3rd Conference on Information-Theoretic Cryptography

ITC 2022, July 5–7, 2022, Cambridge, MA, USA

Edited by

Dana Dachman-Soled

Editors

Dana Dachman-Soled
University of Maryland, College Park, MD, USA
danadach@umd.edu

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

ISBN 978-3-95977-238-9

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-238-9.

Publication date
July, 2022

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.2022.0

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

Papers

3rd Conference on Information-Theoretic Cryptography (ITC 2022).
Editor: Dana Dachman-Soled
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

The third Conference on Information-Theoretic Cryptography (ITC 2022) was held in-person in Cambridge, MA on July 5–7, 2022, with virtual attendance an option. Yael Tauman Kalai and Vinod Vaikuntanathan were the general chairs and Dana Dachman-Soled was the program chair. The conference was held in cooperation with the International Association for Cryptologic Research (IACR).

In its third year, ITC has continued to fill the role in the cryptographic community of disseminating research advances on all aspects of information-theoretic security. The breadth of topics covered by the papers and invited talks reflects the fact that information theoretic techniques are pervasive in essentially all areas of cryptography. ITC has also sought to foster the creation of a community bringing together researchers from coding theory, information theory (classical and quantum), theory of computation, privacy, and cryptography.

This year, the conference received 30 submissions, of which the Program Committee (PC) accepted 17. The 19 PC members were helped in selecting the program by many external reviewers. The small size of the conference afforded the reviewers the opportunity to send anonymous questions to the authors to clarify technical issues. In several cases, this interactive process consisted of multiple rounds and continued until the questions were resolved. The proceedings consist of the revised version of the 17 accepted papers. The revisions were not reviewed, and the authors bear full responsibility for the content.

In addition to the accepted papers, the conference included six invited talks. They were selected by the PC to be "spotlight talks," highlighting the most exciting recent advances in cryptography and theoretical computer science that interest the ITC community.

Many individuals helped make ITC 2022 a success. We offer our sincere thanks to: All the authors who chose to submit their papers to this conference; the PC members for providing thorough reviews, for their dedication to resolving technical issues, and for their active participation in discussing each paper; the external reviewers for providing us with their valuable expert opinions; the ITC Steering Committee, and especially Benny Applebaum, for providing support, guidance, and advice throughout the process; and last but not least, the invited speakers, presenting authors, and participants for committing their time to making the conference a success.

Dana Dachman-Soled

3rd Conference on Information-Theoretic Cryptography (ITC 2022).
Editor: Dana Dachman-Soled
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Steering Committee

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

3rd Conference on Information-Theoretic Cryptography (ITC 2022).
Editor: Dana Dachman-Soled

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

Organization

General Chairs

- Yael Tauman Kalai (MIT and Microsoft Research New England)
- Vinod Vaikuntanathan (MIT)

Program Chair

- Dana Dachman-Soled (University of Maryland)

Program Committee

- Gorjan Alagic (University of Maryland and NIST)
- Gilad Asharov (Bar-Ilan University)
- Marshall Ball (New York University)
- Jeremiah Blocki (Purdue University)
- Anne Broadbent (University of Ottawa)
- Eshan Chattopadhyay (Cornell University)
- Kai-Min Chung (Academia Sinica)
- Ivan Damgård (Aarhus University)
- Mohammad Hajiabadi (University of Waterloo)
- Bhavana Kanakurthi (IISc Bangalore)
- Ilan Komargodski (Hebrew University of Jerusalem and NTT Research)
- Mukul Kulkarni (Technology Innovation Institute (TII Abu Dhabi))
- Feng-Hao Liu (Florida Atlantic University)
- Julian Loss (CISPA Helmholtz Center for Information Security)
- Mohammad Mahmoody (University of Virginia)
- Hemanta Maji (Purdue University)
- Krzysztof Pietrzak (IST Austria)
- Aishwarya Thiruvengadam (IIT Madras)
- Mor Weiss (Bar-Ilan University)

External Reviewers

Anirban Chakrabarti, Seung Geol Choi, Arka Rai Choudhuri, Jesse Goodman, Shreyas Gupta, Yao-Ching Hsieh, Wei-Hsiang Hung, Seunghoon Lee, Ray Li, Jyun-Jie Liao, Fermi Ma, Nicky Mouha, Sai Lakshmi Bhavana Obbattu, Maciej Obremski, Adam O'Neill, Sruthi Sekar, Girisha Shankar, Nikki Sigurdson, Dave Touchette, Benedikt Wagner, Chenkai Weng

3rd Conference on Information-Theoretic Cryptography (ITC 2022).
Editor: Dana Dachman-Soled

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

Multi-Server PIR with Full Error Detection and Limited Error Correction

Reo Eriguchi
Graduate School of Information Science and Technology, The University of Tokyo, Japan
National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Kaoru Kurosawa
Research and Development Initiative, Chuo University, Tokyo, Japan
National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Koji Nuida
Institute of Mathematics for Industry, Kyushu University, Japan
National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

Abstract

An ℓ-server Private Information Retrieval (PIR) scheme allows a client to retrieve the τ-th element a_τ from a database $\boldsymbol{a} = (a_1, \ldots, a_n)$ which is replicated among ℓ servers. It is called t-private if any coalition of t servers learns no information on τ, and b-error correcting if a client can correctly compute a_τ from ℓ answers containing b errors. This paper concerns the following problems: Is there a t-private ℓ-server PIR scheme with communication complexity $o(n)$ such that a client can detect errors with probability $1 - \epsilon$ even if $\ell - 1$ servers return false answers? Is it possible to add error correction capability to it? We first formalize a notion of $(1 - \epsilon)$-fully error detecting PIR in such a way that an answer returned by any malicious server depends on at most t queries, which reflects t-privacy. We then prove an impossibility result that there exists no 1-fully error detecting (i.e., $\epsilon = 0$) PIR scheme with $o(n)$ communication. Next, for $\epsilon > 0$, we construct 1-private $(1 - \epsilon)$-fully error detecting and $(\ell/2 - O(1))$-error correcting PIR schemes which have $n^{o(1)}$ communication, and a t-private one which has $O(n^c)$ communication for any $t \geq 2$ and some constant $c < 1$. Technically, we show generic transformation methods to add error correction capability to a basic fully error detecting PIR scheme. We also construct such basic schemes by modifying certain existing PIR schemes which have no error detection capability.

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

Keywords and phrases Private Information Retrieval, Error Detection, Error Correction

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.1

Related Version *Full Version*: `https://eprint.iacr.org/2022/500`

Funding This research was partially supported by JSPS KAKENHI Grant Numbers JP20J20797 and 19H01109, Japan and JST CREST Grant Numbers JPMJCR19F6 and JPMJCR2113, Japan.

1 Introduction

Private Information Retrieval (PIR) was introduced by Chor, Goldreich, Kushilevitz, and Sudan [7]. In an ℓ-server PIR scheme, a client can retrieve the τ-th element a_τ of a database $\boldsymbol{a} = (a_1, \ldots, a_n)$ replicated among ℓ servers without revealing any information on the index τ to the servers. A trivial solution is that servers send the entire database to the client. However, it results in communication complexity $O(n)$, which is shown to be optimal in the information-theoretic setting when $\ell = 1$ [7]. To get around this, Chor et al. [7] considered ℓ-server PIR schemes for $\ell \geq 2$ in which servers do not collude. More generally, a PIR scheme is called t-private if any coalition of t servers learns no information on τ.

Since then, many ℓ-server PIR schemes have been developed to improve communication cost [1, 3, 4, 6, 7, 10, 11, 14, 22]. Currently, the most communication-efficient schemes are 1-private $2^{O(r)}$-server PIR schemes with sub-polynomial (in n) communication complexity $\mathcal{L}_n[1/r, O_r(1)]$ [6, 10], where $\mathcal{L}_n[s, c]$ denotes a function $\exp(c(\log n)^s(\log\log n)^{1-s})$ and the notation $O_r(\cdot)$ hides constants that depend on r only.[1] To achieve t-privacy for $t \geq 2$, Woodruff and Yekhanin [20] proposed a t-private ℓ-server PIR scheme with communication complexity $n^{\lfloor(2k-1)/t\rfloor^{-1}}\ell^{O(1)}$ for any $1 \leq k \leq \ell$.

As more servers are involved, there is a higher possibility that servers are malicious or fault, or that the databases are not updated simultaneously. It is then important to enable a client to detect or even correct errors when part of servers return false answers. Beimel and Stahl [5] introduced b-error correcting PIR, which enables a client to retrieve a correct value a_τ even if b (or less) servers return false answers. They showed that a b-error correcting PIR scheme can be generically obtained from any k-server PIR scheme if $b \leq (\ell - k)/2$ while the time complexity of error correction is proportional to $\binom{\ell}{k}$. Kurosawa [15] proposed a more time-efficient error correction algorithm specialized for the t-private PIR scheme in [20] and as a result, it performs $\lfloor(\ell - k)/2\rfloor$-error correction in polynomial time in ℓ for any $1 \leq k \leq \ell$.

However, as pointed out in [5], b-error correcting PIR is possible only if $b < \ell/2$. It is therefore important to consider a weaker notion of error detecting to tolerate more malicious servers. Specifically, we define $(1-\epsilon)$-fully error detecting PIR as the one which enables a client to detect errors with probability $1 - \epsilon$ even if $\ell - 1$ out of ℓ answers are false. To the best of our knowledge, there are no fully error detecting PIR schemes in the literature except for the trivial scheme or the one implicitly used in [21] both of which have communication cost $O(n)$. This paper concerns the following problem:

> *Is there a t-private $(1-\epsilon)$-fully error detecting ℓ-server PIR scheme with communication complexity $o(n)$? Is it possible to add error correction capability to it?*

1.1 Our Results

We first formalize the notion of $(1-\epsilon)$-fully error detecting PIR. We then prove an impossibility result that there exists no 1-fully error detecting (i.e., $\epsilon = 0$) PIR scheme with $o(n)$ communication. Next, for $\epsilon > 0$, we construct 1-private $(1-\epsilon)$-fully error detecting and $(\ell/2 - O(1))$-error correcting PIR schemes which has $n^{o(1)}$ communication. For $t \geq 2$, we also propose a t-private one which has $O(n^c)$ communication for some constant $c < 1$. Here, we ignore a factor of $\log \epsilon^{-1}$ in communication cost. Our constructions are based on the following technical contributions:

- We prove that the transformations [5], which add error correction capability to PIR schemes, preserve full error detection capability and even reduces the probability of failure.
- We construct $(1-\epsilon)$-fully error detecting PIR schemes by modifying certain existing schemes.

In what follows we briefly discuss each of these contributions.

Formalization of Fully Error Detecting PIR. Let Π be a t-private ℓ-server PIR scheme. In our model, a set of at most $\ell - 1$ malicious servers T is partitioned into pairwise disjoint subsets $T = T_1 \cup \cdots \cup T_m$ such that $|T_h| \leq t$ for any h, and servers in each T_h can collude to

[1] If $c = O(1)$, $\mathcal{L}_n[1, c]$ is polynomial in n and $\mathcal{L}_n[0, c]$ is polylogarithmic in n. For $0 < s < 1$, $\mathcal{L}_n[s, c]$ is sub-polynomial in n.

generate their false answers. Our model is natural since due to the t-privacy, no malicious server is allowed to see more than t queries and hence its false answer should not depend on more than t queries. We say that Π is $(1-\epsilon)$-fully error detecting if a client can detect errors with probability $1-\epsilon$ for any $T = T_1 \cup \cdots \cup T_m$ satisfying the above condition. We prove that there exists no 1-fully error detecting (i.e., $\epsilon = 0$) PIR scheme with $o(n)$ communication (Theorem 13). This implies that it is necessary to consider $(1-\epsilon)$-fully error detecting PIR with $\epsilon > 0$.

Transformation to Increase Robustness of Fully Error Detecting PIR. To transform a k-server PIR scheme Π to an $\lfloor(\ell-k)/2\rfloor$-error correcting ℓ-server PIR scheme Π', Beimel and Stahl [5] presented a naive method, which executes an independent instance of Π for each group of k servers, and a more refined method, which uses perfect hash families.[2] We prove that the two transformation methods preserve full error detection capability and even reduces the probability of failure. Therefore, they can be used to add $\lfloor(\ell-k)/2\rfloor$-error correction capability to a fully error detecting PIR scheme.More specifically, the method using a perfect hash family transforms a 1-private $(1-\epsilon)$-fully error detecting k-server PIR scheme Π to a 1-private $(1-\epsilon)$-fully error detecting ℓ-server PIR scheme Π' (Theorem 14). The overhead in communication cost is $2^{O(k)}\ell\log\ell$. The naive method can be used to transform a t-private $(1-\epsilon)$-fully error detecting k-server PIR scheme Π to a t-private $(1-\epsilon^M)$-fully error detecting ℓ-server PIR scheme Π', where $M = \lceil(\ell-k+1)/(k+t-2)\rceil$ (Theorem 15). The communication cost of Π' is $\binom{\ell}{k}$ times larger than Π. Although the method in Theorem 14 is more communication-efficient for large k, the naive transformation in Theorem 15 has the following advantages:

- From any 1-private 2-server $(1-\epsilon)$-fully error detecting PIR scheme, we can obtain a 1-private ℓ-server $(1-\epsilon)$-fully error detecting one which has lower communication cost by a factor of $O(\log\ell)$ than if Theorem 14 is applied.
- It works for any $t \geq 1$, where t is the number of servers who can collude.

Constructions of Fully Error Detecting PIR Schemes.

1-Private two-server PIR scheme. Dvir and Gopi [10] showed a 1-private 2-server PIR scheme with communication complexity $\mathcal{L}_n[1/2, O(1)]$ by using a matching vector family and a kind of polynomial interpolation. Based on their scheme, we construct a 1-private $(1-\epsilon)$-fully error detecting 2-server PIR scheme with communication complexity $\mathcal{L}_n[1/2, O(1)] \cdot \log\epsilon^{-1}$ (Theorem 16). Our technical novelty is modifying the scheme [10] in such a way that a client chooses interpolation points at random and carefully analyzing its error detection capability. By applying the naive transformation in Theorem 15, we obtain a 1-private ℓ-server $(1-\epsilon)$-fully error detecting and $\lfloor(\ell-2)/2\rfloor$-error correcting PIR scheme with communication complexity $\mathcal{L}_n[1/2, O(1)] \cdot \ell\log\epsilon^{-1}$ (Corollary 17).

1-Private ℓ-server PIR scheme for larger ℓ. We show that the communication complexity of fully error detecting PIR can be further reduced by increasing the number of servers. We invoke a basic PIR scheme based on a matching vector family shown in [9], which uses Lagrange interpolation to retrieve a_τ. We carefully choose parameters for the matching vector family and let a client choose interpolation points at random. As a result, for any fixed $r \geq 2$, we obtain a 1-private $(1-\epsilon)$-fully error detecting k_r-server PIR scheme with communication complexity $\mathcal{L}_n[1/r, O_r(1)] \cdot \log\epsilon^{-1}$, where k_r is a constant

[2] We note that their method shown in [5, Section 3.1] is a special case of the latter based on perfect hash families.

depending on r (Corollary 19). By applying the transformation in Theorem 14, we obtain a 1-private $(1-\epsilon)$-fully error detecting and $\lfloor(\ell-k_r)/2\rfloor$-error correcting ℓ-server PIR scheme with communication complexity $\mathcal{L}_n[1/r, O_r(1)] \cdot 2^{O(k_r)}\ell \log \ell \log \epsilon^{-1}$ for any $\ell \geq k_r$ (Corollary 20). By setting $r = 3$, we obtain a $(1-\epsilon)$-fully error detecting ℓ-server PIR scheme with communication cost $\mathcal{L}_n[1/3, O(1)] \cdot \ell \log \ell \log \epsilon^{-1}$ for $\ell \geq 2^{17}$.

t-Private ℓ-server PIR scheme for $t \geq 2$ and $\ell \geq 2$. Our construction for $t \geq 2$ is based on the best known t-private $\lfloor(\ell-k)/2\rfloor$-error correcting ℓ-server PIR scheme [20] with communication complexity $O(dn^d\ell \log \ell)$, where $1 \leq k \leq \ell$ and $d = \lfloor(2k-1)/t\rfloor$. Their scheme uses Hermite interpolation [17] to retrieve a_τ. By choosing interpolation points randomly, we obtain a t-private $(1-\epsilon)$-fully error detecting and $\lfloor(\ell-k)/2\rfloor$-error correcting ℓ-server PIR scheme with communication complexity $O(dn^{1/d}\ell \log \ell \log \epsilon^{-1})$ (Theorem 21). We note that the polynomial-time error correction algorithm [15], which was originally proposed for the scheme [20] with no error detection, is applicable to our fully error detecting scheme. Hence, this scheme achieves error correction without the transformations in Theorems 14 and 15.

1.2 Related Work

Beimel and Stahl [5] introduced (k, ℓ)-robust PIR, which allows a client to retrieve a correct value from answers of any k out of ℓ honest servers. They presented generic transformations from any k-server PIR scheme to (k, ℓ)-robust PIR scheme. They also showed that any (k, ℓ)-robust PIR scheme achieves b-error correction for $b \leq (\ell-k)/2$ while the time complexity of error correction is proportional to $\binom{\ell}{k}$. Any (k, ℓ)-robust PIR scheme implies an ℓ-server PIR scheme that detects errors if at most $\ell - k$ servers are malicious, by letting the client recover data from answers of every k servers and check the consistency. However, it cannot be better than the trivial scheme if there are $\ell - 1$ malicious servers.

Yang, Xu, and Bennett [21] proposed a PIR scheme which achieves b-error correction by performing error detection for all subsets of servers of size $b+1$ for $b = \lfloor(\ell-1)/2\rfloor$. Their scheme satisfies our definition of fully error detecting PIR. However, the communication complexity is $O(n)$ and it is not better than the trivial scheme downloading the whole database. Although it can be reduced to $O(\sqrt{n})$ by the balancing technique of [7], the communication complexities of our schemes are still lower than theirs.

Goldberg [12] proposed a list decodable ℓ-server PIR scheme with communication complexity $O(\sqrt{n})$, in which a client outputs a list including a correct value instead of just one. However, the scheme tolerates at most $\ell - \lfloor\sqrt{\ell}\rfloor$ malicious servers and hence it cannot detect errors in the presence of $\ell - 1$ malicious servers. Devet, Goldberg, and Heninger [8] considered a different scenario where a client performs multiple queries and runs a decoding algorithm on multiple answers simultaneously. In this setting, they proposed a list decodable ℓ-server PIR scheme for $\ell - O(1)$ malicious servers with communication complexity $O(\sqrt{n})$.

Sun and Jafar [18, 19] and Banawan and Ulukus [2] considered error correction in the setting where the size of each block of a database is very large, and hence only the download cost is of interest.

2 Preliminaries

Notations. For $m \in \mathbb{N}$, define $[m] = \{1, \ldots, m\}$. For a vector $\boldsymbol{x}$, let x_i denote the i-th entry of $\boldsymbol{x}$. Let $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ be an m-variate polynomial over a finite field $\mathbb{F}_q$ of size q. We say that f is a degree-d polynomial if its total degree is at most d. Define the partial derivative of f with respect to X_j as

$$\partial_{X_j} f = \sum_{\boldsymbol{e}=(e_i)_{i\in[m]}\in I} c_{\boldsymbol{e}} e_j X_j^{e_j-1} \prod_{i\in[m]\setminus\{j\}} X_i^{e_i}$$

if $f = \sum_{\boldsymbol{e}\in I} c_{\boldsymbol{e}} \prod_{i\in[m]} X_i^{e_i}$, where $c_{\boldsymbol{e}} \in \mathbb{F}_q$ and I is a finite set of m-tuples of non-negative integers. For a univariate polynomial f, we denote by ∂f the derivative of f with respect to its unique variable. We write $u \leftarrow_{\$} \mathcal{U}$ if u is randomly chosen from a set $\mathcal{U}$. For two vectors $\boldsymbol{x} = (x_i)_{i\in[m]}, \boldsymbol{y} = (y_i)_{i\in[m]}$ over a ring $\mathcal{U}$, we define $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = \sum_{i\in[h]} u_i v_i$ and $\mathsf{wt}(\boldsymbol{u}) = |\{i \in [m] : u_i \neq 0\}|$. Let $\mathcal{L}_n[s, c]$ denote the function of n defined as

$$\mathcal{L}_n[s, c] = \exp(c(\log n)^s (\log n)^{1-s}),$$

where $0 \leq s \leq 1$ and $c > 0$. Note that if $c = O(1)$, $\mathcal{L}_n[1, c]$ is polynomial in n and $\mathcal{L}_n[0, c]$ is polylogarithmic in n. For $0 < s < 1$, $\mathcal{L}_n[s, c]$ is sub-polynomial in n.

2.1 Lagrange and Hermite Interpolation

Lagrange interpolation recovers a polynomial using its values on given points. Let $\ell \in \mathbb{N}$ and $\mathbb{F}_p$ be a prime field such that $p \geq \ell + 1$. Let $\alpha_1, \ldots, \alpha_\ell$ be ℓ pairwise distinct non-zero elements of $\mathbb{F}_p$ and let $y_j \in \mathbb{F}_p$ for each $j \in [\ell]$. Then, there exists an explicit formula for finding a unique polynomial $g \in \mathbb{F}_p[X]$ such that $\deg g \leq \ell - 1$ and $g(\alpha_j) = y_j$ for all $j \in [\ell]$.

