{0,1\}^d$ the referee reconstructs r_s from the messages of the curator in Π', $(M_i)_{i \notin B}$, and $(M_{i,s})_{i \in B}$ with probability at least $1 - \beta/2^d$. By the union bound, the referee correctly reconstructs all $(r_j)_{j \in \{0,1\}^d}$ with probability at least $1 - \beta$.

Finally, we construct the desired algorithm $\mathcal{A}$ from Π''. The trusted curator simply simulates the referee, the curator, and the agent in Π'', that is, it takes its database with $O((m+n)d)$ samples and partitions it to $(x_1, \ldots, x_{O(md)})$ (the input of the curator) and $y_1, \ldots, y_{O(nd)}$, computes without any interaction a random transcript of Π'' on these inputs, and reconstructs the output $(r_j)_{j \in \{0,1\}^d}$. Since the transcript preserves ϵ-differential privacy and computing the output is post-processing, algorithm $\mathcal{A}$ is ϵ-differential private. ◁

▷ Claim 4.4. If there is exists an ϵ-differentially private algorithm in the trusted curator model for the all-2^d-parity task with strings of length c, then $n = \Omega\left(\frac{c2^d + \ln(1-\beta)}{\epsilon}\right)$.

Proof. The proof is by a simple packing argument. For every strings $(r_j)_{j \in \{0,1\}^d}$, with probability at least $1 - \beta$, the algorithm returns $(r_j)_{j \in \{0,1\}^d}$ when the samples are generated with $(r_j)_{j \in \{0,1\}^d}$. By the group privacy of ϵ-differential privacy, with probability at least $e^{-n\epsilon}(1-\beta)$ the algorithm returns $(r_j)_{j \in \{0,1\}^d}$ when the samples are generated with $(0^c)_{j \in \{0,1\}^d}$. As there are 2^{c2^d} options for $(r_j)_{j \in \{0,1\}^d}$ and the above events are disjoint, $2^{c2^d} e^{-n\epsilon}(1-\beta) \leq 1$, i.e., $n = \Omega\left(\frac{c2^d + \ln(1-\beta)}{\epsilon}\right)$. ◁

▶ **Lemma 4.5.** *Let $m < n$. If there is an ϵ-differentially private protocol for the 1-out-of-2^d-parity task in the (m,n)-hybrid model with $\beta = 1/4$ in which the curator and the referee can exchange many messages and then the referee simultaneously sends one message to each local agent and gets one answer from each agent, then $n = \Omega(c2^d/d\epsilon)$.*

Proof. By Claim 4.3, if there exists an ϵ-differentially protocol in the (m,n)-hybrid model for the 1-out-of-2^d-parity task, then there exists an ϵ-differentially private algorithm in trusted curator model for the all-2^d-parity task with database of size $O(dn)$. Thus, by Claim 4.4 with $\beta = 1/4$, $dn = \Omega(\frac{c2^d}{\epsilon})$. ◀

Lemma 4.5 is valid also if the local agents are allowed to hold a shared (public) random string as this string can be sent by the referee to each agent as part of its message (without adding extra rounds of communication). We summarize the possibility and impossibility results for the 1-out-of-2^d-parity task in the following theorem, where, for convenience, we choose specific parameters that highlight these results.

▶ **Theorem 4.6.** *Let $\epsilon = 1/4$, $\beta = 1/4$. For every integer c, there are $d = \Theta(\log c)$, $m = \Theta(c)$, and $n = \Theta(c^2 \log c)$ such that*

1. *There exists an ϵ-differentially private protocol for the 1-out-of-2^d-parity task with strings of length c in the (m, n)-hybrid model where first each local agent sends one message to the referee and then the referee exchanges one message with the curator.*
2. *There does not exist an ϵ-differentially private protocol for this task in the (m, n)-hybrid model in which the referee first exchanges messages with the curator and then simultaneously exchanges one message with the local agents.*
3. *In any ϵ-differentially private protocol for this task in the local model with n agents the number of rounds is $2^{\Omega(c)} = 2^{\Omega(\sqrt{n/\log n})}$.*
4. *There is no algorithm in the trusted curator model that solves this task with m examples.*

Proof. Item 1 follows directly from Lemma 4.2.

For Item 2, by Lemma 4.5, with $\epsilon = 1/4$, $\beta = 1/4$, and $d = 2\log c + \log\log c$, it must hold that $n = \Omega\left(c2^d/(d\epsilon)\right) = \Omega(c^3)$, contradicting the choice of $n = \Theta(c^2 \log c)$.

For the impossibility result in Item 3, recall that by Fact 2.10 the number of messages sent to the referee in an ϵ-differentially private learning protocol in the local model for parity of strings of length c with respect to the uniform distribution is $2^{\Omega(c)}$. By simple simulation, an ϵ-differentially private protocol in the local model for the 1-out-of-2^d-parity task implies an ϵ-differentially private protocol in the local model for learning parity with respect to the uniform distribution (with the same number of messages). Specifically, since the number of agents is $n = O(c^2 \log c)$, the number of rounds is $2^{\Omega(c)}/(c^2 \log c) = 2^{\Omega(\sqrt{n/\log n})}$.

For Item 4, observe that a curator receiving m input points obtains less than $m+1$ shares of s and hence obtains no information about r_s. Hence, such a curator cannot solve the 1-out-of-2^m-parity task alone, even without privacy constraints. ◀

5 The parity-chooses-secret task

We now present another task that cannot be privately solved neither in the curator model nor in the local model with sub-exponential number of rounds. This task can be solved in the hybrid model; however, it requires interaction, this time first with the curator and then with the local agents. This task (as well as 1-out-of-2^d-parity task) highlights both the information and private-computation gaps between the curator and the local model agents. The local model agents receive enough information to solve the task, but lack the ability to privately solve an essential sub-task. The curator does not receive enough information to solve the task (even non-privately), however the curator can be used to privately solve the hard sub-task. Once the hard sub-task is solved, this information is forwarded to the local agents, which now can solve the task.

▶ **Definition 5.1** (The parity-chooses-secret task)**.** *The inputs in the parity-chooses-secret task are generated as follows:*

1. ***Input:*** *A string $r \in \{0,1\}^c$ and 2^c vectors of $m+1$ bits: a vector $(s_{j,1}, \ldots, s_{j,m+1}) \in \{0,1\}^{m+1}$ for every $j \in \{0,1\}^c$.*

2. *Set* $s_j = s_{j,1} \oplus \cdots \oplus s_{j,m+1}$ *for every for* $j \in \{0,1\}^c$*, i.e.,* s_j *is a random bit shared via an* $m+1$*-out-of-*$m+1$ *secret-sharing scheme, with the shares being* $s_{j,1}, \ldots, s_{j,m+1}$*.*
3. *Each sample* $x_1, \ldots, x_m$ *and* $y_1, \ldots, y_n$ *is generated independently as follows:*
   - *Choose* $x \in_R \{0,1\}^c$ *and* $t \in_R [m+1]$ *and output* $(x, \langle x, r\rangle, t, (s_{j,t})_{j\in\{0,1\}^c})$ *(that is, every point contains a string of length* c*, its inner product with* r*, an integer* t*, and the* t*-th share of each* s_j*).*

The goal of the referee in the parity-chooses-secret task is for every $\left((s_{j,1}, \ldots, s_{j,m+1})\right)_{j\in\{0,1\}^c}$ *and every* r *to recover* s_r *with probability at least* $1-\beta$*, where the probability is over the generation of the inputs in Step 3 and the randomness of the parties.*

We design a protocol for this task and obtain the following lemma (see the full version of this paper for details [10]).

▶ **Lemma 5.2.** *Let* $\beta > 1/m$ *and assume that* $m = \Omega\left(\frac{c\log(1/\beta)}{\epsilon}\right)$ *and* $n = \Omega\left(\frac{m^2}{\epsilon^2}\log(\frac{n}{\epsilon})\right)$*. The parity-chooses-secret task can be solved in the* (m,n)*-hybrid model by an* ϵ*-differentially private protocol with three rounds, where in the first round the curator sends one message to the referee, in the second round the referee sends one message to the local agents, and in the third round each local agent sends one message to the referee.*

We next prove that, unless the database is big, the parity-chooses-secret task requires interaction. Furthermore, we rule-out protocols in which first the referee simultaneously sends one message to each local agent, then receives an answer from each local agent, and finally exchanges (possibly many) messages with the curator. In particular, we rule-out the communication pattern used in Lemma 4.2 for the 1-out-of-2^d-parity task. To prove this result, we first convert a protocol Π for the parity-chooses-secret task with the above communication pattern to a protocol Π' in the hybrid model with the same communication pattern for a similar task (which we call the parity-chooses-secret' task, defined below). We then convert the protocol Π' to a non-interactive protocol Π'' in the local model for another related task, and complete the proof by showing an impossibility result for the related task.

We define the parity-chooses-secret' task as the task in which the input of the curator is generated as in the parity-chooses-secret task and the input of each local agent only contains shares, that is, it is of the form $(t, (s_{j,t})_{j\in\{0,1\}^c})$. The goal of the referee remains the same – to recover s_r.

▷ Claim 5.3. Assume that $m = \Omega\left(\frac{c\log(1/\beta)}{\epsilon}\right)$. If there is an ϵ-differentially private protocol for the parity-chooses-secret task in the (m,n)-hybrid model with error at most β in which in the first round the referee simultaneously sends one message to each local agent, in the second round gets an answer from each agent, and then the referee and the curator exchange (possibly many) messages, then there is a 2ϵ-differentially private protocol for the parity-chooses-secret' task in the (m,n)-hybrid model with error at most 2β with the same communication pattern.

Proof. Let Π be an ϵ-differentially private protocol for the parity-chooses-secret task in the (m,n)-hybrid model with the communication pattern as in the claim in which the referee reconstructs s_r with probability at least $1-\beta$. We construct from Π a 2ϵ-differentially private protocol Π' with the same communication pattern for the the parity-chooses-secret' task in which the referee reconstructs s_r with probability at least $1-2\beta$. In Π', each agent P_i, holding an input $(t, (s_{j,t})_{j\in\{0,1\}^c})$, chooses with uniform distribution a string $x_i \in_{\mathrm{R}} \{0,1\}^c$ and sends two messages of Π, one message, denoted $M_{i,0}$, for the input $(x_i, 0, t, (s_{j,t})_{j\in\{0,1\}^c})$ and one message, denoted $M_{i,1}$, for the input $(x_i, 1, t, (s_{j,t})_{j\in\{0,1\}^c})$. In addition, the agent

sends x_i to the referee. The referee sends the messages that it gets from the local agents (i.e., $(x_i, M_{i,0}, M_{i,1})_{i\in[n]}$) and its random string to the curator. The curator does as follows:

- Privately learns r by executing the ϵ-differentially private algorithm of [43] (see Theorem 2.13) for learning parity with $\alpha = 1/4$, β, and the m inputs $(x, \langle x, r\rangle)$.
- For each agent P_i, computes $b_i = \langle x_i, r\rangle$ and $M_i = M_{i,b_i}$ (that is, the curator chooses the correct message from the two messages the agent sends).
- Simulates the communication between the curator and the referee in Π assuming that the curator gets the messages $(M_i)_{i\in[n]}$ in the first round and reconstructs s_r as the referee reconstructs it in Π.
- Sends s_r to the referee.

As each party executes two ϵ-differentially private algorithms on its input, the resulting protocol is 2ϵ-differentially private. Each agent P_i chooses x_i with uniform distribution (as in the parity-chooses-secret task). Furthermore, as m is big enough, with probability at least $1-\beta$, the curator computes the correct value r. Thus, $(x_i, b_i, t, (s_{j,t})_{j\in\{0,1\}^c})$ is an input distributed as required in the parity-chooses-secret task, and the curator reconstructs s_r with probability at most $1-2\beta$. ◁

We next state a result of [46] showing that the mutual information between the input and output of a differential private algorithm is low. In this result and in the proofs that follow, $H(X)$ is an entropy of a random variable and $I(X;Y)$ is the mutual information between the random variables X and Y.

▶ **Theorem 5.4** (Differential privacy implies low mutual information [46]). *Let $A : X^n \to Y$ be an ϵ-differentially private mechanism. Then for every random variable V distributed on X^n, we have $I(V; A(V)) \le 1.5\epsilon n$.*

▶ **Lemma 5.5.** *Assume that $m = \Omega\left(\frac{c\log(1/\beta)}{\epsilon}\right)$. If there is an ϵ-differentially private protocol for the parity-chooses-secret task in the (m,n)-hybrid model with error at most β in which in the first round the referee simultaneously sends one message to each local agent, in the second round gets an answer from each agent, and then the referee and the curator exchange (possibly many) messages, then $n \ge \frac{(1-2\beta)2^c}{3\epsilon}$.*

Proof. We convert the protocol for the parity-chooses-secret task to a protocol Π'' in the local model with n agents, where the input of each agent contains 2^c bits $(s_j)_{j\in\{0,1\}^c}$. If the inputs of all agents are equal, then for every $r \in \{0,1\}^c$ the referee should output the bit s_r with probability at least $1-\beta$. We will show at the end of the proof that such protocol can exist only if n is big.

By Claim 5.3, under the assumption of the lemma there is a 2ϵ-differentially private protocol Π' for the parity-chooses-secret' task in the (m,n)-hybrid model with error at most 2β and communication pattern is as in the lemma. We construct the following protocol Π'' in the local model with n agents:

- **Input of each agent P_i:** $(s_j)_{j\in\{0,1\}^c}$.
- The referee chooses with uniform distribution 2^c vectors of $m+1$ bits: a vector $(s_{j,1}, \ldots, s_{j,m+1}) \in_R \{0,1\}^{m+1}$ for every $j \in \{0,1\}^c$.
- The referee chooses with uniform distribution m indices $t_1, \ldots, t_m \in_R [m+1]^m$. Let ℓ be an index that does not appear in this list.
- The referee chooses with uniform distribution m strings $(x_1, \ldots, x_m) \in_R (\{0,1\}^c)^m$.
- The referee sends $((s_{j,1}, \ldots, s_{j,m+1}))_{j\in\{0,1\}^c}$ and ℓ to each agent.

- Each agent P_i chooses with uniform distribution an index $t \in [m+1]$. If $t \neq \ell$, it sends to the referee its message in Π' on input $t, (s_{j,t})_{j\in\{0,1\}^c}$. If $t = \ell$, it sends to the referee its message in protocol Π' on input $\ell, \left(s_j \oplus \bigoplus_{k\neq\ell} s_{j,k}\right)_{j\in\{0,1\}^c}$. Denote this message by M_i.
- For every $r \in \{0,1\}^c$, the referee does the following:
  - Computes (without interaction) the communication, denoted $M_{C,r}$, exchanged in Π' between the referee and the curator with input $(x_1, \langle x_1, r\rangle, t_1, (s_{j,t_1})_{j\in\{0,1\}^c}), \ldots, (x_m, \langle x_m, r\rangle, t_m, (s_{j,t_m})_{j\in\{0,1\}^c})$.
  - The referee reconstructs s_r from the messages $M_{C,r}, M_1, \ldots, M_n$ using the reconstruction function of Π'.

Protocol Π'' is 2ϵ-differentially private, since Π' is 2ϵ-differentially private. Furthermore, if all the inputs of the local agents are equal, then $M_{C,r}, M_1, \ldots, M_n$ are distributed as in Π', thus, for every $r \in \{0,1\}^c$, the referee reconstructs s_r with probability at least $1-\beta$.

We complete the proof by showing that n must be large enough in Π'' (hence, also in Π). Assume we execute protocol Π'' when $(s_j)_{j\in\{0,1\}^c}$ is choosen with uniform distribution and denote its input by $(s_j)_{j\in\{0,1\}^c}$ and its output by $(s'_j)_{j\in\{0,1\}^c}$. As the output in Π'' is computed from the transcript of Π', the post-processing property of differential privacy implies that the algorithm that first executes protocol Π' and then computes the output from the transcript is 2ϵ-differentially private. By Theorem 5.4,

$$I\left((s_j)_{j\in\{0,1\}^c}; (s'_j)_{j\in\{0,1\}^c}\right) \leq 3\epsilon n. \tag{1}$$

On the other hand, $\Pr[s_{j_0} = s'_{j_0}] \geq 1-2\beta$ for a given $j_0 \in \{0,1\}^c$, thus $H(s_{j_0}|(s'_j)_{j\in\{0,1\}^c}) \leq H(s_{j_0}|s'_{j_0}) \leq 2\beta$, and $H\left((s_j)_{j\in\{0,1\}^c}|(s'_j)_{j\in\{0,1\}^c}\right) \leq \sum_{j_0\in\{0,1\}^c} H\left(s_{j_0}|(s'_j)_{j\in\{0,1\}^c}\right) \leq 2\beta 2^c$. Thus,

$$\begin{aligned} I\left((s_j)_{j\in\{0,1\}^c}; (s'_j)_{j\in\{0,1\}^c}\right) &= H\left((s_j)_{j\in\{0,1\}^c}\right) - H\left((s_j)_{j\in\{0,1\}^c}|(s'_j)_{j\in\{0,1\}^c}\right) \\ &\geq 2^c - 2\beta 2^c = (1-2\beta)2^c. \end{aligned} \tag{2}$$

Inequalities (1) and (2) imply that $(1-2\beta)2^c \leq 3\epsilon n$. ◀

We next summarize the possibility and impossibility results for parity-chooses-secret.

▶ **Theorem 5.6.** *Let $\epsilon = 1/4$, $\beta = 1/4$. For every integer c, there are $m = \Theta(c)$ and $n = \Theta(c^2 \log c)$ such that*

1. *There is an ϵ-differentially private protocol for the parity-chooses-secret task with strings of length c in the (m,n)-hybrid model where first the curator sends one message to the referee and then the referee simultaneously exchanges one message with each local agent.*
2. *There does not exist an ϵ-differentially private protocol for this task in the (m,n)-hybrid model in which the referee first simultaneously exchanges one message with the local agents and then exchanges messages with the curator.*
3. *In any ϵ-differentially private protocol for this task in the local model with n agents the number of rounds is $2^{\Omega(c)}$.*
4. *There is no algorithm in the trusted curator model that solves this task with m examples.*

Proof. Item 1 follows directly from Lemma 5.2. Item 2 is implied by Lemma 5.5, since $n \ll 2^c$. Item 3 follows from Fact 2.10 as in the proof of Theorem 4.6. For Item 4, observe that a curator receiving m input points obtains less than $m+1$ shares of $(s_1, \ldots, s_j)$ and hence obtains no information about s_r. That is, such a curator cannot solve the parity-chooses-secret task alone, even without privacy constraints. ◀

6 A Negative Result: Basic hypothesis testing

Here, we show that for one of the most basic tasks, differentiating between two discrete distributions $\mathcal{D}_0$ and $\mathcal{D}_1$, the hybrid model gives no significant added power.

▶ **Definition 6.1** (The simple hypothesis-testing task). *Let $0 < \beta < 1$ be a parameter, X be a finite domain, and $\mathcal{D}_0$ and $\mathcal{D}_1$ be two distributions over X. The input of the hypothesis-testing task is composed of i.i.d. samples from $\mathcal{D}_j$ for some $j \in \{0,1\}$ and the goal of the referee is to output $\hat{j}$ s.t. $\Pr[\hat{j} = j] \geq 1-\beta$.*

▶ **Theorem 6.2.** *If there is an ϵ-differentially private protocol in the (m,n)-hybrid model for testing between distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ with success probability $1/2+\gamma$, then either there is an ϵ-differentially private protocol for this task in the curator model that uses m samples and succeeds with probability at least $1/2+\gamma/4$ or there is an ϵ-differentially private protocol for this task in the local model with n agents that succeeds with probability at least $1/2+\gamma/4$.*

Proof. Let Π be a protocol guaranteed by the theorem, that is, when the inputs of the curator and the local agents are drawn from $\mathcal{D}_j$, the referee in Π returns j with probability at least $1/2+\gamma$. Consider an execution of the protocol when the inputs of the curator are drawn from $\mathcal{D}_0$ and the inputs of the local agents are drawn from $\mathcal{D}_1$ and let p be the probability that the referee in Π returns 1 in this case.

We first assume that $p \geq 1/2$ and show that there exists an ϵ-differentially private protocol Π^{local} for this task in the local model with n agents that succeeds with probability at least $1/2+\gamma/4$. The referee in protocol Π^{local} with probability $\gamma/2$ outputs 1 and with probability $1-\gamma/2$ draws m samples from $\mathcal{D}_0$, executes protocol Π, where the referee simulates the messages of the curator using the m samples, and returns the output of Π.

We next analyze this protocol. If the inputs of the local agents are drawn from $\mathcal{D}_1$, then the probability that the referee in protocol Π returns 1 is at least $1/2$ and the probability that the referee in Π^{local} returns 1 is at least $\gamma/2+(1-\gamma/2)\cdot 1/2 = 1/2+\gamma/4$. If the inputs of the local agents are drawn from $\mathcal{D}_0$, then the probability that the referee in Π^{local} returns 0 is at least $(1-\gamma/2)\cdot(1/2+\gamma) \geq 1/2+\gamma/4$ (since $\gamma \leq 1/2$). For the case that $p < 1/2$, it can be shown, using an analogous construction, that there exists an ϵ-differentially private protocol Π^{curator} for this task in the curator model with m samples that succeeds with probability at least $1/2+\gamma/4$. ◀

Recall that the *total variation distance* (also known as the statistical distance) of two discrete distributions $\mathcal{D}_0, \mathcal{D}_1$ over a domain X is $d_{\text{TV}}(\mathcal{D}_0,\mathcal{D}_1) = \sup_{T\subset X} |\mathcal{D}_1(T) - \mathcal{D}_0(T)| = \frac{1}{2}\sum_{x\in X} |\mathcal{D}_1(x) - \mathcal{D}_0(x)|$. The *squared Hellinger distance* between two distributions $\mathcal{D}_0, \mathcal{D}_1$ over a domain X is defined as $d_{H^2}(\mathcal{D}_0,\mathcal{D}_1) = \frac{1}{2}\sum_{x\in X}\left(\sqrt{\mathcal{D}_0(x)} - \sqrt{\mathcal{D}_1(x)}\right)^2$. For the rest of the discussion in this section, fix the domain $X = \{0,1\}$, and some $\alpha > 0$. We define two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ where under $\mathcal{D}_1$ we have $\Pr_{x\sim\mathcal{D}_1}[x=1] = \frac{1}{2}(1+\alpha)$ and under $\mathcal{D}_0$ we have $\Pr_{x\sim\mathcal{D}_0}[x=1] = \frac{1}{2}(1-\alpha)$. It is a fairly simple calculation to see that $d_{\text{TV}}(\mathcal{D}_0,\mathcal{D}_1) = \alpha$ and $\frac{\alpha^2}{2} \leq d_{H^2}(\mathcal{D}_0,\mathcal{D}_1) \leq \alpha^2$. We prove that for some setting of the parameters n and m, the hypothesis-testing task between $\mathcal{D}_0$ and $\mathcal{D}_1$ is impossible in the (m,n)-hybrid model.

The following Theorem follows by combining Theorem 6.2 with [40, Theorem 5.3] and with [19, Theorem 3.5]. See the full version of this paper for more details. [10]

▶ **Theorem 6.3.** *There are constants c_0, c_1 such that in the (m,n)-hybrid model, with $m \leq c_0\left(\frac{1}{\alpha^2} + \frac{1}{\epsilon\alpha}\right)$ and $n \leq c_1 \cdot \frac{1}{\epsilon^2\alpha^2}$, there is no ϵ-differentially private protocol that succeeds in determining whether all $m+n$ input points are drawn from $\mathcal{D}_0$ vs. $\mathcal{D}_1$ w.p. ≥ 0.75.*

References

1 Apple. Learning with privacy at scale. *Apple Machine Learning Journal*, 1, 2017. URL: https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html.

2 Brendan Avent, Aleksandra Korolova, David Zeber, Torgeir Hovden, and Benjamin Livshits. BLENDER: Enabling local search with a hybrid differential privacy model. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 747–764. USENIX Association, 2017. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/avent.

3 C.-A. Azencott. Machine learning and genomics: precision medicine versus patient privacy. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2128):20170350, 2018.

4 Mitali Bafna and Jonathan Ullman. The price of selection in differential privacy. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 151–168. PMLR, 2017.

5 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. Differentially private summation with multi-message shuffling. *CoRR*, abs/1906.09116, 2019. arXiv:1906.09116.

6 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. 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 638–667. Springer, 2019. doi:10.1007/978-3-030-26951-7_22.

7 Raef Bassily, Kobbi Nissim, Adam D. Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman. Algorithmic stability for adaptive data analysis. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 1046–1059, 2016. doi:10.1145/2897518.2897566.

8 Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 2288–2296, 2017. URL: http://papers.nips.cc/paper/6823-practical-locally-private-heavy-hitters.

9 Raef Bassily and Adam D. Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 127–135, 2015. doi:10.1145/2746539.2746632.

10 Amos Beimel, Aleksandra Korolova, Kobbi Nissim, Or Sheffet, and Uri Stemmer. The power of synergy in differential privacy: Combining a small curator with local randomizers. *CoRR*, abs/1912.08951, 2019. arXiv:1912.08951.

11 Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference*, pages 451–468, 2008. doi:10.1007/978-3-540-85174-5_25.

12 Amos Beimel, Kobbi Nissim, and Uri Stemmer. Characterizing the sample complexity of private learners. In *ITCS*, pages 97–110. ACM, 2013.

13 Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. *Theory of Computing*, 12(1):1–61, 2016. doi:10.4086/toc.2016.v012a001.

14 Bonnie Berger and Hyunghoon Cho. Emerging technologies towards enhancing privacy in genomic data sharing. *Genome Biology*, 20, 2019. URL: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1741-0.

15 Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 441–459. ACM, 2017. doi:10.1145/3132747.3132769.

16 Charlotte Bonte, Eleftheria Makri, Amin Ardeshirdavani, Jaak Simm, Yves Moreau, and Frederik Vercauteren. Towards practical privacy-preserving genome-wide association study. *BMC bioinformatics*, 19(1):537, 2018.

17 Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. In Jan Van den Bussche and Marcelo Arenas, editors, *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 435–447. ACM, 2018. doi:10.1145/3196959.3196981.

18 Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In *FOCS*, pages 634–649, 2015.

19 Clément L. Canonne, Gautam Kamath, Audra McMillan, Adam D. Smith, and Jonathan Ullman. The structure of optimal private tests for simple hypotheses. In *STOC*, pages 310–321, 2019.

20 Albert Cheu, Adam D. Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 375–403. Springer, 2019. doi:10.1007/978-3-030-17653-2_13.

21 Catalin Cimpanu. Capital One hacker took data from more than 30 companies, new court docs reveal. *ZDNet*, August 14, 2019. URL: https://www.zdnet.com/article/capital-one-hacker-took-data-from-more-than-30-companies-new-court-docs-reveal/.

22 Privacy Rights Clearinghouse. Data Breaches, 2019. URL: https://www.privacyrights.org/data-breaches.

23 Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3571–3580. Curran Associates, Inc., 2017. URL: http://papers.nips.cc/paper/6948-collecting-telemetry-data-privately.pdf.

24 Aryeh Dvoretzky, Jack Kiefer, and Jacob Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, 27(3):642–669, 1956.

25 Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Preserving statistical validity in adaptive data analysis. In *ACM Symposium on the Theory of Computing (STOC)*. ACM, June 2015.

26 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

27 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 *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 2468–2479. Society for Industrial and Applied Mathematics, 2019. URL: http://dl.acm.org/citation.cfm?id=3310435.3310586.

28 Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1054–1067, New York, NY, USA, 2014. ACM. doi:10.1145/2660267.2660348.

29 Vitaly Feldman and Thomas Steinke. Generalization for adaptively-chosen estimators via stable median. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017*, pages 728–757, 2017. URL: http://proceedings.mlr.press/v65/feldman17a.html.

30 Vitaly Feldman and David Xiao. Sample complexity bounds on differentially private learning via communication complexity. *SIAM J. Comput.*, 44(6):1740–1764, 2015. doi:10.1137/140991844.

31 Marco Gaboardi, Ryan Rogers, and Or Sheffet. Locally private mean estimation: Z-test and tight confidence intervals. *CoRR*, abs/1810.08054, 2018. arXiv:1810.08054.

32 Badih Ghazi, Rasmus Pagh, and Ameya Velingker. Scalable and differentially private distributed aggregation in the shuffled model. *CoRR*, abs/1906.08320, 2019. arXiv:1906.08320.

33 Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2009.

34 Google Research Blog. Tackling urban mobility with technology, Nov 18, 2015. URL: https://europe.googleblog.com/2015/11/tackling-urban-mobility-with-technology.html.