Hermite interpolation is a generalization of Lagrange interpolation, which recovers a polynomial using its derivatives and values on given points. Let $y_{j,w} \in \mathbb{F}_p$ for each $j \in [\ell]$ and $w \in \{0, 1\}$. Then, there exists an explicit formula for finding a unique polynomial $g \in \mathbb{F}_p[X]$ such that $\deg g \leq 2\ell - 1$ and $g(\alpha_j) = y_{j,0}$ and $\partial g(\alpha_j) = y_{j,1}$ for all $j \in [\ell]$ [17].

3 Private Information Retrieval (PIR)

3.1 Definitions

In an ℓ-server PIR scheme, each server has a copy of a database $\boldsymbol{a} = (a_1, \ldots, a_n) \in \{0, 1\}^n$. A client can obtain a_τ by interacting with ℓ servers without revealing any information on τ to the servers.

▶ **Definition 1** (Syntax). *An ℓ-server PIR scheme Π consists of three algorithms $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$, where $\mathcal{Q}$ is probabilistic while $\mathcal{A}$ and $\mathcal{R}$ are deterministic.*

- *A query algorithm $\mathcal{Q}$ takes $\tau \in [n]$ as input and outputs ℓ queries $\mathsf{que}_1, \ldots, \mathsf{que}_\ell$ together with auxiliary information aux. A client computes*

  $$\mathcal{Q}(\tau; r) \to (\mathsf{que}_1, \ldots, \mathsf{que}_\ell; \mathsf{aux})$$

  and then sends que_i to the i-th server for $i \in [\ell]$, where r is a random string.
- *An answer algorithm $\mathcal{A}$ takes as input an index $i \in [\ell]$, a query que_i and a database $\boldsymbol{a} = (a_1, \ldots, a_n) \in \{0, 1\}^n$, and outputs an answer ans_i. The i-th server computes*

  $$\mathcal{A}(i, \mathsf{que}_i, \boldsymbol{a}) \to \mathsf{ans}_i$$

  and then returns ans_i to the client.

- *A reconstruction algorithm $\mathcal{R}$ takes as input ℓ answers $\mathsf{ans}_1, \ldots, \mathsf{ans}_\ell$ and auxiliary information aux, and outputs $\widetilde{a} \in \{0,1\}$. The client computes*

  $$\mathcal{R}(\mathsf{ans}_1, \ldots, \mathsf{ans}_\ell; \mathsf{aux}) \to \widetilde{a}$$

  and outputs $\widetilde{a}$.

We say that Π is correct if for any database $\boldsymbol{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$ and any $\tau \in [n]$, it holds that $\mathcal{R}(\mathsf{ans}_1, \ldots, \mathsf{ans}_\ell; \mathsf{aux}) = a_\tau$, where $(\mathsf{que}_1, \ldots, \mathsf{que}_\ell; \mathsf{aux}) \leftarrow \mathcal{Q}(\tau)$ and $\mathsf{ans}_i \leftarrow \mathcal{A}(i, \mathsf{que}_i, \boldsymbol{a})$ for $i \in [\ell]$. The (total) communication complexity of Π is given by $\sum_{i=1}^{\ell} |\mathsf{que}_i| + \sum_{i=1}^{\ell} |\mathsf{ans}_i|$, where $|\mathsf{que}_i|$ and $|\mathsf{ans}_i|$ are the bit lengths of que_i and ans_i, respectively.

We say that an ℓ-server PIR scheme is t-private if any t servers learn no information on the client's secret index τ even if they collude. Formally,

▶ **Definition 2** (t-Privacy). *An ℓ-server PIR scheme $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ is said to be t-private if for any t indices $i_1, \ldots, i_t \in [\ell]$ and any $\tau, \tau' \in [n]$, the joint distributions of $(\mathsf{que}_{i_1}, \ldots, \mathsf{que}_{i_t})$ and $(\mathsf{que}'_{i_1}, \ldots, \mathsf{que}'_{i_t})$ are perfectly identical, where $(\mathsf{que}_1, \ldots, \mathsf{que}_\ell; \mathsf{aux}) \leftarrow \mathcal{Q}(\tau)$ and $(\mathsf{que}'_1, \ldots, \mathsf{que}'_\ell; \mathsf{aux}') \leftarrow \mathcal{Q}(\tau')$.*

Beimel and Stahl [5] introduced the notion of robust and error correcting PIR.

▶ **Definition 3** (Robust PIR). *An ℓ-server PIR scheme Π is said to be (k,ℓ)-robust if for any $K = \{i_1, \ldots, i_k\} \subseteq [\ell]$, there exists an algorithm $\mathcal{R}_K$ that correctly computes a_τ from k answers $\mathsf{ans}_{i_1}, \ldots, \mathsf{ans}_{i_k}$.*

▶ **Definition 4** (Error correcting PIR). *An ℓ-server PIR scheme $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ is said to be b-error correcting if $\mathcal{R}$ can correctly compute a_τ even if b (or less) answers among $(\mathsf{ans}_1, \ldots, \mathsf{ans}_\ell)$ are false.*

In [5, Theorem 6.2], it is shown that a (k,ℓ)-robust PIR scheme is $\lfloor(\ell-k)/2\rfloor$-error correcting while the time complexity of error correction is proportional to $\binom{\ell}{k}$ since $\mathcal{R}$ needs to perform $\mathcal{R}_K$ for all subsets K of size k.

3.2 Known Transformation from k-Server PIR to (k,ℓ)-Robust PIR

Beimel and Stahl [5] showed a generic transformation from any k-server PIR scheme Π into a (k,ℓ)-robust PIR scheme Π' for any $\ell > k$. Their transformation is based on a minimal perfect hash family.

▶ **Definition 5.** *Let $\ell \geq k$. An (ℓ,k)-minimal perfect hash family $\mathcal{H} = \{h_1, \ldots, h_w\}$ is a family of functions of the form $h_j : [\ell] \to [k]$ such that for each $A \subseteq [\ell]$ of size k, there exists an index j such that $h_j(A) = [k]$.*

Let $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be any k-server PIR scheme and $\mathcal{H} = \{h_1, \ldots, h_w\}$ be an (ℓ,k)-minimal perfect hash family. Beimel and Stahl [5] construct a (k,ℓ)-robust PIR scheme $\Pi' = (\mathcal{Q}', \mathcal{A}', \mathcal{R}'_K)$ where $\mathcal{R}'_K$ is a reconstruction algorithm for a set K of k servers, as follows:

$\mathcal{Q}'(\tau)$. To obtain a_τ, a client executes w times Π independently and generates w query vectors $\mathsf{que}^{(j)} = (\mathsf{que}_1^{(j)}, \ldots, \mathsf{que}_k^{(j)})$, $j \in [w]$ along with auxiliary information $\mathsf{aux}^{(j)}$. For each $i \in [\ell]$, the client sends $\mathsf{que}_i = (\mathsf{que}_{h_1(i)}^{(1)}, \ldots, \mathsf{que}_{h_w(i)}^{(w)})$ to the i-th server.

$\mathcal{A}'(i, \mathsf{que}_i, a)$. The i-th server replies to each query $q = \mathsf{que}_{h_j(i)}^{(j)}$ for $j \in [w]$ as the $h_j(i)$-th server would reply to q in the original k-server PIR scheme Π. The i-th server returns $\mathsf{ans}_i = (\mathcal{A}(h_j(i), \mathsf{que}_{h_j(i)}^{(j)}, \boldsymbol{a}))_{j \in [w]}$ to the client.

$\mathcal{R}'_K(\mathsf{ans}_{i_1}, \ldots, \mathsf{ans}_{i_k}; \mathsf{aux})$. If the client receives answers from a set of k servers $K = \{i_1, \ldots, i_k\} \subseteq [\ell]$, let $h_j \in \mathcal{H}$ be a function such that $h_j(K) = [k]$. Due to the correctness of Π, the client can obtain a_τ from $\{\mathcal{A}(h_j(i), \mathsf{que}^{(j)}_{h_j(i)}, \boldsymbol{a})\}_{i \in K}$ and $\mathsf{aux}^{(j)}$.

A construction of $\mathcal{H}$ with $w = 2^{O(k)} \log \ell$ is also given in [5]. Therefore, the communication complexity of Π' is $c \cdot 2^{O(k)} \ell \log \ell$ if Π has communication complexity c per server. Since each execution of Π is independent, if Π is t-private, so is Π'.

4 Matching Vector Family

4.1 Definitions and Constructions

▶ **Definition 6.** *Let $m \in \mathbb{Z}$ and $S \subseteq \mathbb{Z}_m \setminus \{0\}$. We say that $\mathcal{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$ and $\mathcal{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$, where $\boldsymbol{u}_i, \boldsymbol{v}_i \in \mathbb{Z}_m^h$, form an S-matching vector family if the following condition is satisfied:*

- *$\langle \boldsymbol{u}_i, \boldsymbol{v}_i \rangle = 0$ for every $i \in [n]$;*
- *$\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle \in S$ for every $i \neq j$.*

We say that an S-matching vector family is d-bounded if $s \leq d$ for all $s \in S$ in terms of the usual order on $\mathbb{Z}$.

There exists an explicit construction of a matching vector family.

▶ **Proposition 7** ([13]). *Let $p < q$ be two primes and set $m = pq$. For any integer $n > 1$, there exist a constant θ_m depending on m only and an S-matching vector family $\mathcal{U} = \mathcal{V} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$ over $\mathbb{Z}_m^h$ such that $h = \mathcal{L}_n[1/2, \theta_m]$ and $S = \{p, q, p+q\}$.*

There also exists an explicit construction of a bounded matching vector family.

▶ **Proposition 8** ([9]). *Let $p_1, \ldots, p_r$ be $r \geq 2$ pairwise distinct primes and set $m = p_1 \cdots p_r$. Let u, w be positive integers such that $u \geq w$. For each $i \in [r]$, let e_i be the smallest integer such that $p_i^{e_i} > w^{1/r}$. Set $c = \max_{i \in [r]} p_i^{e_i}$ and $d = m \sum_{i \in [r]} p_i^{-1}$. Then, there exists a d-bounded matching vector family of size n over $\mathbb{Z}_m^h$ such that $n = \binom{u}{w}$ and $h = \binom{u}{\leq c} := \sum_{i=0}^{c} \binom{u}{i}$.*

4.2 Basic PIR Based on a Matching Vector Family

Following [9], we can construct a $(d+1)$-server PIR scheme $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ based on a d-bounded S-matching vector family $\mathcal{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$ and $\mathcal{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ over $\mathbb{Z}_m^h$ as follows. Let q be a prime such that $q = 1 \bmod m$. Let γ be an m-th root of unity of $\mathbb{F}_q$. Let $\alpha_1, \ldots, \alpha_{d+1}$ be distinct elements of $\mathbb{Z}_m$. Let $\boldsymbol{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$ be a database.

$\mathcal{Q}(\tau)$. To obtain a_τ, the client chooses $\boldsymbol{w} \in \mathbb{Z}_m^h$ randomly. He then computes $\boldsymbol{\rho}_i = (\rho_{i1}, \ldots, \rho_{ih}) = \boldsymbol{w} + \alpha_i \boldsymbol{u}_\tau$, sends $\mathsf{que}_i = (\gamma^{\rho_{i1}}, \ldots, \gamma^{\rho_{ih}})$ to the i-th server for $i \in [d+1]$, and stores $\mathsf{aux} = \boldsymbol{w}$.

$\mathcal{A}(i, \mathsf{que}_i, a)$. The i-th server returns

$$\mathsf{ans}_i = \xi_i = \sum_{\sigma \in [n]} a_\sigma (\gamma^{\rho_{i1}})^{v_{\sigma 1}} \cdots (\gamma^{\rho_{ih}})^{v_{\sigma h}} = \sum_{\sigma \in [n]} a_\sigma \gamma^{\langle \boldsymbol{\rho}_i, \boldsymbol{v}_\sigma \rangle}$$

to the client for $i \in [d+1]$, where $v_{\sigma j}$ is the j-th coordinate of $\boldsymbol{v}_\sigma$.

$\mathcal{R}(\mathsf{ans}_1, \ldots, \mathsf{ans}_{d+1}; \mathsf{aux})$. The client computes a_τ from $\xi_1, \ldots, \xi_{d+1}$ as follows. Note that

$$\begin{aligned}\xi_i &= \sum_{\sigma\in[n]} a_\sigma \gamma^{\langle \boldsymbol{\rho}_i, \boldsymbol{v}_\sigma\rangle} \\ &= \sum_{\sigma\in[n]} a_\sigma \gamma^{\langle \boldsymbol{w}+\alpha_i \boldsymbol{u}_\tau, \boldsymbol{v}_\sigma\rangle} \\ &= a_\tau \gamma^{\langle \boldsymbol{w}, \boldsymbol{v}_\tau\rangle} + \sum_{\sigma\neq\tau} a_\sigma \gamma^{\langle \boldsymbol{w}, \boldsymbol{v}_\sigma\rangle} \gamma^{\alpha_i \langle \boldsymbol{u}_\tau, \boldsymbol{v}_\sigma\rangle} \\ &= c_0 + \sum_{s\in S} c_s (\gamma^{\alpha_i})^s\end{aligned}$$

where $c_s = \sum_{\sigma\in[n]:\langle \boldsymbol{u}_\tau, \boldsymbol{v}_\sigma\rangle = s} a_\sigma \gamma^{\langle \boldsymbol{w}, \boldsymbol{v}_i\rangle}$ for each $s \in S$ and $c_0 = a_\tau \gamma^{\langle \boldsymbol{w}, \boldsymbol{u}_\tau\rangle}$. Let $f(x) = c_0 + \sum_{s\in S} c_s x^s$. The degree of f is at most d and $\xi_i = f(\gamma^{\alpha_i})$ for $i \in [d+1]$. By using Lagrange interpolation, the client can compute $f(0) = c_0 = a_\tau \gamma^{\langle \boldsymbol{w}, \boldsymbol{u}_\tau\rangle}$ from $\xi_1, \ldots, \xi_{d+1}$ and obtain a_τ.

5 Formalization of Fully Error Detecting PIR

5.1 Definitions

We formally define error detecting PIR. In a t-private PIR scheme, any t servers learn no information on τ even if they collude, where τ is the secret index of the client. In a t-private error detecting ℓ-server PIR scheme, we require that the client can detect errors even if $\ell - 1$ servers return false answers. We allow only t servers to collude when computing their false answers, which is the same condition as that for t-privacy. Namely a set of malicious servers T is given by a union of pairwise disjoint subsets $T = T_1 \cup \cdots \cup T_m$ in such a way that $|T| \leq \ell - 1$, $|T_h| \leq t$ for $h \in [m]$ and the servers in each T_h can collude. We formalize such malicious servers by using a tampering function f such that

$$f(\mathsf{que}_1, \ldots, \mathsf{que}_\ell, \boldsymbol{a}) = (\widetilde{\mathsf{ans}}_1, \ldots, \widetilde{\mathsf{ans}}_\ell), \tag{1}$$

where que_i is a query sent to the i-th server and $\boldsymbol{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$ is a database.

▶ **Definition 9** (Tampering function). *Let $T_1, \ldots, T_m \subseteq [\ell]$ be pairwise disjoint subsets. We say that a function f given by Eq. (1) is a tampering function for an ℓ-server PIR scheme $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ with respect to $(T_1, \ldots, T_m)$ if for each $i \in [\ell]$, it holds that*

$$\widetilde{\mathsf{ans}}_i = \begin{cases} \mathcal{A}(i, \mathsf{que}_i, \boldsymbol{a}), & \text{if } i \notin T_1 \cup \cdots \cup T_m, \\ f_i(\{\mathsf{que}_{i'}\}_{i'\in T_j}, \boldsymbol{a}), & \text{if } i \in T_j \text{ for some } j \in [m], \end{cases} \tag{2}$$

for some function f_i. We denote the family of all such tampering functions by $\mathcal{F}^{\Pi}_{T_1,\ldots,T_m}$.

▶ **Definition 10** (Error detecting PIR). *We say that an ℓ-server PIR scheme $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ is $(1-\epsilon)$-error detecting with respect to $(T_1, \ldots, T_m)$ if Π is correct and*

$$\Pr[\mathsf{ED}_\Pi(\boldsymbol{a}, \tau, f) = 1] \geq 1 - \epsilon$$

for any database $\boldsymbol{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$, any $\tau \in [n]$ and any $f \in \mathcal{F}^{\Pi}_{T_1,\ldots,T_m}$, where the experiment $\mathsf{ED}_\Pi(\boldsymbol{a}, \tau, f)$ is defined as follows:

1. *Let $(\mathsf{que}_1, \ldots, \mathsf{que}_\ell; \mathsf{aux}) \leftarrow \mathcal{Q}(\tau)$;*
2. *Let $f(\mathsf{que}_1, \ldots, \mathsf{que}_\ell, \boldsymbol{a}) = (\widetilde{\mathsf{ans}}_1, \ldots, \widetilde{\mathsf{ans}}_\ell)$;*
3. *Return 1 if $\mathcal{R}(\widetilde{\mathsf{ans}}_1, \ldots, \widetilde{\mathsf{ans}}_\ell; \mathsf{aux}) \in \{a_\tau, \perp\}$ and return 0 otherwise.*

We say that a t-private ℓ-server PIR scheme Π is $(1-\epsilon)$-fully error detecting if it is $(1-\epsilon)$-error detecting with respect to any tuple of pairwise disjoint subsets $(T_1, \ldots, T_m)$ such that $|T_1 \cup \cdots \cup T_m| \leq \ell - 1$ and $|T_i| \leq t$ for $i \in [m]$.

▶ Remark 11. Although tampering functions are supposed to be deterministic, it can be seen that they capture randomized behavior of malicious servers. This is because the success probability is considered over a random string of $\mathcal{Q}$, which is independent of servers' randomness, and also because servers are assumed to be computationally unbounded.

▶ Remark 12. In a t-private fully error detecting PIR scheme, incorrect answers are allowed to depend on at most t queries. In particular, if $t = 1$, this means that the incorrect answers are generated independently. We note that this somewhat restricted adversarial model is still practically important. For example, consider a situation where a database $\boldsymbol{a}$ is updated frequently. If an honest server i has an old database $\boldsymbol{b} \neq \boldsymbol{a}$, then it returns an incorrect answer $\mathcal{A}(i, \mathsf{que}_i, \boldsymbol{b})$. Such errors can be detected by a 1-private fully error detecting PIR scheme.

5.2 Impossibility of 1-Fully Error Detecting PIR

The trivial scheme clearly achieves 1-full error detection, i.e., $\epsilon = 0$. Theorem 13 shows that we cannot do better than the trivial scheme in the case of $\epsilon = 0$.

▶ **Theorem 13.** *Let $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ be a 1-private 1-fully error detecting ℓ-server PIR scheme for a universe of databases $\{0,1\}^n$. Then, the bit length of an answer of any server is at least n.*

Proof. Suppose that $\mathcal{A}$ outputs a c-bit string and the set of random strings for $\mathcal{Q}$ is $\{0,1\}^\rho$. We show that $c \geq n$. Let $r^{(1)} \in \{0,1\}^\rho$ be any random string for $\mathcal{Q}$ and let $(q_1^{(1)}, \ldots, q_\ell^{(1)}; \mathsf{aux}^{(1)}) = \mathcal{Q}(1; r^{(1)})$. Since Π is 1-private, for any $\tau \in [n] \setminus \{1\}$, there exists $r^{(\tau)} \in \{0,1\}^\rho$ such that $q_1^{(\tau)} = q_1^{(1)}$, where $(q_1^{(\tau)}, \ldots, q_\ell^{(\tau)}; \mathsf{aux}^{(\tau)}) = \mathcal{Q}(\tau; r^{(\tau)})$. We define $q_1 := q_1^{(1)} = q_1^{(2)} = \cdots = q_1^{(n)}$.

We define a function $\phi : \{0,1\}^n \to \{0,1\}^c$ as $\phi(\boldsymbol{a}) = \mathcal{A}(1, q_1, \boldsymbol{a})$. It is sufficient to show that ϕ is injective. Assume that $\mathcal{A}(1, q_1, \boldsymbol{a}) = \mathcal{A}(1, q_1, \boldsymbol{b})$ for some $\boldsymbol{a} \neq \boldsymbol{b} \in \{0,1\}^n$. Let $\tau \in [n]$ be such that $a_\tau \neq b_\tau$. Then, we have that $\mathcal{A}(1, q_1^{(\tau)}, \boldsymbol{a}) = \mathcal{A}(1, q_1^{(\tau)}, \boldsymbol{b})$. Set $T = [\ell] \setminus \{1\}$. Let $f \in \mathcal{F}^{\Pi}_{\{2\},\ldots,\{\ell\}}$ be any tampering function such that

$$f(q_1^{(\tau)}, q_2^{(\tau)}, \ldots, q_\ell^{(\tau)}, \boldsymbol{a}) = (\mathcal{A}(1, q_1^{(\tau)}, \boldsymbol{a}), (\mathcal{A}(i, q_i^{(\tau)}, \boldsymbol{b}))_{i \in T}).$$

Consider the experiment $\mathsf{ED}_\Pi(\boldsymbol{a}, \tau, f)$ in Definition 10. If $r^{(\tau)}$ is chosen, we have that $\mathcal{Q}(\tau; r^{(\tau)}) = (q_1^{(\tau)}, \ldots, q_\ell^{(\tau)}; \mathsf{aux}^{(\tau)})$ at Step 1. At Step 2, it holds that $\widetilde{\mathsf{ans}}_1 = \mathcal{A}(1, q_1^{(\tau)}, \boldsymbol{a})$ and $\widetilde{\mathsf{ans}}_i = \mathcal{A}(i, q_i^{(\tau)}, \boldsymbol{b})$ for $i \in T$. Then, $\mathsf{ED}_\Pi(\boldsymbol{a}, \tau, f)$ returns 0 since

$$\begin{aligned}
\mathcal{R}(\widetilde{\mathsf{ans}}_1, (\widetilde{\mathsf{ans}}_i)_{i \in T}; \mathsf{aux}^{(\tau)}) &= \mathcal{R}(\mathcal{A}(1, q_1^{(\tau)}, \boldsymbol{a}), (\mathcal{A}(i, q_i^{(\tau)}, \boldsymbol{b}))_{i \in T}; \mathsf{aux}^{(\tau)}) \\
&= \mathcal{R}(\mathcal{A}(1, q_1^{(\tau)}, \boldsymbol{b}), (\mathcal{A}(i, q_i^{(\tau)}, \boldsymbol{b}))_{i \in T}; \mathsf{aux}^{(\tau)}) \\
&= b_\tau \notin \{a_\tau, \perp\}.
\end{aligned}$$

Hence $\Pr[\mathsf{ED}_\Pi(\boldsymbol{a}, \tau, f) = 0] \geq \Pr\left[r^{(\tau)} \leftarrow_{\$} \{0,1\}^\rho\right] > 0$, which contradicts the 1-full error detection of Π. ◀

In view of Theorem 13, we will consider $(1-\epsilon)$-fully error detecting PIR with $\epsilon > 0$ in the following sections.