35 Andy Greenberg. Apple's differential privacy is about collecting your data – but not your data. In *Wired*, June 13, 2016.

36 Andy Greenberg. How one of Apple's key privacy safeguards falls short. *WIRED*, Sep 15, 2017. URL: https://www.wired.com/story/apple-differential-privacy-shortcomings/.

37 Alex Hern. Uber employees 'spied on ex-partners, politicians and Beyoncé'. *The Guardian*, Dec 13, 2016. URL: https://www.theguardian.com/technology/2016/dec/13/uber-employees-spying-ex-partners-politicians-beyonce.

38 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 239–248. IEEE Computer Society, 2006. doi:10.1109/FOCS.2006.25.

39 Aaron Johnson and Vitaly Shmatikov. Privacy-preserving data exploration in genome-wide association studies. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1079–1087. ACM, 2013.

40 Matthew Joseph, Jieming Mao, Seth Neel, and Aaron Roth. The role of interactivity in local differential privacy. *CoRR*, abs/1904.03564, 2019. arXiv:1904.03564.

41 Matthew Joseph, Jieming Mao, and Aaron Roth. Exponential separations in local differential privacy through communication complexity. *CoRR*, abs/1907.00813, 2019. arXiv:1907.00813.

42 Matthew Joseph, Aaron Roth, Jonathan Ullman, and Bo Waggoner. Local differential privacy for evolving data. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 2381–2390, 2018. URL: http://papers.nips.cc/paper/7505-local-differential-privacy-for-evolving-data.

43 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. doi:10.1137/090756090.

44 Aaron Mak. Do Tech Companies Really Need to Snoop Into Private Conversations to Improve Their A.I.? *Slate*, Feb 19, 2019. URL: https://slate.com/technology/2019/02/reverse-location-search-warrants-google-police.html.

45 Pascal Massart. The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. *The Annals of Probability*, pages 1269–1283, 1990.

46 Andrew McGregor, Ilya Mironov, Toniann Pitassi, Omer Reingold, Kunal Talwar, and Salil P. Vadhan. The limits of two-party differential privacy. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010*, pages 81–90. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.14.

47 Chris Merriman. Microsoft reminds privacy-concerned Windows 10 beta testers that they're volunteers. In *The Inquirer, http://www.theinquirer.net/2374302*, Oct 7, 2014.

48 Microsoft. Windows Insider Program Agreement, Sep 15, 2017. URL: https://insider.windows.com/en-us/program-agreement.

49 Ilya Mironov. Beyond software: Hardware-based security, May 8, 2019. URL: https://simons.berkeley.edu/talks/beyond-software-hardware-based-security.

50 Mozilla. Firefox Privacy Notice, June 4, 2019. URL: https://www.mozilla.org/en-US/privacy/firefox/#pre-release.

51 Kobbi Nissim and Uri Stemmer, 2017. Personal communication.

52 NYC Department of Transportation. New York City Mobility Report, 2016. URL: `http://www.nyc.gov/html/dot/downloads/pdf/mobility-report-2016-print.pdf`.

53 Phillip Rogaway. The moral character of cryptographic work. *IACR Cryptology ePrint Archive*, 2015:1162, 2015.

54 Stephen Shankland. How Google tricks itself to protect Chrome user privacy. In *CNET*, Oct 31, 2014.

55 Elaine Shi, T.-H. Hubert Chan, Eleanor G. Rieffel, and Dawn Song. Distributed private data analysis: Lower bounds and practical constructions. *ACM Trans. Algorithms*, 13(4):50:1–50:38, 2017. `doi:10.1145/3146549`.

56 Sean Simmons and Bonnie Berger. Realizing privacy preserving genome-wide association studies. *Bioinformatics*, 32(9):1293–1300, 2016.

57 Thomas Steinke and Jonathan Ullman. Tight lower bounds for differentially private selection. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 552–563. IEEE, 2017.

58 Uri Stemmer. Locally private k-means clustering. In *SODA*, pages 548–559. SIAM, 2020.

59 Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and XiaoFeng Wang. Privacy loss in Apple's implementation of differential privacy on MacOS 10.12. *arXiv preprint arXiv:1709.02753*, 2017. `arXiv:1709.02753`.

60 Jonathan Ullman. Tight lower bounds for locally differentially private selection. Technical Report abs/1802.02638, arXiv, 2018. `arXiv:1802.02638`.

61 Salil Vadhan. *The Complexity of Differential Privacy*. Unpublished manuscript, 2016. URL: `https://privacytools.seas.harvard.edu/publications/complexity-differential-privacy`.

62 L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984. `doi:10.1145/1968.1972`.

63 Giridhari Venkatadri, Elena Lucherini, Piotr Sapiezyński, and Alan Mislove. Investigating sources of PII used in Facebook's targeted advertising. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS'19)*, July 2019.

64 Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. Learning your identity and disease from research papers: information leaks in genome wide association study. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pages 534–544. ACM, 2009.

65 Stanley L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

66 Fei Yu and Zhanglong Ji. Scalable privacy-preserving data sharing methodology for genome-wide association studies: an application to idash healthcare privacy protection challenge. *BMC medical informatics and decision making*, 14(1):S3, 2014.

Pure Differentially Private Summation from Anonymous Messages

Badih Ghazi
Google Research, Mountain View, CA, USA
badihghazi@gmail.com

Noah Golowich
Google Research, Mountain View, CA, USA
MIT, Cambridge, MA, USA
nzg@mit.edu

Ravi Kumar
Google Research, Mountain View, CA, USA
ravi.k53@gmail.com

Pasin Manurangsi
Google Research, Mountain View, CA, USA
pasin30055@gmail.com

Rasmus Pagh
Google Research, Mountain View, CA, USA
IT University of Copenhagen, Denmark
Basic Algorithms Research Copenhagen, Denmark
pagh@itu.dk

Ameya Velingker
Google Research, Mountain View, CA, USA
ameyav@google.com

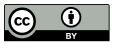
Abstract

The *shuffled* (aka *anonymous*) model has recently generated significant interest as a candidate distributed privacy framework with trust assumptions better than the central model but with achievable error rates smaller than the local model. In this paper, we study *pure* differentially private protocols in the shuffled model for *summation*, a very basic and widely used primitive. Specifically:

- For the binary summation problem where each of n users holds a bit as an input, we give a pure ϵ-differentially private protocol for estimating the number of ones held by the users up to an absolute error of $O_\epsilon(1)$, and where each user sends $O_\epsilon(\log n)$ one-bit messages. This is the first pure protocol in the shuffled model with error $o(\sqrt{n})$ for constant values of ϵ.
  Using our binary summation protocol as a building block, we give a pure ϵ-differentially private protocol that performs summation of real numbers in $[0, 1]$ up to an absolute error of $O_\epsilon(1)$, and where each user sends $O_\epsilon(\log^3 n)$ messages each consisting of $O(\log \log n)$ bits.
- In contrast, we show that for any pure ϵ-differentially private protocol for binary summation in the shuffled model having absolute error $n^{0.5-\Omega(1)}$, the per user communication has to be at least $\Omega_\epsilon(\sqrt{\log n})$ bits. This implies (i) the first separation between the (bounded-communication) multi-message shuffled model and the central model, and (ii) the first separation between pure and approximate differentially private protocols in the shuffled model.

Interestingly, over the course of proving our lower bound, we have to consider (a generalization of) the following question that might be of independent interest: given $\gamma \in (0, 1)$, what is the smallest positive integer m for which there exist two random variables X^0 and X^1 supported on $\{0, \ldots, m\}$ such that (i) the total variation distance between X^0 and X^1 is at least $1 - \gamma$, and (ii) the moment generating functions of X^0 and X^1 are within a constant factor of each other everywhere? We show that the answer to this question is $m = \Theta(\sqrt{\log(1/\gamma)})$.

2012 ACM Subject Classification Security and privacy → Privacy-preserving protocols; Mathematics of computing → Probabilistic algorithms; Theory of computation → Communication complexity



1st Conference on Information-Theoretic Cryptography (ITC 2020).
Editors: Yael Tauman Kalai, Adam D. Smith, and Daniel Wichs; Article No. 15; pp. 15:1–15:23

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

Keywords and phrases Pure differential privacy, Shuffled model, Anonymous messages, Summation, Communication bounds

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.15

Related Version A full version of the paper is available at `https://arxiv.org/abs/2002.01919`.

Funding *Noah Golowich*: Supported at MIT by a Fannie & John Hertz Foundation Fellowship, an MIT Akamai Fellowship, and an NSF Graduate Fellowship.

Acknowledgements We are grateful to Borja Balle, Kunal Talwar, and Vitaly Feldman for helpful discussions.

1 Introduction

Since its introduction by Dwork et al. [19, 18], *differential privacy (DP)* has become widely popular as a rigorous mathematical definition of privacy. This has led to practical deployments at companies such as Apple [28, 4], Google [22, 38], and Microsoft [17], and in government agencies such as the United States Census Bureau [2]. The most widely studied setting with DP is the so-called *central* model (denoted $\mathrm{DP}_{\mathrm{central}}$) where an analyzer observes the crude user data but is supposed to release a differentially private data structure. Many accurate private algorithms have been discovered in the central model; however, the model is limited when the analyst is not to be trusted with the user data. To remedy this, the more appealing *local* model of DP (denoted $\mathrm{DP}_{\mathrm{local}}$) [33] (also [41]) requires the messages sent by each user to the analyst to be private. Nevertheless, the local model suffers from large estimation errors that are known to be on the order of $\sqrt{n}$, where n is the number of users, for a variety of problems including summation, the focus of this work [10, 14]. This has motivated the study of the *shuffled* model of DP (denoted $\mathrm{DP}_{\mathrm{shuffled}}$), which is intended as a middle-ground with trust assumptions better than those of the central model and estimation accuracy better than the local model.

While an analogous setup was first introduced in crytpography by Ishai et al. [31] in their work on cryptography from anonymity, the shuffled model was first proposed for privacy-preserving computations by Bittau et al. [11] in their Encode-Shuffle-Analyze architecture. In this setup, each user sends (potentially several) messages to a trusted shuffler, who randomly permutes all incoming messages before passing them to the analyst. We will treat the shuffler as a black box in this work, though we point out that various efficient cryptographic implementations of the shuffler have been considered, including onion routing, mixnets, third-party servers, and secure hardware (see, e.g., the discussions in [31, 11]). The privacy properties of $\mathrm{DP}_{\mathrm{shuffled}}$ were first studied, independently, by Erlingsson et al. [21] and Cheu et al. [15]. Moreover, several recent works have sought to nail down the trade-offs between accuracy, privacy, and communication [15, 8, 27, 6, 25, 26, 7, 5, 20, 9].

Pure- and Approximate-DP

The two most widely used notions of DP are pure-DP [19] and approximate-DP [18], which we recall next. For any parameters $\epsilon \geq 0$ and $\delta \in [0, 1]$, a randomized algorithm P is (ϵ, δ)-*DP* if for every pair datasets X, X' differing on a single user's data, and for every subset $\mathcal{S}$ of transcripts of P, it is the case that

$$\Pr[P(X) \in \mathcal{S}] \leq e^{\epsilon} \cdot \Pr[P(X') \in \mathcal{S}] + \delta, \tag{1}$$

where the probabilities are taken over the randomness in P. The notion of ϵ-DP is the special case where δ is set to 0 in (1); we use the terms *pure-DP* when $\delta = 0$ and *approximate-DP* when $\delta > 0$. While δ is intuitively an upper bound on the probability that an (ϵ, δ)-DP algorithm fails to be ϵ-DP, this failure event can in principle be catastrophic, revealing all the user inputs to the analyst. Pure-DP protocols are thus highly desirable as they guarantee more stringent protections against the leakage of user data. In the central and local settings, several prior works either obtained pure protocols in regimes where approximate protocols were previously known, or proved separations between pure and approximate protocols (e.g., [30, 16, 37, 39, 13]).

Summation

A basic primitive in data analytics and machine learning is the *summation* (aka *aggregation*) of inputs held by different users. Indeed, private summation is a critical building block in the emerging area of *federated learning* [34], where a machine learning model, say a neural network, is to be trained on data held by many users without having the users send their data over to a central analyzer (see [32] for a recent extensive overview). To do so, private variants of Stochastic Gradient Descent have been developed and their privacy/accuracy trade-offs analyzed (e.g., [1]). The gist of these procedures is the private summation of users' gradient updates. Private summation is also closely related to functions in the widely studied class of *counting queries* (e.g., [40, 12, 30, 29, 37]).

Several recent work studied approximate-$\mathrm{DP}_{\mathrm{shuffled}}$ protocols for summation [15, 8, 27, 6, 26, 7, 5]. For binary summation, Cheu et al. [15] show that the standard randomized response is an (ϵ, δ)-$\mathrm{DP}_{\mathrm{shuffled}}$ protocol for binary summation and that it incurs an absolute error of only $O(\sqrt{\log n})$ for ϵ constant and δ inverse polynomial in n. For real summation in the single-message shuffled model (denoted $\mathrm{DP}^{1}_{\mathrm{shuffled}}$), where each user sends a single message to the shuffler, Balle et al. [8] show that the tight error for approximate protocols is $\Theta(n^{1/6})$. For real summation in the multi-message shuffled model (denoted $\mathrm{DP}^{\geq 1}_{\mathrm{shuffled}}$), where a user can send more than one message, the state-of-the-art approximate protocol was recently obtained in [26, 7][1] and it incurs error at most $O(1/\epsilon)$ with every user sending $O(1 + \frac{\log(1/\delta)}{\log n})$ messages of $O(\log n)$ bits each.

The aforementioned protocols, along with several other results (including the work on "privacy amplification by shuffling" of Erlingsson et al. [21] and Balle et al. [8]), demonstrate the power of the shuffled model over the local model in terms of privacy, as any $(\epsilon, o(1/n))$-$\mathrm{DP}_{\mathrm{local}}$ summation protocol must incur an error of $\Omega_\epsilon(\sqrt{n})$ [14]. However, all of the protocols proposed so far in the shuffled model only achieve an advantage over the local model when allowed approximation. This leads us to the following basic and perplexing question that is the focus of our work:

▶ **Question 1.** *Are there pure-*$\mathrm{DP}_{\mathrm{shuffled}}$ *protocols that achieve better utility than any* $\mathrm{DP}_{\mathrm{local}}$ *protocol?*

1.1 Main Results

We positively answer the above question for the problem of summation. Namely, we give the first pure-$\mathrm{DP}_{\mathrm{shuffled}}$ protocol for binary summation with error depending only on ϵ but independent of n and with logarithmic communication per user.

[1] See also [9] (merger of [7, 6]) together with a novel recursive protocol with $poly(\log\log n)$ error where each user sends $O(\log\log n)$ messages.

▶ **Theorem 2 (Pure Binary Summation via Shuffling).** *For every positive real number ϵ, there is a (non-interactive) ϵ-DP$_{\text{shuffled}}$ protocol for binary summation that has expected error $O_\epsilon(1)$ and where each user sends $O_\epsilon(\log n)$ messages each consisting of a single bit.*

We use the protocol in Theorem 2 as a building block in order to also obtain a protocol with constant error and polylogarithmic communication per user for the more general task of real summation where each user input is a real number in $[0, 1]$.

▶ **Theorem 3 (Pure Real Summation via Shuffling).** *For every positive real number ϵ, there is a (non-interactive) ϵ-DP$_{\text{shuffled}}$ protocol for real summation that has expected error $O_\epsilon(1)$ and where each user sends $O_\epsilon(\log^3 n)$ messages each consisting of $O(\log \log n)$ bits.*

In light of Theorem 2, a natural question is if there is a (non-interactive) pure-DP protocol for binary summation with logarithmic (or even constant) error and constant communication per user, as in the approximate case. We show that no such protocol exists, even for very large (polynomial) errors:

▶ **Theorem 4 (Communication Lower Bound).** *In any non-interactive ϵ-DP$_{\text{shuffled}}$ protocol for binary summation with expected error at most $n^{0.5-\Omega(1)}$, the worst-case per user communication must be $\Omega_\epsilon(\sqrt{\log n})$ bits.*

1.2 Implications

Our results described above imply new separations between different types of DP protocols (e.g., DP$_{\text{central}}$, DP$_{\text{local}}$, DP$^1_{\text{shuffled}}$, and DP$^{\geq 1}_{\text{shuffled}}$), and also give the first accurate pure-DP$_{\text{shuffled}}$ protocol for histograms. We elaborate on these next.

Pure Local vs Shuffled Protocols

In DP$_{\text{local}}$, the tight accuracy for binary summation is known to be $\Theta(\sqrt{n})$ for approximate protocols [41, 10, 14]. Our Theorems 2 and 3 give the first pure-DP$_{\text{shuffled}}$ protocols with error $o(\sqrt{n})$ for binary and real summation respectively, and in fact they only incur constant error for both of these problems. Furthermore, Bun et al. [13] gave a generic transformation from any sequentially interactive approximate-DP$_{\text{local}}$ protocol to a pure-DP$_{\text{local}}$ protocol with essentially the same accuracy and each user communicates only $O(\log \log n)$ bits. In contrast, our Theorem 4 implies that in any such transformation in the shuffled model (if one exists), the per user communication has to be $\Omega(\sqrt{\log n})$.

Pure vs Approximate Shuffled Protocols

Cheu et al. [15] showed that the standard randomized response [41] is an approximate-DP protocol for binary summation that incurs only logarithmic error (for ϵ constant and δ inverse polynomial in n), and where each user sends a single bit. In contrast, our Theorem 4 implies that the communication cost of any pure-DP protocol for binary summation with logarithmic error (and in fact with error as large as $n^{0.5-\Omega(1)}$) is $\Omega(\sqrt{\log n})$ bits. Put together, these two results imply the first separation between the communication complexity of pure-DP$_{\text{shuffled}}$ and approximate-DP$_{\text{shuffled}}$ protocols.

Pure Single-Message vs Multi-Message Shuffled Protocols

As recently shown by [5], any pure-DP$^1_{\text{shuffled}}$ protocol implies a pure-DP$_{\text{local}}$ protocol with the same accuracy. This implies that any pure-DP$^1_{\text{shuffled}}$ protocol for binary summation must incur error $\Omega_\epsilon(\sqrt{n})$. Our Theorem 2 thus implies a huge separation of $\Theta_\epsilon(\sqrt{n})$ between the errors possible for pure-DP$^1_{\text{shuffled}}$ and pure-DP$^{\geq 1}_{\text{shuffled}}$ protocols.

Multi-Message Shuffled vs Central Protocols

It is well-known that the tight error for binary summation in $\mathrm{DP}_{\mathrm{central}}$ is $O(1/\epsilon)$ [19]. Theorem 4 proves that any $\mathrm{DP}_{\mathrm{shuffled}}$ protocol with per user communication $o_\epsilon(\sqrt{\log n})$ bits must incur error $n^{0.5-\Omega(1)}$. It thereby gives the first separation between (bounded-communication) $\mathrm{DP}^{\geq 1}_{\mathrm{shuffled}}$ and $\mathrm{DP}_{\mathrm{central}}$ protocols. Indeed, this is, to the best of our knowledge, the first separation between the accuracy of (bounded-communication) $\mathrm{DP}^{\geq 1}_{\mathrm{shuffled}}$ protocols and those of $\mathrm{DP}_{\mathrm{central}}$ protocols with the same privacy parameters; all previous lower bounds for $\mathrm{DP}_{\mathrm{shuffled}}$ [15, 8, 25] only apply to single-message protocols.

Pure Protocol for Histograms

Our pure binary summation protocol (Theorem 2) implies as a black-box the first pure-DP protocol with polylogarithmic error for computing *histograms* (aka *point functions* or *frequency estimation*), albeit with very large communication (see Appendix A of the full version [24] for more details). It remains a very interesting open question to obtain a communication-efficient and accurate pure-DP protocol for histograms (see Section 5 for more on this and other open questions).

1.3 Overview of Techniques

Binary Summation Protocol

We first explain why all existing summation protocols in the shuffled model with error $o(\sqrt{n})$ are not $O(1)$-DP. First, note that as observed by [5], any pure-$\mathrm{DP}^1_{\mathrm{shuffled}}$ protocol implies a pure-$\mathrm{DP}_{\mathrm{local}}$ protocol with the same accuracy and privacy. Combined with the fact that any $O(1)$-$\mathrm{DP}_{\mathrm{local}}$ protocol for summation must have error $\Omega(\sqrt{n})$, this implies the same lower bound for any pure $O(1)$-$\mathrm{DP}^1_{\mathrm{shuffled}}$ protocol. In particular, this rules out the binary randomized response [41] that was analyzed in the shuffled model by [15]. It also rules out the protocol implied by shuffling RAPPOR [22], and more generally any protocol obtained by the amplification via shuffling approach of [21, 8]. Moreover, in the multi-message shuffled setup, the state-of-the-art real summation protocols of [26, 7], which rely on the Split-and-Mix procedure [31], only give approximate-DP.

A different $\mathrm{DP}^{\geq 1}_{\mathrm{shuffled}}$ protocol for binary summation can be obtained by instantiating the recent $\mathrm{DP}^{\geq 1}_{\mathrm{shuffled}}$ protocols for computing histograms [25], with a domain size of $B = 2$. On a high-level, the two resulting protocols – one of which is based on the Count Min sketch and the other on the Hadamard response – can be seen as special cases of the following common template: each user (i) samples a number ρ of messages that depend on their input, (ii) independently samples a number η of noise messages, and (iii) sends these $\rho + \eta$ messages to the shuffler. Loosely, the analyzer then outputs the number of messages "consistent with" the queried input. However, it can be seen that any protocol following this template will not be pure-DP, as the supports of the distribution of the count observed at the analyzer can shift by 1 when a single user input is changed. The crucial insight in our pure protocol for binary summation will be to *correlate* the input-dependent messages and the noise messages sampled by each user in steps (i) and (ii) above. By doing so, we not only aim to ensure that the supports are identical but that the two densities are also within a small multiplicative factor on any point. We implement this idea using binary messages by having each user send d bits on both inputs 0 and 1. Specifically, the user will start by flipping a suitably biased coin. If it lands as head, the user will send $(d+1)/2$ zeros and $(d-1)/2$ ones when the input is 0, and vice versa when the input is 1. If the coin lands as tail, the user will sample an integer z from

Algorithm 1 Randomizer for binary summation.

procedure BINARYRANDOMIZER$_{\epsilon,n}(x)$
  Let p, d, s be as in Lemma 9 (depending on ϵ, n)
  $a \leftarrow \mathrm{Ber}(p)$
  if $a = 0$ **then**
    if $x = 0$ **then**
      return the multiset with $\left(\frac{d-1}{2}\right)$ ones and $\left(\frac{d+1}{2}\right)$ zeros
    else
      return the multiset with $\left(\frac{d+1}{2}\right)$ ones and $\left(\frac{d-1}{2}\right)$ zeros
  else
    $z \leftarrow \mathrm{DLap}_d(d/2, s)$
    return the multiset with z ones and $(d - z)$ zeros

Algorithm 2 Analyzer for binary summation.

procedure BINARYANALYZER$_{\epsilon,n}(R)$
  Let d be as in Lemma 9 (depending on ϵ, n)
  return $\frac{nd}{2} + \sum_{y \in R} \left(y - \frac{1}{2}\right)$

a truncated discrete Laplace distribution and send z zeros and $d - z$ ones (see Algorithm 1 and Equation (2) for more details). The analyzer (Algorithm 2) then outputs the number of received ones after debiasing. Note that the number of ones received by the analyzer is a random variable taking values between 0 and dn inclusive. To prove that the algorithm is private, we intuitively wish to argue that the noise distribution satisfies the property that its density values on any two adjacent points are within a multiplicative e^ϵ factor. However, the technical challenge stems from the fact that this noise distribution depends on the specific input sequence (and as we discussed above this dependence is necessary!). Instead, we have to analyze the n-fold convolution of the individual responses, and show that the density values of the resulting distribution on any two adjacent points in $\{0, 1, \ldots, dn\}$ are within a multiplicative factor of e^ϵ, for any input sequence. The crux of the proof is to relate the tails of different convolutions of the truncated discrete Laplace distribution (Lemmas 10 and 11). We determine a setting of (i) the mixture probability coefficient (denoted by p in Algorithm 1), (ii) the parameter d, and (iii) the "inverse scaling coefficient" of the truncated discrete Laplace distribution (denoted by s in Algorithm 1), for which the privacy property holds and for which the resulting expected absolute error is $O_\epsilon(1)$.

We point out that the dependence of the error on ϵ that we obtain is $\tilde{O}(1/\epsilon^{3/2})$ for $\epsilon \leq O(1)$ (see Theorem 8). An interesting open question is whether this dependence can be further reduced to $O(1/\epsilon)$, which is the tight error in the central model [19].

Real Summation Protocol

We use our pure private binary summation protocol outlined above as a building block in order to obtain a pure private real summation protocol and prove Theorem 3. We note that Cheu et al. [15] had given a transformation from binary summation to real summation, but their reduction results in a protocol with a very large communication of $\tilde{\Omega}(\sqrt{n})$ bits in order to achieve logarithmic error. We instead give a (different) transformation that results in a protocol with polylogarithmic communication. The high-level idea of our reduction is the following: consider the binary representation of the inputs after rounding them to $O(\log n)$

Algorithm 3 Randomizer for real summation.

procedure RealRandomizer$_{(\epsilon_j)_{j\in\mathbb{N}},n}(x)$
  for $j = 1$ **to** $2\log n$ **do**
    $x[j] \leftarrow j$th most significant bit of x
    $S_j \leftarrow$BinaryRandomizer$_{\epsilon_j,n}(x[j])$ — S_j is a multiset of zeros and ones.
    $R_j \leftarrow \{j\} \times S_j$ — R_j is a multiset of tuples $(j, 0)$ and $(j, 1)$.
  return $\bigcup_{j=1}^{2\log n} R_j$

Algorithm 4 Analyzer for real summation.

procedure RealAnalyzer$_{(\epsilon_j)_{j\in\mathbb{N}},n}(R)$
  for $j = 1$ **to** $2\log_2 n$ **do**
    $R_j \leftarrow \{y_1 \mid y \in R \text{ and } y_0 = j\}$ — Multiset of bit messages for the jth bit.
    $a_j \leftarrow$BinaryAnalyzer$_{\epsilon_j,n}(R_j)$
  return $\sum_{j=1}^{2\log n} a_j/2^j$

bits of precision, then approximate the sum for each bit position independently, and finally combine the estimates into an approximation of the (real-valued) sum of the inputs. Since the bit sum estimates have geometrically decreasing weights, we can afford to increase the error on less significant bits. In terms of privacy, this means that for the jth most significant bit, we run an ϵ_j-DP binary summation protocol where $\epsilon_1, \epsilon_2, \ldots$ is a decreasing sequence. The protocol is illustrated in Algorithms 3 and 4. By carefully choosing the sequence $\epsilon_1, \epsilon_2, \ldots$, we can ensure that the total pure privacy parameter $\sum_j \epsilon_j$ is small, while the total error is a constant times the error for the sum of the most significant bits of the inputs. Intuitively, choosing $\epsilon_1, \epsilon_2, \ldots$ to be a geometrically decreasing sequence (e.g., $\epsilon_j = \frac{0.9^j \cdot \epsilon}{10}$) should suffice for our purposes. However since the communication complexity of our binary summation protocol also depends on the privacy parameter ϵ, such a choice of the sequence would result in $poly(n)$ communication complexity. To overcome this, our actual sequence has a "cut-off" so that the ϵ_j's do not go below $\Theta\left(\frac{\epsilon}{\log n}\right)$. This completes the proof overview.

Lower Bound

We next outline the proof of Theorem 4. Without loss of generality, we consider an arbitrary ϵ-DP$_{\text{shuffled}}$ protocol performing binary summation with error $n^{0.5-\Omega(1)}$, and where every user sends m messages each belonging to the domain $\{1, \ldots, k\}$. We wish to lower bound the number of bits of communication per user in this protocol, which is equal to $m \log k$. We denote by $\mathbf{X}^0$ and $\mathbf{X}^1$ the random multisets of messages sent by a user in this protocol under inputs 0 and 1 respectively. Note that $\mathbf{X}^0$ and $\mathbf{X}^1$ are supported on the set $\Delta_{k,m} := \{(z_1, \ldots, z_k) \in \mathbb{Z}^k_{\geq 0} \mid z_1 + \cdots + z_k = m\}$. Here, z_i captures the number of i messages sent by the user for each $i \in \{1, \ldots, k\}$.