6 Transformation to Increase Robustness of Fully Error Detecting PIR

Beimel ans Stahl [5] presented two generic transformations from a k-server PIR scheme Π to (k,ℓ)-robust (and hence $\lfloor(\ell-k)/2\rfloor$-error correcting) PIR scheme Π'. One is based on a perfect hash family and the other simply executes Π for all groups of k servers. We prove that these methods preserve full error detection capability and can be used to add error correction capability to fully error detecting PIR schemes. Specifically, let Π be a $(1-\epsilon)$-fully error detecting k-server PIR scheme and Π' be an $\lfloor(\ell-k)/2\rfloor$-error correcting ℓ-server PIR scheme obtained by applying one of the transformations [5] to Π. We prove that Π' is $(1-\epsilon')$-fully error detecting for a certain $\epsilon' \le \epsilon$. Although the method based on a perfect hash family is more communication-efficient for large k, the naive method has the following advantages:

- From any 1-private 2-server $(1-\epsilon)$-fully error detecting PIR scheme, we can obtain a 1-private ℓ-server $(1-\epsilon)$-fully error detecting one which has lower communication cost by a factor of $O(\log \ell)$ than if Theorem 14 is applied;
- It works for any $t \ge 1$, where t is the number of servers who can collude.

First, we consider the transformation based on a perfect hash family $\mathcal{H} = \{h_i : [\ell] \to [k] : i \in [w]\}$ (see Definition 5). In Π', a client executes w independent instances $\Pi_1, \ldots, \Pi_w$ of Π and sends to Server $i \in [\ell]$ a query sent to Server $h_j(i) \in [k]$ in Π_j for all $j \in [w]$. We show that if Server i is honest, for any subset $S \subseteq [\ell]$ of size k containing i, the $(1-\epsilon)$-full error detection of Π' follows from that of Π_j, where j is an index such that $h_j(S) = [k]$. We note that this transformation does not provide a t-private $(1-\epsilon)$-fully error detecting ℓ-server PIR scheme for $t \ge 2$, that is, Π' is not necessarily t-private $(1-\epsilon)$-fully error detecting even if Π is. Roughly speaking, this is because the answer of a malicious server may depend on the query which is sent to an honest server. In summary the following theorem holds. See the full version for the proof.

▶ **Theorem 14.** *Suppose that there exists a 1-private $(1-\epsilon)$-fully error detecting k-server PIR scheme $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ such that the communication complexity is c per server. Then, for any $\ell \ge k$, there exists a 1-private $(1-\epsilon)$-fully error detecting ℓ-server PIR scheme Π' with communication complexity $c \cdot 2^{O(k)} \ell \log \ell$. Furthermore, Π' is (k,ℓ)-robust and hence $\lfloor(\ell-k)/2\rfloor$-error correcting.*

Second, the naive transformation executes $p = \binom{\ell}{k}$ independent instances of Π for all groups $S_1, \ldots, S_p$ of k servers. Let T be a set of $\ell - 1$ malicious servers. Suppose that T is partitioned into pairwise disjoint subsets $T = T_1 \cup \cdots \cup T_m$ such that $|T_h| \le t$ for any h and servers in each T_h can collude. For $i \in [p]$, we show that if $|S_i \cap T| \le k-1$, the $(1-\epsilon)$-full error detection of Π' follows from that of the instance of Π corresponding to S_i. More generally, based on the fact that a client's randomness for S_i, S_j $(i \ne j)$ are independent, we show that Π' is even $(1-\epsilon^M)$-fully error detecting if there are M subsets $S_{i_1}, \ldots, S_{i_M}$ such that for every pair S_i, S_j, servers in S_i and in S_j do not receive the same query. We formalize that condition in a combinatorial way and show that the maximum number of M is at least $(\ell - k + 1)/(k + t - 2)$. As a result, Π' is t-private $(1-\epsilon')$-fully error detecting for $\epsilon' = \epsilon^M$. See Appendix A for the proof.

▶ **Theorem 15.** *Suppose that there exists a t-private $(1-\epsilon)$-fully error detecting k-server PIR scheme $\Pi = (\mathcal{Q}, \mathcal{A}, \mathcal{R})$ with communication complexity c. Let $\ell \ge k$. Set*

$$M' = \left\lceil \frac{\ell - k + 1}{k + t - 2} \right\rceil$$

and $\epsilon' = \epsilon^{M'}$. Then, there exists a t-private $(1-\epsilon')$-fully error detecting ℓ-server PIR scheme Π' with communication complexity $c \cdot \binom{\ell}{k}$. Furthermore, Π' is (k,ℓ)-robust and hence $\lfloor(\ell-k)/2\rfloor$-error correcting.

7 1-Private Fully Error Detecting PIR with Sub-polynomial Communication

In this section, we show 1-private $(1-\epsilon)$-fully error detecting ℓ-server PIR schemes Π'_1 and Π'_2 such that:

- For any $\ell \geq 2$, Π'_1 is $\lfloor(\ell-2)/2\rfloor$-error correcting and has the communication complexity $\mathcal{L}_n[1/2, O(1)] \cdot \ell \log \epsilon^{-1}$.
- For any $r \geq 2$ and any $\ell \geq k := r^r 2^{r^2+2r-3}+1$, Π'_2 is $\lfloor(\ell-k)/2\rfloor$-error correcting and has the communication complexity $\mathcal{L}_n[1/r, 2^{O(r)}] \cdot \ell \log \ell \log \epsilon^{-1}$.

7.1 How to Construct Π'_1

In this subsection, we show a 1-private $(1-\epsilon)$-fully error detecting 2-server PIR scheme Π_1 with communication complexity $\mathcal{L}_n[1/2, O(1)] \cdot \log \epsilon^{-1}$. We can obtain Π'_1 by applying Theorem 15 to Π_1. The scheme Π_1 is a variant of the 1-private 2-server PIR scheme of Dvir and Gopi [10] with communication complexity $\mathcal{L}_n[1/2, O(1)]$. Their scheme uses a matching vector family given by Proposition 7 with $p=2$ and $q=3$, and does a sort of polynomial interpolation with fixed points $\beta_1 = \gamma^0$ and $\beta_2 = \gamma^1$. On the other hand, Π_1 uses a matching vector family with $p \geq 3$ and $q = 1 \bmod p$, and does polynomial interpolation with random points $\beta_1 = \gamma^{\alpha_1}$ and $\beta_2 = \gamma^{\alpha_2}$ where α_1, α_2 are randomly chosen from $\{0, 1, \ldots, p-1\}$. A more formal description of Π_1 is shown in Figure 1. We obtain the following theorem. See Appendix B for the proof.

▶ **Theorem 16.** *For any $\epsilon > 0$, Π_1 is a 1-private $(1-\epsilon)$-fully error detecting 2-server PIR scheme with communication complexity $\mathcal{L}_n[1/2, O(1)] \cdot \log \epsilon^{-1}$.*

By applying Theorem 15 to the $(1-\epsilon)$-fully error detecting 2-server scheme Π_1, we obtain a $(1-\epsilon^{\Theta(\ell)})$-fully error detecting ℓ-server scheme Π'_1, which means that the overhead in communication cost is only $O(\ell)$. Note that if we apply Theorem 14 to Π_1, then the overhead is $O(\ell \log \ell)$.

▶ **Corollary 17.** *Let $\epsilon > 0$. For any $\ell \geq 2$, there exists a 1-private $(1-\epsilon)$-fully error detecting and $\lfloor(\ell-2)/2\rfloor$-error correcting ℓ-server PIR scheme Π'_1 for a universe of databases $\{0,1\}^n$ such that the communication complexity is*

$$\mathcal{L}_n[1/2, O(1)] \cdot \ell \log \epsilon^{-1} \tag{3}$$

and the time complexity of its reconstruction algorithm is polynomial in ℓ, n and $\log \epsilon^{-1}$.

7.2 How to Construct Π'_2

In this subsection, for $r \geq 2$ and $k = r^r 2^{r^2+2r-3}+1$, we show a 1-private $(1-\epsilon)$-fully error detecting k-server PIR scheme Π_2 such that the communication complexity is $\mathcal{L}_n[1/r, 2^{O(r)}] \cdot \log \epsilon^{-1}$. We can obtain Π'_2 by applying Theorem 14 to Π_2.

To construct Π_2, we first consider a variant of the 1-private k-server PIR scheme of Section 4.2 such that $\alpha_1, \ldots, \alpha_k$ are chosen randomly (Figure 2). The following theorem holds. See the full version for the proof.

Notations.

- A positive integer λ
- Two primes $p < q$ such that $q \equiv 1 \bmod p$
- $m = pq$ and a $\{p, q, p+q\}$-matching vector family $\mathcal{U}, \mathcal{V}$ over $\mathbb{Z}_m^h$ given by Proposition 7
- A primitive root $\delta \in \mathbb{F}_q^*$ and $\gamma = \delta^{(q-1)/p}$
- The ring homomorphism $\phi : \mathbb{Z}_m \to \mathbb{F}_q$ defined as $\phi(x) = x \bmod q$
- Polynomials $F_{\boldsymbol{a}} \in \mathbb{F}_q[z_1, \ldots, z_h]$ and $G_{\boldsymbol{a}} \in (\mathbb{F}_q^h)[z_1, \ldots, z_h]$ associated with $\boldsymbol{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$ defined as $F_{\boldsymbol{a}}(z_1, \ldots, z_h) = \sum_{\tau \in [n]} a_\tau z_1^{v_{\tau 1}} \cdots z_h^{v_{\tau h}}$, and $G_{\boldsymbol{a}}(z_1, \ldots, z_h) = \sum_{\tau \in [n]} a_\tau \phi(\boldsymbol{v}_\tau) z_1^{v_{\tau 1}} \cdots z_h^{v_{\tau h}}$, where we assume $\boldsymbol{a} \in \mathbb{F}_q^n$, $v_{\tau j}$ is the j-th coordinate of $\boldsymbol{v}_\tau \in \mathbb{Z}_m^h$, and ϕ is applied on vectors entry-wise.

$\mathcal{Q}(\tau)$. Given an input $\tau \in [n]$:

1. For each $j \in [\lambda]$:
   a. Choose two distinct elements $\alpha_1^{(j)}, \alpha_2^{(j)} \in \{0, 1, 2, \ldots, p-1\}$ randomly.
   b. Choose $\boldsymbol{w}^{(j)} \leftarrow_{\$} \mathbb{Z}_m^h$.
   c. Let $(\rho_{i1}^{(j)}, \ldots, \rho_{ih}^{(j)}) = \boldsymbol{w}^{(j)} + \alpha_i^{(j)} \boldsymbol{u}_\tau \in \mathbb{Z}_m^h$ for $i \in \{1, 2\}$.
2. Output $\mathsf{que}_i = (\gamma^{\rho_{i1}^{(j)}}, \ldots, \gamma^{\rho_{ih}^{(j)}})_{j \in [\lambda]}$ for $i \in \{1, 2\}$ together with $\mathsf{aux} = ((\alpha_1^{(j)}, \alpha_2^{(j)})_{j \in [\lambda]}, (\boldsymbol{w}^{(j)})_{j \in [\lambda]}, \boldsymbol{u}_\tau, \boldsymbol{v}_\tau)$.

$\mathcal{A}(i, \mathsf{que}_i, a)$. Given $i \in \{1, 2\}$, a query que_i, and a database $\boldsymbol{a} \in \{0,1\}^n$:

1. Parse $\mathsf{que}_i = (\gamma^{\rho_{i1}^{(j)}}, \ldots, \gamma^{\rho_{ih}^{(j)}})_{j \in [\lambda]}$.
2. For each $j \in [\lambda]$, let $\xi_i^{(j)} = F_{\boldsymbol{a}}(\gamma^{\rho_{i1}^{(j)}}, \ldots, \gamma^{\rho_{ih}^{(j)}})$ and $\boldsymbol{\zeta}_i^{(j)} = G_{\boldsymbol{a}}(\gamma^{\rho_{i1}^{(j)}}, \ldots, \gamma^{\rho_{ih}^{(j)}})$.
3. Output $\mathsf{ans}_i = (\xi_i^{(j)}, \boldsymbol{\zeta}_i^{(j)})_{j \in [\lambda]}$.

$\mathcal{R}(\widetilde{\mathsf{ans}}_1, \widetilde{\mathsf{ans}}_2; \mathsf{aux})$. Given two answers $\widetilde{\mathsf{ans}}_i = (\widetilde{\xi}_i^{(j)}, \widetilde{\boldsymbol{\zeta}}_i^{(j)})_{j \in [\lambda]} \in (\mathbb{F}_q^{h+1})^\lambda$ and auxiliary information $\mathsf{aux} = ((\alpha_1^{(j)}, \alpha_2^{(j)})_{j \in [\lambda]}, (\boldsymbol{w}^{(j)})_{j \in [\lambda]}, \boldsymbol{u}_\tau, \boldsymbol{v}_\tau)$:

1. Let $\mathcal{L} = \emptyset$.
2. For each $j \in [\lambda]$:
   a. Let $\widetilde{\eta}_i^{(j)} = \langle \phi(\boldsymbol{u}_\tau), \widetilde{\boldsymbol{\zeta}}_i^{(j)} \rangle$ and $\beta_i^{(j)} = \gamma^{\alpha_i^{(j)}}$ for $i \in \{1, 2\}$.
   b. Define an invertible matrix $M^{(j)}$ as

$$M^{(j)} = \begin{pmatrix} 1 & 1 & \beta_1^{(j)} & \beta_1^{(j)} \\ 0 & p & 0 & p\beta_1^{(j)} \\ 1 & 1 & \beta_2^{(j)} & \beta_2^{(j)} \\ 0 & p & 0 & p\beta_2^{(j)} \end{pmatrix} \in \mathbb{F}_q^{4 \times 4}.$$

   c. Find $\widetilde{c}_0^{(j)}, \widetilde{c}_p^{(j)}, \widetilde{c}_q^{(j)}, \widetilde{c}_{p+q}^{(j)} \in \mathbb{F}_q$ such that

$$M^{(j)} \begin{pmatrix} \widetilde{c}_0^{(j)} & \widetilde{c}_p^{(j)} & \widetilde{c}_q^{(j)} & \widetilde{c}_{p+q}^{(j)} \end{pmatrix}^\top = \begin{pmatrix} \widetilde{\xi}_1^{(j)} & \widetilde{\eta}_1^{(j)} & \widetilde{\xi}_2^{(j)} & \widetilde{\eta}_2^{(j)} \end{pmatrix}^\top.$$

   d. Add $\widetilde{c}_0^{(j)} \gamma^{-\langle \boldsymbol{w}^{(j)}, \boldsymbol{v}_\tau \rangle}$ to $\mathcal{L}$ if $\widetilde{c}_0^{(j)} \gamma^{-\langle \boldsymbol{w}^{(j)}, \boldsymbol{v}_\tau \rangle} \in \{0, 1\}$. Otherwise, output $\perp$.
3. If $\mathcal{L} = \{s\}$ for some $s \in \{0, 1\}$, output s. Otherwise, output $\perp$.

Figure 1 A 1-private $(1 - \epsilon)$-fully error detecting 2-server PIR scheme Π_1.

▶ **Theorem 18.** *Given a d-bounded matching vector family $\mathcal{U}, \mathcal{V}$ of size n over $\mathbb{Z}_m^h$, there exists a 1-private $(1-\epsilon)$-fully error detecting k-server PIR scheme with communication complexity $O(kh\lambda \log m)$ for $k \geq d+1$ and*

$$\epsilon := \left(\frac{k-1}{m-k+1}\right)^{\lambda}.$$

In the full version, we present a matching vector family that is suitable for the scheme in Theorem 18. The following corollary can be obtained by combining Theorem 18 and that matching vector family. See the full version for the details.

▶ **Corollary 19.** *Let $r \geq 2$ and $\epsilon > 0$. Set $k = r^r 2^{r^2+2r-3} + 1$. Then, there exists a function $n_0 = n_0(r) = \exp(O(2^r r))$ such that the following holds: For any $n \geq n_0$, there exists a 1-private $(1-\epsilon)$-fully error detecting k-server PIR scheme Π_2 for a universe of databases $\{0,1\}^n$ such that the communication complexity is $\mathcal{L}_n[1/r, 2^{O(r)}] \cdot k \log \epsilon^{-1}$ and the time complexity of its reconstruction algorithm is polynomial in k, n and $\log \epsilon^{-1}$.*

By applying Theorem 14 to the k-server PIR scheme Π_2, we obtain a $(1-\epsilon)$-fully error detecting ℓ-server PIR scheme Π_2' while the overhead in communication cost is $2^{O(k)}\ell \log \ell$.

▶ **Corollary 20.** *Let $r \geq 2$ and $\epsilon > 0$. Set $k = r^r 2^{r^2+2r-3} + 1$. For a sufficiently large n (depending on r only) and any $\ell \geq k$, there exists a 1-private $(1-\epsilon)$-fully error detecting and $\lfloor(\ell-k)/2\rfloor$-error correcting ℓ-server PIR scheme Π_2' for a universe of databases $\{0,1\}^n$ such that the communication complexity is*

$$\mathcal{L}_n[1/r, 2^{O(r)}] \cdot 2^{O(k)} \ell \log \ell \log \epsilon^{-1}$$

and the time complexity of its reconstruction algorithm is $\binom{\ell}{k} \cdot \mathsf{poly}\left(k, n, \log \epsilon^{-1}\right)$.

If we set $r = 3$, for any $\ell \geq 2^{17}$, there is a 1-private $(1-\epsilon)$-fully error detecting ℓ-server PIR scheme such that the communication complexity is $\mathcal{L}_n[1/3, O(1)] \cdot \ell \log \ell \log \epsilon^{-1}$, which is lower than the communication complexity (3) of Corollary 17 as functions of n.

8 t-Private Fully Error Detecting PIR with Polynomial Communication

In this section, we show a t-private $(1-\epsilon)$-fully error detecting and $\lfloor(\ell-k)/2\rfloor$-error correcting PIR scheme Π with polynomial (in n) communication. Our scheme Π is the same as the t-private ℓ-server PIR scheme [20] except that it uses Hermite interpolation with random points α_i to achieve error detection (Figure 3). Note that Π has in common with the scheme [20] that a client can compute $\{(g(\alpha_i), \partial g(\alpha_i)) : i \in B\}$ for the unique polynomial g with $g(0) = a_\tau$, where B is a set of honest servers. This property implies that the polynomial-time error correction algorithm of [15] is applicable to Π. We obtain the following theorem. See the full version for the proof.

▶ **Theorem 21.** *Let $\epsilon > 0$ and $\ell \geq k \geq t \geq 1$. Set $d = \lfloor(2k-1)/t\rfloor$. Then, there exists a t-private $(1-\epsilon)$-fully error detecting and $\lfloor(\ell-k)/2\rfloor$-error correcting ℓ-server PIR scheme for a universe of databases $\{0,1\}^n$ such that the communication complexity is $O(dn^{1/d}\ell \log \ell \log \epsilon^{-1})$ and the time complexity of its reconstruction algorithm is polynomial in ℓ, n and $\log \epsilon^{-1}$.*

Notations.

- Positive integers k, d, λ
- $m \geq k$ and a d-bounded S-matching vector family $\mathcal{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n)$, $\mathcal{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)$ over $\mathbb{Z}_m^h$
- A prime field $\mathbb{F}_q$ such that $q \equiv 1 \bmod m$
- A primitive root $\delta \in \mathbb{F}_q^*$ and $\gamma = \delta^{(q-1)/m}$
- A polynomial $F_{\boldsymbol{a}}$ associated with $\boldsymbol{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$ defined as

$$F_{\boldsymbol{a}}(z_1, \ldots, z_h) = \sum_{\tau \in [n]} a_\tau z_1^{v_{\tau 1}} \cdots z_h^{v_{\tau h}} \in \mathbb{F}_q[z_1, \ldots, z_h],$$

where we assume $\boldsymbol{a} \in \mathbb{F}_q^n$ and $v_{\tau j}$ is the j-th coordinate of $\boldsymbol{v}_\tau \in \mathbb{Z}_m^h$

$\mathcal{Q}(\tau)$. Given an input $\tau \in [n]$:

1. For each $j \in [\lambda]$:
   a. Choose k pairwise distinct random elements $\alpha_1^{(j)}, \ldots, \alpha_k^{(j)} \in \mathbb{Z}_m$.
   b. Choose $\boldsymbol{w}^{(j)} \leftarrow_{\$} \mathbb{Z}_m^h$.
   c. Let $(\rho_{i1}^{(j)}, \ldots, \rho_{ih}^{(j)}) = \boldsymbol{w}^{(j)} + \alpha_i^{(j)} \boldsymbol{u}_\tau \in \mathbb{Z}_m^h$ for $i \in [k]$.
2. Output $\mathsf{que}_i = (\gamma^{\rho_{i1}^{(j)}}, \ldots, \gamma^{\rho_{ih}^{(j)}})_{j \in [\lambda]}$ for $i \in [k]$ together with $\mathsf{aux} = ((\alpha_1^{(j)}, \ldots, \alpha_k^{(j)})_{j \in [\lambda]}, (\boldsymbol{w}^{(j)})_{j \in [\lambda]}, \boldsymbol{v}_\tau)$.

$\mathcal{A}(i, \mathsf{que}_i, a)$. Given $i \in [k]$, a query que_i, and a database $\boldsymbol{a} \in \{0,1\}^n$:

1. Parse $\mathsf{que}_i = (\gamma^{\rho_{i1}^{(j)}}, \ldots, \gamma^{\rho_{ih}^{(j)}})_{j \in [\lambda]}$.
2. For each $j \in [\lambda]$, let $\zeta_i^{(j)} = F_{\boldsymbol{a}}(\gamma^{\rho_{i1}^{(j)}}, \ldots, \gamma^{\rho_{ih}^{(j)}})$.
3. Output $\mathsf{ans}_i = (\zeta_i^{(j)})_{j \in [\lambda]}$.

$\mathcal{R}(\widetilde{\mathsf{ans}}_1, \ldots, \widetilde{\mathsf{ans}}_k; \mathsf{aux})$. Given k answers $\widetilde{\mathsf{ans}}_i = (\widetilde{\zeta}_i^{(j)})_{j \in [\lambda]} \in \mathbb{F}_q^\lambda$ and auxiliary information $\mathsf{aux} = ((\alpha_1^{(j)}, \ldots, \alpha_k^{(j)})_{j \in [\lambda]}, (\boldsymbol{w}^{(j)})_{j \in [\lambda]}, \boldsymbol{v}_\tau)$:

1. Let $\mathcal{L} = \emptyset$.
2. Choose any subset $A \subseteq [k]$ of size $d+1$.
3. For each $j \in [\lambda]$:
   a. Compute a degree-d polynomial $\widetilde{g}^{(j)}(x) \in \mathbb{F}_q[x]$ such that $\widetilde{g}^{(j)}(\gamma^{\alpha_i^{(j)}}) = \widetilde{\zeta}_i^{(j)}$ for all $i \in A$, using Lagrange interpolation.
   b. Add $\widetilde{g}^{(j)}(0)\gamma^{-\langle \boldsymbol{w}^{(j)}, \boldsymbol{v}_\tau \rangle}$ to $\mathcal{L}$ if $\widetilde{\zeta}_i^{(j)} = \widetilde{g}^{(j)}(\gamma^{\alpha_i^{(j)}})$ for all $i \notin A$ and $\widetilde{g}^{(j)}(0)\gamma^{-\langle \boldsymbol{w}^{(j)}, \boldsymbol{v}_\tau \rangle} \in \{0,1\}$. Otherwise, output $\perp$.
4. If $\mathcal{L} = \{s\}$ for some $s \in \{0,1\}$, output s. Otherwise, output $\perp$.

Figure 2 A 1-private $(1-\epsilon)$-fully error detecting k-server PIR scheme Π.

Notations.

- A prime field $\mathbb{F}_p$ such that $p \geq \ell + 1$.
- A positive integer k
- $d = \lfloor (2k-1)/t \rfloor$ and $m \in \mathbb{N}$ such that $\binom{m}{d} \geq n$
- An injection $E : [n] \to \{0,1\}^m$ such that $\mathsf{wt}(E(\tau)) = d$
- $\boldsymbol{u}^{E(\tau)} = \prod_{j:E(\tau)_j=1} u_j$ for $\boldsymbol{u} = (u_j)_{j\in[m]}$, where $E(\tau) = (E(\tau)_j)_{j\in[m]}$
- A polynomial $F_{\boldsymbol{a}}$ associated with $\boldsymbol{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$ defined as

$$F_{\boldsymbol{a}}(z_1, \ldots, z_m) = \sum_{\tau\in[n]} a_\tau \boldsymbol{z}^{E(\tau)} \in \mathbb{F}_p[z_1, \ldots, z_m],$$

where $\boldsymbol{z} = (z_1, \ldots, z_m)$ and we assume $\boldsymbol{a} \in \mathbb{F}_p^n$

$\mathcal{Q}(\tau)$. Given an input $\tau \in [n]$:

1. Choose ℓ pairwise distinct random non-zero elements $\alpha_1, \ldots, \alpha_\ell \in \mathbb{F}_p$.
2. Choose $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t \leftarrow_{\$} \mathbb{F}_p^m$.
3. Let $\boldsymbol{q}_i = E(\tau) + \alpha_i \boldsymbol{v}_1 + \cdots + \alpha_i^t \boldsymbol{v}_t$ for $i \in [\ell]$.
4. Set $\boldsymbol{w}_i = \boldsymbol{v}_1 + 2\alpha_i \boldsymbol{v}_2 + \cdots + t\alpha^{t-1} \boldsymbol{v}_t$.
5. Output $\mathsf{que}_i = \boldsymbol{q}_i$ for $i \in [\ell]$ together with auxiliary information $\mathsf{aux} = ((\alpha_i)_{i\in[\ell]}, (\boldsymbol{w}_i)_{i\in[\ell]})$.

$\mathcal{A}(i, \mathsf{que}_i, a)$. Given $i \in [\ell]$, a query $\mathsf{que}_i = \boldsymbol{q}_i$, and a database $\boldsymbol{a} \in \{0,1\}^n$:

1. Let $\zeta_i^{(0)} = F_{\boldsymbol{a}}(\boldsymbol{q}_i)$ and $\zeta_{ij}^{(1)} = \partial_{z_j} F_{\boldsymbol{a}}(\boldsymbol{q}_i)$ for $j \in [m]$.
2. Output $\mathsf{ans}_i = (\zeta_i^{(0)}, (\zeta_{ij}^{(1)})_{j\in[m]})$.

$\mathcal{R}(\widetilde{\mathsf{ans}}_1, \ldots, \widetilde{\mathsf{ans}}_\ell; \mathsf{aux})$. Given ℓ answers $\widetilde{\mathsf{ans}}_i = (\widetilde{\zeta}_i^{(0)}, (\widetilde{\zeta}_{ij}^{(1)})_{j\in[m]}) \in \mathbb{F}_p^{m+1}$ and auxiliary information $\mathsf{aux} = ((\alpha_i)_{i\in[\ell]}, (\boldsymbol{w}_i)_{i\in[\ell]})$:

1. Let $\widetilde{\boldsymbol{\zeta}}_i^{(1)} = (\widetilde{\zeta}_{i1}^{(1)}, \ldots, \widetilde{\zeta}_{im}^{(1)})$ for all $i \in [\ell]$.
2. Let $\widetilde{\xi}_i^{(0)} = \widetilde{\zeta}_i^{(0)}$ and $\widetilde{\xi}_i^{(1)} = \langle \widetilde{\boldsymbol{\zeta}}_i^{(1)}, \boldsymbol{w}_i \rangle$ for all $i \in [\ell]$.
3. Choose any subset $A \subseteq [\ell]$ of size at least $(td+1)/2$.
4. Compute a polynomial $\widetilde{g}(x) \in \mathbb{F}_p[x]$ of degree at most td such that $\widetilde{\xi}_i^{(0)} = \widetilde{g}(\alpha_i)$ and $\widetilde{\xi}_i^{(1)} = \partial\widetilde{g}(\alpha_i)$ for all $i \in A$, using Hermite interpolation.
5. Output $\widetilde{g}(0)$ if $\widetilde{\xi}_i^{(0)} = \widetilde{g}(\alpha_i)$ and $\widetilde{\xi}_i^{(1)} = \partial\widetilde{g}(\alpha_i)$ for all $i \notin A$ and $\widetilde{g}(0) \in \{0,1\}$. Otherwise, output $\perp$.

Figure 3 A t-private $(1-\epsilon)$-fully error detecting ℓ-server PIR scheme.

References

1 A. Ambainis. Upper bound on the communication complexity of private information retrieval. In *Automata, Languages and Programming*, pages 401–407, 1997.

2 K. Banawan and S. Ulukus. The capacity of private information retrieval from byzantine and colluding databases. *IEEE Transactions on Information Theory*, 65(2):1206–1219, 2019.

3 A. Beimel and Y. Ishai. Information-theoretic private information retrieval: A unified construction. In *Automata, Languages and Programming*, pages 912–926, 2001.

4 A. Beimel, Y. Ishai, E. Kushilevitz, and J.-F. Raymond. Breaking the o(n/sup 1/(2k-1)/) barrier for information-theoretic private information retrieval. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 261–270, 2002.

5 A. Beimel and Y. Stahl. Robust information-theoretic private information retrieval. *Journal of Cryptology*, 20(3):295–321, 2007.

6 Y.M. Chee, T. Feng, S. Ling, H. Wang, and L.F. Zhang. Query-efficient locally decodable codes of subexponential length. *computational complexity*, 22(1):159–189, 2013.

7 B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–982, 1998.

8 C. Devet, I. Goldberg, and N. Heninger. Optimally robust private information retrieval. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 269–283, 2012.

9 Z. Dvir, P. Gopalan, and S. Yekhanin. Matching vector codes. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 705–714, 2010.

10 Z. Dvir and S. Gopi. 2-server pir with subpolynomial communication. *Journal of the ACM*, 63(4):1–15, 2016.

11 K. Efremenko. 3-query locally decodable codes of subexponential length. *SIAM Journal on Computing*, 41(6):1694–1703, 2012.

12 I. Goldberg. Improving the robustness of private information retrieval. In *2007 IEEE Symposium on Security and Privacy (SP'07)*, pages 131–148, 2007.

13 Vince Grolmusz. Superpolynomial size set-systems with restricted intersections mod 6 and explicit ramsey graphs. *Combinatorica*, 20(1):71–86, 2000.

14 T. Itoh and Y. Suzuki. Improved constructions for query-efficient locally decodable codes of subexponential length. *IEICE Transactions on Information and Systems*, E93.D(2):263–270, 2010.

15 K. Kurosawa. How to correct errors in multi-server pir. In *Advances in Cryptology – ASIACRYPT 2019*, pages 564–574, 2019.

16 UV Linnik. On the least prime in an arithmetic progression. II. The Deuring–Heilbronn phenomenon. *Rec. Math. [Mat. Sbornik] N.S.*, 15(3):347–368, 1944.

17 A. Spitzbart. A generalization of Hermite's interpolation formula. *The American Mathematical Monthly*, 67(1):42–46, 1960.

18 H. Sun and S.A. Jafar. The capacity of private information retrieval. *IEEE Transactions on Information Theory*, 63(7):4075–4088, 2017.

19 H. Sun and S.A. Jafar. The capacity of robust private information retrieval with colluding databases. *IEEE Transactions on Information Theory*, 64(4):2361–2370, 2018.

20 D. Woodruff and S. Yekhanin. A geometric approach to information-theoretic private information retrieval. In *20th Annual IEEE Conference on Computational Complexity (CCC'05)*, pages 275–284, 2005.

21 E.Y. Yang, Jie Xu, and K.H. Bennett. Private information retrieval in the presence of malicious failures. In *Proceedings 26th Annual International Computer Software and Applications*, pages 805–810, 2002.

22 S. Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM (JACM)*, 55(1):1–16, 2008.

A Proof of Theorem 15

We first show a lemma used in the proof of Theorem 15.

▶ **Lemma 22.** *Let $T_1, \ldots, T_m$ be m pairwise disjoint subsets of $[\ell]$ such that $T_1 \cup \cdots \cup T_m = [\ell] \setminus \{1\}$ and $|T_i| \leq t$ for all $i \in [m]$. Consider the following conditions on a family $\mathcal{F} = \{S_1, \ldots, S_N\}$ of subsets of $[\ell]$:*

1. *For any $i \in [\ell]$, $|S_i| = k$ and $1 \in S_i$;*
2. *For any $h \in [m]$, there exists at most one $i \in [N]$ such that $T_h \cap S_i \neq \emptyset$.*

Define M be the maximum size of $\mathcal{F}$ satisfying the above conditions. Then,

$$M \geq M' = \left\lceil \frac{\ell - k + 1}{k + t - 2} \right\rceil .$$

Proof of Lemma 22. It is sufficient to construct a family $\mathcal{F}$ with $|\mathcal{F}| \geq M'$ satisfying the conditions 1 and 2. We consider the following algorithm to generate a sequence $u_1, u_2, \ldots$ in $[m]$:

1. Set $i = 1$ and $u_0 = 0$;
2. Repeat the following:
   a. If $\sum_{u_{i-1}+1 \leq h \leq m} |T_h| \geq k - 1$, let u_i be the smallest element such that $|T_{u_{i-1}+1}| + |T_{u_{i-1}+2}| + \cdots + |T_{u_i}| \geq k - 1$. Otherwise, go to Step 3.
   b. Set $i \leftarrow i + 1$.
3. Output $u_1, \ldots, u_{i-1}$.

Let $u_0 = 0, u_1, \ldots, u_N$ be an output of the above algorithm. For each $i \in [N]$, choose any subset S'_i of size $k - 1$ such that $S'_i \subseteq T_{u_{i-1}+1} \cup \cdots \cup T_{u_i}$. We define $\mathcal{F} = \{S'_i \cup \{1\} : i \in [N]\}$. It easily follows from the definition that $\mathcal{F}$ satisfies the conditions 1 and 2. Since $\sum_{u_{i-1}+1 \leq h \leq u_i - 1} |T_h| \leq k - 2$ and $|T_{u_i}| \leq t$, it holds that $\sum_{u_{i-1}+1 \leq h \leq u_i} |T_h| \leq k + t - 2$ for any $i \in [N]$. By adding them up, we have that

$$\ell - 1 - \sum_{u_N + 1 \leq h \leq m} |T_h| = \sum_{1 \leq h \leq u_N} |T_h| \leq N(k + t - 2)$$

and hence

$$|\mathcal{F}| = N \geq \frac{\ell - k + 1}{k + t - 2}$$

since $\sum_{u_N+1 \leq h \leq m} |T_h| \leq k - 2$. ◀

We consider a PIR scheme $\Pi' = (\mathcal{Q}', \mathcal{A}', \mathcal{R}')$ where $(\mathcal{Q}', \mathcal{A}')$ runs $\binom{\ell}{k}$ independent instances of $(\mathcal{Q}, \mathcal{A})$ between a client and every subset of k servers. The communication complexity of Π' is $c \cdot \binom{\ell}{k}$. Since each execution of $\mathcal{Q}(\tau)$ is done independently, Π' is also t-private. Furthermore for each execution of $(\mathcal{Q}, \mathcal{A})$, $\mathcal{R}'$ runs $\mathcal{R}$ on the corresponding input. If $\mathcal{R}$ outputs the same value a for every execution, then $\mathcal{R}'$ outputs a. Otherwise $\mathcal{R}'$ outputs $\perp$. Then it is clear that Π' is correct. It also follows that it is (k, ℓ)-robust and hence $\lfloor (\ell - k)/2 \rfloor$-error correcting.

We prove that Π' is $(1 - \epsilon')$-fully error detecting. Without loss of generality, suppose that the first server is honest and all the other servers are malicious. Consider pairwise disjoint subsets $T_1, \ldots, T_m$ such that $T_1 \cup \ldots \cup T_m = [\ell] \setminus \{1\}$ and $|T_h| \leq t$ for $h \in [m]$. We assume that all the servers in each T_h can collude.

Let $p = \binom{\ell}{k}$ and let $S_1, \ldots, S_p$ be all k-sized subsets of $[\ell]$. Let $\mathcal{F}$ be a family of subsets of $[\ell]$ with $|\mathcal{F}| = M \geq M'$ given by Lemma 22. By rearranging the order, we may assume that $\mathcal{F} = \{S_1, \ldots, S_M\}$. Let Π_j denote the instance of Π executed by the client and servers in S_j.

During the execution of Π_j, the client generates $\mathcal{Q}(\tau; r_j) \to (\{\mathsf{que}_i^{(j)} : i \in S_j\}; \mathsf{aux}^{(j)})$, where r_j is a random string and $\mathsf{que}_i^{(j)}$ is sent to the i-th server for $i \in S_j$. The i-th server then receives $\mathsf{que}'_i = \{\mathsf{que}_i^{(j)} : j \in [p] \text{ with } i \in S_j\}$. In each Π_j, the i-th server with $i \in S_j$ returns

$$\widetilde{\mathsf{ans}}_i^{(j)} = \begin{cases} \mathsf{ans}_1^{(j)} = \mathcal{A}(1, \mathsf{que}_1^{(j)}, \boldsymbol{a}), & \text{if } i = 1, \\ f_i^{(j)}(\{\mathsf{que}'_{i'}\}_{i' \in T_h}, \boldsymbol{a}), & \text{if } i \in T_h \end{cases}$$

for some function $f_i^{(j)}$, where $\boldsymbol{a} = (a_1, \ldots, a_n)$ is a database. It then follows from our definition of $\mathcal{R}'$ that

$$\begin{aligned} & \Pr[\mathcal{R}' \text{ outputs some } a' \notin \{a_\tau, \perp\}] \\ \leq\ & \Pr\left[\forall j \in [M] : \mathcal{R}(\mathsf{ans}_1^{(j)}, \{\widetilde{\mathsf{ans}}_i^{(j)}\}_{i \in S_j \setminus \{1\}}; \mathsf{aux}^{(j)}) \notin \{a_\tau, \perp\}\right]. \end{aligned}$$

Since $\epsilon^M \leq \epsilon'$, it is enough to show that

$$p_0 := \Pr\left[\forall j \in [M] : \mathcal{R}(\mathsf{ans}_1^{(j)}, \{\widetilde{\mathsf{ans}}_i^{(j)}\}_{i \in S_j \setminus \{1\}}; \mathsf{aux}^{(j)}) \notin \{a_\tau, \perp\}\right] \leq \epsilon^M.$$

Now fix $r' = (r_{M+1}, \ldots, r_p)$ arbitrarily. Then $\mathsf{que}_i^{(j)}$ is a fixed constant for any $j \in \{M+1, \ldots, p\}$ and $i \in S_j$. Let $j \in [M]$, $i \in S_j \setminus \{1\}$ and let $h \in [m]$ be the unique index such that $i \in T_h$. Since $\mathcal{F}$ satisfies the condition 2 in Lemma 22, we have that $T_h \cap S_{j'} = \emptyset$ for any $j' \in [M] \setminus \{j\}$. Therefore, we can write

$$\widetilde{\mathsf{ans}}_i^{(j)} = f_i^{(j)}(\{\mathsf{que}'_{i'}\}_{i' \in T_h}, \boldsymbol{a}) = g_{i,r'}(\{\mathsf{que}_{i'}^{(j)}\}_{i' \in T_h}, \boldsymbol{a})$$

using some function $g_{i,r'}$. In particular, $\{\widetilde{\mathsf{ans}}_i^{(j)}\}_{i \in S_j \setminus \{1\}}$ and $\{\widetilde{\mathsf{ans}}_i^{(j')}\}_{i \in S_{j'} \setminus \{1\}}$ are independent if $j \neq j' \in [M]$. Let $\mathcal{X}$ denote the random variable which represents $r' = (r_{M+1}, \ldots, r_p)$. Then, for any fixed $r' = (r_{M+1}, \ldots, r_p)$, we have that

$$\begin{aligned} & \Pr_{r_1, \ldots, r_M}\left[\forall j \in [M] : \mathcal{R}(\mathsf{ans}_1^{(j)}, \{\widetilde{\mathsf{ans}}_i^{(j)}\}_{i \in S_j \setminus \{1\}}; \mathsf{aux}^{(j)}) \notin \{a_\tau, \perp\} \mid \mathcal{X} = r'\right] \\ \leq\ & \prod_{j \in [M]} \Pr_{r_j}\left[\mathcal{R}(\mathsf{ans}_1^{(j)}, \{\widetilde{\mathsf{ans}}_i^{(j)}\}_{i \in S_j \setminus \{1\}}; \mathsf{aux}^{(j)}) \notin \{a_\tau, \perp\} \mid \mathcal{X} = r'\right] \\ \leq\ & \epsilon^M \end{aligned}$$

since Π is $(1-\epsilon)$-fully error detecting. Therefore, it holds that

$$\begin{aligned} p_0 &= \sum_{r'} \Pr\left[\mathcal{X} = r'\right] \Pr_{r_1, \ldots, r_M}\left[\forall j \in [M] : \mathcal{R}(\mathsf{ans}_1^{(j)}, \{\widetilde{\mathsf{ans}}_i^{(j)}\}_{i \in S_j}; \mathsf{aux}^{(j)}) \notin \{a_\tau, \perp\} \,\middle|\, \mathcal{X} = r'\right] \\ &\leq \sum_{r'} \Pr\left[\mathcal{X} = r'\right] \times \epsilon^M \\ &\leq \epsilon^M. \end{aligned}$$

B Proof of Theorem 16

Using the notations in Figure 1, the 1-privacy of Π_1 follows from the fact that a random vector $\boldsymbol{w}^{(j)}$ masks $\boldsymbol{u}_\tau$ for each $j \in [\lambda]$.