Using the pure privacy of the protocol, we can argue that the ratio of the moment generating functions (MGFs) of $\mathbf{X}^0$ and $\mathbf{X}^1$ cannot take a very large or a very small value. Specifically, using the fact that the MGF of a sum of independent random variables is equal to the product of the individual MGFs, we derive a simple yet powerful property that should be satisfied by any ϵ-DP protocol in the shuffled model: the ratio of the MGFs of $\mathbf{X}^0$ and $\mathbf{X}^1$ should always lie in the interval $[e^{-\epsilon}, e^{\epsilon}]$. We will refer to such random variables as having an e^{ϵ}-*bounded MGF ratio* (see Section 4.1 for more details). We remark that while MGFs

have been used before in DP by Abadi et al. [1] and subsequent works on Renyi DP (starting from [36]), these usages are in a completely different context compared to ours. In particular, these prior works keep track of the moments in order to bound the privacy parameters under composition of protocols. To the best of our knowledge, MGFs have neither been used in lower bounds for DP nor in the shuffled model before.

Then, using the accuracy of the protocol, we can deduce that the total variation distance between $\mathbf{X}^0$ and $\mathbf{X}^1$ has to be large. We do so by invoking a result from the literature [14, 25] showing that for any binary summation protocol that incurs an absolute error of α, the total variation distance between $\mathbf{X}^0$ and $\mathbf{X}^1$ must be at least $1 - \Theta(\alpha/\sqrt{n})$ (see Theorem 16 for more details). Since $\alpha = n^{0.5-\Omega(1)}$ in our case, we get a lower bound of $1 - n^{-\Omega(1)}$ on the total variation distance between $\mathbf{X}^0$ and $\mathbf{X}^1$.

Equipped with these two ingredients, the task of lower bounding the per user communication cost of the protocol reduces to lower bounding the following quantity:

▶ **Definition 5.** *Given parameters $\epsilon > 0$ and $\gamma \in [0, 1]$, we define $C_{\epsilon,\gamma}$ as the minimum value of $m \log k$ for which there exist two random variables supported on $\Delta_{k,m}$ that are at total variation distance is at least $1 - \gamma$ but that have an e^ϵ-bounded MGF ratio.*

Note that any lower bound on the value of $C_{\epsilon,\gamma}$ can be used to infer a lower bound on the per user communication cost. In order to prove Theorem 4, and given our setting of $\gamma = 1/n^{\Omega(1)}$, it is thus enough for us to show that $C_{\epsilon,\gamma} \geq \Omega_\epsilon(\sqrt{\log(1/\gamma)})$. To prove this bound, it suffices to show that if two random variables $\mathbf{X}^0, \mathbf{X}^1$ have an e^ϵ-bounded MGF ratio, then their total variation distance must be at least $1 - \exp(O_\epsilon(m^2 \log k))$. For each $\mathbf{x} \in \Delta_{k,m}$, we view $\Pr[\mathbf{X}^0 = \mathbf{x}]$ and $\Pr[\mathbf{X}^1 = \mathbf{x}]$ as variables. The e^ϵ-bounded MGF ratio constraints can then be written as infinitely many linear inequalities over these variables. Moreover, the total variation distance between $\mathbf{X}^0$ and $\mathbf{X}^1$ can be written as a maximum of linear combinations of these same variables. We therefore get a linear program with infinitely many constraints, and we would like to show that any solution to it has "cost" (i.e., total variation distance) at least $1 - \exp(O_\epsilon(m^2 \log k))$. We do so by giving a dual solution with cost at most $1 - \exp(O_\epsilon(m^2 \log k))$, which by weak duality implies our desired bound (see Section 4 for more details).

A natural question is if the lower bound $C_{\epsilon,\gamma} \geq \Omega_\epsilon(\sqrt{\log(1/\gamma)})$ outlined above can be improved, as that would immediately lead to an improved communication complexity lower bound. However, we show that the lower bound is tight, even in the special case where $k = 2$. Namely, we give two random variables supported on $\Delta_{2,m}$ with $m = \Theta_\epsilon(\sqrt{\log(1/\gamma)})$ that are at total variation distance at least $1 - \gamma$ but that have an e^ϵ-bounded MGF ratio. Our construction is based on truncations of discrete Gaussian random variables (see Section 4.3 for more details). We note that this limitation only applies to the approach of lower bounding the per user communication complexity via lower bounding $C_{\epsilon,\gamma}$. It remains possible that other approaches might give better lower bounds. For instance, one might be able to proceed by giving a necessary condition for the accuracy of binary summation protocols that is stronger than the total variation distance bound that we used, or a necessary condition for pure privacy that is better than our e^ϵ-bounded MGF ratio property.

1.4 Organization

We start with some notation and background in Section 2. Our protocol for binary summation is presented and analyzed in Section 3. In Section 4, we prove our lower bound (Theorem 4). Any deferred proofs from these sections appear in the full version of the paper [24]. We conclude with some interesting open questions in Section 5. Our full proof for real summation (Theorem 3) and our corollary for histograms are deferred to the full version [24].

2 Preliminaries

Shuffled Model of Privacy

Let n be the number of users and let $\mathcal{X}$ be the domain. For each i in $[n] := \{1, \ldots, n\}$, we denote by x_i the input held by the ith user, and further assume that $x_i \in \mathcal{X}$. In the binary summation case, we have that $\mathcal{X} = \{0, 1\}$ while in the real summation case, we let $\mathcal{X}$ be the set $[0, 1]$ of real numbers. A protocol $P = (R, S, A)$ in the shuffled model consists of three algorithms: (i) the *local randomizer* $R(\cdot)$ whose input is the data of one user and whose output is a sequence of messages, (ii) the *shuffler* $S(\cdot)$ whose input is the concatenation of the outputs of the local randomizers and whose output is a uniform random permutation of its inputs, and (iii) the *analyzer* $A(\cdot)$ whose input is the output of the shuffler and whose output is the output of the protocol. The privacy in the shuffled model is guaranteed with respect to the input to the analyzer, i.e., the output of the shuffler.

▶ **Definition 6** (DP in the shuffled model, [21, 15]). *A protocol $P = (R, S, A)$ is (ϵ, δ)-$\mathrm{DP}_{\mathrm{shuffled}}$ if, for any dataset $X = (x_1, \ldots, x_n)$, the algorithm $S(R(x_1), \ldots, R(x_n))$ is (ϵ, δ)-DP. In the special case where $\delta = 0$, we say that the protocol P is ϵ-$\mathrm{DP}_{\mathrm{shuffled}}$.*

Note that the $\mathrm{DP}_{\mathrm{local}}$ model corresponds to the case where S is the identity function.

▶ **Definition 7** (Non-Interactive Protocols). *Let k and m be positive integers. In a* non-interactive *(aka* one-round*) protocol, each of the n users (i.e., randomizers) receives an input b and outputs at most m messages each consisting of $\log k$ bits, according to a certain distribution (depending on b), and using private randomness. We say that such a protocol has a* communication complexity *of $m \log k$.*

It is often convenient to view each message as a number in $[k]$. We use $\mathbf{X}^b \in \mathbb{Z}^k_{\geq 0}$ to denote the random variable whose sth coordinate X^b_s denotes the number of s-messages output by the randomizer on input b. Note that it is always the case that $\sum_{s \in [k]} X^b_s = m$, i.e., $\mathrm{supp}(\mathbf{X}^b) \subseteq \Delta_{k,m} := \{(z_1, \ldots, z_k) \in \mathbb{Z}^k_{\geq 0} \mid z_1 + \cdots + z_k = m\}$.

3 Pure Binary Summation Protocol via Shuffling

In this section we prove Theorem 2, restated formally below.

▶ **Theorem 8.** *For every sufficiently large n and $O(1) \geq \epsilon > 1/n^{2/3}$, there is an ϵ-$\mathrm{DP}_{\mathrm{shuffled}}$ protocol for summation for inputs $x_1, \ldots, x_n \in \{0, 1\}$ where each user sends $O\left(\frac{\log n}{\epsilon}\right)$ one-bit messages to the analyzer and has expected error at most $O\left(\frac{\sqrt{\log 1/\epsilon}}{\epsilon^{3/2}}\right)$.*

We remark that the assumption $\epsilon > \frac{1}{n^{2/3}}$ is made w.l.o.g., because, for $\epsilon \leq \frac{1}{n^{2/3}}$, there is a trivial algorithm that achieves square error of $O(1/\epsilon^{3/2})$: the analyzer just always outputs 0.

Throughout this section we assume that for some absolute constant C, $\epsilon \leq C$, and thus in particular e^ϵ can be bounded above by an absolute constant. (The constant C can be arbitrary.) It is well-known that any ϵ-$\mathrm{DP}_{\mathrm{central}}$ protocol for summation has error $\Omega(1/\epsilon)$ [40]. Thus the error in Theorem 8 is suboptimal by a factor of at most $\tilde{O}(1/\sqrt{\epsilon})$.

The remainder of the section is organized as follows. In Section 3.1, we present the protocol used to prove Theorem 8. In Section 3.2, we prove the accuracy and privacy guarantees of Theorem 8, and in Section 3.3 we outline the proof of a technical lemma needed in the privacy analysis.

3.1 The Protocol

To described the protocol, we will use the truncated version of the discrete Laplace distribution, for which we condition the support to be on $[\mu - w/2, \mu + w/2]$ where $w \geq 1$ is the "width" of the support. We denote such a distribution by $\mathrm{DLap}_w(\mu, s)$. More specifically, its probability mass function satisfies

$$\Pr_{Z \sim \mathrm{DLap}_w(\mu,s)}[Z = z] = \begin{cases} \frac{1}{C_w(\mu,s)} \cdot e^{-|z-\mu|/s} & \text{if } \mu - w/2 \leq z \leq \mu + w/2 \text{ and } z \in \mathbb{Z} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Here $C_w(\mu, s) = \sum_{z \in [\mu - w/2, \mu + w/2] \cap \mathbb{Z}} e^{-|z-\mu|/s}$ is simply the normalization factor.

Our randomizer and analyzer are presented in Algorithm 1 and Algorithm 2, respectively. The protocol has 3 parameters: the number of messages d, the "inverse scaling exponent" s, and the "noise probability" p. We always assume that d is a positive odd integer[2]. These parameters will be chosen later (in Lemma 9).

3.2 Privacy Analysis

For $b \in \{0, 1\}$, we write $\mathcal{R}_b$ to denote the distribution[3] on the number of ones output by the randomizer on input b. (This distribution depends on d, s, p but we do not include them in the notation to avoid being cumbersome.) Notice that we can decompose $\mathcal{R}_b$ as a mixture $p \cdot \mathrm{DLap}_d(d/2, s) + (1 - p) \cdot \mathbf{1}(\frac{d-1}{2} + b)$, where we use $\mathbf{1}(\vartheta)$ to denote the distribution that is ϑ with probability 1.

To prove the privacy guarantee of Theorem 8, we first note that we may focus only on the neighboring datasets $(0, \ldots, 0, 0)$ and $(0, \ldots, 0, 1)$; this follows since we may assume (due to symmetry) that more than half of the bits are zero and we can then condition out the results from the 1 bits that they share. (See the proof of Theorem 8 for a formalization of this.) For these datasets, Lemma 9 below bounds the ratio of the probabilities of ending up with a particular union of outputs from these two datasets.

▶ **Lemma 9.** *There is a sufficiently small constant $c_0 \in (0, 1)$ so that the following holds. For any sufficiently large $n \in \mathbb{N}$ and any $c_0 \geq \epsilon > \frac{1}{n^{2/3}}$, let $s = \frac{10}{\epsilon}$, $p = \frac{100\, e^{100\epsilon} \log(1/(1-e^{-0.1\epsilon}))}{n(1-e^{-0.1\epsilon})}$, and $d = 4\left\lceil \frac{1000\, e^{100\epsilon}}{(1-e^{-0.1\epsilon})} \cdot \log\left(\frac{n}{1-e^{-0.1\epsilon}}\right)\right\rceil + 3$. Then, for all $t \in \{0, \ldots, dn\}$, we have*

$$\frac{\Pr_{Z_1,\ldots,Z_n \sim \mathcal{R}_0}[Z_1 + \cdots + Z_n = t]}{\Pr_{Z_1,\ldots,Z_{n-1} \sim \mathcal{R}_0, Z_n \sim \mathcal{R}_1}[Z_1 + \cdots + Z_n = t]} \in [e^{-\epsilon}, e^{\epsilon}]. \tag{3}$$

This means that, for the above selection of parameters, the protocol is ϵ-DP. Using Lemma 9, we prove Theorem 8.

Proof of Theorem 8. We may assume without loss of generality that $\epsilon \leq c_0$, as otherwise we may set ϵ to $\min\{\epsilon, c_0\}$ instead.

We use the local randomizer $\textsc{BinaryRandomizer}_{\epsilon,n}$ of Algorithm 1 and the analyzer $\textsc{BinaryAnalyzer}_{\epsilon,n}$ of Algorithm 2, with the parameters s, d, p given by the expressions in Lemma 9, except with n replaced by $\lceil (n+1)/2 \rceil$. Explicitly, we have $s = \frac{10}{\epsilon}$, $p =$

[2] We only assume that d is odd for convenience, so that $\left(\frac{d-1}{2}\right)$ and $\left(\frac{d+1}{2}\right)$ are integers. Using an even d and replacing these two quantities with $d/2 - 1$ and $d/2 + 1$ also works, provided that the proofs are adjusted appropriately.

[3] This is the distribution of $\mathbf{X}^b$ defined in Section 2.

$\frac{100\,e^{100\epsilon}\log(1/(1-e^{-0.1\epsilon}))}{\lceil (n+1)/2\rceil(1-e^{-0.1\epsilon})}$ and $d = 4\left\lceil \frac{1000\,e^{100\epsilon}}{(1-e^{-0.1\epsilon})}\cdot\log\left(\frac{\lceil (n+1)/2\rceil}{1-e^{-0.1\epsilon}}\right)\right\rceil + 3$. We prove the accuracy guarantee first, which is a simple consequence of the choices of p, d made in Lemma 9, followed by the privacy guarantee, which uses Lemma 9.

Proof of accuracy

Fix a dataset $X = (x_1, \ldots, x_n) \in \{0,1\}^n$. Let $Y \in \mathbb{R}$ be the count released by the analyzer. Moreover, for $1 \le i \le n$, let $Z_1, \ldots, Z_n$ be i.i.d. random variables distributed according to $\nu = \mathrm{DLap}_d(d/2, s)$. It is easy to check that $\mathrm{Var}[Z_i] \le 2s^2$. Moreover, let $M \in [n]$ be the number of users for whom the Bernoulli random variable a is equal to 1. In particular, $M \sim \mathrm{Binom}(n, p)$. The expected absolute error is given by

$$\begin{aligned}
\mathbb{E}\left[\left|Y - \sum_{i=1}^{n} x_i\right|\right] &\le \sum_{m=0}^{n} \Pr[M = m]\cdot\left(\frac{m}{2} + \mathbb{E}\left[\left|Z_1 + \cdots + Z_m - \frac{md}{2}\right|\right]\right) \\
\text{(by Jensen's inequality)} \quad &\le \mathbb{E}[M/2] + \sum_{m=0}^{n}\Pr[M=m]\cdot\sqrt{\mathbb{E}\left[\left(Z_1+\cdots+Z_m-\frac{md}{2}\right)^2\right]} \\
&\le pn/2 + \sum_{m=0}^{n}\Pr[M=m]\cdot\sqrt{m}\cdot\sqrt{2s^2} \\
&\le pn/2 + \sqrt{2}s\cdot\sqrt{pn}. \qquad (4)
\end{aligned}$$

Since $p = \frac{100\,e^{100\epsilon}\log(1/(1-e^{-0.1\epsilon}))}{\lceil (n+1)/2\rceil(1-e^{-0.1\epsilon})}$, we have $pn/2 + \sqrt{2pn}\cdot s = O\left(\frac{\sqrt{\log 1/\epsilon}}{\epsilon^{3/2}}\right)$. Combined with (4), this gives us the desired upper bound on the expected error of the protocol.

Proof of privacy

Let $X = (x_1, \ldots, x_{n-1}, x_n) \in \{0,1\}^n$ and $X' = (x_1, \ldots, x_{n-1}, x'_n) \in \{0,1\}^n$ be two neighboring datasets. By symmetry, without loss of generality, we may assume that $x_n = 0$ and at least $n_0 := \lceil n-1\rceil/2$ of the values of $x_1, \ldots, x_{n-1}$ are also 0. By permuting the users, we may also assume without loss of generality that $x_1 = x_2 = \cdots = x_{n_0} = 0$. For $1 \le i \le n$, let $Y_i \in [0, d]$ denote the (random) number of 1s output by user i when their input is x_i. Also let $Y'_n \in [0,1]$ denote the (random) number of 1's output by user n when its input is x'_n. By [8, Lemma A.2], to show that $\frac{\Pr[Y_1+\cdots+Y_{n-1}+Y_n=t]}{\Pr[Y_1+\cdots+Y_{n-1}+Y'_n=t]} \in [e^{-\epsilon}, e^{\epsilon}]$ for all $t \in \mathbb{N}$, it suffices to show that for all $t_0 \in \mathbb{N}$, $\frac{\Pr[Y_1+\cdots+Y_{n_0}+Y_n=t_0]}{\Pr[Y_1+\cdots+Y_{n_0}+Y'_n=t_0]} \in [e^{-\epsilon}, e^{\epsilon}]$. Now the validity of the latter is an immediate consequence of Lemma 9 with the parameter n of Lemma 9 equal to $n_0 + 1$. ◀

From now on, we will use ν and ω_b as abbreviations for $\mathrm{DLap}_d(d/2, s)$ and $\mathbf{1}(\frac{d-1}{2} + b)$ respectively, where d, s are defined as in Lemma 9.

Let us denote by $P_{m,k}$ the probability that m independent random variables from the noise distribution ν sums up to k; more formally, $P_{m,k} := \Pr_{Z_1,\ldots,Z_m\sim\nu}[Z_1 + \cdots + Z_m = k]$. For convenience, we define $P_{0,0} = 1$ and $P_{0,k} = 0$ for all $k \ne 0$.

As we will see in the proof of Lemma 9 below, expansions of the numerator and denominator of the left hand side of (3) result in similar terms involving $P_{m,k}$, except occasionally with (i) k differing by one or (ii) m differing by 1 and k differing by $\left(\frac{d-1}{2}\right)$ or $\left(\frac{d-3}{2}\right)$. Hence, to bound the ratio between the two, we have to find some relation between $P_{m,k}, P_{m,k-1}, P_{m+1,k+\left(\frac{d-1}{2}\right)}$, and $P_{m+1,k+\left(\frac{d-3}{2}\right)}$. The exact inequality we will use here is stated below and its proof overview is given in Section 3.3.

▶ **Lemma 10.** *For any sufficiently large $n \in \mathbb{N}$, let ϵ, d and s be as in Lemma 9. Then the following hold: For any integers $\frac{10\log(1/(1-e^{-0.1\epsilon}))}{1-e^{-0.1\epsilon}} \le m \le n-1$, and $\ell_1, \ell_2 \in \left\{\frac{d-1}{2}, \frac{d-3}{2}\right\}$, if $p \ge \frac{100\,e^{100\epsilon}}{n(1-e^{-0.1\epsilon})}$, then we have*

$$e^{-\epsilon}p(1-e^{-\epsilon/2})\cdot\left(P_{m+1,k+\ell_1}+\frac{(n-m-1)}{m+1}\cdot P_{m+1,k+\ell_2}\right)+e^{0.2\epsilon}\cdot P_{m,k-1}\ge P_{m,k}.$$

We also need the following lemma, which can be interpreted as an anti-concentration result. Recall that $P_{i,j} = \Pr_{Z_1,\ldots,Z_i\sim\nu}[Z_1+\cdots+Z_i=j]$. For any $a\in\mathbb{N}$, if also $Z'_1,\ldots,Z'_a\sim\nu$, then as $\mathbb{E}[Z'_1+\cdots+Z'_a]=da/2$ and the distribution of $Z'_1+\cdots+Z'_a$ has sufficient mass at its expectation, one expects that $P_{i+a,j+a}=\Pr[Z_1+\cdots+Z_i+Z'_1+\cdots+Z'_a=j+da/2]$ is not too much smaller than $P_{i,j}$. Lemma 11 says that in fact $\Pr[Z_1+\cdots+Z_i+Z'_1+\cdots+Z'_a=j+da/2-d/2]$ is not too much smaller than $P_{i,j}$. Its proof is deferred to the full version [24].

▶ **Lemma 11.** *For any $i,j,a\in\mathbb{N}_0$ such that $a\le s^2/1000$, we have $P_{i+a,j+a\left(\frac{d-1}{2}\right)}\ge\frac{\sqrt{a}}{40s^3}\cdot P_{i,j}$.*

With Lemmas 10 and 11 ready, we can now prove Lemma 9 as follows.

Proof of Lemma 9. Let $c_0\in(0,1)$ be some sufficiently small positive constant, to be specified later. We would like to show that, for all $t\in\{0,\ldots,dn\}$, the following hold:

$$\Pr_{Z_1,\ldots,Z_n\sim\mathcal{R}_0}[Z_1+\cdots+Z_n=t]\le e^{\epsilon}\cdot\Pr_{Z_1,\ldots,Z_{n-1}\sim\mathcal{R}_0,Z_n\sim\mathcal{R}_1}[Z_1+\cdots+Z_n=t],\tag{5}$$

and

$$\Pr_{Z_1,\ldots,Z_{n-1}\sim\mathcal{R}_0,Z_n\sim\mathcal{R}_1}[Z_1+\cdots+Z_n=t]\le e^{\epsilon}\cdot\Pr_{Z_1,\ldots,Z_n\sim\mathcal{R}_0}[Z_1+\cdots+Z_n=t].\tag{6}$$

Due to space constraints, we will only prove (5) here; (6) can be proved in a similar manner. To prove (5), let us first decompose the probability on the left and the right hand sides based on whether Z_n is sampled from the noise distribution ν. This gives

$$\begin{aligned}&\Pr_{Z_1,\ldots,Z_n\sim\mathcal{R}_0}[Z_1+\cdots+Z_n=t]\\&=p\cdot\Pr_{Z_1,\ldots,Z_{n-1}\sim\mathcal{R}_0,Z_n\sim\nu}[Z_1+\cdots+Z_n=t]\\&\quad+(1-p)\cdot\Pr_{Z_1,\ldots,Z_{n-1}\sim\mathcal{R}_0}\left[Z_1+\cdots+Z_{n-1}=t-\left(\frac{d-1}{2}\right)\right],\end{aligned}$$

and

$$\begin{aligned}&\Pr_{Z_1,\ldots,Z_{n-1}\sim\mathcal{R}_0,Z_n\sim\mathcal{R}_1}[Z_1+\cdots+Z_n=t]\\&=p\cdot\Pr_{Z_1,\ldots,Z_{n-1}\sim\mathcal{R}_0,Z_n\sim\nu}[Z_1+\cdots+Z_n=t]\\&\quad+(1-p)\cdot\Pr_{Z_1,\ldots,Z_{n-1}\sim\mathcal{R}_0}\left[Z_1+\cdots+Z_{n-1}=t-\left(\frac{d+1}{2}\right)\right].\end{aligned}$$

Moreover, observe that, by expanding based on the number of variables among $Z_1,\ldots,Z_{n-1}$ that uses the noise distribution (i.e., i below), we have

$$\Pr_{Z_1,\ldots,Z_{n-1}\sim\mathcal{R}_0,Z_n\sim\nu}[Z_1+\cdots+Z_n=t]$$

$$= \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} \cdot P_{i+1,t-(n-1-i)\left(\frac{d-1}{2}\right)},$$

and

$$\Pr_{Z_1,\ldots,Z_{n-1}\sim\mathcal{R}_0}\left[Z_1+\cdots+Z_{n-1} = t-\left(\frac{d-1}{2}\right)\right]$$
$$= \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} \cdot P_{i,t-(n-i)\left(\frac{d-1}{2}\right)}.$$

Finally, we have

$$\Pr_{Z_1,\ldots,Z_{n-1}\sim\mathcal{R}_0}\left[Z_1+\cdots+Z_{n-1} = t-\left(\frac{d+1}{2}\right)\right]$$
$$= \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} \cdot P_{i,t-(n-i)\left(\frac{d-1}{2}\right)-1}.$$
$$= \frac{1}{e^\epsilon} \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} \cdot e^{\epsilon/2} \cdot P_{i,t-(n-i)\left(\frac{d-1}{2}\right)-1}$$
$$+ \frac{1}{e^\epsilon} \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} \cdot (e^\epsilon - e^{\epsilon/2}) P_{i,t-(n-i)\left(\frac{d-1}{2}\right)-1}$$
$$\geq \frac{1}{e^\epsilon} \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} \cdot e^{\epsilon/2} \cdot P_{i,t-(n-i)\left(\frac{d-1}{2}\right)-1}$$
$$+ \frac{1}{e^\epsilon} \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} \cdot (e^{\epsilon/2} - 1) P_{i,t-(n-i)\left(\frac{d-1}{2}\right)-1}$$
$$\geq \frac{1}{e^\epsilon} \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} \cdot e^{\epsilon/2} \cdot P_{i,t-(n-i)\left(\frac{d-1}{2}\right)-1}$$
$$+ \frac{1}{e^\epsilon} \cdot \sum_{i=0}^{n-1} \binom{n-1}{i+1} p^{i+1} (1-p)^{n-2-i} \cdot (e^{\epsilon/2} - 1) P_{i+1,t-(n-i-1)\left(\frac{d-1}{2}\right)-1}$$
$$\geq \frac{1}{e^\epsilon} \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} e^{\epsilon/2} \cdot P_{i,t-(n-i)\left(\frac{d-1}{2}\right)-1}$$
$$+ \frac{1}{e^\epsilon} \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} \cdot \frac{p(n-1-i)}{i+1} \cdot (e^{\epsilon/2} - 1) P_{i+1,t-(n-i-1)\left(\frac{d-1}{2}\right)-1}.$$

Using the above expressions, we may write the difference between the right hand side and the left hand side of (5) as

$$e^\epsilon \cdot \Pr_{Z_1,\ldots,Z_{n-1}\sim\mathcal{R}_0, Z_n\sim\mathcal{R}_1}[Z_1+\cdots+Z_n = t] - \Pr_{Z_1,\ldots,Z_n\sim\mathcal{R}_0}[Z_1+\cdots+Z_n = t]$$
$$\geq (e^\epsilon - 1) \cdot p \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} \cdot P_{i+1,t-(n-1-i)\left(\frac{d-1}{2}\right)}$$
$$+ (1-p) \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} e^{\epsilon/2} \cdot P_{i,t-(n-i)\left(\frac{d-1}{2}\right)-1}$$
$$+ (1-p) \cdot \sum_{i=0}^{n-1} \binom{n-1}{i} p^i (1-p)^{n-1-i} \cdot \frac{p(n-1-i)}{i+1} \cdot (e^{\epsilon/2} - 1) P_{i+1,t-(n-i-1)\left(\frac{d-1}{2}\right)-1}$$

$$
\begin{aligned}
&- (1-p)\cdot\sum_{i=0}^{n-1}\binom{n-1}{i}p^i(1-p)^{n-1-i}\cdot P_{i,t-(n-i)\left(\frac{d-1}{2}\right)}\\
&\geq (1-p)\sum_{i=0}^{n-1}\binom{n-1}{i}p^i(1-p)^{n-1-i}\Delta_i, \qquad (7)
\end{aligned}
$$

where

$$
\begin{aligned}
\Delta_i := p(1-e^{-\epsilon/2})\cdot&\left(P_{i+1,t-(n-1-i)\left(\frac{d-1}{2}\right)}+\frac{n-1-i}{i+1}\cdot P_{i+1,t-(n-i-1)\left(\frac{d-1}{2}\right)-1}\right)\\
&+e^{\epsilon/2}\cdot P_{i,t-(n-i)\left(\frac{d-1}{2}\right)-1}-P_{i,t-(n-i)\left(\frac{d-1}{2}\right)},
\end{aligned}
$$

and we have used that $e^{\epsilon}-1\geq e^{\epsilon/2}-1\geq 1-e^{-\epsilon/2}$ for $\epsilon\geq 0$.