First we show the correctness of Π_1. Fix a database $\boldsymbol{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$ and a client's index τ. Note that $\xi_i^{(j)}$ at Step 2 of $\mathcal{A}$ in Figure 1 is computed as

$$\xi_i^{(j)} = \sum_{\sigma \in [n]} a_\sigma \gamma^{\langle \boldsymbol{w}^{(j)}, \boldsymbol{v}_\sigma \rangle + \alpha_i^{(j)} \langle \boldsymbol{u}_\tau, \boldsymbol{v}_\sigma \rangle} = c_0^{(j)} + \sum_{s \in \{p, q, p+q\}} c_s^{(j)} (\gamma^{\alpha_i^{(j)}})^s$$

for each $j \in [\lambda]$, where $c_0^{(j)} = a_\tau \gamma^{\langle \boldsymbol{w}^{(j)}, \boldsymbol{v}_\tau \rangle}$ and $c_s^{(j)} = \sum_{\sigma \in [n]: \langle \boldsymbol{u}_\tau, \boldsymbol{v}_\sigma \rangle = s} a_\sigma \gamma^{\langle \boldsymbol{w}^{(j)}, \boldsymbol{v}_\sigma \rangle}$. Similarly, $\boldsymbol{\zeta}_i^{(j)}$ at Step 2 of $\mathcal{A}$ is written as

$$\boldsymbol{\zeta}_i^{(j)} = \sum_{\sigma \in [n]} a_\sigma \phi(\boldsymbol{v}_\sigma) \gamma^{\langle \boldsymbol{w}^{(j)}, \boldsymbol{v}_\sigma \rangle + \alpha_i^{(j)} \langle \boldsymbol{u}_\tau, \boldsymbol{v}_\sigma \rangle}.$$

Hence, $\eta_i^{(j)}$ at Step 2(a) of $\mathcal{R}$ is computed as follows:

$$\begin{aligned}
\eta_i^{(j)} &:= \langle \phi(\boldsymbol{u}_\tau), \boldsymbol{\zeta}_i^{(j)} \rangle \\
&= \sum_{\sigma \in [n]} a_\sigma \phi(\langle \boldsymbol{u}_\tau, \boldsymbol{v}_\sigma \rangle) \gamma^{\langle \boldsymbol{w}^{(j)}, \boldsymbol{v}_\sigma \rangle} (\gamma^{\alpha_i^{(j)}})^{\langle \boldsymbol{u}_\tau, \boldsymbol{v}_\sigma \rangle} \\
&= a_\tau \phi(0) \gamma^{\langle \boldsymbol{w}^{(j)}, \boldsymbol{v}_\tau \rangle} + \sum_{s \in \{p, q, p+q\}} c_s^{(j)} \phi(s) (\gamma^{\alpha_i^{(j)}})^s \\
&= \sum_{s \in \{p, q, p+q\}} c_s^{(j)} \phi(s) (\gamma^{\alpha_i^{(j)}})^s.
\end{aligned}$$

On the other hand, the matrix $M^{(j)}$ computed by $\mathcal{R}$ is written as

$$M^{(j)} = \begin{pmatrix}
1 & (\beta_1^{(j)})^p & (\beta_1^{(j)})^q & (\beta_1^{(j)})^{p+q} \\
0 & \phi(p)(\beta_1^{(j)})^p & \phi(q)(\beta_1^{(j)})^q & \phi(p+q)(\beta_1^{(j)})^{p+q} \\
1 & (\beta_2^{(j)})^p & (\beta_2^{(j)})^q & (\beta_2^{(j)})^{p+q} \\
0 & \phi(p)(\beta_2^{(j)})^p & \phi(q)(\beta_2^{(j)})^q & \phi(p+q)(\beta_2^{(j)})^{p+q}
\end{pmatrix}$$

since $(\beta_i^{(j)})^p = (\gamma^p)^{\alpha_i^{(j)}} = 1$, $(\beta_i^{(j)})^q = \beta_i^{(j)}$ in $\mathbb{F}_q$, and $\phi(p) = p, \phi(q) = 0, \phi(p+q) = p$ due to the definition of ϕ. We therefore have that

$$M^{(j)} \begin{pmatrix} c_0^{(j)} \\ c_p^{(j)} \\ c_q^{(j)} \\ c_{p+q}^{(j)} \end{pmatrix} = \begin{pmatrix} \xi_1^{(j)} \\ \eta_1^{(j)} \\ \xi_2^{(j)} \\ \eta_2^{(j)} \end{pmatrix}. \tag{4}$$

Furthermore, it holds that $\det M^{(j)} = p^2 (\beta_1^{(j)} - \beta_2^{(j)})^2 \neq 0$ since $\alpha_1^{(j)} \neq \alpha_2^{(j)}$. Therefore $\mathcal{R}$ correctly recovers $c_0^{(j)}$, $c_p^{(j)}$, $c_q^{(j)}$, $c_{p+q}^{(j)}$ for each $j \in [\lambda]$. Hence $\mathcal{L} = \{a_\tau\}$ and $\mathcal{R}$ outputs a_τ. Thus the correctness of Π_1 holds.

We next show that Π_1 is $(1-\epsilon)$-fully error detecting. The reconstruction algorithm $\mathcal{R}$ computes $\widetilde{c}_0, \widetilde{c}_p, \widetilde{c}_q, \widetilde{c}_{p+q}$ such that

$$M^{(j)} \begin{pmatrix} \widetilde{c}_0^{(j)} \\ \widetilde{c}_p^{(j)} \\ \widetilde{c}_q^{(j)} \\ \widetilde{c}_{p+q}^{(j)} \end{pmatrix} = \begin{pmatrix} \widetilde{\xi}_1^{(j)} \\ \widetilde{\eta}_1^{(j)} \\ \widetilde{\xi}_2^{(j)} \\ \widetilde{\eta}_2^{(j)} \end{pmatrix} \tag{5}$$

for all $j \in [\lambda]$. Without loss of generality, we may assume that $(\widetilde{\xi}_1^{(j)}, \widetilde{\boldsymbol{\eta}}_1^{(j)}) = (\xi_1^{(j)}, \boldsymbol{\eta}_1^{(j)})$ and $(\widetilde{\xi}_2^{(j)}, \widetilde{\boldsymbol{\eta}}_2^{(j)}) \neq (\xi_2^{(j)}, \boldsymbol{\eta}_2^{(j)})$. Namely the first server is honest and the second server is malicious. From Eqs. (4) and (5), it holds that

$$M^{(j)} \begin{pmatrix} \widetilde{c}_0^{(j)} - c_0^{(j)} \\ \widetilde{c}_p^{(j)} - c_p^{(j)} \\ \widetilde{c}_q^{(j)} - c_q^{(j)} \\ \widetilde{c}_{p+q}^{(j)} - c_{p+q}^{(j)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \mu_2^{(j)} \\ \nu_2^{(j)} \end{pmatrix} \quad (\forall j \in [\lambda])$$

where $\mu_2^{(j)} = \widetilde{\xi}_2^{(j)} - \xi_2^{(j)}$ and $\nu_2^{(j)} = \widetilde{\eta}_2^{(j)} - \eta_2^{(j)}$. By multiplying the adjugate matrix $\mathrm{adj}(M^{(j)})$ from the left, we obtain the first entry as

$$(\det M^{(j)})(\widetilde{c}_0^{(j)} - c_0^{(j)}) = M_{13}^{(j)}\mu_2^{(j)} + M_{14}^{(j)}\nu_2^{(j)} \ (\forall j \in [\lambda]),$$

where $M_{k\ell}^{(j)}$ is the (k,ℓ)-entry of $\mathrm{adj}(M^{(j)})$. Let $\widetilde{c}_0^{(j)} = a'\gamma^{\langle \boldsymbol{w}^{(j)}, \boldsymbol{v}_\tau \rangle}$, and define $\Delta := a' - a_\tau$ and $\delta^{(j)} := \gamma^{\langle \boldsymbol{w}^{(j)}, \boldsymbol{v}_\tau \rangle}$. By calculating the adjacent matrix $\mathrm{adj}(M^{(j)})$ directly, we have

$$p^2(\beta_1^{(j)} - \beta_2^{(j)})^2 \Delta\delta^{(j)} = (p\mu_2^{(j)} - \nu_2^{(j)})p(\beta_1^{(j)} - \beta_2^{(j)})\beta_1^{(j)} \tag{6}$$

for any $j \in [\lambda]$. Let $x_1 = \beta_1^{(j)} = \gamma^{\alpha_1^{(j)}}$. Then we have

$$p^2(x_1 - \beta_2^{(j)})^2 \Delta\delta^{(j)} - (p\mu_2^{(j)} - \nu_2^{(j)})p(x_1 - \beta_2^{(j)})x_1 = 0$$

and hence

$$p(x_1 - \beta_2^{(j)}) \left((p\Delta\delta^{(j)} - p\mu_2^{(j)} + \nu_2^{(j)})x_1 - p\Delta\delta^{(j)}\beta_2^{(j)} \right) = 0.$$

Since $x_1 = \beta_1^{(j)} \neq \beta_2^{(j)}$, it must hold that

$$(p\Delta\delta^{(j)} - p\mu_2^{(j)} + \nu_2^{(j)})x_1 - p\Delta\delta^{(j)}\beta_2^{(j)} = 0. \tag{7}$$

Now suppose that $\Delta = a' - a_\tau \neq 0$. Then it must hold that $p\Delta\delta^{(j)} - p\mu_2^{(j)} + \nu_2^{(j)} \neq 0$ since $p\Delta\delta^{(j)}\beta_2^{(j)} \neq 0$. Furthermore $x_1 \neq \beta_2^{(j)}$ is randomly chosen independently from the other values in Eq. (7). Therefore Eq. (7) holds for all $j \in [\lambda]$ with probability at most $(p-1)^{-\lambda}$. This means that $\mathcal{R}$ adds $a' \neq a_\tau$ to $\mathcal{L}$ at Step 2(d) with probability at most $\epsilon = (p-1)^{-\lambda}$ since Δ takes at most one value. Therefore our PIR scheme is $(1-\epsilon)$-fully error detecting.

Finally, a prime q satisfying $q \equiv 1 \bmod p$ can be chosen as $q = p^{O(1)}$ from Linnik's theorem [16]. It then holds that $\lambda \log q = O(\lambda \log p) = O(\log \epsilon^{-1})$ and the communication complexity is given by $O(h\lambda \log q) = \mathcal{L}_n[1/2, \theta_m] \cdot \log \epsilon^{-1}$ from Proposition 7, where θ_m is a constant depending on $m = pq$ only. Since $m = pq$ can be chosen as a constant, we conclude that the communication complexity is $\mathcal{L}_n[1/2, O(1)] \cdot \log \epsilon^{-1}$.

Property-Preserving Hash Functions and Combinatorial Group Testing

Kazuhiko Minematsu
NEC, Tokyo, Japan
Yokohama National University, Japan

Abstract

Property-preserving hash (PPH) function is a class of hash functions that allows an evaluation of the property of inputs from their hash values. Boyle et al. at ITCS 2019 recently introduced it and considered the robustness of PPH against an adversary who accesses the internal randomness of PPH, and proposed two robust PPH constructions for a weak form of Hamming distance predicate. The second construction received attention for its short hash value, although it relies on an ad-hoc security assumption. The first construction, which is entirely hash-based and based on the classical collision-resistance assumption, has been largely overlooked. We study their first construction and discover its close connection to a seemingly different field of hash/MAC-based (adversarial) error detection using the theory of Combinatorial Group Testing (CGT). We show some consequences of this discovery. In particular, we show that some existing proposals in the field of CGT-based error detection can be converted into a PPH for the Hamming distance property, and they immediately improve and generalize Boyle et al.'s hash-based PPH proposal. We also show that the idea of Boyle et al. is useful in the context of a variant of CGT problem.

2012 ACM Subject Classification Security and privacy → Hash functions and message authentication codes

Keywords and phrases Hash function, Property-Preserving Hash, Combinatorial Group Testing, Provable Security

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.2

Related Version *Previous Version*: `https://eprint.iacr.org/2022/478`

1 Introduction

Compressing a large amount of data into small digests while maintaining some of their properties is one of the fundamental goals in computer science. Popular algorithms such as Bloom filter [2] or Cukoo hashing [31] offer such property-preserving hashing for approximate set membership property. Locality-sensitive hash (LSH) functions [22] allow for compressing two inputs independently and evaluating if they are close or not with respect to some metric from their digests. These algorithms are randomized and usually studied in the setting where inputs are independent of the internal random coin. In real-world use cases, this may not be enough, because we often need to consider an adversary who has an incentive to corrupt the algorithm by giving maliciously crafted inputs. Such an adversary would somehow try to learn the internal random coin, however, basic property-preserving hashing algorithms have no guarantee against such attacks.

Based on this motivation, Boyle et al. [4] (BLV19) initiated the study of *robust* property-preserving hash (PPH) functions that resist such attacks. They proposed two constructions for the Gap Hamming predicate, which is a weak form of Hamming distance predicate. The first construction was entirely hash-based and relied on the classical collision resistance of the hash function. The second was based on an ad-hoc assumption related to the hardness of decoding linear codes. The second construction has a much shorter hash size than the

first and stipulated research on pairing or lattice-based PPHs for (exact) Hamming distance predicate with similar hash size, as shown by Fleischhacker and Simkin [18] and Fleischhacker et al. [17]. In contrast, the first construction has been largely overlooked since the proposal, possibly because of its large hash size. However, its simplicity and computational efficiency, and high-security reliance may make it attractive in practical applications.

1.1 Our Contributions

The core idea of BLV19's first construction (hereafter PPH1) is subsampling. That is, taking hash values for a predetermined set of input subsequences and concatenating them. We discover that this idea of PPH1 is closely related to a different application field of hash functions or message authentication codes (MACs), namely hash/MAC function with detection capability [19, 14, 25]. Such a hash/MAC function can be interpreted as an application of classical Combinatorial Group Testing (CGT) to the detection of data corruptions, where detection means to pinpoint the parts of the input data that have been corrupted. For convention, we say *CGT-based hash function* to mean this application. Our finding derives several interesting consequences.

In more detail, we show that known schemes in the aforementioned field of CGT-based hash can be interpreted as a robust PPH (in the sense of BLV19) for the Hamming distance predicate, with a much better compression rate, i. e. smaller hash size, than PPH1. Note that PPH1 only preserves a weaker Gap Hamming distance predicate. CGT-based hash functions also have smaller computation time for hashing and preserve more informative properties than plain Hamming distance predicate. The security (robustness) against the adversary is also proved with minor modifications to the original proofs for the security notions of CGT-based hash. Moreover, a MAC-based PPH is also naturally derived from the known CGT-based MACs [19, 25, 26]. As well as the hash-based counterpart, a MAC-based PPH preserves the Hamming distance property and fulfills a weak form of robustness defined by BLV19. Moreover, by using the technique of [25], the resulting scheme can significantly reduce the hashing time thanks to its special structure for the internal MAC function. Finally, we show that the construction of PPH1, which uses a bipartite expander graph, is also useful in building a test matrix for CGT that aims at estimating the number of defectives rather than detecting them [7]. Therefore, the connection is not one-way.

We came up with this finding while studying the existing CGT-based hash/MAC schemes. Our study did not present any new scheme. Once found, the connection may look rather obvious, while we are not aware in the literature and present a formal security analysis. We think bridging two seemingly different areas is important and will help understanding and development of both areas.

2 Preliminaries

2.1 Notations

Let $[i]$ denote $\{1, \ldots, i\}$ for $i \geq 1$. The set of all binary strings is denoted by $\{0,1\}^*$, which includes the empty string ε. We write the bit length of $X \in \{0,1\}^*$ by $|X|$, where $|\varepsilon| = 0$. The Hamming weight of a binary string X is denoted by $\mathsf{Hw}(X)$. For $X = (X_1, \ldots, X_n) \in \mathcal{X}^n$ for some finite set $\mathcal{X}$ and $V = (V_1, \ldots, V_n) \in \{0,1\}^n$, let $X_{|V}$ be the sub-sequence of X indexed by V. That is, $X_{|V}$ is $(X_{i_1}, \ldots, X_{i_v})$, where $\mathsf{Hw}(V) = v$ and $i_j \in [n]$ is the index of the j-th bit set at V. For $X, X' \in \mathcal{X}^n$, let $\mathcal{D}(X, X') \coloneqq (\mathsf{test}(X_1, X'_1), \ldots, \mathsf{test}(X_n, X'_n))$, where test is a test function (i. e., $\mathsf{test}(A, B) = 1$ if $A = B$ and 0 otherwise). For $X, Y \in \{0,1\}^n$, $X \vee Y$ denotes the bitwise Boolean sum (logical OR) of X and Y.

For a keyed function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ for key space $\mathcal{K}$, we may write $F_K(X)$ instead of $F(K, X)$. Let $\mathrm{negl}(\lambda)$ for a security parameter $\lambda \in \mathbb{N}$ be a negligible function. For a finite set $\mathcal{X}$ we write $X \xleftarrow{\$} \mathcal{X}$ to mean X is uniformly sampled over $\mathcal{X}$. For a probabilistic polynomial-time (PPT) adversary A and the oracles $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_c$, $\mathsf{A}^{\mathcal{O}_1,\ldots,\mathcal{O}_c} \to Y$ denotes the event that A possibly adaptively queries to $\mathcal{O}_i$ in an arbitrarily order and outputs Y. The set of all functions of domain $\mathcal{X}$ and range $\mathcal{Y}$ is written as $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ and a uniformly random function (URF) $\mathsf{R} : \mathcal{X} \to \mathcal{Y}$ is a random function that uniformly distributes over $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$.

2.2 Advantage Functions

Let $F, G : \mathcal{X} \to \mathcal{Y}$ be two (possibly randomized) functions. Let A be an adversary who tries to distinguish F from G using oracle access, and outputs a binary decision. We define

$$\mathbf{Adv}^{\mathtt{IND}}_{F,G}(\mathsf{A}) \coloneqq \Pr[\mathsf{A}^F \to 1] - \Pr[\mathsf{A}^G \to 1]$$

as the advantage of A in distinguishing F and G. Similarly, let $\mathbf{Adv}^{\mathtt{Event}}_F(\mathsf{A})$ denote the probability of an event Event invoked by A in the game.

2.3 Matrix Representation

Let $\mathbb{M}$ be an $n \times m$ binary matrix. We write $\mathbb{M}_{i,*}$ to denote the i-th row, and $\mathbb{M}_{*,j}$ to denote the j-th column, and $\mathbb{M}_{i,j}$ to denote the entry at i-th row and j-th column. For simplicity we may abbreviate $\mathbb{M}_{i,*}$ to $\mathbb{M}_i$. The rows and columns of $\mathbb{M}$ are interchangeably seen as sets, e.g. $\mathbb{M}_i = \{j \in [m] : \mathbb{M}_{i,j} = 1\}$, and $a \in \mathbb{M}_i$ means $\mathbb{M}_{i,a} = 1$. This may also apply to any binary string.

2.4 Property-Preserving Hash Function

Following BLV19, we describe the basics of property-preserving hash functions. For a finite set $\mathcal{Z}$, a $\mathcal{Z}$-valued property for a pair of messages in message space $\mathcal{X}$ is a function $P : \mathcal{X} \times \mathcal{X} \to \mathcal{Z}$. When $|\mathcal{Z}| = 2$ it is called a predicate. A property is called *promise* if it is undefined for some inputs and otherwise called *total*. We only consider the latter case. This paper focuses on two-input properties, but generalization is possible.

▶ **Definition 1.** *Let $\mathcal{X} = \mathcal{B}^n$ for some set $\mathcal{B}$ and a positive integer n. let $\mathsf{Hd}(X, X')$ for $X, X' \in \mathcal{X}$ be the generalized Hamming distance defined as*

$$\mathsf{Hd}(X, X') \coloneqq \mathsf{Hw}(\mathcal{D}(X, X')).$$

As we may see a binary string as a set, we also have $\mathsf{Hd}(X, X') = |\mathcal{D}(X_i, X'_i)|$. The Hamming predicate with threshold $d \in [n]$ is defined as

$$\mathsf{HAM}^d(X, X') \coloneqq \begin{cases} 1 & \textit{if } \mathsf{Hd}(X, X') \geq d \\ 0 & \textit{otherwise} \end{cases}$$

Moreover, the (generalized) Gap-Hamming predicate with parameter (d, ϵ) for $\epsilon \in (0, 1)$ is defined as

$$\mathsf{GapHAM}^{d,\epsilon}(X, X') \coloneqq \begin{cases} 1 & \textit{if } \mathsf{Hd}(X, X') \geq d \cdot (1 + \epsilon) \\ 0 & \textit{if } \mathsf{Hd}(X, X') \leq d \cdot (1 - \epsilon) \\ \textit{undefined} & \textit{otherwise} \end{cases}$$

As shown above, Gap-Hamming is a premise predicate.

For simplicity, we assume $\mathcal{B} = \{0, 1\}^b$ for some $b \geq 1$. When $b = 1$, $\mathsf{Hd}(*, *)$ corresponds to the classical Hamming distance between the bit strings.