By Lemma 10 with $m=i, k=t-(n-i)\left(\frac{d-1}{2}\right), \ell_1=\left(\frac{d-1}{2}\right), \ell_2=\left(\frac{d-3}{2}\right)$, we have

$$
\Delta_i\geq (e^{0.3\epsilon}-1)P_{i,t-(n-i)\left(\frac{d-1}{2}\right)}\geq 0, \qquad (8)
$$

for all i such that $\frac{10\log(1/(1-e^{-0.1\epsilon}))}{1-e^{-0.1\epsilon}}\leq i\leq n-1$. Let $i_0:=\frac{10\log(1/(1-e^{-0.1\epsilon}))}{1-e^{-0.1\epsilon}}$. It remains to lower bound the terms in (7) for $0\leq i<i_0$. To do so, we will "borrow" the additional mass of $(e^{0.3\epsilon}-1)P_{i,t-(n-i)\left(\frac{d-1}{2}\right)}$ from the terms with $i\geq i_0$. To show that this borrowing gives sufficient positive mass from the terms $P_{i,t-(n-i)\left(\frac{d-1}{2}\right)}$ with $i\geq i_0$, we will use Lemma 11.

Next, let $i_{\max}\in\{0,1,\ldots,i_0-1\}$ and $i_{\min}\in\{i_0,i_0+1,\ldots,2p(n-1)\}$ be defined so that:

$$
\begin{aligned}
&P_{i_{\max},t-(n-i_{\max})\left(\frac{d-1}{2}\right)}\geq P_{i,t-(n-i)\left(\frac{d-1}{2}\right)} \quad \forall i\in\{0,1,\ldots,i_0-1\}\\
&P_{i_{\min},t-(n-i_{\min})\left(\frac{d-1}{2}\right)}\leq P_{i,t-(n-i)\left(\frac{d-1}{2}\right)} \quad \forall i\in\{i_0,i_0+1,\ldots,2p(n-1)\}.
\end{aligned}
$$

As $p=\frac{100\,e^{100\epsilon}\log(1/(1-e^{-0.1\epsilon}))}{n(1-e^{-0.1\epsilon})}$ and $\epsilon<c_0\leq 1$, we have that as long as c_0 is sufficiently small,

$$
2p(n-1)\leq\frac{100e^{100}\log(5/\epsilon)}{0.1\epsilon}\leq\frac{1}{4\epsilon^2}=s^2/1000.
$$

It follows from Lemma 11 with $a=i_{\min}-i_{\max}\leq 2p(n-1)$ that

$$
P_{i_{\min},t-(n-i_{\min})\left(\frac{d-1}{2}\right)}\geq\frac{1}{40s^3}P_{i_{\max},t-(n-i_{\max})\left(\frac{d-1}{2}\right)}.
$$

Let $M\sim\mathrm{Binom}(n-1,p)$ be a binomial random variable. Then, as (8) holds for $n-1\geq i\geq i_0$, we have

$$
\begin{aligned}
&\sum_{i=0}^{n-1}\binom{n-1}{i}p^i(1-p)^{n-1-i}\Delta_i\\
&\geq-\sum_{i=0}^{i_0-1}\binom{n-1}{i}p^i(1-p)^{n-1-i}P_{i,t-(n-i)\left(\frac{d-1}{2}\right)}\\
&\quad+\sum_{i=i_0}^{2p(n-1)}\binom{n-1}{i}p^i(1-p)^{n-1-i}(e^{0.3\epsilon}-1)\cdot P_{i,t-(n-i)\left(\frac{d-1}{2}\right)}\\
&\geq-P_{i_{\max},t-(n-i_{\max})\left(\frac{d-1}{2}\right)}\sum_{i=0}^{i_0-1}\binom{n-1}{i}p^i(1-p)^{n-1-i} \qquad (9)\\
&\quad+0.3\epsilon\cdot P_{i_{\min},t-(n-i_{\min})\left(\frac{d-1}{2}\right)}\sum_{i=i_0}^{2p(n-1)}\binom{n-1}{i}p^i(1-p)^{n-1-i}
\end{aligned}
$$

$$\geq P_{i_{\max},t-(n-i_{\max})\left(\frac{d-1}{2}\right)} \cdot \left(\frac{0.3\epsilon}{40s^3} \cdot \Pr[i_0 \leq M \leq 2p(n-1)] - \Pr[M < i_0] \right). \tag{10}$$

By the Chernoff bound, for sufficiently large n and since $pn = n \cdot \frac{100\, e^{100\epsilon} \log(1/(1-e^{-0.1\epsilon}))}{n(1-e^{-0.1\epsilon})} \geq 100$, we have

$$\Pr[M > 2p(n-1)] \leq \exp(-p(n-1)/3) \leq \exp(-pn/4) < 1/2.$$

Moreover, since $i_0 = \frac{10\log(1/(1-e^{-0.1\epsilon}))}{1-e^{-0.1\epsilon}} \leq pn/3 \leq p(n-1)/2$ and $\epsilon \leq 1$ in the current case,

$$\Pr[M < i_0] \leq \exp(-p(n-1)/8) \leq \exp(-pn/10) \leq \exp\left(\frac{10}{1-e^{-0.1\epsilon}}\right) < 1/4.$$

Hence, recalling $s = \frac{10}{\epsilon}$, $d = 4\left\lceil \frac{1000\, e^{100\epsilon}}{(1-e^{-0.1\epsilon})} \cdot \log\left(\frac{n}{1-e^{-0.1\epsilon}}\right)\right\rceil + 3$, $p = \frac{100\, e^{100\epsilon} \log(1/(1-e^{-0.1\epsilon}))}{n(1-e^{-0.1\epsilon})}$, as well as the assumption $\epsilon > 1/n^{2/3}$,

$$\begin{aligned}
&\frac{0.3\epsilon}{40s^3} \cdot \Pr[i_0 \leq M \leq 2p(n-1)] - \Pr[M < i_0] \\
&\geq \frac{0.3\epsilon}{160s^3} - \exp(-1/(2\epsilon)) \\
&\geq c\epsilon^4 - \exp(-1/(2\epsilon)),
\end{aligned}$$

for some sufficiently small positive absolute constant c. The above quantity is positive as long as $\exp(1/(2\epsilon)) \geq \frac{1}{c\epsilon^4}$, i.e., as long as $\epsilon \leq c'$ for some absolute constant $c' > 0$ (which holds as long as we select $c_0 \leq c'$). From this and (7), we can conclude that (5) holds. ◀

3.3 Proof Overview of Lemma 10

In this subsection, we give an overview of the proof of Lemma 10; we defer the full proof to the full version [24]. Throughout this section, we will use the several additional notation:

- First, we will overload the notation and use $\nu(z)$ to denote the probability mass function of ν at z, i.e., $\nu(z) := \Pr_{Z\sim\nu}[Z = z]$.
- We often represent a sequence of integers $a_1, \ldots, a_m$ as a vector $\mathbf{a} = (a_1, \ldots, a_m)$. For such a vector, we use $\nu(\mathbf{a})$ as a shorthand for the product $\nu(a_1) \cdots \nu(a_m)$, and $\mathrm{zero}(\mathbf{a})$ as a shorthand for the number of zero coordinates, i.e., $\mathrm{zero}(\mathbf{a}) := |\{i \in [m] \mid a_i = 0\}|$.
- We use $S_{m,k,d}$ to denote the set of all sequences of integers $a_1, \ldots, a_m$ between 0 and d (inclusive) whose sum is k; more formally, $S_{m,k,d} := \Delta_{m,k} \cap [0,d]^m$. Since d will be fixed throughout, we omit d and simply write $S_{m,k}$.
- Next, for any $i \in \mathbb{R}$, we use $S_{m,k}^{\mathrm{zero}<i}$ (resp. $S_{m,k}^{\mathrm{zero}\geq i}$) to denote the sets of sequences in $S_{m,k}$ whose number of zero-coordinates is less than (resp., at least) i. More formally, $S_{m,k}^{\mathrm{zero}<i} := \{\mathbf{a} \in S_{m,k} \mid \mathrm{zero}(\mathbf{a}) < i\}$ and $S_{m,k}^{\mathrm{zero}\geq i} := \{\mathbf{a} \in S_{m,k} \mid \mathrm{zero}(\mathbf{a}) \geq i\}$.

To prove Lemma 10, let us observe that we may expand $P_{m,k}$ as

$$P_{m,k} = \sum_{\mathbf{a}\in S_{m,k}} \nu(\mathbf{a}) = \sum_{\mathbf{a}\in S_{m,k}^{\mathrm{zero}<i}} \nu(\mathbf{a}) + \sum_{\mathbf{a}\in S_{m,k}^{\mathrm{zero}\geq i}} \nu(\mathbf{a}),$$

where i will be chosen later in the proof.

We will bound the two terms on the right separately. More specifically, we will show that

$$\sum_{\mathbf{a}\in S_{m,k}^{\mathrm{zero}<i}} \nu(\mathbf{a}) \leq e^{0.5\epsilon} \cdot P_{m,k-1}, \tag{11}$$

and that for $\ell_1, \ell_2 \in \{\frac{d-1}{2}, \frac{d-3}{2}\}$,

$$\sum_{\mathbf{a} \in S_{m,k}^{\text{zero} \geq i}} \nu(\mathbf{a}) \leq p(1 - e^{-0.5\epsilon}) \cdot \left(P_{m+1,k+\ell_1} + \frac{(n-m+1)}{m+1} \cdot P_{m+1,k+\ell_2} \right). \tag{12}$$

Once we have these two inequalities, Lemma 10 immediately follows. The intuition behind the two inequalities is quite simple. For (11), since each sequence $\mathbf{a} \in S_{m,k}^{\text{zero}<i}$ contains few zeros, we should be able to pick a non-zero a_i and decrease it by one and end up with a sequence in $S_{m,k-1}$ instead; since the discrete Laplace distribution's mass (i.e., ν) on a_i and on $a_i - 1$ differs (multiplicatively) by a factor of at most $e^{1/s}$, the mass of the modified sequence also differs from the original sequence by a factor of $e^{1/s}$.

For (12), the intuition is pretty similar. We start with a sequence $\mathbf{a} \in S_{m,k}^{\text{zero} \geq i}$ and we will modify it to end up with a sequence in $S_{m+1,k+\ell}$ where ℓ is either $\left(\frac{d-1}{2}\right)$ or $\left(\frac{d-3}{2}\right)$. The intuition here is that since $\mathbf{a}$ contains many zero coordinates, there are many ways for us to divide ℓ among these zero coordinates and an additional coordinate, which would result naturally in a sequence in $P_{m+1,k+\ell}$.

To turn the intuition into a formal proof, we need to be careful about "double counting" a modified sequence. As an example, for (11), suppose we would like to modify a sequence in $S_{m,k}^{\text{zero}<i}$ to one in $S_{m,k-1}$ by decreasing any non-zero coordinate. Then, it is possible that two sequences $(1, 0, a_3, \ldots, a_n)$ and $(0, 1, a_3, \ldots, a_n)$ results in the same sequence $(0, 0, a_3, \ldots, a_n)$.

In order to avoid such "double counting", we divide our proofs into two parts. First, we show that we may replace $S_{m,k}^{\text{zero}<i}$ (resp. $S_{m,k}^{\text{zero}\geq i}$) with the set of sequences whose first coordinate is non-zero (resp., whose first few coordinates are zeros).

4 Lower Bound for Binary Summation

We next prove our lower bound on the communication complexity of any non-interactive pure-$\text{DP}_{\text{shuffled}}$ protocol that can perform bit addition with small error (Theorem 4). To prove our lower bound, first recall the following notion from probability theory.

▶ **Definition 12** (Moment Generating Function). *Let $\mathbf{Y}$ be a $\mathbb{R}^k$-valued random variable for some $k \in \mathbb{N}$. Its* moment generating function (MGF) *is defined as $\mathbf{M}_{\mathbf{Y}}(\mathbf{t}) = \mathbb{E}[e^{\langle \mathbf{t}, \mathbf{Y} \rangle}]$.*

Throughout this section, we will be dealing with pairs of random variables whose MGFs are within a certain factor of each other. The following definition will be particularly handy.

▶ **Definition 13** (Bounded MGF ratio). *We say that two $\mathbb{R}^k$-valued random variables $\mathbf{Y}, \mathbf{Y}'$ have e^ϵ-*bounded MGF ratio *if and only if, for all $\mathbf{t} \in \mathbb{R}^k$ we have that $\frac{\mathbf{M}_{\mathbf{Y}}(\mathbf{t})}{\mathbf{M}_{\mathbf{Y}'}(\mathbf{t})} \in [e^{-\epsilon}, e^{\epsilon}]$.*

Furthermore, let $\text{SD}(\mathbf{Y}, \mathbf{Y}')$ denote the *total variation distance* between them.

Our proofs follow the outline from Section 1.3. Specifically, in Section 4.1, we prove that a pure-$\text{DP}_{\text{shuffled}}$ protocol implies bounded MGF ratio. Then, in Section 4.2, we give a lower bound on $C_{\epsilon,\gamma}$ from Definition 5, which then implies Theorem 4. Finally, in Section 4.3, we provide an example which shows that our lower bound for the question is tight.

4.1 Pure-DP Implies Bounded MGF Ratio

We start by proving a necessary (but not sufficient) condition on ϵ-DP protocols in terms of the MGFs of $\mathbf{X}^0, \mathbf{X}^1$. A straightforward observation we will use is the following:

▶ **Observation 14.** *Let $\mathbf{Y}, \mathbf{Y}'$ be two random variables with the same support $\text{supp}(\mathbf{Y}) = \text{supp}(\mathbf{Y}') \subseteq \mathbb{R}^k$ such that $\frac{\Pr[\mathbf{Y}=\mathbf{v}]}{\Pr[\mathbf{Y}'=\mathbf{v}]} \in [e^{-\epsilon}, e^{\epsilon}]$. Then, $\mathbf{Y}, \mathbf{Y}'$ satisfies e^ϵ-bounded MGF ratio.*

Furthermore, recall a (well-known) multiplicative property of MGF that, if $\mathbf{Y}, \mathbf{Y}' \in \mathbb{R}^k$ are two independent random variables, then $\mathbf{M}_{\mathbf{Y}+\mathbf{Y}'}(\mathbf{t}) = \mathbf{M}_{\mathbf{Y}}(\mathbf{t}) \cdot \mathbf{M}_{\mathbf{Y}'}(\mathbf{t})$ for all $\mathbf{t} \in \mathbb{R}^k$. With this in mind, we can now prove the desired result:

▶ **Lemma 15.** *For any ϵ-DP protocol, $\mathbf{X}^0, \mathbf{X}^1$ must satisfy e^ϵ-bounded MGF ratio.*

Proof. Consider two input vectors $0 \dots 00$ and $0 \dots 01$. Let $\mathbf{Y}^0, \mathbf{Y}^1 \in \mathbb{Z}^k$ denote the views of the shuffled output on the corresponding input vectors, where $\mathbf{Y}^0_j$ denote the number of j's received by the analyzer for the input vector $0 \dots 00$ and $\mathbf{Y}^1_j$ denote the number of j's received by the analyzer for the input vector $0 \dots 01$. Notice that $\mathbf{Y}^0$ is a sum of n i.i.d. copies of $\mathbf{X}^0$ and $\mathbf{Y}^1$ is a sum of $(n-1)$ i.i.d. copies of $\mathbf{X}^0$ and a copy of $\mathbf{X}^1$. Observe also that ϵ-DP implies that $\mathbf{Y}^0, \mathbf{Y}^1$ satisfy the condition in Observation 14. From this, we have

$$[e^{-\epsilon}, e^{\epsilon}] \ni \frac{\mathbf{M}_{\mathbf{Y}^0}(\mathbf{t})}{\mathbf{M}_{\mathbf{Y}^1}(\mathbf{t})} = \frac{(\mathbf{M}_{\mathbf{X}^0}(\mathbf{t}))^n}{(\mathbf{M}_{\mathbf{X}^0}(\mathbf{t}))^{n-1} \cdot \mathbf{M}_{\mathbf{X}^1}(\mathbf{t})} = \frac{\mathbf{M}_{\mathbf{X}^0}(\mathbf{t})}{\mathbf{M}_{\mathbf{X}^1}(\mathbf{t})}. \qquad \blacktriangleleft$$

4.2 From Bounded MGF Ratio to Communication Lower Bound

We will now use the bounded MGF ratio property to bound the communication complexity of any non-interactive protocol for summation that incurs small error. To do so, let us recall below a known result that any protocol that can perform binary summation to within a small error must have large statistical distance between $\mathbf{X}^0$ and $\mathbf{X}^1$:

▶ **Theorem 16** ([14]). *Any non-interactive protocol that can perform binary summation to within an expected absolute error of α must satisfy* $\mathrm{SD}(\mathbf{X}^0, \mathbf{X}^1) \geq 1 - O\left(\frac{\alpha}{\sqrt{n}}\right)$.

Thanks to Lemma 15 and Theorem 16, to prove our lower bound (Theorem 4), it now suffices to show that, for any $\mathbf{Y}, \mathbf{Y}'$ whose supports lie in $\Delta_{k,m}$ that satisfy e^ϵ-bounded MGF ratio and $\mathrm{SD}(\mathbf{Y}, \mathbf{Y}')$ is large, then $m \log k$ must be large. The main lemma of this subsection, which is a quantitative version of the aforementioned statement, is stated formally below.

▶ **Lemma 17.** *Let $\mathbf{Y}, \mathbf{Y}'$ be two random variables supported on $\Delta_{k,m}$ with e^ϵ-bounded MGF ratio. Then,* $\mathrm{SD}(\mathbf{Y}, \mathbf{Y}') \leq 1 - 2^{-O_\epsilon(m^2 \log k)}$.

It is straightforward to see that Lemma 17, Lemma 15, and Theorem 16 together implies Theorem 4. We devote the rest of this subsection to the proof of Lemma 17.

Dual Approach and Proof of Lemma 17

For notational convenience, we use $p_{\mathbf{y}}$ and $p'_{\mathbf{y}}$ to denote $\Pr[\mathbf{Y} = \mathbf{y}]$ and $\Pr[\mathbf{Y}' = \mathbf{y}]$ respectively.

Before we formalize the proof below, let us first present an informal overview. Recall that $1 - \mathrm{SD}(\mathbf{Y}, \mathbf{Y}')$ is equal to $\sum_{\mathbf{y} \in \Delta_{k,m}} \min\{p_{\mathbf{y}}, p'_{\mathbf{y}}\} = \min_{S \subseteq \Delta_{k,m}} \left\{\sum_{\mathbf{y} \in S} p_{\mathbf{y}} + \sum_{\mathbf{y} \in \Delta_{k,m} \setminus S} p'_{\mathbf{y}}\right\}$. Hence, it suffices for us to show that, for every $S \subseteq \Delta_{k,m}$, we have

$$\sum_{\mathbf{y} \in S} p_{\mathbf{y}} + \sum_{\mathbf{y} \in \Delta_{k,m} \setminus S} p'_{\mathbf{y}} \geq 2^{-O_\epsilon(m^2 \log k)}. \tag{13}$$

We will give a "dual certificate" for this statement. Notice that since the total probability of each of $\mathbf{Y}, \mathbf{Y}'$ must be one, we have $\sum_{\mathbf{y} \in \Delta_{k,m}} p_{\mathbf{y}} = 1$ and $\sum_{\mathbf{y} \in \Delta_{k,m}} p'_{\mathbf{y}} = 1$. We also have the non-negativity constraints: $p_{\mathbf{y}}, p'_{\mathbf{y}} \geq 0$ for all $\mathbf{y} \in \Delta_{k,m}$. Finally, the e^ϵ-bounded MGF ratio property between $\mathbf{Y}$ and $\mathbf{Y}'$ translates to the following linear inequalities for all $\mathbf{t} \in \mathbb{R}^k$:

$$\sum_{\mathbf{y} \in \Delta_{k,m}} e^{\langle \mathbf{t}, \mathbf{y} \rangle} \cdot p'_{\mathbf{y}} - \sum_{\mathbf{y} \in \Delta_{k,m}} e^{\langle \mathbf{t}, \mathbf{y} \rangle - \epsilon} \cdot p_{\mathbf{y}} \geq 0, \quad \sum_{\mathbf{y} \in \Delta_{k,m}} e^{\langle \mathbf{t}, \mathbf{y} \rangle} \cdot p_{\mathbf{y}} - \sum_{\mathbf{y} \in \Delta_{k,m}} e^{\langle \mathbf{t}, \mathbf{y} \rangle - \epsilon} \cdot p'_{\mathbf{y}} \geq 0. \tag{14}$$

Hence, we have a system of linear inequalities and we would like to certify a linear inequality (13). We may do so by writing (13) as a linear combination of the constraints.

As a wishful thinking, if we could somehow "extract" only the $p_{\mathbf{y}}$ and $p'_{\mathbf{y}}$ terms from (14), we would be done because we would simply have $e^{\epsilon} \cdot p_{\mathbf{y}} \geq p'_{\mathbf{y}} \geq e^{-\epsilon} \cdot p_{\mathbf{y}}$ which can easily be combined with the total probability and non-negativity constraints to get a good bound on $\sum_{\mathbf{y} \in S} p_{\mathbf{y}} + \sum_{\mathbf{y} \in \Delta_{k,m} \setminus S} p'_{\mathbf{y}}$. Of course, such extraction is not possible since, for any value $\mathbf{t}$ we plug into (14), we always get non-zero coefficients for all vectors in $\Delta_{k,m}$, not just $\mathbf{y}$.

With this in mind, our relaxed goal is to select $\mathbf{t} = \tau(\mathbf{y})$ for each $\mathbf{y}$ in such a way that the coefficient of $\mathbf{y}$ from its own inequality (i.e., $\mathbf{t} = \tau(\mathbf{y})$) "dominates" the coefficients of $\mathbf{y}$ from other inequalities (i.e., $\mathbf{t} = \tau(\mathbf{y}')$ for $\mathbf{y}' \neq \mathbf{y}$). A formal version of the statement is proved below. Note that $e^{\beta(\mathbf{y})}$ should be thought of as the "scaling factor" for the inequality for $\mathbf{y}$.

▶ **Lemma 18.** *For any $\epsilon > 0$, there exist mappings $\tau : \Delta_{k,m} \to \mathbb{R}^k$ and $\beta : \Delta_{k,m} \to \mathbb{R}$ such that the following hold for all $\mathbf{y} \in \Delta_{k,m}$:*

$$0 \geq \langle \tau(\mathbf{y}), \mathbf{y} \rangle + \beta(\mathbf{y}) \geq \zeta := -O_{\epsilon}(m^2 \log k), \tag{15}$$

and

$$e^{\langle \tau(\mathbf{y}), \mathbf{y} \rangle + \beta(\mathbf{y})} \geq 2e^{\epsilon} \cdot \sum_{\mathbf{y}' \in \Delta_{k,m} \setminus \{\mathbf{y}\}} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle + \beta(\mathbf{y}')}. \tag{16}$$

Proof. Let $\rho = \epsilon + 10 \ln(k+1) + 10$. We pick $\tau(\mathbf{y}) = \rho \cdot 2\mathbf{y}$ and $\beta(\mathbf{y}) = \rho \cdot \left(-\|\mathbf{y}\|_2^2 - m^2\right)$. It is obvious to see that (15) holds. To prove (16), let us first observe the following identity:

$$\langle \tau(\mathbf{y}'), \mathbf{y} \rangle + \beta(\mathbf{y}') = \langle \tau(\mathbf{y}), \mathbf{y} \rangle + \beta(\mathbf{y}) - \rho \cdot \|\mathbf{y} - \mathbf{y}'\|_2^2. \tag{17}$$

Thus, we may rewrite the right hand side of (16) as

$$\begin{aligned} &\sum_{\mathbf{y}' \in \Delta_{k,m} \setminus \{\mathbf{y}\}} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle + \beta(\mathbf{y}')} \\ &\overset{(17)}{=} e^{\langle \tau(\mathbf{y}), \mathbf{y} \rangle + \beta(\mathbf{y})} \cdot \left(\sum_{i=1}^{2m^2} e^{-\rho i} \cdot \left| \{\mathbf{y}' \in \Delta_{k,m} \mid \|\mathbf{y} - \mathbf{y}'\|_2^2 = i\} \right| \right). \end{aligned} \tag{18}$$

Furthermore, we have $\left|\{\mathbf{y}' \in \Delta_{k,m} \mid \|\mathbf{y} - \mathbf{y}'\|_2^2 = i\}\right| \leq \left|\{\mathbf{z} \in \mathbb{Z}^k \mid \|\mathbf{z}\|_2^2 = i\}\right| \leq 2^i \cdot \binom{k+i-1}{i} \leq (2e(k+1))^i$. Plugging this back into (18), we have

$$\begin{aligned} \sum_{\mathbf{y}' \in \Delta_{k,m} \setminus \{\mathbf{y}\}} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle + \beta(\mathbf{y}')} &\leq e^{\langle \tau(\mathbf{y}), \mathbf{y} \rangle + \beta(\mathbf{y})} \cdot \left(\sum_{i=1}^{2m^2} \left(e^{-\rho} \cdot 2e(k+1) \right)^i \right) \\ &\text{(From our choice of } \rho) \leq e^{\langle \tau(\mathbf{y}), \mathbf{y} \rangle + \beta(\mathbf{y})} \cdot \frac{1}{2e^{\epsilon}}. \end{aligned}$$

◀

With Lemma 18 ready, we can now prove Lemma 17.

Proof of Lemma 17. Let τ, β be as in Lemma 18. Consider any set $S \subseteq \Delta_{k,m}$.

For every $\mathbf{y}' \in S$, $\mathbf{M}_{\mathbf{Y}}(\tau(\mathbf{y}')) \geq e^{-\epsilon} \cdot \mathbf{M}_{\mathbf{Y}'}(\tau(\mathbf{y}'))$ is equivalent to

$$\sum_{\mathbf{y} \in \Delta_{k,m}} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}')} \cdot p_{\mathbf{y}} - \sum_{\mathbf{y} \in \Delta_{k,m}} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}') - \epsilon} \cdot p'_{\mathbf{y}} \geq 0. \tag{19}$$

Similarly, for every $\mathbf{y}' \in \Delta_{k,m} \setminus S$, $\mathbf{M}_{\mathbf{Y}'}(\tau(\mathbf{y}')) \geq e^{-\epsilon} \cdot \mathbf{M}_{\mathbf{Y}}(\tau(\mathbf{y}'))$ can be rearranged as

$$\sum_{\mathbf{y} \in \Delta_{k,m}} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}')} \cdot p'_{\mathbf{y}} - \sum_{\mathbf{y} \in \Delta_{k,m}} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}') - \epsilon} \cdot p_{\mathbf{y}} \geq 0. \tag{20}$$

By adding (19) for all $\mathbf{y}' \in S$ with (20) for all $\mathbf{y}' \in \Delta_{k,m} \setminus S$, we have

$$\sum_{\mathbf{y} \in \Delta_{k,m}} \left(\sum_{\mathbf{y}' \in S} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}')} - \sum_{\mathbf{y}' \in \Delta_{k,m} \setminus S} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}') - \epsilon} \right) p_{\mathbf{y}}$$
$$+ \sum_{\mathbf{y} \in \Delta_{k,m}} \left(\sum_{\mathbf{y}' \in \Delta_{k,m} \setminus S} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}')} - \sum_{\mathbf{y}' \in S} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}') - \epsilon} \right) p'_{\mathbf{y}} \geq 0. \tag{21}$$

Now, for all $\mathbf{y} \in S$, we can upper bound the coefficient of $p'_{\mathbf{y}}$ in (21) by

$$\sum_{\mathbf{y}' \in \Delta_{k,m} \setminus S} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}')} - \sum_{\mathbf{y}' \in S} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}') - \epsilon}$$
$$\leq \sum_{\mathbf{y}' \in \Delta_{k,m} \setminus \{\mathbf{y}\}} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}')} - e^{\langle \tau(\mathbf{y}), \mathbf{y} \rangle - \beta(\mathbf{y}) - \epsilon} \overset{(16)}{\leq} -0.5 e^{\langle \tau(\mathbf{y}), \mathbf{y} \rangle - \beta(\mathbf{y}) - \epsilon} \overset{(15)}{\leq} -e^{\zeta - 1 - \epsilon}.$$

Similarly, for all $\mathbf{y} \in \Delta_{k,m} \setminus S$, the coefficient of $p_{\mathbf{y}}$ in (21) is at most $-e^{\zeta - 1 - \epsilon}$.

Moreover, for all $\mathbf{y} \in S$, we can upper bound the coefficient in (21) of $p_{\mathbf{y}}$ by

$$\sum_{\mathbf{y}' \in S} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}')} - \sum_{\mathbf{y}' \in \Delta_{k,m} \setminus S} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}') - \epsilon}$$
$$\leq \sum_{\mathbf{y}' \in \Delta_{k,m}} e^{\langle \tau(\mathbf{y}'), \mathbf{y} \rangle - \beta(\mathbf{y}')} \overset{(16)}{\leq} \left(1 + \frac{1}{2e^{\epsilon}}\right) e^{\langle \tau(\mathbf{y}), \mathbf{y} \rangle - \beta(\mathbf{y})} \overset{(15)}{\leq} 2.$$

Similarly, for all $\mathbf{y} \in S$, the coefficient of $p'_{\mathbf{y}}$ in (21) is at most 2.