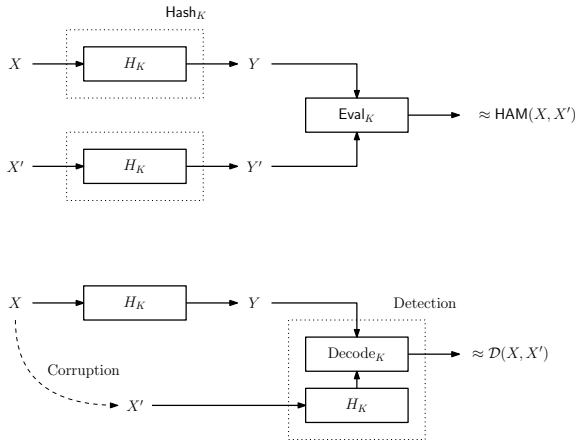

Figure 1 PPH (top) and CGT-based MAC/Hash scheme defined at Section 4 (bottom).

▶ **Definition 2** (Property-preserving Hash Function). *Let $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a keyed function. If H is a property-preserving hash (PPH) function for a property $P : \mathcal{X} \times \mathcal{X} \to \mathcal{Z}$, it requires the following algorithms:*

- *Sampling (key-generation) function* $\mathsf{Sample}(\lambda)$ *for the security parameter λ that efficiently samples K over $\mathcal{K}$ (we hereafter assume λ as a fixed constant and simply write $K \xleftarrow{\$} \mathcal{K}$ to represent this)*
- *Hash output evaluation function* $\mathsf{Hash} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *that returns $H(K, X)$ on input (K, X)*
- *Property evaluation function* $\mathsf{Eval} : \mathcal{K} \times \mathcal{Y} \times \mathcal{Y} \to \mathcal{Z}$

We say H is η-compressing if $\log |\mathcal{Y}| \leq \eta \log |\mathcal{X}|$ for $0 < \eta < 1$.

The role of Eval is to give an estimate of $P(X_1, X_2)$ for some $X_1, X_2 \in \mathcal{X}$, using the key K and their corresponding hash values.

2.5 Robustness Notions for PPHs

Let $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a PPH for a property $P : \mathcal{X} \times \mathcal{X} \to \mathcal{Z}$ and let $H.\mathsf{hash}$ and $H.\mathsf{Eval}$ be the corresponding Hash and Eval functions. The goal of PPH has some similarities to the existing objects, such as LSH [22], which is a randomized hashing algorithm that preserves the distance between the inputs for some metric. However, the security against the adversary who may query the hashing and evaluation processes or even access the internal randomness has not been formally studied. Based on this observation, BLV19 introduced multiple robustness notions of PPH to capture such security, using the security parameter λ. Each robustness notion has a different adversary model, as shown below.

▶ **Definition 3** (Non-Robust PPH [4]). *If H is a non-robust PPH for a property P, for any PPT adversary* A*,*

$$\begin{aligned}&\mathbf{Adv}^{\mathtt{NR\text{-}PPH}}_{H}(\mathsf{A})\\ &:= \Pr[K \xleftarrow{\$} \mathcal{K} : \mathsf{A} \to (X, X'), \mathsf{Eval}(K, H_K(X), H_K(X')) \neq P(X, X')] \leq \mathrm{negl}(\lambda).\end{aligned}$$

▶ **Definition 4** (Evaluation-Oracle-Robust PPH [4]). *If H is an Evaluation-Oracle (EO) robust PPH for a property P, for any PPT adversary* A*,*

$$\begin{aligned}&\mathbf{Adv}^{\mathtt{EO\text{-}PPH}}_{H}(\mathsf{A})\\ &:= \Pr[K \xleftarrow{\$} \mathcal{K}; \mathsf{A}^{\mathcal{O}_{H.\mathsf{Eval}}} \to (X, X') : \mathsf{Eval}(K, H_K(X), H_K(X')) \neq P(X, X')]] \leq \mathrm{negl}(\lambda).\end{aligned}$$

▶ **Definition 5** (Double-Oracle-Robust PPH [4]). *If H is an Double-Oracle (DO) robust PPH for a property P, for any PPT adversary* A*,*

$$\begin{aligned}&\mathbf{Adv}^{\mathtt{DO\text{-}PPH}}_{H}(\mathsf{A})\\&:= \Pr[K \xleftarrow{\$} \mathcal{K}; \mathsf{A}^{\mathcal{O}_{H.\mathsf{Hash}}, \mathcal{O}_{H.\mathsf{Eval}}} \to (X, X') : \mathsf{Eval}(K, H_K(X), H_K(X')) \neq P(X, X')]]\\&\leq \mathrm{negl}(\lambda).\end{aligned}$$

▶ **Definition 6** (Direct-Access Robust PPH [4]). *If H is a Direct-Access (DA) PPH for a property P, for any PPT adversary* A*,*

$$\begin{aligned}&\mathbf{Adv}^{\mathtt{DA\text{-}PPH}}_{H}(\mathsf{A})\\&:= \Pr[K \xleftarrow{\$} \mathcal{K}; (X, X') \leftarrow \mathsf{A}(K) : \mathsf{Eval}(K, H_K(X), H_K(X')) \neq P(X, X')]] \leq \mathrm{negl}(\lambda).\end{aligned}$$

A DA-robust PPH may be simply called a robust PPH.

In DA-PPH the adversary is given K, hence can also simulate $\mathcal{O}_{\mathsf{Hash}}$ and $\mathcal{O}_{\mathsf{Eval}}$.

For example, classical universal hash function [35] can be interpreted as a non-robust PPH for the collision property (i. e. $P(X, X') = 1$ iff $X = X'$). BLV19 provides in-depth discussions on these notions and relations to the existing ideas such as one-way communication protocols. They are not relevant to our work, so refer to BLV19 for more details.

2.6 Constructions of PPHs

As described in Introduction, BLV19 showed two constructions of PPH for Gap-Hamming predicate (Definition 1), which we call PPH1 and PPH2. PPH1 is entirely based on a collision-resistant hash function. The construction takes hash values for multiple subsets of the input, where the subsets are specified by a class of bipartite expander graph, and the output is the concatenation of these hash values (see Sections 4.2 and 5 for more details). While intuitive, its large hash size is problematic. PPH2 supports a smaller gap and better efficiency than PPH1 (although the compression rate is constant for both schemes) and is based on a new assumption related to the hardness of syndrome decoding of LDPC codes. BLV19 stipulated research on PPHs. Fleischhacker and Simkin [18] showed a PPH for the (exact, rather than Gap) Hamming distance predicate with a hash length for threshold t, which is much better than PPH1. The security is based on a new bilinear discrete-logarithm assumption in pairing friendly curves. Fleischhacker et al. [17] proposed a PPH for the Hamming distance predicate whose security is proved under a standard lattice hardness assumption while having a larger hash size than [18]. These studies have been done with PPH2 in mind.

Compared to PPH2, PPH1 has not been studied since the proposal, possibly for its larger hash size. However, hash-based constructions may be worth studying for their simplicity, computational efficiency, and classical security. The reduction to classical symmetric-key cryptographic assumption is also meaningful in the context of post-quantum cryptography. These observations made us turn our attention to PPH1. As a result, we discover a connection between the idea of PPH1 and the classical Combinatorial Group Testing.

3 Combinatorial Group Testing

We provide some basic ideas about Combinatorial Group Testing (CGT). It is a method to detect defectives among a set of items by using a set of *group tests*. A group test specifies a subset $\mathcal{S}$ of the whole samples $\mathcal{M}$ and returns a binary output indicating if $\mathcal{S}$ contains a

defective. Originally Dorfman invented CGT in WWII [15] to effectively find blood samples infected by syphilis. We write $\mathcal{M} = [n]$ and $\mathcal{I} \subseteq \mathcal{M}$ to denote the (indices of) whole items and the defectives, and in the classical setting we know $|\mathcal{I}| \leq d$ for some $d \in [n]$. In the non-adaptive setting, which is relevant for our case, the set of k group tests is determined by a $k \times n$ binary matrix $\mathbb{M}$, where $\mathbb{M}_{i,j} = 1$ means j-th item is included in the i-th test.

The fundamental goal of CGT is to detect all the defectives, using as few tests as possible. In the case of non-adaptive CGT, we detect all the defectives when the test matrix satisfies a property called d-disjunct [15].

▶ **Definition 7.** *A $k \times n$ binary matrix $\mathbb{M}$ is d-disjunct if, for any $\mathcal{S} \subseteq [n]$ and $|\mathcal{S}| \in [d]$, $\mathbb{M}_{*,j} \not\subseteq \bigvee_{h \in \mathcal{S}} \mathbb{M}_{*,h}$ holds for any $j \notin \mathcal{S}$. That is, a sum of any distinct $i \leq d$ columns of $\mathbb{M}$ does not cover any other column.*

An $n \times n$ identity matrix is n-disjunct. When the test matrix is d-disjunct and $|\mathcal{I}| \leq d$, it is known that the so-called *naive decoder* can detect all the defectives. Algorithm 1 shows the naive decoder for n items, k tags using $k \times n$ test matrix $\mathbb{M}$, taking the test result $R = (R_1, \ldots, R_k) \in \{0,1\}^k$, where $R_i = 1$ denotes that i-th test is positive (indicating the test contains defectives).

Algorithm 1 Naive Decoder.

```
1: procedure DECODE(𝕄, R)          ▷ k × n binary matrix 𝕄, test result R ∈ {0,1}^k
2:     𝒫 ← [n]
3:     for i = 1, ..., k do
4:         if R_i = 1 then
5:             𝒫 ← 𝒫 \ 𝕄_i
6:     return 𝒫
```

Naturally, we want to minimize the number of rows (k) of a d-disjunct matrices given d and n. Disjunct matrix has extensively been studied from the viewpoint of combinatorics or designs or codes; refer to Du and Hwang [15] for classical constructions and the bounds. Most importantly it is known that $k = O(d^2 \log n)$. Porat and Rothchild (PR11) [32], and Cheraghchi and Nakos (CN20) [9] presented polynomial-time constructions that achieve this bound, i. e., these constructions are order-optimal.

Since the inception, the most typical application of CGT is biology, such as DNA screening [28]. One can also find a recent surge of research on applying CGT to COVID-19 PCR testing (e. g. [27] and there are lots of many others). Moreover, CGT has many other applications in the field of computer science, such as image compression [21], streaming computation [10], digital forensics [19] and (aggregate) MAC with detection capability [25, 26, 20, 30].

4 Constructing PPHs from CGT

4.1 CGT-Hash as Direct-Access Robust PPH

4.1.1 Defining CGT-Hash

We describe a class of CGT-based hash function (recall that this is to detect data corruptions rather than telling the binary verification result). For $\mathcal{X} = \mathcal{B}^n$, $\mathcal{B} = \{0,1\}^b$ for some fixed b, let $G : \mathcal{K} \times [k] \times \mathcal{X} \to \mathcal{T}$ for $\mathcal{T} = \{0,1\}^t$ be the atomic (keyed) hash function, and let $\mathbb{M}$ be a $k \times n$ binary matrix. Taking G and $\mathbb{M}$ as parameters, $\mathsf{CGT\text{-}Hash}[G, \mathbb{M}] : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ for $\mathcal{Y} = \mathcal{T}^k$ is defined as

$$\mathsf{CGT\text{-}Hash}[G, \mathbb{M}](K, X) = (Y_1, \ldots, Y_k), \quad Y_i = G(K, i, X_{|\mathbb{M}_i})$$

Note that $\mathbb{M}$ is a fixed, public matrix. This could be used to detect corruptions on $X \in \mathcal{X}$: first we take $Y = \mathsf{CGT\text{-}Hash}[G, \mathbb{M}](K, X)$ with key K and store Y to a tamper-free storage. Later, we take $Y' = \mathsf{CGT\text{-}Hash}[G, \mathbb{M}](K, X')$ for X' which is a possibly corrupted version of X by the adversary, and try to detect the locations of corruptions (namely, $\mathcal{D}(X, X')$) via Y and Y'. CGT-Hash is an application of CGT described in Section 3. From the definition of d-disjunct matrix, if X consists of n items and the adversary can corrupt at most d items, by using d-disjunct matrix $\mathbb{M}$ and the naive decoder (Alg. 1), we can detect $\mathcal{D}(X, X')$ without an error, if G is secure. CGT-Hash will bring more useful information than just taking one hash for the entire X. Moreover, it significantly reduces the size of tamper-free storage than taking hash values for each item. In other words, it allows a trade-off between the size of hash values and the resolution of detection.

Effectively the same idea has been seen in *Corruption-localizing hashing* by Crescenzo et al. [14] and follow-up works [11, 3], where different attack models and different decoders are used. From a practical viewpoint, a basic form of hash-based corruption detection (say by taking hash values for all the files in a server) has been extensively used by major commercial integrity protection/management products, such as TripWire[1] or Splunk[2]. In principle, the use of CGT-Hash will reduce the size of tamper-free storage in these applications.

4.1.2 Connection to PPH

We can use CGT-Hash as a PPH for Hamming distance property. By writing $H(K, *)$ to denote $\mathsf{CGT\text{-}Hash}[G, \mathbb{M}](K, *)$, Hash procedure just performs $H_K(X)$ for input X, and the Eval procedure taking (K, Y, Y') s.t. $Y = H(K, X)$ and $Y' = H(K, X')$ for some $X \neq X' \in \mathcal{X}$ first obtains $\mathcal{D}(Y, Y')$ and performs the naive decoder. If decoder output is larger than d, Eval returns 1 and 0 otherwise. Formally, the following theorem shows that CGT-Hash is a DA-robust PPH for the Hamming distance predicate if G (given the random key K) is collision-resistant.

▶ **Theorem 8.** *In the aforementioned problem setting, if $\mathbb{M}$ is d-disjunct and G is collision-resistant, $\mathsf{CGT\text{-}Hash}[G, \mathbb{M}]$ is a DA-robust PPH for the (generalized) Hamming predicate with threshold d (HAM^d) over $\mathcal{B}^n$.*

Proof. In the game of DA-Robust PPH, suppose the adversary A uses q Hash queries, with total σ blocks and τ time. Note that Eval oracle does not need the key, hence this can be accessed for free. The i-th Hash query and the corresponding tag vector are denoted by $X^{(i)}$ and $T^{(i)} = (T_1^{(i)}, \ldots, T_k^{(i)})$. Let Cons (for consistency) be the event that $[\forall i, j \in [q], i \neq j, \forall h \in [k] : X^{(i)}_{|\mathbb{M}_h} \neq X^{(j)}_{|\mathbb{M}_h} \Leftrightarrow T_h^{(i)} \neq T_h^{(j)}]$.

It is clear that to invoke

$$T_h^{(i)} = T_h^{(j)}$$

for some distinct $i, j \in [q]$ and $h \in [k]$ s.t. $X^{(i)}_{|\mathbb{M}_h} \neq X^{(j)}_{|\mathbb{M}_h}$, A needs to invoke a collision on G. If $T_h^{(i)} \neq T_h^{(j)}$ we must have $X^{(i)}_{|\mathbb{M}_h} \neq X^{(j)}_{|\mathbb{M}_h}$ for any G. Thus, invalidation of Cons is equivalent to finding a non-trivial collision on G.

We also observe that, as long as Cons holds, the problem exactly matches with non-adaptive CGT with *error-free* tests. Moreover, whenever the difference is larger than d, Eval will always detect this fact because the decoder output is always larger than d from the

[1] https://www.tripwire.com/
[2] https://www.splunk.com/

property of the naive decoder (i. e., it never evicts non-corrupted items). A group testing scheme with this property is called a strict group testing [13], and any non-adaptive CGT scheme with a naive decoder and a disjunct test matrix is a strict group testing. This means that HAM^d property is always preserved. Therefore,

$$\mathbf{Adv}^{\mathtt{DA\text{-}PPH}}_{\mathsf{CGT\text{-}Hash}[G,\mathbb{M}]}(\mathsf{A}) \le \mathbf{Adv}^{\mathtt{Coll}}_{G}(\mathsf{B})$$

holds for some collision adversary B against G using qt queries with total σ blocks, $\tau' = \tau + O(\sigma)$ time. ◀

By using order-optimal d-disjunct matrices derived by PR11 or CN20, the compression rate of $\mathsf{CGT\text{-}Hash}[G, \mathbb{M}]$ is kt/bn for $k = O(d^2 \log n)$. The evaluation Eval runs in time $O(nk)$ relying upon the naive decoder. We note that, the evaluation complexity can be greatly improved if *efficient* decoder is available for the test matrix. For example, the disjunct matrix construction by Indyk et al. [23] achieves $k = O(d^2 \log n)$ tests with $O(d^4 \log n)$ decoding time, and that by Ngo et al. [29] achieves $k = O(d^2 \log n + d \log n \cdot \log \log_d n)$ tests with $O(k \log^2 n)$ decoding time. The construction by CN20 also has an efficient decoder. More precisely, it presented a construction achieving $k = O(d^2 \min\{\log n, (\log_d n)^2\})$ tests with $O(k + d \log^2(n/d))$ decoding time.

4.2 Beyond Predicate

Theorem 8 is easily extended to a more informative property defined below:

$$\mathsf{DETECT}^d(X, X') \coloneqq \begin{cases} \mathcal{D}(X, X') & \text{if } \mathsf{Hd}(X, X') \le d \\ 0 & \text{otherwise.} \end{cases}$$

The DETECT^d property tells exact indices of different components when the difference is smaller than the threshold, and otherwise tells that the difference is indeed larger than d. Such a property will facilitate further investigation and is useful for some applications, e. g., biometrics or digital forensics.

4.2.1 Comparison with PPH1

BLV19's first construction (PPH1) is a PPH for the Gap-Hamming property with parameter (d, ϵ), taking $b = 1$. It utilizes a bipartite expander graph with a certain regularity. Using a hash G of t-bit output, its output is $k \cdot t$ bits, where $k = O(n/\log n)$ [4, Lemma 47], $d = O(n/t)$. Direct comparison is not possible, however, $\mathsf{CGT\text{-}Hash}$ (for the same input, using the same G) implements a robust PPH for the exact Hamming distance predicate with $k \cdot t$-bit output, where $k = O(d^2 \log n)$. Thus, the size reduces by a factor of about $n/(d^2 \log^2 n)$, which is non-negligible when we focus on the case $d \ll n$.

We remark that PPH for relatively small d is still usable: it allows to check if two inputs are close even the adversary has full knowledge of the hashing scheme, which is a typical and intuitive application of PPH to (e. g.) biometrics.

4.3 CGT-MAC as Double-Oracle Robust PPH

We consider a MAC-based counterpart to $\mathsf{CGT\text{-}Hash}$. Namely, using a variable-input-length pseudorandom function (PRF) $F : \mathcal{K} \times [k] \times \mathcal{X} \to \mathcal{T}$,

$$\mathsf{CGT\text{-}MAC}[F, \mathbb{M}](K, X) = (Y_1, \ldots, Y_k), \quad Y_i = F(K, i, X_{|\mathbb{M}_i})$$

Table 1 Comparison of Hash-based PPHs. We assume that the baseline hash G runs in $O(n)$ time for n-bit/item input.

	Property	Output	Hash time	Eval time
PPH1 [4]	$\mathsf{GapHAM}^{d,\epsilon}$	$O(\frac{nt}{\log n})$	$O(\frac{n^2}{\log n})$	$O(\frac{n}{\log n})$
CGT-Hash	$\mathsf{HAM}^d, \mathsf{DETECT}^d$	$O(d^2 t \log n)$	$O(d^2 n \log n)$	$O(d^2 n \log n)$[1)]

1) Improved to $\mathrm{poly}(d^2 \log n)$ if efficient decoders are available (see Section 4.1)

as in the same manner to CGT-Hash. Eval is identical to that of CGT-Hash and does not involve the key. We cannot give K to the adversary, as there is no security guarantee for PRF once the key is known. Thus Robust PPH is impossible in the first place. The next strong robustness notion is DO-PPH. Assuming F is a monolithic, black-box PRF, proving DO-PPH security for the above CGT-MAC is almost identical to the DA-PPH security proof of CGT-Hash, hence we omit it here. Such an instantiation of CGT-MAC scheme is the same as the core component of what Goodrich et al. [19] proposed for data forensics applications.

To achieve a further efficiency improvement from the above scheme, we take a CGT-based MAC scheme proposed by Minematsu [25], called GTM. It is a blockcipher-based scheme, however, instead of a conventional variable-input-length PRF (or MAC) such as CMAC, GTM uses a variant of vector-input PRF [34] that accepts an empty string as a component of an input vector, where a vector consists of (a fixed number of) bit strings. Namely, if the i-th row of the test matrix is $(1, 0, 1)$, the input to the underlying vector-input PRF is (X_1, ε, X_3), rather than conventional (X_1, X_3), together with additional index i. Such a PRF is easily built on any variable-input-length PRF via an input encoding, but if we adopt a structure similar to PMAC [1, 33] and simply skip the computation for j-th component if it is an empty string, it allows a significant computational improvement in our application while maintaining appropriate security. Concretely, CGT-MAC uses F which is defined as

$$F(K, i, X_{|\mathbb{M}_i}) = E'\left(K_2, i, \sum_{j \in \mathbb{M}_i} E(K_1, j, X_j)\right). \tag{1}$$

To learn why this F can be interpreted as a vector-input PRF, please refer to [25]. The above F is based on two PRFs, $E : \mathcal{K}_1 \times [k] \times \mathcal{X} \to \mathcal{C}$ for $\mathcal{C} = \{0,1\}^c$ for some positive c, and $E' : \mathcal{K}_2 \times [k] \times \mathcal{C} \to \mathcal{T}$. The key is $K = (K_1, K_2)$. As mentioned above, it is closely related to PMAC. Let $\mathsf{GTM}[F, \mathbb{M}]$ denote the PPH using F and the test matrix $\mathbb{M}$ that is d-disjunct. The Hash and Eval procedures are defined as in the same manner to CGT-Hash. We show that $\mathsf{GTM}[F, \mathbb{M}]$ is a DO-PPH for HAM^d.