Plugging these back into (21), we have

$$0 \leq 2 \left(\sum_{\mathbf{y} \in S} p_{\mathbf{y}} + \sum_{\mathbf{y} \in \Delta_{k,m} \setminus S} p'_{\mathbf{y}} \right) - e^{\zeta - 1 - \epsilon} \left(\sum_{\mathbf{y} \in S} p'_{\mathbf{y}} + \sum_{\mathbf{y} \in \Delta_{k,m} \setminus S} p_{\mathbf{y}} \right).$$

Using the fact that $\sum_{\mathbf{y} \in \Delta_{k,m}} p_{\mathbf{y}} = \sum_{\mathbf{y} \in \Delta_{k,m}} p'_{\mathbf{y}} = 1$, we can simplify the RHS above to

$$2e^{\zeta - 1 - \epsilon} \leq (2 + e^{\zeta - 1 - \epsilon}) \left(\sum_{\mathbf{y} \in S} p_{\mathbf{y}} + \sum_{\mathbf{y} \in \Delta_{k,m} \setminus S} p'_{\mathbf{y}} \right).$$

This means that

$$\left(\sum_{\mathbf{y} \in S} p_{\mathbf{y}} + \sum_{\mathbf{y} \in \Delta_{k,m} \setminus S} p'_{\mathbf{y}} \right) \geq \frac{2e^{\zeta - 1 - \epsilon}}{2 + e^{\zeta - 1 - \epsilon}} \overset{(15)}{\geq} 2^{-O_{\epsilon}(m^2 \log k)}.$$

This establishes (13) and hence we have $\mathrm{SD}(\mathbf{Y}, \mathbf{Y}') \leq 1 - 2^{-O_{\epsilon}(m^2 \log k)}$ as desired. ◀

4.3 Limitations of the Lower Bound Approach

In this subsection, we argue that the bound we achieve in Lemma 17 is essentially tight, even for $k = 2$. In other words, our approach of using only bounded MGF ratio property and the total variation distance bound from Theorem 16 cannot give any lower bound better than $O_\epsilon(\sqrt{\log n})$. Specifically, the main lemma of this section is stated below.

▶ **Lemma 19.** *For every $\epsilon > 0$ and $\gamma \in (0, 0.5)$, there exist two random variables $\mathbf{Y}, \mathbf{Y}'$ supported on (subsets of) $\Delta_{2,m}$ for some $m = O_\epsilon(\sqrt{\log(1/\gamma)})$ such that $\mathrm{SD}(\mathbf{Y}, \mathbf{Y}') \geq 1 - \gamma$ and that $\mathbf{Y}, \mathbf{Y}'$ satisfy the e^ϵ-bounded MGF ratio property.*

Similar to when we analyze our binary summation protocol in Section 3, it suffices to prove the following one-dimensional version of the above statement, where the two random variables are from $\{0, 1, \ldots, m\}$ rather than $\Delta_{2,m}$.

▶ **Lemma 20.** *For every $\epsilon > 0$ and $\gamma \in (0, 0.5)$, there exist two random variables Y^0 and Y^1 supported on $\{0, \ldots, m\}$ for some $m = O_\epsilon(\sqrt{\log(1/\gamma)})$ such that $\mathrm{SD}(Y^0, Y^1) \geq 1 - \gamma$ and that Y^0, Y^1 satisfy e^ϵ-bounded MGF ratio property.*

4.3.1 Discrete Gaussian Distributions

Our construction for Lemma 20 is based on the discrete Gaussian distribution, which can be defined as follows. Let the (one-dimensional) Gaussian function centered at c with parameter s as $\rho_{s,c}(x) := \exp\left(-\frac{\pi(x-c)^2}{s^2}\right)$ for all $x \in \mathbb{R}$. For any countable $A \subseteq \mathbb{R}$, let $\rho_{s,c}(A) := \sum_{x \in A} \rho_{s,c}(x)$. When $\rho_{s,c}(A)$ is finite, the discrete Gaussian distribution over A centered at c with parameter s, denoted by $\mathcal{D}_{A,s,c}$, has $\mathcal{D}_{A,s,c}(x) = \frac{\rho_{s,c}(x)}{\rho_{s,c}(A)}$ for all $x \in A$.

We will use a special case of a well-known property of lattices (cf. [35, 23, 3]). Recall that a one-dimensional lattice is $a\mathbb{Z} := \{at \mid t \in \mathbb{Z}\}$ for some $a \in \mathbb{R}^+$. Informally, the property states that, if for sufficiently large s , "shifting" the discrete Gaussian distribution by c does not change its normalization factor too much. This is formalized below.

▶ **Lemma 21** (e.g. [23, Lemma 2.6])**.** *For any constants $a, \delta \in \mathbb{R}^+$, there exists a sufficiently large constant $s^* = s^*(a, \delta)$ such that, for any $c \in \mathbb{R}$, we have $\frac{\rho_{s^*,c}(a\mathbb{Z})}{\rho_{s^*,0}(a\mathbb{Z})} \in [e^{-\delta}, 1]$.*

We will also use the following (rather straightforward) observation that, similar to the (continuous) Gaussian distribution, we may choose a sufficiently large truncation point $\ell^* a$ for which the total mass of all points x with $|X - c| > \ell^* a$ is arbitrarily small.

▶ **Observation 22.** *For any constants $a, \delta \in \mathbb{R}^+$, let $s^* = s^*(a, \delta)$ be as in Lemma 21. Then, for any $\lambda > 0$, there exists a sufficiently large positive integer $\ell^* = \ell^*(a, \delta, \lambda)$ such that, for any $c \in \mathbb{R}$, we have $\Pr_{X \sim \mathcal{D}_{a\mathbb{Z},s^*,c}}[|X - c| > \ell^* a] \leq \lambda$.*

4.3.2 Proof Overview of Lemma 20

Distributions of both Y^0, Y^1 will place $\frac{\gamma}{2}$ probability masses at each of 0 and m, and these two points shared by the supports of Y^0 and Y^1. (This ensures that the total variation distance of Y^0 and Y^1 are at least $1 - \gamma$.) In the middle, we then place discrete Gaussian distributions centered at $c = m/2$ for Y^0 and Y^1, with that of Y^0 only supported on even numbers whereas that of Y^1 supported on odd numbers. These discrete Gaussian distributions are truncated so that the supports are within the range of $[c - w, c + w]$ for some parameter w.

The intuition behind the construction is as follows. First, when $|t| \geq O_\epsilon(\sqrt{\log(1/\gamma)})$, it is not hard to see that the MGFs at t are dominated by the terms corresponding to the points

0 or m. Our parameters are selected in such a way that, when this is not the case, it must be that $|t| \ll w$. In this case, we observe that the MGFs of discrete Guassian distributions are simply proportional to normalization terms of other discrete Gaussian distributions, shifted by $O(t)$ (and truncated appropriately). Since $|t| \ll w$, we can then apply Lemma 21 and Observation 22 to get a good bound on these terms. This concludes the main ideas in the proof; due to space constraints, the full proof is deferred to the full version [24].

5 Conclusions and Open Questions

We gave the first pure-$\text{DP}_{\text{shuffled}}$ protocols for binary and real summation with constant error. We further prove a communication lower bound for any non-interactive protocols for binary summation. While these have advanced our understanding of pure-$\text{DP}_{\text{shuffled}}$ protocols, many questions remain open after this work. Specifically, the immediate open questions are:

- Can we improve the error guarantee in the (binary and real) summation protocols to achieve the asymptotically optimal error $1/\epsilon$, which can be achieved in $\text{DP}_{\text{central}}$ [19]?
- What is the optimal per user communication complexity of non-interactive $\text{DP}_{\text{shuffled}}$ protocols for binary and real summation? As we have shown, the communication complexity for binary summation lies between $O_\epsilon(\log n)$ and $\Omega_\epsilon(\sqrt{\log n})$. On the other hand, for real summation, the only lower bound is the trivial $\Omega(\log n)$ bound (which holds even without privacy concerns) whereas our upper bound is $O_\epsilon(\log^3 n)$.
- In Appendix A of the full version [24], we show that our binary summation protocol also yields a pure-DP protocol for histograms with error $O_\epsilon(\log B \log n)$ but with $O_\epsilon(B \log n)$ per user communication complexity. The latter is in contrast to the approximate-DP multi-message protocol of [25], which has a per user communication complexity of only $O_\epsilon(poly(\log n, \log B))$ and incurs a similar error. It is thus an interesting open question to come up with (or rule out) a pure-DP protocol with a smaller communication complexity.
- Can we exploit interactivity to break our $\Omega_\epsilon(\sqrt{\log n})$ communication lower bound? Alternately, can we prove any non-trivial lower bound that holds also with interaction?

On a high-level, it would also be interesting to develop tools to help prove guarantees for pure-$\text{DP}_{\text{shuffled}}$ protocols. In the case of approximate-DP, there are amplification theorems [21, 8] that can yield an approximate-$\text{DP}_{\text{shuffled}}$ protocol from a DP_{local} protocol. Although this may not be optimal in some cases (as shown by the multi-message protocols in [25, 7, 26]), such theorems can be conveniently applied to a large class of protocols and yield good approximate-DP guarantees. On the other hand, our proofs in this work are specific to our carefully designed protocols. It would be much more convenient if one can give a unifying theorem that proves pure privacy guarantees for any protocol with easily verifiable conditions.

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. doi:10.1145/2976749.2978318.

2 John M Abowd. The US Census Bureau adopts differential privacy. In *KDD*, pages 2867–2867, 2018.

3 Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. Discrete Gaussian leftover hash lemma over infinite domains. In *ASIACRYPT*, pages 97–116, 2013. doi:10.1007/978-3-642-42033-7_6.

4 Apple Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 2017.

5 Victor Balcer and Albert Cheu. Separating local & shuffled differential privacy via histograms. *arXiv: 1911.06879*, 2019.

6 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. Differentially private summation with multi-message shuffling. *arXiv: 1906.09116*, 2019.

7 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. Improved summation from shuffling. *arXiv: 1909.11225*, 2019.

8 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *CRYPTO*, pages 638–667, 2019. doi:10.1007/978-3-030-26951-7_22.

9 Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. Private summation in the multi-message shuffle model. *arXiv: 2002.00817*, 2020.

10 Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *CRYPTO*, pages 451–468, 2008.

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 Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.

13 Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. In *PODS*, pages 435–447, 2018.

14 T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Optimal lower bound for differentially private multi-party aggregation. In *ESA*, pages 277–288, 2012. doi:10.1007/978-3-642-33090-2_25.

15 Albert Cheu, Adam D. Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *EUROCRYPT*, pages 375–403, 2019. doi:10.1007/978-3-030-17653-2_13.

16 Anindya De. Lower bounds in differential privacy. In *TCC*, pages 321–338, 2012. doi:10.1007/978-3-642-28914-9_18.

17 Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *NIPS*, pages 3571–3580, 2017.

18 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.

19 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

20 Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *arXiv:2001.03618*, 2020.

21 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.

22 Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067, 2014.

23 Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008. doi:10.1145/1374376.1374407.

24 Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Pure differentially private summation from anonymous messages. *arXiv: 2002.01919*, 2020. arXiv:2002.01919.

25 Badih Ghazi, Noah Golowich, Ravi Kumar, Rasmus Pagh, and Ameya Velingker. On the power of multiple anonymous messages. Cryptology ePrint Archive, Report 2019/1382, 2019.

26 Badih Ghazi, Pasin Manurangsi, Rasmus Pagh, and Ameya Velingker. Private aggregation from fewer anonymous messages. *arXiv: 1909.11073*, 2019.

27 Badih Ghazi, Rasmus Pagh, and Ameya Velingker. Scalable and differentially private distributed aggregation in the shuffled model. *arXiv: 1906.08320*, 2019.

28 Andy Greenberg. Apple's "differential privacy" is about collecting your data – but not your data. *Wired, June*, 13, 2016.

29 Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, pages 61–70, 2010.

30 Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *STOC*, pages 705–714, 2010.

31 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *FOCS*, pages 239–248, 2006.

32 Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid 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, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, 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. *arXiv: 1912.04977*, 2019.

33 Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Rashkodnikova, and Adam Smith. What can we learn privately? In *FOCS*, pages 531–540, 2008.

34 Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv: 1610.05492*, 2016.

35 Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SICOMP*, 37(1):267–302, 2007. doi:10.1137/S0097539705447360.

36 Ilya Mironov. Rényi differential privacy. In *CSF*, pages 263–275, 2017. doi:10.1109/CSF.2017.11.

37 Aleksandar Nikolov, Kunal Talwar, and Li Zhang. On the geometry of differential privacy: the sparse and approximate cases. In *STOC*, pages 351–360, 2013.

38 Stephen Shankland. How Google tricks itself to protect Chrome user privacy. *CNET, October*, 2014.

39 Thomas Steinke and Jonathan Ullman. Between pure and approximate differential privacy. *arXiv: 1501.06095*, 2015.

40 Salil Vadhan. *The Complexity of Differential Privacy*, pages 347–450. Springer International Publishing, 2017.

41 Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *JASA*, 60(309):63–69, 1965.

On Locally Decodable Codes in Resource Bounded Channels

Jeremiah Blocki
Purdue University, West Lafayette, IN, USA
jblocki@purdue.edu

Shubhang Kulkarni
Purdue University, West Lafayette, IN, USA
kulkar17@purdue.edu

Samson Zhou
Carnegie Mellon University, Pittsburgh, PA, USA
samsonzhou@gmail.com

Abstract

Constructions of locally decodable codes (LDCs) have one of two undesirable properties: low rate or high locality (polynomial in the length of the message). In settings where the encoder/decoder have already exchanged cryptographic keys and the channel is a probabilistic polynomial time (PPT) algorithm, it is possible to circumvent these barriers and design LDCs with constant rate and small locality. However, the assumption that the encoder/decoder have exchanged cryptographic keys is often prohibitive. We thus consider the problem of designing explicit and efficient LDCs in settings where the channel is *slightly* more constrained than the encoder/decoder with respect to some resource e.g., space or (sequential) time. Given an explicit function f that the channel cannot compute, we show how the encoder can transmit a random secret key to the local decoder using $f(\cdot)$ and a random oracle $\mathsf{H}(\cdot)$. We then bootstrap the private key LDC construction of Ostrovsky, Pandey and Sahai (ICALP, 2007), thereby answering an open question posed by Guruswami and Smith (FOCS 2010) of whether such bootstrapping techniques are applicable to LDCs in channel models weaker than just PPT algorithms. Specifically, in the random oracle model we show how to construct explicit constant rate LDCs with locality of polylog in the security parameter against various resource constrained channels.

2012 ACM Subject Classification Theory of computation → Error-correcting codes

Keywords and phrases Locally Decodable Codes, Resource Bounded Channels

Digital Object Identifier 10.4230/LIPIcs.ITC.2020.16

Related Version `https://arxiv.org/pdf/1909.11245.pdf`

Funding This research was supported in part by the National Science Foundation (CCF Award #1910659).

Acknowledgements We would like to thank anonymous reviewers for helpful feedback that improved the presentation of this paper.

1 Introduction

Consider the classical one-way communication setting where two parties, the *sender* and *receiver*, communicate over a *noisy channel* that may *corrupt* parts of any message sent over it. An *error correcting code* is an invertible transformation mapping messages into *codewords* that are then transmitted over the noisy channel. The goal is to ensure that the decoder can (w.h.p.) reliably recover the entire message from the corrupted codeword. For locally decodable codes (LDCs) we have an even stronger goal: The decoder should be able

to reliably recover *any* individual bit of the original message (w.h.p.) by examining at most ℓ bits of the corrupted codeword. An ideal LDC should have a good rate (i.e., the codeword should not be much longer than the original message) and small locality ℓ.

Historically, there have been two major lines of work associated with modelling the channel behavior. In Shannon's *symmetric channel* model, the channel each bit of the codeword independently at random with some fixed probability. By contrast, in Hamming's *adversarial channel* model the channel corrupts the codeword in a worst case manner subject to an upper bound on the total number of corruptions.

Unsurprisingly, when we work in Shannon's channel model it is much easier to design LDCs with good rate/locality. By contrast, state of the art LDC constructions for Hamming channels either have very high locality e.g., $\ell = 2^{\mathcal{O}\left(\sqrt{\log n \log\log n}\right)}$ [29] or poor rate e.g., Hadamard codes have constant locality $\ell = \mathcal{O}(1)$ but the codeword has exponential length. Unfortunately, in many real-world settings independent random noise is not a realistic model of channel behavior e.g., burst-errors are common in reality, but unlikely in Shannon's model. Thus, coding schemes designed to work in Shannon's channel model are not necessarily suitable in practice. By contrast, coding schemes designed to work in Hamming's adversarial setting must be able to handle *any* error pattern.

Our central motivating goal is to find classes of adversarial channels that are expressive enough to model any error patterns that would arise in nature, yet admit LDCs with good decoding algorithms. LDCs have found remarkable applications throughout various fields, notably private information retrieval schemes [9, 16, 30], psuedo-random generator constructions [7, 38], self-correcting computations [18, 21], PCP systems [8] and fault tolerant storage systems [27].

Lipton [31] introduced the *adversarial computationally bounded* model, where the channel was viewed as a Hamming channel restricted to bounded corruption by a *probabilistic polynomial time* (PPT) algorithm. The notion of adversaries being computationally bounded is well-motivated by real-world channels that have some sort of limitations on their computations i.e., we expect error patterns encountered in nature to be modeled by some (possibly unknown) PPT algorithm. We argue that even Lipton's channel significantly overestimates the capability of the channel. For example, if the channel has reasonably small latency, say 10 seconds, and the world's fastest single core processor can evaluate 10 billion instructions per second then the depth of any (parallel) computation performed by the channel is at most 100 billion operations.

This view of modelling the channel as more restricted than just PPT was further explored by Guruswami and Smith [23] who studied channels that could be described by simple (low-depth) circuits. Remarkably, even such a simple restriction allowed them to design codes that enjoyed no public/private key setup assumptions, while matching the Shannon capacity using polynomial time encoding/decoding algorithms. With such positive results, it is natural to ask whether similar results may be expected for LDCs.

1.1 Contributions

We introduce *resource bounded adversarial channel* models which admit LDCs with good locality whilst still being expressive enough to plausibly capture any error pattern for most real-world channels. We argue that these resource bounded channel models are already sufficiently expressive to model any corruption pattern that might occur in nature e.g., burst-errors, correlated errors. For example, observe that the channel must compute the entire error pattern *before* the codeword is delivered to the receiver. Thus, the channel can

be viewed as *sequentially time bounded* e.g., the channel may perform arbitrary computation in parallel but the total depth of computation is bounded by the latency of the channel. The notion of a space bound (or space-time bound) channel can be similarly motivated.

We introduce *safe functions* as a general way to characterize LDC friendly channels. Intuitively, a function f is "safe" for a class of channels if the channel is not able to predict $f(x)$ given x. We show how to construct safe functions for several classes of resource bounded channels including time bounded, space bounded, and cumulative memory cost bounded channels in the parallel random oracle model. For example, in the random oracle model the function $\mathsf{H}^{t+1}(x)$ is a safe function for the class of sequentially time-bounded adversaries i.e., it is not possible to evaluate the function using fewer than t sequential calls to the random oracle H. We also discuss how to construct safe functions for the class of space (resp. space-time) bounded channels using random oracles.

Furthermore, we give a general framework for designing good locally decodable codes against resource bounded adversarial channels by using safe functions to bootstrap existing private-key LDC constructions. Our framework assumes no a priori private or public key setup assumptions, and constructs explicit LDCs over the binary alphabet[1] with constant rate against *any* class of resource bounded adversaries admitting *safe functions*.

Our local decoder can decode correctly with arbitrarily high constant probability after examining at most $\mathcal{O}\left(f(\kappa)\right)$ bits of the corrupted codeword, where κ is the security parameter[2] and $f(\kappa)$ is any function such that $f(\kappa) = \omega(\log \kappa)$ e.g., $f(\kappa) = \log^{1+\varepsilon} \kappa$ or $f(\kappa) = \log \kappa \log\log \kappa$. By contrast, state of the art LDC constructions for Hamming channels have very high locality e.g., $2^{\mathcal{O}\left(\sqrt{\log n \log\log n}\right)}$ [29]. Our codes are robust against a constant fraction of corruptions, and are (essentially) *non-adaptive* i.e., the local decoding algorithm can decode after submitting just two batches of queries.

Our constructions stand at the intersection of coding theory and cryptography, using well-known tools and techniques from cryptography to provide notions of (information theoretic) randomness and security for communication protocols between sender/receiver. To prove the security of our constructions, we introduce a *two-phase distinguisher hybrid argument*, which may be of independent interest for other coding theoretic problems in these resource bounded channel models.

1.2 Technical Overview

Private LDCs. Our starting point is the private locally decodable codes of [33]. These LDCs permit nearly optimal query complexity, asymptotically positive rate and reliable decoding with high probability, but make the strong assumption that the sender and receiver have already exchanged a secret key sk that is unknown to the PPT adversarial channel over which they communicate. In our setting the sender and the receiver do not have access to any secret key. Our constructions thus *reduce* the general setting (no setup assumptions) against resource bounded channels to the shared private key setting against these channels, so that we can bootstrap private LDC constructions.

[1] Note that small alphabet sizes are attractive for practical channels designed to transmit bits efficiently.

[2] In this paper we use the security parameter κ in an asymptotic sense e.g., for any attacker running in time $\mathtt{poly}(\kappa)$ there is a negligible function $\mathtt{negl}(\kappa)$ upper bounding the probability that the attacker succeeds. In particular, the function $\mathtt{negl}(\kappa) = 2^{-\log^{1+\varepsilon} \kappa)}$ is negligible, but does not provide κ-bits of concrete security i.e., any attacker running in time t succeeds with probability at most $t2^{-\kappa}$.

Bootstrapped Encoder/Decoder. Our encoder uses the following high level template: (1) samples a random seed r (2) computes a predetermined *safe function* $f(r)$ on the seed and extracts a secret key sk from $f(r)$ (e.g., using a random oracle) (3) Uses the private LDC encoder to encode the message using sk (4) appends a reliable encoding (error-correcting code composed with a repetition code) of the random seed r to the codeword. The local decoder (1) decodes the random seed r (random sampling + majority vote). (2) Evaluates the safe function $f(r)$ to recover the secret key sk. (3) Uses the private LDC decoder with the secret key sk to recover the desired bit of the original message.

Security Proof. We remark that there are a few subtle challenges that arise while proving the security of our bootstrapped constructions. We want to prove that the channel fails to produce a corrupted codeword that fools the local decoding algorithm. Towards this goal we might try to prove that the channel cannot distinguish the derived key sk from a truly random key, even when given the nonce r. However, this is insufficient to prove that the local decoder is successful because the local decoder *is* able to recover sk from the f. We introduce a novel *two-phase distinguisher game* to address these challenges. In particular, we consider an attacker-distinguisher pair who tries to predict whether or not the secret encoding key sk is derived from the nonce r ($b = 0$) or was selected uniformly at random ($b = 1$). In phase 1 the (resource bounded) attacker generates a corrupted codeword which is given to the distinguisher in phase 2 who must then guess whether $b = 1$ or $b = 0$. The distinguisher is computationally unbounded, but is not allowed to query the random oracle. If f is a safe function then the advantage of any such attacker-distinguisher pair can be shown to be negligible. We demonstrate that any channel which succeeds at fooling our local decoder yields an attacker-distinguisher pair for this two phase game – the distinguisher works by simulating the private LDC decoder to distinguish between the two aforementioned encodings. It follows that the channel cannot fool the local decoder (except with negligible probability).

1.3 Related Work

Many existing code constructions consider an underlying channel that can only introduce a bounded number of errors, but has an unlimited time to adversarially decide the positions of these errors. These codes are therefore resilient to any possible error pattern with a bounded number of corruptions, corresponding to Hamming's error model, and are safe for data transmission. However, this resiliency to the worst-case error leads to coding limitations and some possibly undesirable tradeoffs. On one hand, current constructions for LDCs that focus on efficient encoding can obtain any constant rate $R < 1$ while simultaneously being robust to any constant fraction $\delta < 1 - R$ of errors and using $2^{\mathcal{O}\left(\sqrt{\log n \log\log n}\right)}$ queries for decoding [29]. On the other hand, codes that focus on low query complexity obtain blocklength that is subexponential in the message length while using a constant number of queries $q \leq 3$ [39, 20, 19]. Finally, if exactly $q = 2$ queries are desired, any code *must* use blocklength exponential in the message length [28]. Avoiding such drastic tradeoffs between blocklength and query complexity would be attractive for other natural channels in contrast to Hamming's error model. For example, Shannon introduces a model in which each symbol has some independent probability of being corrupted; this probability is generally fixed across all symbols and known a priori. However, this probabilistic channel may be too weak to capture natural phenomenon such as bursts of consecutive error.

Thus it is reasonable to believe that many natural channels lie between these two extremes; in particular, Lipton [31] argues that many reasonable channels are computationally bounded and can be modeled as PPT algorithms. In this model, [31] introduced an analog to classical error-correcting codes that is robust to a fraction of errors beyond the rates provably tolerable by *any* code in the adversarial Hamming channel model. Similarly, a line of work [31, 32, 23, 37] have improved upon the error rate limits of classical error-correcting codes in slight variants of Lipton's computationally bounded channel model. A weakness of the codes introduced by [31] is the strong cryptographic assumption that the sender and receiver share a *secret* random string unknown to the channel. This weakness is ameliorated by [32], who observe that if a message is encoded by digitally signing a code that is *list-decodable* with a secret key, then an adversarial PPT is unlikely to produce valid signatures. Conversely, the decoder can select the unique message from the list of possible messages with a valid signature, effectively producing public-key error-correcting codes against computationally bounded channels. Subsequently, [23] further removes the public-key setup assumption specifically for the channel in which either the error is independent of the actual message being sent, or the errors can be described by polynomial size circuits. Their results are based on the idea that the sender can choose a permutation and some key that is computable by the decoder but not by the channel, since it operates with low complexity. In some loose sense, their results are an example of our framework when the channel has bounded circuit complexity, i.e. the bounded resource is circuit complexity of the error.

[33] obtain LDCs with constant information and error rates over the binary alphabet against computationally bounded errors, using a small number of queries to the corrupted word; specifically they can achieve any $\omega(\log \kappa)$ query complexity, where κ is the desired security parameter. However, their results not only assume the existence of one-way functions, but also once again assume a predetermined private key known to both the encoder and decoder but not the channel, similar to [31]. Analogous to the improvements of [32] for classical error codes, [24, 25] construct public-key LDCs, assuming the existence of Φ-hiding schemes [15] and IND-CPA secure cryptosystems.

Ben-Sasson et al. [10] introduce the concept of *relaxed locally decodable codes* (RLDCs) as an alternative means of decreasing the tradeoffs between rate and locality in classical LDCs. In contrast to LDCs, the decoding algorithm for RLDCs is allowed to output $\perp$ sometimes to reveal that the correct value is unknown, though it is limited in the fraction of outputs in which it can output $\perp$. The RLDCs proposed by Ben-Sasson et al. [10] obtain constant query complexity and blocklength $n = k^{1+\epsilon}$. Subsequently, Gur et al. [22] construct *relaxed locally correctable codes* (RLCCs) with attractive properties but significant tradeoffs; they propose codes with constant query complexity and error rate but block length roughly quartic in the message length as well as codes with constant error rate and linear block length, but quasipolynomial ($(\log n)^{\mathcal{O}(\log \log n)}$) query complexity. These parameters are significantly better than classical locally correctable codes and their results immediately extend to RLDCs, since the original message is embedded within the initial part of the encoding. However, these tradeoffs are still undesirable.

Recently, Blocki et al. [12] study RLDCs and RLCCs on adversarial but computationally bounded channels in an effort to reduce these tradeoffs. They obtain RLDCs and RLCCs over the binary alphabet, with constant information rate, and poly-logarithmic locality. Moreover, their codes require no public-key or private-key cryptographic setup; the only setup assumption required is the selection of the public parameters (seed) for a collision-resistant hash function.