▶ **Theorem 9.** *Let F be a keyed function built on E, E' as Eq. (1). If $\mathbb{M}$ is d-disjunct and E, E' are PRFs, $\mathsf{GTM}[F, \mathbb{M}]$ is a DO-robust PPH for the (generalized) Hamming predicate with threshold d (HAM^d) over $\mathcal{B}^n$.*

Proof. Since Eval is a keyless function, the proof approach is almost identical to that of CGT-Hash, and the proof is similar to that of [25]. We first analyze the information-theoretic setting. Let R_F be a function having the same domain and range as F, and uses two b-bit URFs R, R' instead of E and E'. Using a similar argument as the proof of Theorem 8, we observe that breaking the DO-PPH security implies the invalidation of Cons event. Let A be an adversary using q queries to $\mathsf{GTM}[\mathsf{R}_F, \mathbb{M}].\mathsf{Hash}$ oracle and infinite computation time (note that queries to the key-less $\mathsf{GTM}[\mathsf{R}_F, \mathbb{M}].\mathsf{Eval}$ oracle are simulatable hence we omit).

We have

$$\mathbf{Adv}^{\mathtt{DO\text{-}PPH}}_{\mathsf{GTM}[\mathsf{R}_F,\mathbb{M}]}(\mathsf{A}) \leq \mathbf{Adv}^{\mathtt{Cons}}_{\mathsf{GTM}[\mathsf{R}_F,\mathbb{M}].\mathsf{Hash}}(\mathsf{A}'),$$

for some adversary A' using q Hash queries. Let Ideal be the idealized version of $\mathsf{GTM}[\mathsf{R}_F, \mathbb{M}]$, namely using an independent URF instead of $F(K, i, *)$, for each $i \in [k]$. Using the standard hybrid argument, we have

$$\mathbf{Adv}^{\mathtt{Cons}}_{\mathsf{GTM}[\mathsf{R}_F,\mathbb{M}]}(\mathsf{A}') \leq \mathbf{Adv}^{\mathtt{IND}}_{\mathsf{GTM}[\mathsf{R}_F,\mathbb{M}].\mathsf{Hash},\mathsf{Ideal}.\mathsf{Hash}}(\mathsf{B}) + \mathbf{Adv}^{\mathtt{Cons}}_{\mathsf{Ideal}.\mathsf{Hash}}(\mathsf{A}'') \tag{2}$$

Since each component function of Ideal is an independent random function, as in the case of $\mathsf{CGT\text{-}Hash}$, the last term of the right hand side of Eq. (2) is reduced to the collision probability (in the same component function; we do not have to care about collision between different components). Let $\mathbf{Adv}^{\mathtt{Coll}}_{\mathsf{Ideal}.\mathsf{hash}}(\mathsf{A})$ denote such a collision probability by A. We have

$$\mathbf{Adv}^{\mathtt{Cons}}_{\mathsf{Ideal}.\mathsf{hash}}(\mathsf{A}'') \leq \mathbf{Adv}^{\mathtt{Coll}}_{\mathsf{Ideal}.\mathsf{hash}}(\mathsf{A}'') \leq \frac{q^2}{2^{t+1}}, \tag{3}$$

where the last equation follows from the standard collision analysis. For the first term of the r.h.s. of Eq. (2), define $X^{(i)}$ and $T^{(i)} = (T^{(i)}_1, \ldots, T^{(i)}_k)$ as the i-th Hash query and its response, and let $S^{(i)} = (S^{(i)}_1, \ldots, S^{(i)}_k) \in (\{0,1\}^c)^k$ be the inputs to R', i. e., $S^{(i)}_j = \sum_{h \in \mathbb{M}_j} \mathsf{R}(K_1, h, X^{(i)}_h)$. Based on the analysis [25] (which is also a variant of PMAC's security proof [33]), we show that the indistinguishability is reduced to the collision between S, namely the first term of the r.h.s. of Eq. (2) is bounded by

$$\binom{q}{2} \cdot \Pr[\exists h, h' \in [q], i \in [k] : X^{(h)}_{|\mathbb{M}_i} \neq X^{(h')}_{|\mathbb{M}_i}, S^{(h)}_i = S^{(h')}_i] \leq \binom{q}{2} \cdot \frac{1}{2^c} \leq \frac{q^2}{2^{c+1}}, \tag{4}$$

where the probability is defined by A'' and $\mathsf{GTM}[\mathsf{R}_F, \mathbb{M}]$. The second inequality follows from the observation that each component of keyed message hashing procedure, $Y \leftarrow \sum_{j \in \mathbb{M}_i} \mathsf{R}(j, X_j)$, is XOR-universal, that is, for any $X^{(h)}$ and $X^{(h')}$ and $i \in [k]$ such that $X^{(h)}_{|\mathbb{M}_i} \neq X^{(h')}_{|\mathbb{M}_i}$, the sum $S^{(h)}_i \oplus S^{(h')}_i$ is uniform. Note that we do not have to count collisions between different component functions thanks to the "finalization" by R' taking the index of component as a part of input (often called a domain separation). From Eqs. (2), (3), and (4), we have

$$\mathbf{Adv}^{\mathtt{DO\text{-}PPH}}_{\mathsf{GTM}[\mathsf{R}_F,\mathbb{M}]}(\mathsf{A}) \leq \frac{q^2}{2^{c+1}} + \frac{q^2}{2^{t+1}}. \tag{5}$$

Eq. (5) immediately tells that the computational security of $\mathsf{GTM}[F, \mathbb{M}]$ assuming that E and E' are PRFs. More precisely, $\mathbf{Adv}^{\mathtt{DO\text{-}PPH}}_{\mathsf{GTM}[F,\mathbb{M}]}(\mathsf{A})$ for any A with q Hash queries with t time is at most the sum of PRF advantages of E and E' (i. e., the advantage in distinguishing them from the URF), and the bound shown above. This bound is shown via the standard information-theoretic-to-computational hybrid. ◀

4.3.1 Reduced Hashing Time

A significant difference from the generic $\mathsf{CGT\text{-}MAC}$ is the time complexity for hashing: assuming F runs in $O(n)$ for n-block input (where a block is b-bit), which holds for the most of popular constructions such as HMAC or CMAC, $\mathsf{CGT\text{-}MAC}[F, \mathbb{M}]$ needs $O(kn)$ time for hashing an input. In contrast, the hashing of GTM needs n calls of E and k calls of E' – hence $O(k + n)$ time – by caching the outputs of E (See [25]). If we use the parameters

of Table 1, hashing time is reduced from $O(d^2 n \log n)$ to $O((d^2 \log n) + n)$, assuming the MAC runs in $O(n)$ for n input items. Typically $k \ll n$, as otherwise, the hashing does not compress well, hence this simple trick of [25] allows to reduce the cost of hashing to that of single MAC/PRF computation[3]. We also note that this computation cost does not depend on the contents of $\mathbb{M}$.

4.4 XOR-GTM

Minematsu and Kamiya [26] (MK19) proposed a new approach to CGT-based MAC, dubbed XOR-GTM. It enables to detect d corruptions among n data items using a significantly smaller number of MAC tags than $O(d^2 \log n)$, namely what CGT-MAC or GTM can achieve with a disjunct matrix.

We only briefly describe the scheme. The scheme has almost the same procedure as GTM, using a test matrix $\mathbb{M}$ and PRF E, except that the final E' is a tweakable block cipher [24] taking $i \in [k]$ as a tweak. Here, a tweakable block cipher is an extension of a conventional block cipher that takes a chosen public value, called tweak, as a part of input in addition to the message block. On the detection of corruptions, it first decrypts the received tags using the inverse of E'. Next, it takes linear combinations of the decrypted tags, following the "extended" test matrix $\mathbb{M}^R$, which is a submatrix of row span of $\mathbb{M}$. It also takes the same combinations from the received message and compares the linear combinations. MK19 showed that if $\mathbb{M}^R$ is d-disjunct, this scheme allows detecting of up to d corrupted items. Since the communication cost (number of tags) only depends on $\mathbb{M}$, not $\mathbb{M}^R$, this implies the communication cost can be possibly smaller than the limit of the number of rows of d-disjunct matrix. In other words, what is needed here is a disjunct matrix of small rank for $\mathbb{M}^R$. MK19 presented several such instances using error-correcting codes derived by finite geometry.

4.4.1 Robustness of XOR-GTM

It is natural to expect DO-PPH security for XOR-GTM in the same manner as GTM. However, unlike GTM, the Evaluation oracle of XOR-GTM involves the key. This fact makes the security proof for DO-PPH not directly derived from the original proof of XOR-GTM. Most importantly, the original security proof of XOR-GTM requires that the forward evaluation of E' is pseudorandom, but does not require the pseudorandomness of the inverse. Technically speaking, MK19 requires Tweakable Pseudorandom Permutation (TPRP, for CPA-secure TBC) but not Tweakable Strong PRP (TSPRP, for CCA-secure TBC). Proving DO-PPH security appears to require the latter. This difference comes from the fact that the adversary in the game defined for XOR-GTM (called Decoder Unforgeability, DUF) in MK19 does not have freedom in choosing the tag given to the decoder, while the adversary in the game of DO-PPH notion has no restriction on the choice of tags given to the evaluation oracle. Considering that XOR-GTM significantly reduces the number of tags beyond $O(d^2 \log n)$, proving DO-PPH for XOR-GTM assuming TSPRP for E' is an interesting open question.

[3] A more detailed comparison of GTM and CGT-MAC is possible by instantiating F as a variant of PMAC using b-bit block cipher, and letting $c = b$ and E and E' be the same b-bit block cipher. CGT-MAC needs $w = O(kn)$ block cipher calls, where w denotes the weight of $\mathbb{M}$, and GTM needs $(k + n)$ calls.

5 Expander Graph-based Constructions for PPH and CGT

As stated earlier, PPH1 relies on a bipartite expander graph. The graph has node sets $L = [n]$ and $R = [k]$, by defining the i-th test as its neighbors in L for $i \in R$. We find the relationship between this expander-based PPH with a variant of CGT problem that aims at determining the number of defectives rather than identifying them. This problem has been actively studied for its practical and theoretical importance, say [8, 6, 16, 12] for example.

Among these studies, a CGT scheme proposal by Bshouty and Haddad-Zaknoon [7] (BH21) has an interesting overlap with BLV19's PPH. In detail, they first proposed the construction of matrix using a bipartite expander matrix in the same manner as BLV19. Let $\mathcal{M} = [n]$ be the item set and $\mathcal{I} \subseteq \mathcal{M}$ be the set of defectives. The proposed matrix allows a distinguisher $\mathsf{A}(\ell, \Delta)$ who uses $m(\ell, \Delta)$ tests to distinguish whether $|\mathcal{I}| < \ell/\Delta^2$ (A returns 0) or $|\mathcal{I}| \geq \ell/\Delta$ (A returns 1) without error, for certain parameters ℓ and $\Delta > 1$. Given the maximum value for $|\mathcal{I}|$, $|\mathcal{I}| < D$, the proposed test (for n items and the defective set $\mathcal{I}$) runs $\mathsf{A}(D/\Delta^i, \Delta)$ for $i = 0, 1, \ldots, \lceil (\log D/\log \Delta) \rceil$, and outputs $\hat{d} = D/\Delta^{i+1}$ for the smallest i such that $\mathsf{A}(D/\Delta^i, \Delta) \to 1$. The $\hat{d}$ guarantees $|\mathcal{I}|/\Delta \leq \hat{d} \leq |\mathcal{I}|\Delta$. Using a known construction of bipartite expander graphs, the number of tests is $(D/\Delta^2) \cdot 2^{O(\log^3(\log n))}$. This construction of $\mathsf{A}(\ell, \Delta)$ in BH21 is essentially identical to the idea of PPH1. We note that the problem setting of BH21 requires D as prior information on $|\mathcal{I}|$ and if D is not known there is no non-trivial testing to estimate $|\mathcal{I}|$ [5]. BH21 can be a variant of hash-based PPH whose goal is to give an estimate of the generalized Hamming distance with a predetermined margin, however, such a variant is not covered by the original definition of PPH by BLV19. Therefore, we cannot expect a direct translation from CGT scheme into a PPH as we did in the previous sections. It may be interesting to further explore the relationships.

6 Conclusions

This paper has shown the connection between the property-preserving hash (PPH) functions and the adversarial error detection schemes combining the classical Combinatorial Group Testing (CGT) and hash/MAC functions. Our findings brought several implications and improvements to the hash/MAC-based PPHs, which has been initially proposed by Boyle et al. [4] but largely overlooked since the proposal. PPH is still in its infancy as a research field, and we believe that the connection we have discovered will be useful for developing PPH. Moreover, we hope that our results will also encourage the CGT research community to look into research on PPH.

References

1 John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 384–397. Springer, Heidelberg, April / May 2002. doi:10.1007/3-540-46035-7_25.

2 Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM*, 13(7):422–426, 1970.

3 Annalisa De Bonis and Giovanni Di Crescenzo. Combinatorial Group Testing for Corruption Localizing Hashing. In *COCOON*, volume 6842 of *Lecture Notes in Computer Science*, pages 579–591. Springer, 2011.

4 Elette Boyle, Rio LaVigne, and Vinod Vaikuntanathan. Adversarially robust property-preserving hash functions. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 16:1–16:20. LIPIcs, January 2019. doi:10.4230/LIPIcs.ITCS.2019.16.

5 Nader H. Bshouty, Vivian E. Bshouty-Hurani, George Haddad, Thomas Hashem, Fadi Khoury, and Omar Sharafy. Adaptive Group Testing Algorithms to Estimate the Number of Defectives. In *ALT*, volume 83 of *Proceedings of Machine Learning Research*, pages 93–110. PMLR, 2018.

6 Nader H. Bshouty, George Haddad, and Catherine A. Haddad-Zaknoon. Bounds for the Number of Tests in Non-adaptive Randomized Algorithms for Group Testing. In *SOFSEM*, volume 12011 of *Lecture Notes in Computer Science*, pages 101–112. Springer, 2020.

7 Nader H. Bshouty and Catherine A. Haddad-Zaknoon. Optimal deterministic group testing algorithms to estimate the number of defectives. *Theor. Comput. Sci.*, 874:46–58, 2021.

8 Chao L. Chen and William H. Swallow. Using Group Testing to Estimate a Proportion, and to Test the Binomial Model. *Biometrics*, 46(4):1035–1046, 1990.

9 Mahdi Cheraghchi and Vasileios Nakos. Combinatorial group testing and sparse recovery schemes with near-optimal decoding time. In *61st FOCS*, pages 1203–1213. IEEE Computer Society Press, November 2020. doi:10.1109/FOCS46700.2020.00115.

10 Graham Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. *ACM Trans. Database Syst.*, 30(1):249–278, 2005.

11 Giovanni Di Crescenzo and Gonzalo R. Arce. Data Forensics Constructions from Cryptographic Hashing and Coding. In *IWDW*, volume 7128 of *Lecture Notes in Computer Science*, pages 494–509. Springer, 2011.

12 Peter Damaschke and Azam Sheikh Muhammad. Competitive Group Testing and Learning Hidden Vertex Covers with Minimum Adaptivity. *Discret. Math. Algorithms Appl.*, 2(3):291–312, 2010.

13 Peter Damaschke, Azam Sheikh Muhammad, and Gábor Wiener. Strict group testing and the set basis problem. *J. Comb. Theory, Ser. A*, 126:70–91, 2014.

14 Giovanni Di Crescenzo, Shaoquan Jiang, and Reihaneh Safavi-Naini. Corruption-localizing hashing. In Michael Backes and Peng Ning, editors, *ESORICS 2009*, volume 5789 of *LNCS*, pages 489–504. Springer, Heidelberg, September 2009. doi:10.1007/978-3-642-04444-1_30.

15 D. Du and F. Hwang. *Combinatorial Group Testing and Its Applications*. Applied Mathematics. World Scientific, 2000.

16 Moein Falahatgar, Ashkan Jafarpour, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Estimating the number of defectives with group testing. In *ISIT*, pages 1376–1380. IEEE, 2016.

17 Nils Fleischhacker, Kasper Green Larsen, and Mark Simkin. Property-Preserving Hash Functions from Standard Assumptions. *CoRR*, abs/2106.06453, 2021. arXiv:2106.06453.

18 Nils Fleischhacker and Mark Simkin. Robust property-preserving hash functions for hamming distance and more. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 311–337. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77883-5_11.

19 Michael T. Goodrich, Mikhail J. Atallah, and Roberto Tamassia. Indexing information for data forensics. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 206–221. Springer, Heidelberg, June 2005. doi:10.1007/11496137_15.

20 Shoichi Hirose and Junji Shikata. Aggregate Message Authentication Code Capable of Non-Adaptive Group-Testing. *IEEE Access*, 8:216116–216126, 2020.

21 Edwin S. Hong and Richard E. Ladner. Group testing for image compression. *IEEE Trans. Image Process.*, 11(8):901–911, 2002.

22 Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *STOC*, pages 604–613. ACM, 1998.

23 Piotr Indyk, Hung Q. Ngo, and Atri Rudra. Efficiently Decodable Non-adaptive Group Testing. In *SODA*, pages 1126–1142. SIAM, 2010.

24 Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, Heidelberg, August 2002. doi:10.1007/3-540-45708-9_3.

25 Kazuhiko Minematsu. Efficient message authentication codes with combinatorial group testing. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015, Part I*, volume 9326 of *LNCS*, pages 185–202. Springer, Heidelberg, September 2015. doi:10.1007/978-3-319-24174-6_10.

26 Kazuhiko Minematsu and Norifumi Kamiya. Symmetric-key corruption detection: When XOR-MACs meet combinatorial group testing. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *ESORICS 2019, Part I*, volume 11735 of *LNCS*, pages 595–615. Springer, Heidelberg, September 2019. doi:10.1007/978-3-030-29959-0_29.

27 Leon Mutesa, Pacifique Ndishimye, Yvan Butera, Jacob Souopgui, Annette Uwineza, Robert Rutayisire, Ella Larissa Ndoricimpaye, Emile Musoni, Nadine Rujeni, Thierry Nyatanyi, Edouard Ntagwabira, Muhammed Semakula, Clarisse Musanabaganwa, Daniel Nyamwasa, Maurice Ndashimye, Eva Ujeneza, Ivan Emile Mwikarago, Claude Mambo Muvunyi, Jean Baptiste Mazarati, Sabin Nsanzimana, Neil Turok, and Wilfred Ndifon. A pooled testing strategy for identifying sars-cov-2 at low prevalence. *Nature*, 589(7841):276–280, January 2021. doi:10.1038/s41586-020-2885-5.

28 Hung Q. Ngo and Ding-Zhu Du. A Survey on Combinatorial Group Testing Algorithms with Applications to DNA Library Screening. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 2000.

29 Hung Q. Ngo, Ely Porat, and Atri Rudra. Efficiently Decodable Error-Correcting List Disjunct Matrices and Applications - (Extended Abstract). In *ICALP (1)*, volume 6755 of *Lecture Notes in Computer Science*, pages 557–568. Springer, 2011.

30 Yoshinori Ogawa, Shingo Sato, Junji Shikata, and Hideki Imai. Aggregate message authentication codes with detecting functionality from biorthogonal codes. In *ISIT*, pages 868–873. IEEE, 2020.

31 Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *J. Algorithms*, 51(2):122–144, 2004.

32 Ely Porat and Amir Rothschild. Explicit Nonadaptive Combinatorial Group Testing Schemes. *IEEE Trans. Inf. Theory*, 57(12):7982–7989, 2011.

33 Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, Heidelberg, December 2004. doi:10.1007/978-3-540-30539-2_2.

34 Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006. doi:10.1007/11761679_23.

35 Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.

Protecting Distributed Primitives Against Leakage: Equivocal Secret Sharing and More

Carmit Hazay
Bar-Ilan University, Ramat-Gan, Israel

Muthuramakrishnan Venkitasubramaniam
Georgetown University, Washington, DC, USA

Mor Weiss
Bar-Ilan University, Ramat-Gan, Israel

Abstract

Leakage-resilient cryptography aims to protect cryptographic primitives from so-called "side channel attacks" that exploit their physical implementation to learn their input or secret state. Starting from the works of Ishai, Sahai and Wagner (CRYPTO'03) and Micali and Reyzin (TCC'04), most works on leakage-resilient cryptography either focus on protecting general computations, such as circuits or multiparty computation protocols, or on specific non-interactive primitives such as storage, encryption and signatures. This work focuses on leakage-resilience for the middle ground, namely for *distributed and interactive* cryptographic primitives.

Our main technical contribution is designing the *first* secret-sharing scheme that is *equivocal*, resists *adaptive* probing of a *constant fraction* of bits from each share, while incurring only a *constant* blowup in share size. Equivocation is a strong leakage-resilience guarantee, recently introduced by Hazay et al. (ITC'21). Our construction is obtained via a general compiler which we introduce, that transforms *any* secret-sharing scheme into an equivocal scheme against adaptive leakage. An attractive feature of our compiler is that it respects additive reconstruction, namely, if the original scheme has additive reconstruction, then the transformed scheme has linear reconstruction.