2 Preliminaries

2.1 Notation

We use the notation $[n]$ to represent the set $\{1, 2, \ldots, n\}$. For any $x, y \in \Sigma^n$, let $\mathsf{HAM}(x)$ denote the Hamming weight of x, i.e. the number of non-zero coordinates of x. Let $\mathsf{HAM}(x, y) = \mathsf{HAM}(x - y)$ denote the Hamming distance between the vectors x and y. All logarithms will be base 2. For n vectors $x_1, \ldots, x_n$, we use $\mathsf{majority}(x_1 \cdots x_n)$ to denote the vector that appears most frequently. If such a vector is not unique, then an arbitrary vector of highest frequency is chosen. For any vector $x \in \Sigma^n$, let $x[i]$ be the i^{th} coordinate of x. We also let $x \circ y$ denote the concatenation of x with y and $x \oplus y$ denote the bitwise XOR of x and y. For a randomized function $f(\cdot)$, the notation $f(\cdot; R)$ will be used to denote that $f(\cdot)$ uses random coins R as its randomness. A function $\mathsf{negl}(\kappa)$ is said to be *negligible* in κ if $\mathsf{negl}(\kappa) \in o\left(\left|\frac{1}{\mathsf{poly}(\kappa)}\right|\right)$ for any non-zero polynomial $\mathsf{poly}(\cdot)$. Finally, we distinguish between inputs and parameters to a function f as follows: $f(\text{inputs} \cdots)[\text{parameters} \cdots]$.

2.2 Locally Decodable Codes

We consider the setting where sender $\mathcal{S}$ encodes a *message* x into a *codeword* y using an *encoding algorithm* so that y is sent over noisy channel $\mathcal{C}$, which then hands over the possibly corrupted codeword y' to $\mathcal{R}$, who then uses a *decoding algorithm* to obtain the original message. We denote $x \in \Sigma^k$ and $y \in \Sigma^K$ where Σ is the alphabet. We denote the alphabet size by $q = |\Sigma|$. We consider the model where y' corresponds to y with some symbols replaced with others in Σ. The term *corruptions* refers to such symbol replacements within y, with a single corruption meaning a single symbol replacement, so that $y' \in \Sigma^K$. The encoding and decoding algorithms are denoted by $\mathsf{Enc} : \Sigma^k \to \Sigma^K$ and $\mathsf{Dec} : \Sigma^K \to \Sigma^k$. We use the terms sender, encoder, and encoding algorithm interchangeably, and similarly for receiver, decoder, and decoding algorithm.

A *code* is an encoder-decoder pair. The *information rate* or simply *rate* of the code is the ratio k/K, so that a lower rate corresponds to a larger amount of information redundancy introduced by the code. The message length, codeword length, and alphabet size characterize a *coding scheme*. Coding schemes with high rate and low alphabet size are desired.

An error correcting code allows the decoder to recover the entire original message x by reading the entire y'. It is also possible to construct codes that only need to read a few symbols of y' rather than the entire message to recover a small part of the message. Such codes are called *locally decodable codes* (LDC), and will be the focus of this work. An LDC has *locality* ℓ, *error rate* ρ and *error correction probability* p if any character of x may be recovered with probability at least p by making at most ℓ queries to y', even when the channel corrupts ρ fraction of all symbols of y to generate y'. We use the terms *query complexity* and locality interchangeably. When ρ and p are clear from context (as constants), the scheme may be referred to as an ℓ-LDC. Naturally, LDCs with low locality, high error rate, and high error correction probability are desired.

2.3 Definitions

The focus of this work will be the construction of LDCs (Section 2.4) for *resource-bounded* channels (Section 4.1). In this section, we present several building blocks that we will require in our constructions – LDC**s*, *private*-LDC*s* and *safe functions*. We first give two classical definitions pertaining to LDCs that compactly summarize our discussion in Section 2.2.

▶ **Definition 1.** *A* $(K,k)_q$-coding scheme $C[K,k,q] = (\mathsf{Enc},\mathsf{Dec})$ *is a pair of encoding* $\mathsf{Enc}:\Sigma^k \to \Sigma^K$ *and decoding* $\mathsf{Dec}:\Sigma^K \to \Sigma^k$ *algorithms where* $|\Sigma| = q$. *The* information rate *of the scheme is defined as* $\frac{k}{K}$.

▶ **Definition 2.** *A* $(K,k)_q$*-coding scheme* $C[K,k,q] = (\mathsf{Enc},\mathsf{Dec})$ *is an* (ℓ,ρ,p)-locally decodable code (LDC) *if* Dec*, with query access to a word* y' *such that* $\mathsf{HAM}(\mathsf{Enc}(x),y') \le \rho K$*, on input index* $i \in [k]$*, makes at most* ℓ *queries to* y' *and outputs* x_i *with probability at least* p *over the randomness of the decoder.*

Next, we present a simple variant of LDCs which we denote by LDC*s. These will be very similar to LDCs except that they are required to decode the entire original message while making as few queries to the corrupted codeword as possible. They are defined with respect to the same setting as in Section 2.2.

▶ **Definition 3.** *A* $(K,k)_q$*-coding scheme* $C[K,k,q] = (\mathsf{Enc},\mathsf{Dec})$ *is an* (ℓ,ρ,p)-LDC* *if* Dec*, with query access to a word* y' *such that* $\mathsf{HAM}(\mathsf{Enc}(x),y') \le \rho K$*, makes at most* ℓ *queries to* y' *and outputs* x *with probability at least* p *over the randomness of the decoder.*

We remark that it will be typically desired that for an LDC* $\mathsf{C}[K,k,q]$, the locality be $\mathcal{O}(k)$ even when K is very large. We now move on to define *one-time private*-LDC*s* analogous to Definition 2 as an alternative to that given by Ostrovsky, Pandey and Sahai [33]. Ostrovsky et al. also give explicit constructions of one-time private-LDCs, and so we restate their result according to our new definition. Refer to the full version [13] for an overview of [33].

$\mathtt{priv}-\mathtt{LDC}-\mathtt{Sec}-\mathtt{Game}[\mathcal{A},x,\kappa,\rho,p]$:

1. The challenger generates a secret key $\mathsf{sk} \leftarrow \mathsf{GenKey}(1^\kappa)$, computes the codeword $y \leftarrow \mathsf{Enc}(x,\kappa,\mathsf{sk})$ for the message x and sends the codeword y to the attacker.
2. The attacker outputs a corrupted codeword $y' \leftarrow \mathcal{A}(x,y,\kappa,\rho,p,k,K)$ where $y' \in \Sigma^K$ should have hamming distance at most ρK from y.
3. The output of the experiment is determined as follows:

$$\mathtt{priv}-\mathtt{LDC}-\mathtt{Sec}-\mathtt{Game}[\mathcal{A},x,\kappa,\rho,p] = \begin{cases} 1 & \text{if } \exists i \le k \text{ s.t. } \Pr[\mathsf{Dec}^{y'}(i,\kappa,\mathsf{sk}) = x_i] < p \\ 0 & \text{otherwise} \end{cases}$$

If the output of the experiment is 1 (resp. 0), the attacker $\mathcal{A}$ is said to *win* (resp. *lose*) against C.

Figure 1 $\mathtt{priv}-\mathtt{LDC}-\mathtt{Sec}-\mathtt{Game}$ defining the interaction between an attacker and an honest party.

▶ **Definition 4** (One-Time Private-LDC)**.** *A triplet of probabilistic algorithms* $\mathsf{C}[K,k,\kappa] = (\mathsf{GenKey},\mathsf{Enc},\mathsf{Dec})$ *is an* $(\ell,\rho,p,\epsilon,\mathbb{C})$-private locally decodable code (private LDC) *against a class* $\mathbb{C}$ *if* Dec *makes at most* ℓ *queries and for all attackers* $\mathcal{A} \in \mathbb{C}$ *and all messages* $x \in \Sigma^k$*,*

$$\Pr[\mathtt{priv}-\mathtt{LDC}-\mathtt{Sec}-\mathtt{Game}[\mathcal{A},x,\kappa,\rho,p] = 1] \le \epsilon$$

where the probability is taken over the random coins of $\mathcal{A}$ *and* GenKey*. If* $\mathbb{C}$ *is the set of all (computationally unbounded) attackers we say that the scheme is a* (ℓ,ρ,p,ϵ)-private LDC*.*

A construction of Ostrovsky et al. [33] yields a constant-rate One-Time Private-LDC with constant rate, low locality $\ell = \omega(\log \kappa)$ and negligible failure probability whenever $k = \mathrm{poly}(\kappa)$.

▶ **Theorem 5** (One-Time Private-LDC Existence). *[33] Let $f(\kappa)$ be any function such that $f(\kappa) = \omega(\log \kappa)$. Then, for security parameter κ and for all $K > k > 0$ such that $k = \text{poly}(\kappa)$ where* poly *is any non-zero polynomial, there exists a $(K, k)_2$ coding scheme that is a one-time $(\ell_{\mathsf{OPS}}, \rho_{\mathsf{OPS}}, p_{\mathsf{OPS}}, \epsilon_{\mathsf{OPS}})-$private* LDC *where $\ell_{\mathsf{OPS}} = f(\kappa)$, ρ_{OPS} is a constant, $p_{\mathsf{OPS}} = 1$, and $\epsilon_{\mathsf{OPS}} \leq k \left(\frac{e}{4}\right)^{-\rho_{\mathsf{OPS}} \ell_{\mathsf{OPS}}}$ is negligible in the security parameter.*

Our contributions in the subsequent sections will assume that the coding scheme and channel all have access to a random oracle. Furthermore, we assume that the channel is a pROM algorithm with respect to this random oracle (refer to the initial discussion in Section 4.1 for an overview of the pROM model). The following definition establishes a notion of *privacy* against classes (i.e. sets) of adversarial channels in terms of "hard to compute" functions.

▶ **Definition 6** (Safe Function). *We say that a function $f : \{0,1\}^n \to \{0,1\}^*$ is* δ-safe *for a class $\mathbb{C}$ of algorithms if for all $\mathcal{A} \in \mathbb{C}$ we have*

$$\Pr\left[\mathcal{A}(x) = f(x)\right] \leq \delta$$

where the probability is taken over the random coins of $\mathcal{A}$ and the selection of an input $x \in \{0,1\}^n$. If the function $f = f^{\mathsf{H}(\cdot)}$ is defined using a random oracle, then the probability $\Pr\left[\mathcal{A}^{\mathsf{H}(\cdot)}(x) = f^{\mathsf{H}(\cdot)}(x)\right]$ is also taken over the selection of the random oracle $\mathsf{H}(\cdot)$.

We will use the notation $\mathbb{S}_{\mathbb{C}}$ to denote a $\delta-$safe function for class $\mathbb{C}$. In the above definition, we usually think of δ as being a negligibly small parameter. We remark that in the parallel random oracle model, one can construct functions with sharp thresholds on the required resources. For example, the function $\mathsf{H}^{t+1}(x)$ is trivial to compute using at most $t+1$ sequential queries to $\mathsf{H} : \{0,1\}^* \to \{0,1\}^2$, but *any* parallel algorithm making at most q queries over t rounds succeeds with probability at most $\delta = (t^2 + tq)/2^w$.

Precomputation. Definition 6 can be extended to consider an attacker who is allowed to perform precomputation with the random oracle $\mathsf{H}(\cdot)$ before receiving the input x. In particular, we could consider a pair of oracle algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{A}_1^{\mathsf{H}(\cdot)}(m)$ outputs an m-bit hint $\sigma \in \{0,1\}^m$ for $\mathcal{A}_2$ after making at most q queries to $\mathsf{H}(\cdot)$. We could modify the definition to require that for all $\mathcal{A}_2 \in \mathbb{C}$ we have

$$\Pr\left[\mathcal{A}_2^{\mathsf{H}(\cdot)}(x, \mathcal{A}_1^{\mathsf{H}(\cdot)}(m)) = f^{\mathsf{H}(\cdot)}(x)\right] \leq \delta$$

where the randomness is taken over the selection of x, the random oracle $\mathsf{H}(\cdot)$, and the random coins of $\mathcal{A}_2$. Here, $\mathcal{A}_1^{\mathsf{H}(\cdot)}(m)$ (precomputation) is not necessarily constrained to be in the same class $\mathbb{C}$ as $\mathcal{A}_2$.

We remark that for $k = m/w$, a precomputing attacker can succeed with probability at least $k/2^n$ by having $\mathcal{A}_1^{\mathsf{H}(\cdot)}(m)$ output the hint $\sigma = f^{\mathsf{H}(\cdot)}(1), \ldots, f^{\mathsf{H}(\cdot)}(k)$. Then $\mathcal{A}_2^{\mathsf{H}(\cdot)}(x, \sigma)$ first checks if $x \in \{1, \ldots, k\}$ and, if so, simply returns the output $f^{\mathsf{H}(\cdot)}(x)$ which is already recorded in the hint σ. Thus, we need the length n of the random nonce x to be sufficiently large to resist brute-force precomputation attacks. By contrast, if the attacker does not get to perform any precomputation then δ can be negligible even when $n = \mathcal{O}(1)$.

All of the safe functions we consider would also be secure under this stronger notion. For example, $\mathsf{H}^{t+1}(x)$ is δ-safe for $\delta = \mathcal{O}\left((qt + t^2)/2^w + qt/2^n\right)$ where x is a random n bit string, $\mathcal{A}_1$ makes at most q total random oracle queries, and $\mathcal{A}_2$ makes at most q total queries in at most t rounds to $\mathsf{H}(\cdot)$. In our LDC constructions we select a random nonce of length $\Omega(\log^{1+\varepsilon} \kappa)$ to ensure that a precomputing attacker fails.

2.4 Our Model

We first define an experiment to model the interaction between a code and an algorithm from a class of pROM algorithms adversarial against the code. For random oracle $\mathsf{H}(\cdot)$, let $\mathsf{C} = (\mathsf{Enc}^{\mathsf{H}(\cdot)}, \mathsf{Dec}^{\mathsf{H}(\cdot)})$ be a $(K, k)_q$-coding scheme in the random oracle model and let $\mathbb{C}$ be a class of pROM algorithms. Then, the interaction of $\mathcal{A}^{\mathsf{H}(\cdot)} \in \mathbb{C}$ having error rate ρ, with the code C is defined in Figure 2 (analogous to `priv-LDC-Sec-Game` defined in Figure 1). Here, the security parameter κ, and the decoding probability p are also given as inputs to the game. We now formally define a notion of LDCs analogous to Definition 2, but with respect to general classes of adversarial (pROM) channels.

$\mathtt{LDC-Sec-Game}[\mathcal{A}, x, \mathsf{H}, \kappa, \rho, p]$:

1. The challenger computes $y \leftarrow \mathsf{Enc}^{\mathsf{H}(\cdot)}(x, \kappa)$ encoding the message x and sends $y \in \Sigma^K$ to the attacker.
2. The channel $\mathcal{A}^{\mathsf{H}(\cdot)}$ outputs a corrupted codeword $y' \leftarrow \mathcal{A}^{\mathsf{H}(\cdot)}(x, y, \kappa, \rho, p, k, K)$ where $y' \in \Sigma^K$ should have hamming distance at most ρK from y.
3. The output of the experiment is determined as follows:

$$\mathtt{LDC-Sec-Game}[\mathcal{A}, x, \mathsf{H}, \kappa, \rho, p] = \begin{cases} 1 & \text{if } \exists i \leq k \text{ such that } \Pr[\mathsf{Dec}^{y', \mathsf{H}(\cdot)}(i, \kappa) = x_i] < p \\ 0 & \text{otherwise} \end{cases}$$

If the output of the experiment is 1 (resp. 0), the channel is said to *win* (resp. *lose*).

Figure 2 $\mathtt{LDC-Sec-Game}$ defining the interaction between an attacker and an honest party.

▶ **Definition 7.** *Let $\mathbb{C}$ be a class of pROM algorithms. A $(K, k)_q$-coding scheme $\mathsf{C}[K, k, q] = (\mathsf{Enc}^{\mathsf{H}(\cdot)}, \mathsf{Dec}^{\mathsf{H}(\cdot)})$ is an $(\ell, \rho, p, \epsilon, \mathbb{C})$-*locally decodable code (LDC) *if $\mathsf{Dec}^{\mathsf{H}(\cdot)}$ makes at most ℓ queries and for all $\mathcal{A}^{\mathsf{H}(\cdot)} \in \mathbb{C}$ and all messages $x \in \Sigma^k$ we have*

$$\Pr[\mathtt{LDC-Sec-Game}[\mathcal{A}, x, \mathsf{H}, \kappa, \rho, p] = 1] \leq \epsilon$$

where the probability is taken over the random coins of $\mathcal{A}^{\mathsf{H}(\cdot)}$ and the selection of the random oracle H.

3 Constructions

We begin by discussing the use of safe functions in Section 3.1 and give several examples of constructing such functions in Section 4. We then show how allowing an encoder/decoder pair with enough resources to compute safe functions can effectively generate a random shared secret key between the pair. This secret key can then be bootstrapped into existing private LDC constructions to give codes against resource bounded adversaries. We give our final framework in Section 3.2 and the main proofs in Sections 3.3 and 3.4.

3.1 Using Safe Functions

Let $\mathbb{C}$ be a class of algorithms with safe function $\mathbb{S}_{\mathbb{C}}$. For some input $x \in \{0,1\}^n$ to $\mathcal{A}^{\mathsf{H}(\cdot)} \in \mathbb{C}$, we will be interested in bounding the probability of the undesirable event where the $\mathcal{A}^{\mathsf{H}(\cdot)}$queries the random oracle at any string of the form $y \circ \mathbb{S}_{\mathbb{C}}(x)$ with $y \in \{0,1\}^{\lceil \log_2 \alpha \rceil}$. In the absence of such an event, $\mathsf{H}(\mathbb{S}_{\mathbb{C}}(x))$ would information theoretically appear random to $\mathcal{A}^{\mathsf{H}(\cdot)}$. Lemma 8 shows that such an event may only happen with negligible probability $q\epsilon$ where q is the total number of random oracle queries.

▶ **Lemma 8.** *For a some class $\mathbb{C}$ of pROM algorithms with $\delta-$safe function $\mathbb{S}_{\mathbb{C}}\{0,1\}^n \to \{0,1\}^*$, let $\mathsf{bad}_{\mathcal{A}}$ be the event that on some input $x \in \{0,1\}^n$, $\mathcal{A}^{\mathsf{H}(\cdot)} \in \mathbb{C}$ queries the random oracle at $\alpha \circ \mathbb{S}_{\mathbb{C}}(x)$ for any $\alpha > 0$. Then $\Pr[\mathsf{bad}_{\mathcal{A}}] \leq q\delta$, where q is the number of oracle queries made by $\mathcal{A}^{\mathsf{H}(\cdot)}$.*

Proof. We prove the claim by a reduction argument. By way of contradiction, suppose there exists a $\mathcal{B}^{\mathsf{H}(\cdot)} \in \mathbb{C}$ such that on input string x, $\mathcal{B}^{\mathsf{H}(\cdot)}$ makes q queries to the random oracle $\mathsf{H}(\cdot)$ and $\Pr[\mathsf{bad}_{\mathcal{B}}] > q\epsilon$. We construct an adversary $\mathcal{A}^{\mathsf{H}(\cdot)}$ as follows: on input x, the adversary

- Simulates $\mathcal{B}^{\mathsf{H}(\cdot)}$ with input x
- Keeps track of all q queries by which $\mathcal{B}^{\mathsf{H}(\cdot)}$ queries the random oracle
- On termination of $\mathcal{B}^{\mathsf{H}(\cdot)}$, returns the suffix of length $|\mathbb{S}_{\mathbb{C}}(\mathrm{x})|$ from one of the q queries selected uniformly at random

However, we know that $\mathcal{B}^{\mathsf{H}(\cdot)}$ queries the random oracle at $\alpha \circ \mathbb{S}_{\mathbb{C}}(x)$ with probability $> q\delta$. Since $\mathcal{A}^{\mathsf{H}(\cdot)}$ picks one of $\mathcal{B}^{\mathsf{H}(\cdot)}$'s queries at random, $\Pr[\mathcal{A}^{\mathsf{H}(\cdot)}(x) = \mathbb{S}_{\mathbb{C}}(x)] > \delta$, which contradicts the definition of $\delta-$safe function. ◀

Assuming that $\mathcal{A}^{\mathsf{H}(\cdot)}$never queries the random oracle at any point of the form $y \circ \mathbb{S}_{\mathbb{C}}(x)$ with $y \in \{0,1\}^{\lceil \log_2 \alpha \rceil}$ (for some $\alpha > 0$) we can view each $\mathsf{H}(y \circ \mathbb{S}_{\mathbb{C}}(x))$ as a fresh w-bit string. Thus, we can obtain a random $w\alpha$-bit string by concatenating all of the labels $\mathsf{H}(y \circ \mathbb{S}_{\mathbb{C}}(x))$ for each $y \in \{0,1\}^{\lceil \log_2 \alpha \rceil}$. This motivates the following definition of an *expansion family* which will be used in subsequent sections.

▶ **Definition 9** (Expansion Family)**.** *For random oracle $\mathsf{H}(\cdot)$ the expansion family of functions $\{\mathsf{E}_\alpha^{\mathsf{H}(\cdot)}\}_{\alpha=1}^{\infty}$ where each function $\mathsf{E}_\alpha^{\mathsf{H}(\cdot)} : \{0,1\}^* \to \{0,1\}^{\alpha w}$ is defined as $\mathsf{E}_\alpha^{\mathsf{H}(\cdot)}(x) = \mathsf{H}(1 \circ x) \circ \mathsf{H}(2 \circ x) \circ \cdots \circ \mathsf{H}(\alpha \circ x)$, where the prefix $i \in [\alpha]$ of x for each oracle query in the definition is expressed in binary using $\lceil \log_2 \alpha \rceil$ bits.*

3.2 Framework for LDCs against Resource Bounded Channels

Our aim in this section is to achieve LDCs having no asymptotic loss in rate, query complexity, or success probability of private locally decodable codes. In contrast to the private LDC setting, we assume no private (or public) key setup assumptions. We also aim for LDCs that may be used for multiple (polynomial) rounds of communication, a notion which we describe later in the section. Let $\mathsf{C}_{\mathsf{ldc}^*}$ be a LDC*and $\mathsf{C}_{\mathsf{priv}}$ be a private-LDC. Then, against classes of pROM algorithms permitting $\delta-$safe functions, our encoder will use $\mathsf{C}_{\mathsf{ldc}^*}$ to bootstrap off of $\mathsf{C}_{\mathsf{priv}}$ even in the absence of shared private randomness with the decoder.

Framework Overview: The encoding algorithm first samples a random seed r of modest length (k_{ldc^*}). By embedding an encoding of r (via $\mathsf{C}_{\mathsf{ldc}^*}$) in our final codeword, we can ensure that our decoder will also have access to r. Let the channel, over which the communication happens, belong to a class $\mathbb{C}$ of pROM algorithms (w.r.t. random oracle $\mathsf{H}(\cdot)$) permitting some δ-safe function $\mathbb{S}_{\mathbb{C}} : \{0,1\}^{k_{\mathsf{ldc}^*}} \to \{0,1\}^*$. Even though the channel has access to the seed r, it will be unable to compute $\mathbb{S}_{\mathbb{C}}(\mathsf{r})$ by definition of the safe function. Thus $\mathsf{H}(\mathbb{S}_{\mathbb{C}}(\mathsf{r}))$ is effectively a random string to the channel. We can expand this randomness via an expansion function (Definition 9), and use $\mathsf{GenKey}_{\mathsf{priv}}$ with this randomness to compute a key. The computed key is effectively secret from the channel and can be used in conjunction with $\mathsf{Enc}_{\mathsf{priv}}$ to obtain an encoding of any input message. Note that since the decoder also has access to r, it may also compute the secret key using exactly the same procedure and use this key in conjunction with $\mathsf{Dec}_{\mathsf{priv}}$ to perform the required decoding. Thus the use of $\mathsf{C}_{\mathsf{ldc}^*}$,

safe and expansion functions on a random seed reduces the setting to that of $\mathsf{C}_{\mathsf{priv}}$. Our framework is parameterized by $[\mathbb{S}_{\mathbb{C}}, \mathsf{C}_{\mathsf{ldc}^*}, \mathsf{C}_{\mathsf{priv}}]$.

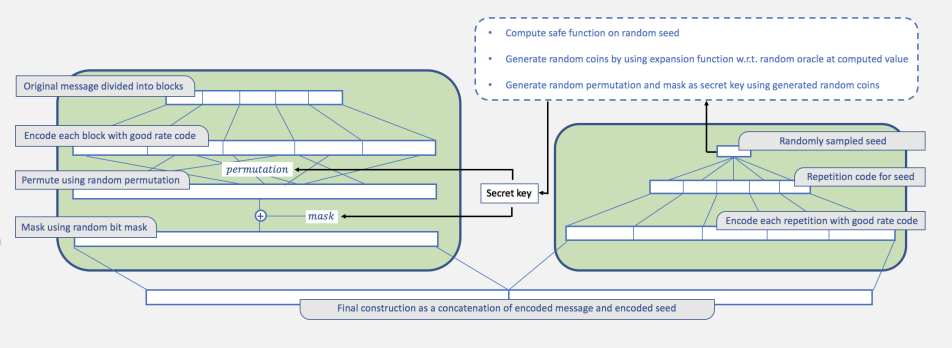


Figure 3 Instantiation of framework for LDCs against adversaries permitting safe functions.

Explicit Constructions: We provide explicit constructions of LDCs against adversarial pROM channels permitting $\delta-$safe functions by instantiating the framework discussed above. Figure 3 gives an overview of the instantiation. For private LDCs we use the constructions of Theorem 5. Furthermore, we instantiate $\mathsf{C}_{\mathsf{ldc}^*}$ as follows: The encoder encodes the seed with a standard constant rate error correcting code – we instantiate this with Justesen codes – composed with a repetition code. The local decoder then randomly samples seed-encodings and takes a majority vote over the decoded samples to determine the seed. We show that these encoding and decoding algorithms result in a (ℓ, ρ, p)-LDC^* where, for a parameter α and message of length k, $\ell = \mathcal{O}(\alpha k)$, $\rho = \mathcal{O}(1)$ and $p \geq 1 - e^{-\alpha}$. The prior results are obtainable for any desired codeword length $K = \Omega(k)$. We refer the reader to the full version [13] for a formal explanation of this LDC^* instantiation.

Detailed descriptions of our encoder $(\mathsf{Enc}_{\mathsf{final}}^{\mathsf{H}(\cdot)})$ and decoder $(\mathsf{Dec}_{\mathsf{final}}^{\mathsf{H}(\cdot)})$, given a message x, security parameter κ, and random oracle $\mathsf{H}(\cdot)$, may be described in Figure 4. In particular, our framework lead to the following theorem.

$\mathsf{Enc}_{\mathsf{final}}^{\mathsf{H}(\cdot)}(\mathbf{x}, \kappa)[\mathbb{S}_{\mathbb{C}}, \mathsf{C}_{\mathsf{ldc}^*}, \mathsf{C}_{\mathsf{final}}]$:

1. Sample a random seed of length k_{ldc^*}.
   $\mathsf{r} \leftarrow \{0,1\}^{k_{\mathsf{ldc}^*}}$
2. Encode random seed using an LDC.
   $Y_{\mathsf{ldc}^*} := \mathsf{Enc}_{\mathsf{ldc}^*}(\mathsf{r})$
3. Generate randomness uncomputable by channel via safe and expansion functions.
   $R := \mathsf{E}_{\tau}^{\mathsf{H}(\cdot)}(\mathbb{S}_{\mathbb{C}}(\mathsf{r}))$
4. Generate a secret key from the randomness.
   $\mathsf{sk}_{\mathsf{final}} := \mathsf{GenKey}_{\mathsf{priv}}(\kappa; R)$
5. Use private LDC encoder with generated key.
   $Y_{\mathsf{priv}} := \mathsf{Enc}_{\mathsf{priv}}(x, \kappa, \mathsf{sk}_{\mathsf{final}})$
6. **Output** $Y_{\mathsf{priv}} \circ Y_{\mathsf{ldc}^*}$

$\mathsf{Dec}_{\mathsf{final}}^{\mathsf{H}(\cdot), \mathbf{Y}'_{\mathsf{priv}} \circ \mathbf{Y}'_{\mathsf{ldc}^*}}(\mathbf{i}, \kappa)[\mathbb{S}_{\mathbb{C}}, \mathsf{C}_{\mathsf{ldc}^*}, \mathsf{C}_{\mathsf{final}}]$:

1. Decode the original random seed.
   $\mathsf{r} := \mathsf{Dec}_{\mathsf{ldc}^*}^{Y'_{\mathsf{ldc}^*}}(\cdot)$
2. Compute randomness used by encoder.
   $R := \mathsf{E}_{\tau}^{\mathsf{H}(\cdot)}(\mathbb{S}_{\mathbb{C}}(\mathsf{r}))$
3. Compute secret key used by encoder.
   $\mathsf{sk}_{\mathsf{final}} := \mathsf{GenKey}_{\mathsf{OPS}}(\kappa; R)$
4. Use private LDC decoder with computed key.
   Output $\mathsf{Dec}_{\mathsf{priv}}^{Y'_{\mathsf{priv}}}(i, \mathsf{sk}_{\mathsf{final}})$

Figure 4 Encoding and decoding algorithms for our LDC construction.

▶ **Theorem 10.** *Let* $\mathsf{C}_{\mathsf{ldc}^*}[K_{\mathsf{ldc}^*}, k_{\mathsf{ldc}^*}] = (\mathsf{Enc}_{\mathsf{ldc}^*}, \mathsf{Dec}_{\mathsf{ldc}^*})$ *be an* $(\ell_{\mathsf{ldc}^*}, \rho_{\mathsf{ldc}^*}, p_{\mathsf{ldc}^*})-\mathsf{LDC}^*$ *and* $\mathsf{C}_{\mathsf{priv}}[K_{\mathsf{priv}}, k_{\mathsf{priv}}, \kappa] = (\mathsf{Enc}_{\mathsf{priv}}, \mathsf{Dec}_{\mathsf{priv}}, \mathsf{GenKey}_{\mathsf{priv}})$ *be an* $(\ell_{\mathsf{priv}}, \rho_{\mathsf{priv}}, p_{\mathsf{priv}}, \epsilon_{\mathsf{priv}})-$*private* LDC. *Then for any class* $\mathbb{C}$ *of* pROM *algorithms admitting a* $\delta-$*safe function* $\mathbb{S}_{\mathbb{C}} : \{0,1\}^{k_{\mathsf{ldc}^*}} \to \{0,1\}^*$, *the* $(K_{\mathsf{final}}, k_{\mathsf{final}})_2$ *coding scheme in the random oracle model* $\mathsf{C}_{\mathsf{final}}[\mathbb{S}_{\mathbb{C}}, \mathsf{C}_{\mathsf{ldc}^*}, \mathsf{C}_{\mathsf{priv}}] = (\mathsf{Enc}^{\mathsf{H}(\cdot)}_{\mathsf{final}}, \mathsf{Dec}^{\mathsf{H}(\cdot)}_{\mathsf{final}})$ *is an* $(\ell_{\mathsf{final}}, \rho_{\mathsf{final}}, p_{\mathsf{final}}, \epsilon_{\mathsf{final}})$-LDC *with* $k_{\mathsf{final}} = k_{\mathsf{priv}}$, $K_{\mathsf{final}} = K_{\mathsf{ldc}^*} + K_{\mathsf{priv}}$, $\ell_{\mathsf{final}} = \ell_{\mathsf{final}} + \ell_{\mathsf{ldc}^*}$, $\rho_{\mathsf{final}} = \frac{1}{K_{\mathsf{ldc}^*} + K_{\mathsf{priv}}} \min\{\rho_{\mathsf{ldc}^*} K_{\mathsf{ldc}^*}, \rho_{\mathsf{priv}} K_{\mathsf{priv}}\}$, $p_{\mathsf{final}} \geq 1 - (2 - p_{\mathsf{priv}} - p_{\mathsf{ldc}^*})$, $\epsilon_{\mathsf{final}} \leq \epsilon_{\mathsf{priv}} + q\delta$. *Here* q *is an upper bound on the number of queries any algorithm* $\mathcal{A}^{\mathsf{H}(\cdot)} \in \mathbb{C}$ *makes to the random oracle* $\mathsf{H}(\cdot)$.

The final codeword generated by $\mathsf{Enc}^{\mathsf{H}(\cdot)}_{\mathsf{final}}$ is simply the concatenation of the codewords generated by $\mathsf{Enc}_{\mathsf{priv}}$ and $\mathsf{Enc}_{\mathsf{ldc}^*}$, resulting in $K_{\mathsf{final}} = K_{\mathsf{ldc}^*} + K_{\mathsf{priv}}$. By construction, the only queries $\mathsf{Dec}^{\mathsf{H}(\cdot)}_{\mathsf{final}}$ makes to the corrupted codeword are during the executions of $\mathsf{Dec}_{\mathsf{ldc}^*}$ and $\mathsf{Dec}_{\mathsf{priv}}$. This gives the locality $\ell_{\mathsf{final}} = \ell_{\mathsf{ldc}^*} + \ell_{\mathsf{priv}}$. Furthermore for correct overall decoding, it is necessary that the individual codes are correctly decoded. Thus the total errors that the code can tolerate is bounded by the maximum number of errors any individual one of the codes can tolerate. This gives the claimed (worst case) error rate. We emphasize that the proofs of the bounds on the decoder's success probability and the security of the framework is much more involved than the above discussion and is included in Section 3.3 and 3.4. In particular, we show that no adversary admitting δ-safe functions can distinguish between the encodings of $\mathsf{Enc}^{\mathsf{H}(\cdot)}_{\mathsf{final}}$ and those of $\mathsf{Enc}_{\mathsf{priv}}$ with random strings appended to them. Furthermore, even the decoder, who has no computational restrictions and gets the appropriate secret key used during the respective encoding processes may not make this distinction, thereby effectively reducing the security of $\mathsf{C}_{\mathsf{final}}$ to that of $\mathsf{C}_{\mathsf{priv}}$ with negligible loss. The following two corollaries exhibit decoding probability vs locality tradeoffs when our framework is instantiated with LDC^*s and private-LDCs from the discussion of explicit constructions earlier in the section. We defer details of these to the full version [13].

▶ **Corollary 11.** *For security parameter* κ, *a class* $\mathbb{C}$ *of* pROM *adversaries admitting* $\delta-$*safe function* $\mathbb{S}_{\mathbb{C}} : \{0,1\}^{\log^{1+\varepsilon}\kappa} \to \{0,1\}^*$ *where* $\varepsilon > 0$ *and for every* $k > 0$ *such that* $k = \mathrm{poly}(\kappa)$ *where* poly *is any non-zero polynomial, there exists a* $(\beta k, k)_2$ *coding scheme in the random oracle model that is an* $(\ell, \rho, p, \epsilon, \mathbb{C})-\mathsf{LDC}$ *where* $\ell = (\alpha+1)\log^{1+\varepsilon}\kappa$ *(such that* $\alpha \geq 17$*),* ρ *is a constant,* p *is a constant dependent on* α, *and* $\epsilon \leq \mathsf{negl}(\kappa) + q\delta$. *Here* β *is a constant,* $\mathsf{negl}(\kappa)$ *is a negligible function of* κ *and* q *is an upper bound on the total queries any algorithm in* $\mathbb{C}$ *makes to the random oracle.*

▶ **Corollary 12.** *For security parameter* κ, *a class* $\mathbb{C}$ *of* pROM *adversaries admitting* $\delta-$*safe function* $\mathbb{S}_{\mathbb{C}} : \{0,1\}^{\log^{1+\varepsilon}\kappa} \to \{0,1\}^*$ *where* $\varepsilon > 0$ *and for every* $k > 0$ *such that* $k = \mathrm{poly}(\kappa)$ *where* poly *is any non-zero polynomial, there exists a* $(\beta k, k)_2$ *coding scheme in the random oracle model that is an* $(\ell, \rho, p, \epsilon, \mathbb{C})-\mathsf{LDC}$ *where* $\ell = (1+24\log^{1+\varepsilon}\kappa)\log^{1+\varepsilon}\kappa$, ρ *is a constant,* $p \geq (1 - \mathsf{negl}_1(\kappa))$, *and* $\epsilon \leq \mathsf{negl}_2(\kappa) + q\delta$. *Here* β *is a constant,* $\mathsf{negl}_1(\kappa)$ *and* $\mathsf{negl}_2(\kappa)$ *are negligible functions of* κ, *and* q *is an upper bound on the total queries any algorithm in* $\mathbb{C}$ *makes to the random oracle.*

Precomputation. We remark that Steps 1-4 of $\mathsf{Enc}^{\mathsf{H}(\cdot)}_{\mathsf{final}}$ may be precomputed. This may be advantageous in some settings to speed up encoding time as the sender may precompute multiple $(\mathsf{sk}_{\mathsf{final}}, Y_{\mathsf{ldc}^*})$ pairs. When a message is ready to be encoded, the sender then simply needs to generate Y_{priv} using $\mathsf{sk}_{\mathsf{final}}$ and append Y_{ldc^*} to generate the final codeword. However, we do note that this precomputation must be done after the selection of the random oracle, and that such precomputation is not possible for $\mathsf{Dec}^{\mathsf{H}(\cdot)}_{\mathsf{final}}$.

Multi-round Communication. Existing constructions of private LDCs [33] are secure only for a single *round* of communication (see the full version [13] for details on the round-based game between the encoder/decoder and the channel in the private LDC setting). We may generalize our model to be in terms of rounds as well, where each round runs an instance of the experiment $\mathtt{LDC}-\mathtt{Sec}-\mathtt{Game}$ defined in Section 2.4. We remark that our codes work for this generalized model as well. In every round of the experiment, the encoder can sample a fresh random seed r. This is not directly possible in the existing private LDC constructions as an attacker listening to the decoder's queries may learn information about the secret key after a single round of communication. For this Ostrovsky et al. introduce a new construction which hides the secret key behind a layer of encryption, which in turn increases the locality of their final constructions to $\omega(\log^2 \kappa)$.

3.3 Two-Phase Hybrid Distinguisher Argument

To prove the security of the LDC framework in section 3.2, our approach is to argue the following: if any channel wins the `LDC-Sec-Game` against an instantiation of our LDC constructions $(\mathsf{Enc}_{\mathsf{final}}^{\mathsf{H}(\cdot)}, \mathsf{Dec}_{\mathsf{final}}^{\mathsf{H}(\cdot)})$, then this channel can win the `priv-LDC-Sec-Game` against its constituent private-LDC (contradicting its security guarantee).

Standard Hybrid Argument Failure: A natural attempt to prove this, yet one that fails, is to use the following standard hybrid argument. In the first hybrid we use our original encoding scheme $\mathsf{Enc}_{\mathsf{final}}^{\mathsf{H}(\cdot)}$ to obtain a codeword $\mathsf{Y}_{\mathsf{priv}}{}^{(0)} \circ \mathsf{Y}_{\mathsf{ldc}}{}^{(0)}$. In the second hybrid, we replace the second component with an encoding of a random unrelated nonce to get $\mathsf{Y}_{\mathsf{priv}}{}^{(1)} \circ \mathsf{Y}_{\mathsf{ldc}^*}{}^{(1)}$. Here $\mathsf{Y}_{\mathsf{ldc}^*}{}^{(1)}$ is an encoding of some random nonce which is sampled completely independent of the message encoding $\mathsf{Y}_{\mathsf{priv}}{}^{(1)}$. We would like to argue that the two hybrids are indistinguishable and conclude that a resource bounded channel cannot fool the local decoder from original encoding scheme (first hybrid) – since we cannot fool the private-LDC local decoder in the second hybrid. However, if the distinguisher $\mathcal{D}$ is able to evaluate the safe-function then the hybrids are trivially distinguishably. On the other hand, if we assume that the distinguisher $\mathcal{D}$ is resource bounded like the channel then indistinguishability does not suffice to argue that the local decoder i.e., fooling the decoder does not yield a resource bounded distinguisher $\mathcal{D}$ since the decoder is not constrained in the same way as the resource bounded channel.

Two-Phase Argument Overview: We address the previous issue by introducing a *two-phase distinguisher game* defined over adversary/distinguisher pairs. In the first phase of this game, a random coin toss $b \in \{0,1\}$ randomly selects one of the hybrid encoders to encode a message. The selected hybrid hands its encoding $\mathsf{Y}_{\mathsf{priv}}{}^{(b)} \circ \mathsf{Y}_{\mathsf{ldc}^*}{}^{(b)}$ to the adversary $\mathcal{A}^{\mathsf{H}(\cdot)}$ which outputs a corrupted codeword $\mathsf{Y}_{\mathsf{hyb}}{}^{(b)'}$. In the second phase, the distinguisher $\mathcal{D}$ is given the initial message x, the corrupted codeword $\mathsf{Y}_{\mathsf{hyb}}{}^{(b)'}$, along with the secret key $\mathsf{sk}^{(b)}$ used to obtain $\mathsf{Y}_{\mathsf{priv}}{}^{(b)}$, and tries to predict the value of b, i.e., which hybrid encoder was used. An important point to note is that $\mathcal{D}$ is not constrained in any way. However, it is not given access to the random oracle. We show (Lemma 14) that for any such attacker-distinguisher pair, the distinguisher succeeds at guessing which hybrid encoding was used with at most negligible probability. The two phase hybrid argument allows us to reason about our original goal: the probability that the channel fools the honest decoder. In particular, a channel that wins the `LDC-Sec-Game` with non-negligible probability can be used in phase 1 in conjunction with a distinguisher that can simulate the decoding algorithm (with the correct key) in phase 2 to distinguish between the hybrids with non-negligible probability. This gives the required contradiction (Lemma 15). We formally define the two hybrid encoders in Figure 5.

$\mathsf{Enc}_{\mathbf{0}}^{\mathsf{H}(\cdot)}(\mathbf{x}, \kappa)[\mathbb{S}_{\mathbb{C}}, \mathsf{C}_{\mathsf{ldc}^*}, \mathsf{C}_{\mathsf{priv}}]$: (**same as Figure 3**)

1. Sample a random seed of length k_{ldc^*}.
   $\mathsf{r}^{(0)} \leftarrow \{0,1\}^{k_{\mathsf{ldc}^*}}$
2. Encode random seed using an LDC^*.
   $\mathsf{Y}_{\mathsf{ldc}}{}^{(0)} := \mathsf{Enc}_{\mathsf{ldc}^*}(\mathsf{r}^{(0)})$
3. Generate randomness uncomputable by channel via safe and expansion functions.
   $\mathsf{R}^{(0)} := \mathsf{E}_{\tau}^{\mathsf{H}(\cdot)}(\mathbb{S}_{\mathbb{C}}(\mathsf{r}^{(0)}))$
4. Generate a secret key from the randomness.
   $\mathsf{sk}^{(0)} := \mathsf{GenKey}_{\mathsf{priv}}(\kappa; \mathsf{R}^{(0)})$
5. Use private LDC encoder with generated key.
   $\mathsf{Y}_{\mathsf{priv}}{}^{(0)} := \mathsf{Enc}_{\mathsf{priv}}(x, \kappa, \mathsf{sk}^{(0)})$
6. **Output** $\mathsf{Y}_{\mathsf{priv}}{}^{(0)} \circ \mathsf{Y}_{\mathsf{ldc}}{}^{(0)}$

$\mathsf{Enc}_{\mathbf{1}}(\mathbf{x}, \mathsf{sk}^{(\mathbf{1})}, \kappa)[\mathbb{S}_{\mathbb{C}}, \mathsf{C}_{\mathsf{ldc}^*}, \mathsf{C}_{\mathsf{priv}}]$:

1. Sample a random seed of length k_{ldc^*}.
   $\mathsf{r}^{(1)} \leftarrow \{0,1\}^{k_{\mathsf{ldc}^*}}$
2. Encode random seed using an LDC^*.
   $\mathsf{Y}_{\mathsf{ldc}^*}{}^{(1)} := \mathsf{Enc}_{\mathsf{ldc}^*}(\mathsf{r}^{(1)})$
3. Use private LDC encoder with input key.
   $\mathsf{Y}_{\mathsf{priv}}{}^{(1)} := \mathsf{Enc}_{\mathsf{priv}}(x, \kappa, \mathsf{sk}^{(1)})$
4. **Output** $\mathsf{Y}_{\mathsf{priv}}{}^{(1)} \circ \mathsf{Y}_{\mathsf{ldc}^*}{}^{(1)}$

Figure 5 Hybrid encoding algorithms. By design, $\mathsf{Enc}_0^{\mathsf{H}(\cdot)}$ is the same as our proposed LDC construction.

Let $\mathcal{A}^{\mathsf{H}(\cdot)}$ be an adversarial channel belonging to a class $\mathbb{C}$ of pROM algorithms w.r.t random oracle $\mathsf{H}(\cdot)$ permitting $\delta-$safe functions. Furthermore, let $\mathcal{D} : \left(\{0,1\}^*\right)^4 \to \{0,1\}$ be a computationally unbounded algorithm. We will term $\mathcal{A}^{\mathsf{H}(\cdot)}$ and $\mathcal{D}$ as *attacker* and *distinguisher* respectively. Using the *hybrid* encoders in Figure 5, we define the *indistinguishability experiment* $\mathtt{Exp}_{\mathcal{A},\mathcal{D},\mathsf{H},\kappa,x}$ over all *attacker-distinguisher pairs* $(\mathcal{A}^{\mathsf{H}(\cdot)}, \mathcal{D})$. Note that in this experiment, $\mathcal{D}$ is provided with the secret key that the selected hybrid used during encoding, and does not have access to the random oracle. With respect to this experiment, we define the *advantage* of the attacker-distinguisher pair as follows:

$$\mathsf{Adv}_{\mathcal{A},\mathcal{D}} := \max_{x} \left| \Pr[\mathtt{Exp}_{\mathcal{A},\mathcal{D},\mathsf{H},\kappa,x} = 1] - \frac{1}{2} \right|$$

where the probability is taken over the randomness of $\mathcal{D}$, $\mathcal{A}^{\mathsf{H}(\cdot)}$, and the selection of the random oracle $\mathsf{H}(\cdot)$. Our first aim will be to show that the advantage of any attacker-distinguisher pair, as defined above, is negligible at best.

Let $(\mathcal{A}^{\mathsf{H}(\cdot)}, \mathcal{D})$ be any attacker-distinguisher pair and hybrid encoders be instantiated with parameters $[\mathbb{S}_{\mathbb{C}}, \mathsf{C}_{\mathsf{ldc}^*}, \mathsf{C}_{\mathsf{priv}}]$. For security parameter κ and message x, consider an execution of the indistinguishability experiment $\mathtt{Exp}_{\mathcal{A},\mathcal{D},\mathsf{H},\kappa,x}$. Let $\mathsf{bad}_{\mathcal{A}}$ be the event that the attacker queries the random oracle at position $c \circ \mathbb{S}_{\mathbb{C}}(\mathsf{r}^{(b)})$ where r is the random seed chosen by the selected hybrid encoder $\mathsf{Enc}_b^{\mathsf{H}(\cdot)}$ and c is any constant expressed in binary. Furthermore, let succ be the event where the attacker-distinguisher pair succeed in distinguishing the hybrid encodings in the experiment, i.e., the event where $\mathtt{Exp}_{\mathcal{A},\mathcal{D},\mathsf{H},\kappa,x} = 1$.

The next proposition follows from the observation that conditioning on the event $\mathsf{bad}_{\mathcal{A}}$ not occurring, the secret key sk_b used during the encoding process remains (information theoretically) private to both the adversary and the distinguisher. To the pair, $\mathsf{Enc}_0^{\mathsf{H}(\cdot)}$ appears information theoretically identical to Enc_1 which gets a secret key as its input, and thus any advantage on distinguishing the encoding schemes would allow the pair to distinguish between random strings.

▶ **Proposition 13.** $\Pr[\mathsf{succ} | \overline{\mathsf{bad}}_{\mathcal{A}}] = 1/2$

$\mathtt{Exp}_{\mathcal{A},\mathcal{D},\mathsf{H},\kappa,x}$: \\message x and security parameter κ:

Phase I

1. Encode message with both hybrids. Let $\mathsf{sk}^{(0)}$ and $\mathsf{sk}^{(1)}$ be the secret keys used by first and second hybrid respectively.
   $\mathsf{Y_{hyb}}^{(0)} := \mathsf{Enc}_0^{\mathsf{H}(\cdot)}(x,\kappa)$.
   $\mathsf{Y_{hyb}}^{(1)} := \mathsf{Enc}_1(x,\kappa,\mathsf{sk}^{(1)})$.
2. Flip an unbiased coin to randomly select a hybrid encoding.
   $b \leftarrow \{0,1\}$
3. Hand the selected encoding to the channel to get corrupted codeword.
   $\mathsf{Y_{hyb}}^{(b)'} := \mathcal{A}^{\mathsf{H}(\cdot)}(x,\kappa,\mathsf{Y_{hyb}}^{(b)})$

Phase II

1. Distinguisher, given the message, secret key, corrupted codeword, and security parameter, guesses the coin toss.
   $b' := \mathcal{D}(x,\mathsf{sk}^{(b)},\mathsf{Y_{hyb}}^{(b)'},\kappa)$
2. $\mathtt{Exp}_{\mathcal{A},\mathcal{D},\mathsf{H},x,\kappa} = \begin{cases} 1 & \text{iff } b' = b \\ 0 & \text{otherwise} \end{cases}$

Figure 6 Indistinguishability experiment for the attacker-distinguisher pair.

The following lemma shows that the advantage for any attacker-distinguisher pair is negligible.

▶ **Lemma 14.** $\mathsf{Adv}_{\mathcal{A},\mathcal{D}} \leq \frac{q\delta}{2}$ *for any execution of the game* $\mathtt{Exp}_{\mathcal{A},\mathcal{D},\mathsf{H},x,\kappa}$. *Here* q *is an upper bound on the number of queries* $\mathcal{A}^{\mathsf{H}(\cdot)}$ *makes to the random oracle.*

Proof. Consider some execution of the game $\mathtt{Exp}_{\mathcal{A},\mathcal{D},\mathsf{H},x,\kappa}$. Using conditional probability to partition the event space, the advantage of the attacker-distinguisher pair is:

$$\mathsf{Adv}_{\mathcal{A},\mathcal{D}} = \left|\Pr[\mathsf{succ}] - \frac{1}{2}\right| = \left|\Pr[\mathsf{succ}|\mathsf{bad}_{\mathcal{A}}]\Pr[\mathsf{bad}_{\mathcal{A}}] + \Pr[\mathsf{succ}|\overline{\mathsf{bad}}_{\mathcal{A}}]\Pr[\overline{\mathsf{bad}}_{\mathcal{A}}] - \frac{1}{2}\right|$$

By Proposition 13, we may view the event of succ conditioned on $\mathsf{bad}_{\mathcal{A}}$ not occurring as an unbiased random choice. Thus $\mathsf{Adv}_{\mathcal{A},\mathcal{D}} = \left|\Pr[\mathsf{succ}|\mathsf{bad}_{\mathcal{A}}]\Pr[\mathsf{bad}_{\mathcal{A}}] + \frac{1}{2}(1-\Pr[\mathsf{bad}_{\mathcal{A}}]) - \frac{1}{2}\right|$. This allows us to bound the advantage of the attacker-distinguisher pair by a factor of the probability of event bad occurring by $\mathsf{Adv}_{\mathcal{A},\mathcal{D}} = \Pr[\mathsf{bad}_{\mathcal{A}}]\left|\Pr[\mathsf{succ}|\mathsf{bad}_{\mathcal{A}}] - \frac{1}{2}\right| \leq \Pr[\mathsf{bad}_{\mathcal{A}}]\frac{1}{2}$. Therefore by Lemma 8, the advantage of the attacker-distinguisher pair for the execution of $\mathtt{Exp}_{\mathcal{A},\mathcal{D},\mathsf{H},x,\kappa}$ is at most $\frac{q\delta}{2}$. ◀

3.4 Security and Decoding Probability of Constructions

Note that $\mathsf{Enc}_0^{\mathsf{H}(\cdot)}$ is identical to $\mathsf{Enc}_{\mathsf{final}}^{\mathsf{H}(\cdot)}$ and Enc_1 is identical to $\mathsf{Enc}_{\mathsf{priv}}$ with random strings appended to its output. Consider a $(\ell_{\mathsf{priv}},\rho_{\mathsf{priv}},p_{\mathsf{priv}},\epsilon_{\mathsf{priv}})-$private LDC instance $\mathsf{C}_{\mathsf{priv}}[k_{\mathsf{priv}},K_{\mathsf{priv}}] = (\mathsf{Enc}_{\mathsf{priv}},\mathsf{Dec}_{\mathsf{priv}},\mathsf{GenKey}_{\mathsf{priv}})$ and an instantiation of our constructions $\mathsf{C}_{\mathsf{final}}[\mathbb{S}_{\mathsf{C}},\mathsf{C}_{\mathsf{ldc}^*},\mathsf{C}_{\mathsf{final}}] = (\mathsf{Enc}_{\mathsf{final}}^{\mathsf{H}(\cdot)},\mathsf{Dec}_{\mathsf{final}}^{\mathsf{H}(\cdot)})$. With respect to these instances, we define $\epsilon_{\mathsf{final}}$ as the following:

$$\epsilon_{\mathsf{final}} := \Pr[\mathtt{LDC}-\mathtt{Sec}-\mathtt{Game}[\mathcal{A},x,\mathsf{H},\kappa,\rho,p] = 1 \text{ against } \mathsf{C}_{\mathsf{final}}]$$

Consider the codes $\mathsf{C}_0 = (\mathsf{Enc}_0^{\mathsf{H}(\cdot)},\mathsf{Dec}_{\mathsf{final}}^{\mathsf{H}(\cdot)})$ and $\mathsf{C}_1 = (\mathsf{Enc}_1,\mathsf{Dec}_{\mathsf{priv}^*})$ formed by our hybrid encoders. Here $\mathsf{Dec}_{\mathsf{priv}^*}$ is defined identical to $\mathsf{Dec}_{\mathsf{priv}}$ except that it ignores the strings

appended to the output of $\mathsf{Enc}_{\mathsf{priv}}$ during the encoding execution of Enc_1. With respect to these codes, we define the following:

$$\epsilon_0 := \max_{\mathcal{A}^{\mathsf{H}(\cdot)} \in \mathbb{C}} \Pr[\mathtt{priv} - \mathtt{LDC} - \mathtt{Sec} - \mathtt{Game}[\mathcal{A}, x, \kappa, \rho_{\mathsf{final}}, p_{\mathsf{final}}] = 1 \text{ against } \mathsf{C}_0]$$

$$\epsilon_1 := \max_{\mathcal{A}^{\mathsf{H}(\cdot)} \in \mathbb{C}} \Pr[\mathtt{priv} - \mathtt{LDC} - \mathtt{Sec} - \mathtt{Game}[\mathcal{A}, x, \kappa, \rho_{\mathsf{final}}, p_{\mathsf{final}}] = 1 \text{ against } \mathsf{C}_1]$$

Note that by our definitions, $\epsilon_0 = \epsilon_{\mathsf{final}}$ and $\epsilon_1 \leq \epsilon_{\mathsf{priv}}$. The second observation follows from the following:

$$\begin{aligned}
\epsilon_1 &= \max_{\mathcal{A}^{\mathsf{H}(\cdot)} \in \mathbb{C}} \Pr[\mathtt{priv} - \mathtt{LDC} - \mathtt{Sec} - \mathtt{Game}[\mathcal{A}, x, \kappa, \rho_{\mathsf{final}}, p_{\mathsf{final}}] = 1 \text{ against } \mathsf{C}_1] \\
&\leq \max_{\mathcal{A}^{\mathsf{H}(\cdot)} \in \mathbb{C}} \Pr[\mathtt{priv} - \mathtt{LDC} - \mathtt{Sec} - \mathtt{Game}[\mathcal{A}, x, \kappa, \rho_{\mathsf{final}}, p_{\mathsf{priv}}] = 1 \text{ against } \mathsf{C}_1] \\
&\leq \max_{\mathcal{A} \in \mathbb{C}} \Pr[\mathtt{priv} - \mathtt{LDC} - \mathtt{Sec} - \mathtt{Game}[\mathcal{A}, x, \kappa, \rho_{\mathsf{priv}}, p_{\mathsf{priv}}] = 1 \text{ against } \mathsf{C}_{\mathsf{priv}}] = \epsilon_{\mathsf{priv}}
\end{aligned}$$

where the first inequality follows because $p_{\mathsf{final}} \leq p_{\mathsf{priv}}$, while the second inequality follows since $\rho_{\mathsf{final}} K_{\mathsf{final}} \leq \rho_{\mathsf{priv}} K_{\mathsf{priv}}$ i.e., the attacker gets to make more corruptions against $\mathsf{C}_{\mathsf{priv}}$. Lemma 15 upper bounds $|\epsilon_0 - \epsilon_1| \leq q\delta$ and it immediately follows that $\epsilon_{\mathsf{final}} \leq \epsilon_{\mathsf{priv}} + q\delta$.

▶ **Lemma 15.** *$|\epsilon_0 - \epsilon_1| \leq q\delta$. Here q is an upper bound on the number of queries the attacker makes to the random oracle.*

Proof. Recall that an attacker wins the $\mathtt{LDC} - \mathtt{Sec} - \mathtt{Game}[\mathcal{A}, x, \mathsf{H}, \kappa, \rho, p]$ if there exists some index which the corresponding decoder fails to decode with probability at least p. Suppose for sake of contradiction that $|\epsilon_0 - \epsilon_1| > q\delta$ for some attacker $\mathcal{A}^{\mathsf{H}(\cdot)}$. Consider the distinguisher D' in Figure 7. With respect to the indistinguishability experiment, D' takes as input the original message x, the corrupted codeword y'_b, the key used by hybrid b during encoding, and the security parameter κ.

Distinguisher $\mathsf{D}'(x, y'_b, \mathsf{sk}_b, \kappa)$:

1. Computes ϵ_b by enumerating over all i, running $\mathsf{Dec}^{y'_b}_{\mathsf{priv}}(i, \kappa)$ and checking whether $\mathsf{Dec}_{\mathsf{priv}}$ fails to decode correctly with probability at least p_{priv}.
2. return $b' = \begin{cases} 1 & \text{with probability } \epsilon_b \\ 0 & \text{otherwise} \end{cases}$

Figure 7 Distinguisher that uses the $\mathsf{Dec}_{\mathsf{priv}}$ decoding algorithm.

Note that the computationally intensive step 1 of D' is possible since we assume no computational restrictions. Thus by conditional probability, the advantage of distinguisher D' paired with any $\mathcal{A}^{\mathsf{H}(\cdot)} \in \mathbb{C}$ may be given by

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A},\mathcal{D}'} &= \left| \Pr[\mathsf{succ}] - \frac{1}{2} \right| = \frac{1}{2} \left| \Pr[\mathsf{succ} | b = 0] - \Pr[\overline{\mathsf{succ}} | b = 1] \right| \\
&= \frac{1}{2} \left| (1 - \epsilon_0) - (1 - \epsilon_1) \right| = \frac{1}{2} |\epsilon_1 - \epsilon_0| \,,
\end{aligned}$$

where the penultimate equality is by definition of the distinguisher D'. Our initial assumption $|\epsilon_0 - \epsilon_1| > q\delta$ then implies that $\mathsf{Adv}_{\mathcal{A},\mathcal{D}} > \frac{q\delta}{2}$, contradicting Lemma 14. ◀

The following proposition is a direct consequence of Lemma 15 and the observation that $\epsilon_1 \leq \epsilon_{\mathsf{priv}}$.

▶ **Proposition 16.** *$\epsilon_0 \leq \epsilon_{\mathsf{priv}} + q\delta$ where q is an upper bound to the number of queries that the attacker makes to the random oracle.*

Finally, we complete the proof by showing that that $\epsilon_{\mathsf{final}} \leq \epsilon_0$ in Lemma 17. Combined with proposition 16 this completes the proof since $\epsilon_{\mathsf{final}} \leq \epsilon_0 + q\delta$.

▶ **Lemma 17.** $\epsilon_{\mathsf{final}} \leq \epsilon_0$

Proof. Let fail_i denote the event that $\mathsf{Dec}_{\mathsf{final}}^{\mathsf{H}(\cdot)}$ incorrectly decodes x_i for $i \in [k]$. We define succ to be the event that $\mathtt{priv} - \mathtt{LDC} - \mathtt{Sec} - \mathtt{Game}[\mathcal{A}, x, \kappa, \rho_{\mathsf{final}}, p_{\mathsf{final}}] = 1$ against C_0 to simplify notation. It suffices to argue that $\Pr[\mathsf{fail}_i|\overline{\mathsf{succ}}] \leq (1 - p_{\mathsf{priv}}) + (1 - p_{\mathsf{ldc}^*})$ for any $i \in [k]$ since $\Pr[\mathsf{succ}] = \epsilon_0$. Let key be the event that $\mathsf{Dec}_{\mathsf{final}}^{\mathsf{H}(\cdot)}$ recovers the correct seed $\mathsf{r}^{(0)}$ from ${\mathsf{Y}_{\mathsf{ldc}}}^{(0)}$. We first observe that

$$\begin{aligned}\Pr[\mathsf{fail}_i|\overline{\mathsf{succ}}] &= \Pr[\mathsf{fail}_i|\overline{\mathsf{succ}}, \mathsf{key}]\Pr[\mathsf{key}|\overline{\mathsf{succ}}] + \Pr[\mathsf{fail}_i|\overline{\mathsf{succ}}, \overline{\mathsf{key}}]\Pr[\overline{\mathsf{key}}|\overline{\mathsf{succ}}] \\ &\leq \Pr[\mathsf{fail}_i|\overline{\mathsf{succ}}, \mathsf{key}] + \Pr[\overline{\mathsf{key}}|\overline{\mathsf{succ}}]\end{aligned}$$

Second we observe that $\Pr[\overline{\mathsf{key}}|\overline{\mathsf{succ}}] \leq 1 - p_{\mathsf{ldc}^*}$ since there are at most $\rho_{\mathsf{final}}K_{\mathsf{final}} \leq \rho_{\mathsf{ldc}^*}K_{\mathsf{ldc}^*}$ errors in the second part of the codeword ${\mathsf{Y}_{\mathsf{ldc}}}^{(0)}$. Finally, observe that by definition we have $\Pr[\mathsf{fail}_i|\overline{\mathsf{succ}}, \mathsf{key}] \leq 1 - p_{\mathsf{priv}}$. The claim now directly follows. ◀

4 Constructing Safe Functions

In this section we provide several examples of safe functions in the parallel random oracle model (pROM) [6]. We first define the parallel random oracle model and introduce several cost metrics that measure the resources used by a pROM algorithm $\mathcal{A}^{\mathsf{H}(\cdot)}$.

4.1 Parallel Random Oracle Model

Computation in the pROM proceeds in rounds. Each round ends when the algorithm $\mathcal{A}$ outputs a batch of random oracle queries to be answered in parallel and a new round begins when the attacker receives the answer(s) to this batch of queries. In between rounds the $\mathcal{A}$ may perform arbitrary computation. Formally, in the initial round the pROM algorithm $\mathcal{A}$ takes input x, performs some arbitrary computation, and outputs a state σ_1 and list $\vec{u}_1 = (u_1^1, \ldots, u_{q_1}^1)$ of random oracle queries. In general, we then have $(\vec{u}_{i+1}, \sigma_{i+1}) = \mathcal{A}(\sigma_i, \vec{a}_i)$ where $\vec{a}_i = (\mathsf{H}(u_1^i), \ldots, \mathsf{H}(u_{q_i}^i)$ are the answers to the q_i random oracle queries $\vec{u}_i = (u_1^i, \ldots, u_{q_i}^i)$ asked in the previous round. The execution ends in round t if the algorithm $\mathcal{A}$ returns an output value $y = \sigma_t$ along with an empty batch of random oracle queries $\vec{u}_t = \emptyset$. We use $\mathsf{Trace}_{\mathcal{A},R,\mathsf{H}}(x) = (\sigma_1, \sigma_2 \cdots, \sigma_t, \vec{u}_1, \ldots, \vec{u}_t)$ to denote the sequence of states (and oracle queries) output when we run the pROM attacker $\mathcal{A}(x)$ on input x fixing the random oracle $\mathsf{H}(\cdot)$ and fixing $\mathcal{A}$'s random coins R.

Cost Metrics. Figure 8 defines the *resources* we will consider as characterizing the cost of a particular execution trace $\mathcal{T} = \mathsf{Trace}_{\mathcal{A},R,\mathsf{H}}(x)$. We can define the time (resp. space) cost as $\mathsf{time}(\mathcal{T}) = t$ (resp. $\mathsf{space}(\mathcal{T}) = \max_{i \leq t} |\sigma_i|$). Similarly, the space time cost measures the product $\mathsf{space} - \mathsf{time}(\mathcal{T}) = t \cdot \max_{i \leq t} |\sigma_i|$ and cumulative memory complexity measures $\mathsf{CMC}(\mathcal{T}) = \sum_{i=0}^{t} |\sigma_i|$. Intuitively, cumulative memory complexity captures the amortized space time complexity of a function that we want to evaluate many times in parallel [6]. Finally, the cumulative query cost is $\mathsf{CQ}(\mathcal{T}) = \sum_{i=1}^{t} |\vec{u}_i|$. For a resource $\mathcal{R}$ listed in Figure 8, the term $\mathcal{R}$ *complexity* will refer to a upper bound on resource $\mathcal{R}$.

Resource	Notation	Definition
Time	$\mathsf{time}(\mathcal{T})$	t
Space	$\mathsf{space}(\mathcal{T})$	$\max_{i=0}^{t} \lvert\sigma_i\rvert$
Space-Time	$\mathsf{ST}(\mathcal{T})$	$\mathsf{space}(\mathcal{T}) \cdot \mathsf{time}(\mathcal{T})$
Cumulative memory	$\mathsf{CMC}(\mathcal{T})$	$\sum_{i=0}^{t} \lvert\sigma_i\rvert$
Cumulative query	$\mathsf{CQ}(\mathcal{T})$	$\sum_{i=0}^{t} \vec{u}_i$

Figure 8 Resource Definitions.

▶ **Definition 18.** *(Resource Bounded Algorithms) We use $\mathcal{C}_{\mathsf{CQ},q}$ to refer to the set of all pROM algorithms $\mathcal{A}$ with the property that for all inputs x, random oracles $\mathsf{H}(\cdot)$, and all random strings R, we have $\mathsf{CQ}(\mathsf{Trace}_{\mathcal{A},R,\mathsf{H}}(x)) \leq q$. We use $\mathcal{C}_{\mathsf{space},M,} \subset \mathcal{C}_{\mathsf{CQ},q}$ to refer to the subset of all pROM algorithms $\mathcal{A}$ with the additional constraint that for all inputs x, random oracles $\mathsf{H}(\cdot)$, and all random strings R, we have $\mathsf{CQ}(\mathsf{Trace}_{\mathcal{A},R,\mathsf{H}}(x)) \leq q$ and $\mathsf{space}(\mathsf{Trace}_{\mathcal{A},R,\mathsf{H}}(x)) \leq M$. Similarly, $\mathcal{C}_{\mathsf{time},T,q} \subset \mathcal{C}_{\mathsf{CQ},q}$ (resp. $\mathcal{C}_{\mathsf{space-time},S,q} \subset \mathcal{C}_{\mathsf{CQ},q}$) refers to the subset of all pROM algorithms $\mathcal{A}$ with the additional constraint that for all inputs x, random oracles $\mathsf{H}(\cdot)$, and all random strings R, we have $\mathsf{time}(\mathsf{Trace}_{\mathcal{A},R,\mathsf{H}}(x)) \leq T$ (resp. $\mathsf{space} - \mathsf{time}(\mathsf{Trace}_{\mathcal{A},R,\mathsf{H}}(x)) \leq S$). The definition of $\mathcal{C}_{\mathsf{CMC},M,q}$ is symmetric – we add the additional constraint that $\mathsf{CMC}(\mathsf{Trace}_{\mathcal{A},R,\mathsf{H}}(x)) \leq M$ for all $x, R, \mathsf{H}(\cdot)$.*

The assumption that the channel is resource constrained with respect to one or more of the above resources (time, space, cmc, etc.) is natural in most real word settings. For example, if a low latency channel uses $\mathcal{A}^{\mathsf{H}(\cdot)}$ to compute the corruptions to an encoded message then we can plausibly assume that the attacker $\mathcal{A} \in \mathcal{C}_{time,M,q}$ is time bounded – M denotes the maximum number of sequential evaluations of $\mathsf{H}(\cdot)$ before the corrupted codeword must be delivered. It would also be reasonable to assume that the total number of random oracle queries q is polynomial in the relevant parameters. One can also argue that in most practical settings the channel $\mathcal{A}$ will have other resource constraints e.g., space-bounded etc. In general one can define complexity classes for various combinations of resource constraints – see Definition 19.

▶ **Definition 19.** *For constraints $\mathcal{M} = (\mathcal{M}_1, \ldots, \mathcal{M}_p)$ on resources $\mathcal{R} = (\mathcal{R}_1, \ldots, \mathcal{R}_p)$ listed in Figure 8, the constraint class $C_{\mathcal{R},\mathcal{M}}$ is the set of all pROM $\mathcal{A}^{\mathsf{H}(\cdot)}$ such that $\mathcal{A}^{\mathsf{H}(\cdot)}$ is $\mathcal{R}$-bounded with constraints $\mathcal{M}$. Here, a pROM algorithm is said to be* $\mathcal{R}$-bounded *with* constraints $\mathcal{M}$ *if for all $i \leq p$ and on all inputs x, random coins R, and random oracles $\mathsf{H}(\cdot)$, we have*

$$\mathcal{R}_i\big(\mathsf{Trace}_{\mathcal{A},R,\mathsf{H}}(x)\big) \leq \mathcal{M}_i$$

SCRYPT. Alwen et al. [5] proved that Percival's [35] memory hard function `scrypt` is maximally memory hard. In particular, $\mathtt{scrypt}_N$ can be computed in sequential time N, but any pROM attacker evaluating the scrypt function has cumulative memory complexity at least $\Omega(N^2 w)$, where w is the length of the output. Thus, `scrypt` could be used to obtain safe functions for the classes $\mathcal{C}_{\mathsf{CMC},S,q}$ and $\mathcal{C}_{\mathsf{space-time},S,q}$ – observe that $\mathsf{CMC}(\mathcal{T}) \leq \mathsf{space} - \mathsf{time}(\mathcal{T})$ for any execution trace $\mathcal{T}$.

4.2 Sequentially Hard Function

The hash iteration function $f(x) = \mathsf{H}(x)^{t+1}$, defined recursively as $\mathsf{H}(x)^{t+1} = \mathsf{H}(\mathsf{H}(x)^t)$ where $\mathsf{H}(x)^1 = \mathsf{H}(x)$, is a simple example of a safe function for the class $\mathcal{C}_{\mathsf{time},T=t,q}$ of time bounded attackers – see Claim 20. The trade-off is sharp since it is trivial to compute $f(x)$ in sequential time $t+1$. This is a desirable property in our context since the encoder/decoder both need to compute $f(x)$ for a random input x.

We remark that the proof of Claim 20 is very similar to an argument of Cohen and Pietrzak [17]. Our bound is slightly tighter, but less general. Cohen and Pietrzak [17] proved that any pROM algorithm running in time t can produce an arbitrary H-sequence with probability at most $\mathcal{O}\left(\frac{q^2}{2^w}\right)$. We can reduce the bound to $\mathcal{O}\left(\frac{qt}{2^w}\right)$ since the attacker needs to compute a specific H-sequence i.e., $L_1, \ldots, L_{t+1}$ with $L_i = \mathsf{H}(x)^i$. In general, we may have $q \ll t$.

▷ Claim 20. Let $f(x) = \mathsf{H}(x)^{t+1}$ and let $\epsilon = (t+1)t/2^{w+1} + (qt+1)2^{-w}$ then the function f is ϵ-safe for the class $\mathcal{C}_{\mathsf{time},T=t,q}$.

Proof. (Sketch) Let $L_i := \mathsf{H}(x)^i$. We remark that if $L_1, \ldots, L_{j-1}$ are all distinct then

$$\mathbf{Pr}\left[L_j = \mathsf{H}(L_{j-1}) \in \{L_1, \ldots, L_{j-1}\right] \leq (j-1)2^{-w}$$

Thus, the probability of the event $\mathtt{COL}$ that $L_i = L_j$ for some $1 \leq i < j \leq t+1$ is at most $2^{-w}\sum_{j=1}^{t+1}(j-1) = (t+1)t/2^{w+1}$. We say that a particular random oracle query u in round i is lucky if the output is $\mathsf{H}(u) = L_j$ but the label L_{j-1} had not previously been observed as the output to any earlier random oracle query. If i denotes the maximum index such that L_i has been observed as a random oracle output, then the probability that a particular query u is lucky is at most $\Pr[\mathsf{H}(u) \in \{L_{i+2}, \ldots, L_{t+1}\} | \overline{\mathtt{COL}}] = (t-i)2^{-w} \leq t2^{-w}$.

Conditioning on the event $\overline{\mathtt{COL}}$ that no collisions occur, we can apply union bounds to show that, except with probability $qt/2^2$, there are no lucky queries. If there are no lucky queries, then after t sequential rounds the output $L_{t+1} = f(x)$ can be viewed as uniformly random and the probability that the attacker outputs $f(x)$ is at most 2^{-w} in this case. ◀

If we let r denote the maximum number of sequential calls to $\mathsf{H}(\cdot)$ that can be evaluated in a second[3] then we could set $t = r \times L_{max}$, where L_{max} denotes the maximum latency of the channel. Note that the encoder/decoder would need require time marginally higher than the latency $L_{max} + 1/r \approx L_{max}$ to compute $H^{t+1}(x)$.

4.3 Graph Labeling Functions

We define a labeling function $f_{G,H}(x)$ on a graph G, hash function H, and input x.

▶ **Definition 21.** *Given a DAG $G = (V = [N], E)$ and a random oracle function $H : \Sigma^* \to \Sigma^w$ over an alphabet Σ, we define the labeling of graph G as $L_{G,H} : \Sigma^* \to \Sigma^*$. In particular, given an input x the (H, x) labeling of G is defined recursively by*

$$L_{G,H,x}(v) = \begin{cases} H(v \circ x), & \mathsf{indeg}(v) = 0 \\ H\left(v \circ L_{G,H,x}(v_1) \circ \cdots \circ L_{G,H,x}(v_d)\right), & \mathsf{indeg}(v) > 0, \end{cases}$$

[3] Bonneau and Schechter [14] estimated that SHA256 can be evaluated $r \approx 10^7$ times per second on a single core processor

where $v_1, \ldots, v_d$ are the parents of v in G, according to some predetermined lexicographical order. We define $f_{G,H}(x) = \{L_{G,H,x}(s)\}_{s \in \mathsf{sinks}(G)}$. If there is a single sink node s_G then $f_{G,H}(x) = L_{G,H,x}(s_G)$. We omit the subscripts G, H, x when the dependency on the graph G and hash function H is clear.

The graph labeling function can be used to construct safe functions for several different classes of resource bounded adversaries. In particular, the resources necessary to compute $f_{G,H}$ in the pROM are tightly linked to the black pebbling cost of the DAG G.

Parallel Black Pebbling Game. A legal (parallel) pebbling $P = (P_0, P_1, \ldots, P_t)$ of a DAG $G = (V, E)$ consists of a sequence of pebbling configurations $P_i \subseteq V$ – representing the set of labels $L_{G,H,x}(v)$ which are stored in memory at time i. We start with no pebbles on the graph $P_0 = \emptyset$, and can remove pebbles from the graph (free memory) at any time. For any newly pebbled node $v \in P_{i+1} \setminus P_i$, it must be the case that $\mathsf{parents}(v) \subseteq P_i$ where $\mathsf{parents}(v) := \{u \ : \ (u, v) \in E\}$. Intuitively, this is because we cannot compute $L_{G,H,x}(v)$ unless each of the dependent values $L_{G,H,x}(u)$ for each $u \in \mathsf{parents}(v)$ is already available in memory. In the parallel version of the black pebbling game, there is no constraint on the number of new pebbles $\left|P_{i+1} \setminus P_i\right|$ that can be placed on the graph in each round.

The space cost of a pebbling P is defined as $\mathsf{space}(P) := \max_i |P_i|$ and the space complexity of a graph is $\mathsf{space}(G) = \min_P \mathsf{space}(P)$. The space-time (resp. cumulative cost) cost of a pebbling P is the product $\mathsf{space} - \mathsf{time}(P) := \mathsf{time}(P) \times \mathsf{space}(P)$ (resp. $\mathtt{CC}(G) = \sum_i |P_i|$). We remark that $\mathtt{CC}(G) \leq \mathsf{space} - \mathsf{time}(G)$. For constant degree graphs G with N nodes it is known that $\mathsf{space}(G) = \mathcal{O}\left(N/\log N\right)$ and that $\mathtt{CC}(G) = \mathcal{O}\left(N^2 \log\log N/\log N\right)$ [1]. One can also construct graphs G s.t. $\mathtt{CC}(G) = \Omega(N^2/\log N)$ [3, 2] and Paul et al. [34] constructed a constant indegree graph G with $\mathsf{space}(G) = \Omega(N/\log N)$ [34, 4] – this last bound is tight as Hopcroft et al. [26] showed that *any* static DAG G on N nodes with constant indegree can be pebbled with at most $\mathsf{space}(G) = \mathcal{O}\left(N/\log N\right)$ pebbles.

Pebbling Reductions. In the full version [13] we prove that if $\mathsf{space}(G) \geq m$ and $S = mw/2$ then $f_{G,H}$ is safe for the class $\mathcal{C}_{\mathsf{space},S,q}$. The pebbling reduction is conceptually very similar to the reduction of Alwen and Serbinenko [6] who proved that $\mathsf{CMC}(f_{G,H}) = \Omega(\mathtt{CC}(G) \cdot w)$ i.e., if the graph G has high cumulative pebbling cost then $f_{G,H}$ is safe for the class $\mathcal{C}_{\mathsf{CMC},M,q}$, and by extension safe for the class $\mathcal{C}_{\mathsf{space}-\mathsf{time},M,q} \subseteq \mathcal{C}_{\mathsf{CMC},M,q}$. In particular, given an execution trace $\mathsf{Trace}_{\mathcal{A},R,\mathsf{H}}(x)$ for an algorithm $\mathcal{A}^{H(\cdot)}(x)$ computing $f_{G,H}(x)$ we can (with high probability) extract a legal pebbling $P = (P_1, \ldots, P_t)$ for G and then use an extractor argument to show that $|\sigma_i|/w \geq |P_i|/2$ during each round i – otherwise we could derive a contradiction by using the extractor to compress the random oracle. Thus, to construct a safe function one simply needs to find a graph G with sufficiently large pebbling cost.

4.4 Brief Note on Candidate Constructions without Random Oracles

Recall that the proof of correctness for our LDC constructions on space bounded channels uses the random oracle model inherently through an extractor argument showing that any space bounded channel that fools a decoding algorithm can also essentially predict a random string. Obtaining the same results without random oracles is an open challenge. We remark that there are several candidate constructions of safe-functions in the standard model e.g., using *time-lock puzzles* [36]. One may also be able to use the framework of Bitansky et al. [11] to construct safe-functions without random oracles e.g., Bitansky et al. construct explicit time-lock puzzles from the minimal assumption that "inherently sequential" languages exist. It is plausible that the same construction would also yield space-bound (or space-time bound) puzzles from minimal assumptions. See the full version [13] for additional assumptions.

References

1 Joël Alwen and Jeremiah Blocki. Efficiently computing data-independent memory-hard functions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 241–271, Santa Barbara, CA, USA, August 14–18 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53008-5_9.

2 Joël Alwen, Jeremiah Blocki, and Ben Harsha. Practical graphs for optimal side-channel resistant memory-hard functions. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1001–1017, Dallas, TX, USA, October 31 – November 2 2017. ACM Press. doi:10.1145/3133956.3134031.

3 Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Depth-robust graphs and their cumulative memory complexity. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 3–32, Paris, France, April 30 – May 4 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-56617-7_1.

4 Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Sustained space complexity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 99–130, Tel Aviv, Israel, April 29 – May 3 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78375-8_4.

5 Joël Alwen, Binyi Chen, Krzysztof Pietrzak, Leonid Reyzin, and Stefano Tessaro. Scrypt is maximally memory-hard. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 33–62, Paris, France, April 30 – May 4 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-56617-7_2.

6 Joël Alwen and Vladimir Serbinenko. High parallel complexity graphs and memory-hard functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 595–603, Portland, OR, USA, June 14–17 2015. ACM Press. doi:10.1145/2746539.2746622.

7 L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. Bpp has subexponential time simulations unless exptime has publishable proofs. In *[1991] Proceedings of the Sixth Annual Structure in Complexity Theory Conference*, pages 213–219, June 1991. doi:10.1109/SCT.1991.160263.

8 László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 21–31, 1991. doi:10.1145/103418.103428.

9 Amos Beimel and Yuval Ishai. Information-theoretic private information retrieval: A unified construction. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming*, pages 912–926, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

10 Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust pcps of proximity, shorter pcps, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006. A preliminary version appeared in the Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC).

11 Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In Madhu Sudan, editor, *ITCS 2016: 7th Conference on Innovations in Theoretical Computer Science*, pages 345–356, Cambridge, MA, USA, January 14–16 2016. Association for Computing Machinery. doi:10.1145/2840728.2840745.

12 Jeremiah Blocki, Venkata Gandikota, Elena Grigorescu, and Samson Zhou. Relaxed locally correctable codes in computationally bounded channels. In *IEEE International Symposium on Information Theory, ISIT*, page (to appear), 2019.

13 Jeremiah Blocki, Shubhang Kulkarni, and Samson Zhou. On locally decodable codes in resource bounded channels. *CoRR*, abs/1909.11245, 2019.

14 Joseph Bonneau and Stuart E. Schechter. Towards reliable storage of 56-bit secrets in human memory. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014: 23rd USENIX Security Symposium*, pages 607–623, San Diego, CA, USA, August 20–22 2014. USENIX Association.

15 Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 402–414, 1999.

16 Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, November 1998. doi:10.1145/293347.293350.

17 Bram Cohen and Krzysztof Pietrzak. Simple proofs of sequential work. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 451–467, Tel Aviv, Israel, April 29 – May 3 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78375-8_15.

18 A. Deshpande, R. Jain, T. Kavitha, S. V. Lokam, and J. Radhakrishnan. Better lower bounds for locally decodable codes. In *Proceedings 17th IEEE Annual Conference on Computational Complexity*, pages 184–193, May 2002. doi:10.1109/CCC.2002.1004354.

19 Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching vector codes. *SIAM J. Comput.*, 40(4):1154–1178, 2011.

20 Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM J. Comput.*, 41(6):1694–1703, 2012.

21 Peter Gemmell, Richard Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 33–42, New York, NY, USA, 1991. ACM. doi:10.1145/103418.103429.

22 Tom Gur, Govind Ramnarayan, and Ron D. Rothblum. Relaxed locally correctable codes. In *9th Innovations in Theoretical Computer Science Conference, ITCS*, pages 27:1–27:11, 2018.

23 Venkatesan Guruswami and Adam Smith. Optimal rate code constructions for computationally simple channels. *J. ACM*, 63(4):35:1–35:37, September 2016. doi:10.1145/2936015.

24 Brett Hemenway and Rafail Ostrovsky. Public-key locally-decodable codes. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Proceedings*, pages 126–143, 2008.

25 Brett Hemenway, Rafail Ostrovsky, Martin J. Strauss, and Mary Wootters. Public key locally decodable codes with short keys. In *14th International Workshop, APPROX, and 15th International Workshop, RANDOM, Proceedings*, pages 605–615, 2011.

26 John Hopcroft, Wolfgang Paul, and Leslie Valiant. On time versus space. *J. ACM*, 24(2):332–337, April 1977. doi:10.1145/322003.322015.

27 Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 80–86, New York, NY, USA, 2000. ACM. doi:10.1145/335305.335315.

28 Iordanis Kerenidis and Ronald de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *J. Comput. Syst. Sci.*, 69(3):395–420, 2004.

29 Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-rate locally correctable and locally testable codes with sub-polynomial query complexity. *J. ACM*, 64(2):11:1–11:42, 2017.

30 E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 364–373, October 1997. doi:10.1109/SFCS.1997.646125.

31 Richard J. Lipton. A new approach to information theory. In *STACS 94*, pages 699–708, Berlin, Heidelberg, 1994.

32 Silvio Micali, Chris Peikert, Madhu Sudan, and David A. Wilson. Optimal error correction against computationally bounded noise. In Joe Kilian, editor, *Theory of Cryptography*, pages 1–16, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

33 Rafail Ostrovsky, Omkant Pandey, and Amit Sahai. Private locally decodable codes. In *Automata, Languages and Programming*, pages 387–398, 2007.

34 Wolfgang J. Paul, Robert Endre Tarjan, and James R. Celoni. Space bounds for a game on graphs. In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, STOC '76, pages 149–160, New York, NY, USA, 1976. ACM. doi:10.1145/800113.803643.

35 C. Percival. Stronger key derivation via sequential memory-hard functions. In *BSDCan 2009*, 2009.

36 Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. Technical report, Massachusetts Institute of Technology, USA, 1996.

37 Ronen Shaltiel and Jad Silbak. Explicit list-decodable codes with optimal rate for computationally bounded channels. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 45:1–45:38, 2016.

38 Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the xor lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001. doi:10.1006/jcss.2000.1730.

39 Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *J. ACM*, 55(1):1:1–1:16, 2008.