We extend our compiler to a general paradigm for protecting distributed primitives against leakage, and show its applicability to various primitives, including secret sharing, verifiable secret sharing, function secret sharing, distributed encryption and signatures, and distributed zero-knowledge proofs. For each of these primitives, our paradigm transforms *any* construction of the primitive into a scheme that resists adaptive party corruptions, as well as adaptive probing leakage of a constant fraction of bits in each share when the share is stored in memory (but not when it is used in computations). Moreover, the transformation incurs only a constant blowup in the share size, and respects additive reconstruction – an important feature for several of these primitives, such as function secret sharing and distributed encryption.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives; Theory of computation → Cryptographic protocols

Keywords and phrases Leakage Resilience, Secret Sharing, Equivocal Secret Sharing, Verifiable Secret Sharing, Function Secret Sharing, Threshold Encryption, Distributed Zero-Knowledge Proofs

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.3

Related Version *Full Version*: `https://eprint.iacr.org/2022/497` [58]

Funding *Carmit Hazay*: Supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, and by ISF grant No. 1316/18.
Mor Weiss: Supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office.

Acknowledgements We thank the anonymous ITC'22 reviewers for helpful comments.

1 Introduction

Starting with the works of Kocher et al. [65, 66], and more recently in works on Meltdown and Spectre attacks [71, 64]), it has been demonstrated time and again that cryptographic primitives are susceptible to side-channel attacks.

The goal of leakage-resilient cryptography is to protect against such attacks. The focus of this work is on constructing lightweight leakage-resilient variants of central distributed and interactive cryptographic primitives. We first survey the main results in the field.

There has been a long and successful line of works on protecting either *general computations* (initiated by Ishai, Sahai and Wagner [60], and Micali and Reyzin [74]), or *specific non-interactive primitives* such as storage (e.g., in [42, 32, 2]), signatures [20], and encryption schemes [41, 2, 77, 39, 22, 24]. (We refer the interested reader to the excellent survey by Kalai and Reyzin [62] for a discussion of works in the field.)

1.0.1 Previous works on leakage-resilience

Works protecting general computations from leakage constrain the permissible leakage in various ways, either restricting the complexity class from which leakage functions can be chosen [60, 46, 79, 76, 75, 3], or assuming leakage functions operate on disjoint sets of wires of the circuit [50, 61, 51, 43, 33, 34, 52] (the so called "Only Computation Leaks" – or OCL – model). Due to their generality, these works are usually complex, rely on additional computational assumptions (such as public-key cryptography [50, 61, 52] or even indistinguishability obfuscation [52]), or incur high (polynomial) overheads. There are also works on leakage-resilient Multi-Party Computation (MPC) protocols [18], as well as works on interactive protocols (such as MPC [10, 16, 11] and zero-knowledge proofs [49, 10]) that achieve a weaker leakage-resilience guarantee known as *leakage tolerance*, which allows the *ideal-world adversary* to obtain leakage on the inputs of the honest parties.

On the other extreme, there are leakage-resilient constructions of storage [42, 32, 2, 41, 23, 40]), signatures [20], and encryption schemes [41, 2, 77, 39, 22, 24]. While focusing on these primitives admits simpler and more efficient constructions than in the general case, these constructions are naturally more limited since they offer solutions for specific (and non-interactive) tasks only, and do not readily extend to the distributed setting in which multiple parties wish to jointly compute on their inputs in a leakage-resilient manner.

There are only a handful of works on the middle ground, namely on protecting *distributed (and interactive) primitives* from leakage, despite the centrality of such primitives in designing cryptographic protocols.[1] Specifically, Boyle et al. [19] use randomness extractors to construct coin-tossing and verifiable secret sharing secure against a single leakage query (i.e., non-adaptive leakage), and [49, 10] design leakage-tolerant zero-knowledge proofs (as noted above, leakage tolerance is a weaker notion than leakage-resilience).

1.0.2 Leakage-resilient secret sharing

Unlike other distributed primitives, the leakage-resilient notion of *secret sharing* – an important cryptographic primitive with various applications in different areas of cryptography – has received extensive attention from the community in recent years [44, 35, 53, 54, 8, 81, 4,

[1] We note that one could protect such primitives from leakage using works protecting general computations, or by employing generic leakage-resilient MPC protocols. However, as discussed above these are complex, and either incur high costs or require additional computational assumptions.

67, 1, 29, 69, 72, 73, 82, 28]. Focusing on secret sharing, which only protects *information* (as opposed to protecting *computation*, as in the works on general leakage-resilient computation discussed above) allows to obtain simpler and more efficient constructions.

Works on Leakage-Resilient Secret Sharing Schemes (LR-SSSs) can be roughly divided into two categories. The first category, initiated by [8] (though some results appeared already in earlier works, e.g., [12]), analyzes the leakage-resilience properties of *existing* secret-sharing schemes. They show that linear secret sharing schemes[2] resist *non-adaptive local* leakage, namely the adversary can make a single leakage query, which returns few leakage bits computed on each share independently.

Proving leakage-resilience of existing primitives has several advantages. First, applications can still exploit specific design features or additional properties, such as additivity or linearity of reconstruction (which are very useful, as discussed in Section 2 below). Moreover, classic cryptographic primitives are usually more efficient than special-purpose leakage-resilient schemes. However, restricting oneself to existing primitives has a major disadvantage: we are restricted by the leakage-resilience properties of a scheme, which was *not designed with the goal of protecting against leakage*. In some cases, it is unknown whether leakage resilience can be enhanced (e.g., to allow *adaptive* leakage in which the adversary makes several leakage queries, each depending on the answers to previous queries). There are also inherent limitations to the leakage-resilience of some schemes, in terms of the leakage bound and reconstruction threshold (see Section A). For example, Shamir's secret sharing scheme over fields of characteristic 2 does not resist leakage of even *a single bit* from each share [55].

These limitations of existing secret sharing schemes have motivated the second category of LR-SSSs which focuses on designing *new* schemes – thus offering the possibility to overcome the inherent limitations of existing schemes. The goal of this line of works is to resist leakage from a wide range of leakage functions and arbitrary thresholds [1, 81, 67, 29, 69, 28]. The advantage of designing new schemes is clear: we can (potentially) protect against a wide range of leakage classes, and in particular ones against which classic secret sharing schemes are insecure, such as adaptive leakage queries [67, 29, 69, 28], or global leakage (for restricted classes of permissible leakage functions) [12, 69]. Unfortunately, the stronger leakage-resilience guarantee does not come without a price: these constructions are typically complex [67, 29, 69, 28]; some works are considerably less efficient than standard (non-LR) schemes [53, 54, 29]; and most (if not all) works do not maintain the design benefits of classic constructions (such as linear reconstruction) and thus cannot be used in many applications of standard secret sharing [44, 35, 53, 54, 1, 81, 4, 67, 29, 69, 28]. Some of these works describe a general compiler that transforms any secret sharing schem into a leakage-resilient one [81, 1, 29], while others describe specially-tailored constructions [53, 67, 69, 28].

The literature on LR-SSSs can be viewed as studying two extremes: either focusing on *existing* schemes and analyzing their leakage-resilience guarantees, or alternatively *pushing leakage-resilience "to the limit"*, mostly by designing new specialized (and often also complex and significantly less efficient) schemes.[3] But what can we obtain by only *slightly modifying* existing schemes? In particular,

Can we obtain LR-SSSs that surpass the leakage-resilience of classic schemes while maintaining their structural features?

[2] Roughly, in a linear secret sharing over a finite field each share is a linear combination of the secret and random field elements.

[3] As noted above, the works of [81, 1, 29] construct general compilers. However, [81, 1] resists only non-adaptive leakage and the construction of [29] has low rate.

More generally, we study the leakage-resilience of distributed cryptographic primitives such as Verifiable Secret Sharing (VSS), Function Secret Sharing (FSS), Threshold Encryption (TES), threshold signatures, and distributed Zero-Knowledge (dZK) proofs, asking the following:

Do there exist simple leakage-resilient variants of distributed cryptographic primitives such as VSS, FSS, TES, threshold signatures, and dZK?

We focus on a setting in which the adversary can maliciously and adaptively corrupt parties (i.e., the adversary does not need to commit ahead of time to the set of corrupted parties). Similar to the case of secret sharing, our goal is to devise *simple* schemes that *preserve the main design features* of existing (non-LR) schemes. To the best of our knowledge, there are no leakage-resilient constructions of FSS, TES, distributed signatures, and dZK proofs, even disregarding the desiderata of simple schemes. Existing leakage-resilient VSS schemes [19] are only secure against static corruptions (i.e., when the set of corrupted parties is determined at the onset of the computation), and employ randomness extractors that do not preserve linearity, which – as explained below – is one of our main goals in this work. (See Section A for a detailed discussion of the VSS scheme of [19], and a comparison with our results.)

Considering leakage-resilience in the more general context of distributed primitives also helps in identifying the important structural features of existing schemes which we should strive to preserve. For example, applications of FSS require linear sharing (namely, that the reconstruction procedure is linear), and applications of TES in MPC require that parties hold additive shares of a secret key.[4] Thus, in this work we focus on designing leakage-resilient secret sharing schemes – and more generally, distributed primitives – with *linear reconstruction*.

1.0.3 Our leakage model: adaptive probing-resilience for data at rest

We focus on protecting cryptographic primitives against *local probing leakage*. We provide information-theoretic security against adversaries that can *adaptively* probe the shares, as well as corrupt *any* unauthorized set (i.e., a subset of parties such that their collective shares reveal nothing about the shared secret).[5] Our focus on probing resilience stems from our goal of preserving linear reconstruction, which implies we cannot hope to protect against leakage classes that are significantly more general than probing.

For distributed primitives beyond secret sharing, we focus on protecting *data at rest*. That is, we protect the shares from leakage throughout the lifetime of the process, *except* when they are directly computed on. Our leakage model is related to several leakage models (in particular, that of [8]), and can be seen as a combination of the leakage model of [60] that restricts the function class from which leakage functions can be chosen, and the "Only Computation Leaks" (OCL) model of [74] which assumes that different memory components leak separately. (As noted in Section 1.1 below, our constructions actually achieve a stronger notion known as OCL+ [9].)

---

[4] For example, in the TES based on Diffie-Helman, parties hold public-key shares of the form $g^{\mathrm{SK}_1}, \ldots, g^{\mathrm{SK}_m}$, where g is a public group generator, and $\mathrm{SK}_1, \ldots, \mathrm{SK}_m$ are the secret-key shares. In this case, the public key is $g^{\sum_i \mathrm{SK}_i}$ and the secret key $\mathrm{SK} = \sum_i \mathrm{SK}_i$ is unknown to any party. See the full version [58] for further details.

[5] For primitives with computational security (such as FSS and TES), we provide the best possible security, namely computational security for party corruptions, and *information-theoretic* for leakage. See the full version [58] for further details on the security definition.

Our focus on protecting data at rest is motivated by many application scenarios in which following a short sharing phase – in which one party locally generates and distributes shares to the other parties – the shares are stored in memory for extended time periods, during which the adversary can adaptively leak on them. Since leakage may accumulate over time, and the adversary might be able to infiltrate the devices of certain parties (in effect making them – passively or actively – corrupted), it is important to incorporate party corruptions into the security model. We expect the periods of time during which shares are computed on – which involve fast local computations, or fast interactive protocols requiring all parties to be online simultaneously – to be much shorter than the lifetime of the system (indeed, state-of-the-art practical secure multi-party computation protocols last milliseconds [83]). Therefore, an adversary would not be able to launch a meaningful leakage attack during these short computation phases.

Example: Function Secret Sharing (FSS). We demonstrate our leakage model and its motivation through the example of FSS. FSS consists of a generation phase in which a function f is shared between multiple parties by generating function keys; and a local evaluation phase – whose duration we expect to typically be marginal compared to the lifetime of the system – in which each party can locally compute from its function key a share of the output $f(x)$, for any input x. These output shares form an additive (or, more generally, linear) secret sharing of $f(x)$. During this process, we protect the function keys and output shares from leakage, *except* during the evaluation phase in which the output shares are (locally) computed from the function keys. (We stress that once the evaluation phase terminates, function keys and output shares are again protected from leakage.)

In FSS applications, the function and output shares are often stored in memory for extended time periods, thus naturally necessitating leakage-resilience for data at rest. Indeed, in one FSS application, statistics (such as website traffic statistics [17]) – in the form of output shares – are accumulated over time. In particular, output shares are stored in memory for long periods of time. Another application is "FSS as a service", namely when the function keys are distributed between multiple servers (each server holds a single function key), and users can ask the servers to evaluate the function on inputs of their choice. More specifically, a data-owner generates function keys $k_1, \ldots, k_m$ for a secret function f, and stores k_i on server i. Users can then ask the servers for the value $f(x)$ by sending x to each server i, who locally evaluates its function key k_i on x (using the FSS-evaluation procedure), and sends the output share y_i back to the user, who can then recover $f(x)$ from the output shares $y_1 \ldots, y_m$. Thus, the user learns $f(x)$ but does not learn f, and none of the servers learn f or $f(x)$. In this scenario, function keys may be stored on the servers indefinitely, and output shares y_i might likewise be stored for extended time periods, e.g., for backup purposes, so it is important to protect both from leakage.

Our leakage model is further motivated by other situations in which shares are stored in memory for a long time, e.g., when using Beaver triples [7] that are generated in an offline phase and later consumed during the execution of an MPC protocol. We describe further applications (e.g., for TES) in the full version [58], and note that in some cases (e.g., for certain TES constructions) we are able to protect *the entire process* – including computation – from leakage.

1.0.4 Beyond leakage-resilience

In the context of secret sharing, we also explore a stronger leakage-resilience guarantee known as *equivocation*, recently introduced by [57] and employed towards constructing zero-knowledge proof systems. Roughly, equivocation guarantees that there exists an efficient

simulator that can answer adaptive leakage queries, as well as adaptively simulate the views of corrupted parties. Moreover, at some point during the simulation, the simulator is given an *arbitrary* secret, and is then able to generate an entire secret sharing of this secret which is consistent with the leakage and the views of corrupted parties. That is, the leakage can be efficiently "explained" as leakage on the shares of *any* arbitrary secret. More specifically, we consider *semi-honest* adversaries that can *adaptively* obtain the shares of any unauthorized set, as well as adaptively probe a constant fraction of bits from the shares of every other party.

Unfortunately, the existing equivocal SSS [57] is a degenerate one-party scheme,[6] so the adversary cannot obtain the share in full. Therefore, we ask:

Can we construct multiparty *equivocal SSSs secure against probing leakage?*

1.1 Our Results

We construct the *first multiparty equivocal secret sharing scheme*. Our scheme can withstand *any number* of corrupted parties, as well as leakage of a *constant fraction* of bits from each of the other shares. Our scheme is also *tamper-resilient* against arbitrary tampering of every share that modifies a constant fraction of bits (below the error-correction bound) in the share. We then *extend our technique* to obtain leakage-resilience and tampering-resilience for *other distributed primitives*.

1.1.1 Equivocal secret-sharing

We design a compiler that transforms standard (possibly non-leakage-resilient) SSSs into LR-SSSs with the following features. First, it obtains the stronger guarantee of equivocation. Second, it respects additive reconstruction, namely if the underlying SSS has additive reconstruction, then the resultant LR-SSS will have linear reconstruction. Third, our compiler guarantees leakage-resilience against *any* access structure, in the following sense. Given any SSS for an access structure Acc, the resultant LR-SSS is leakage-resilient against an adversary obtaining the shares of an unauthorized set $T \notin \mathsf{Acc}$, as well as the answers to *adaptive* probing queries that can leak a *constant* fraction of bits from each share. Finally, the resultant scheme is also tampering-resilient in the following sense: the correct secret will be reconstructed, even if an external adversary can arbitrarily modify a constant fraction of bits *in each share*. In fact, we obtain the stronger guarantee that *the original share can be efficiently recovered* from the modified share. (Thus, necessarily, we can only handle tampering of a fraction of bits which is *below* the error-correction bound. See Section A for a comparison with other tampering-resilient notions.) Our compiler is also conceptually simple: roughly, it protects the shares by encoding them using a linear code with equivocation properties (known as an *RPE* [30, 6, 31, 5], see Section 2).

Our construction is summarized in the following theorem, where (Acc, ℓ)-equivocation means the adversary can *adaptively* obtain the shares of any unauthorized set $T \notin \mathsf{Acc}$, as well as *adaptively* probe ℓ bits from every other share, and τ-tampering-resilience guarantees that the correct secret will be reconstructed even if an external adversary can arbitrarily modify τ bits in *every* share. (See the full version [58] for the formal statement.)

[6] In such schemes there is a single party and only one share. Secrecy holds under the supported leakage class, but not if the share is revealed in full. This is a degenerate scheme since it is not useful as a secret-sharing mechanism, but has application to, e.g., construction of zero-knowledge probabilistic proof systems.

▶ **Theorem 1** (Equivocal SSS from standard SSS – informal). *Let* SSS *be an m-party secret sharing scheme for secrets in $\mathcal{S}$ and an access structure* Acc*, with shares of length N. Then there exists an $\ell = \Omega(N)$ for which there exists an m-party secret sharing scheme* SSS′ *for secrets in $\mathcal{S}$ and access structure* Acc*, with ℓ-tampering-resilience, and (Acc, ℓ)-equivocation. Moreover, the secret shares have length $O(N)$. Furthermore, if* SSS *has additive reconstruction, then* SSS′ *has linear reconstruction.*

In Table 1 (Page 12) we compare our equivocal SSS with existing LR-SSSs. Applying Theorem 1 to Shamir's secret sharing, we obtain the following (see the full version [58] for the formal statement).

▶ **Corollary 2** (Equivocal SSS – informal). *There exists an $\ell \in \mathbb{N}$ such that for any $t < m \in \mathbb{N}$ there exists an m-party (Acc, ℓ)-equivocal and ℓ-tampering-resilient secret sharing scheme, where* Acc *consists of all subsets of $[m]$ of size more than t. Moreover, $\ell = \Omega(N)$, where N is the share length.*

Corollary 2 demonstrates that by applying our transformation to Shamir's secret sharing scheme, we can circumvent the impossibility result of [55] on the leakage-insecurity of Shamir's scheme over fields of characteristic 2, as well as the lower bounds of [8] and [78] (which are discussed in Section A), while still preserving the main features of Shamir's scheme.

1.1.2 Leakage-resilient distributed primitives

We extend our compiler to a general paradigm for securing cryptographic primitives against adaptive probing leakage, and show its applicability to a wide range of primitives in which a secret is distributed between multiple parties. Specifically, we obtain leakage-resilient variants of verifiable secret sharing, function secret sharing, threshold encryption and signatures, and distributed zero-knowledge proofs. In all our constructions, the blowup in the share size is constant (compared to the original, non-leakage-resilient scheme), and the resultant scheme is secure against adaptive probing of a constant fraction of bits from each share separately even if the adversary adaptively corrupts parties. (We note that in certain primitives – such as TES and FSS – each party may store several shares of different secrets, in which case we assume that each share leaks separately. As discussed above, this is similar to the restriction on leakage in the OCL model. However, our constructions satisfy a stronger guarantee known as OCL+ [9] which allows to alternately leak from the shares, where each leakage query might depend on the responses to previous queries. In particular, the leakage on different shares is not necessarily independent.) All our constructions are also tamper-resilient against tampering of a constant fraction of bits in each share. (See the full version [58] for the constructions and formal theorem statements.)

We note that *defining* leakage-resilience for these primitives is highly *non-trivial*. This is especially true for computational primitives such as TES and FSS, in which we *need to combine the computational* security guarantee for *fully-revealed shares*, with an *information-theoretic guarantee for leakage* on the other shares.[7] Even for information-theoretic objects such as dZK proofs, we encounter subtleties since standard definitions are only *statically*-secure. As an additional contribution, we generalize these definitions to the adaptive setting,

[7] We note that similar security models – with information-theoretic security against leakage, and computational security for the underlying primitive itself – have been considered in the leakage resilience literature, e.g., for public-key primitives in the bounded retrieval model [42, 32]. However, due to the distributed nature of the primitives we consider, our security definitions are more intricate than in these aforementioned works.

and prove that existing protocols (e.g., a protocol of [13]) are adaptively-secure, a result which may be of independent interest. See Section 2, and the formal definitions in the full version [58], for a discussion of the subtleties of the definitions.

1.2 Applications

In this section we describe several applications of equivocal SSSs.

1.2.1 Deniable Secret Sharing Schemes

Deniability in the context of encryption. The notion of deniable encryption, introduced by Canetti et al. [26], ensures private communication in the presence of attackers that are also provided with all the private information: the plaintext, the random bits used for encryption, and any secret keys the parties have. Specifically, deniable encryption introduces an additional efficient "faking" algorithm that is not part of the standard secure communication definitions. This new algorithm generates "fake" internal states for the participating parties, that are indistinguishable from the real states, and are consistent with the (public) communication transcript and any plaintext of the parties' choice. In addition to being an interesting object, deniable encryption has important applications, the most immediate one being preventing vote-buying in electronic voting schemes. Deniability is a very strong security guarantee. Specifically, deniable encryption schemes require stronger hardness assumptions than classic encryption schemes [80, 27].[8]

Deniable secret-sharing. The notion of deniability can be highly useful in applications using secret sharing, where a dealer distributes the shares of a sensitive database – e.g., a clinic sharing a database containing the medical records of its patients – amongst a set of parties or servers.[9] Clearly, if the attacker obtains sufficiently many shares (specifically, of an authorized set), it can fully recover the secret. On the other hand, obtaining only the shares of an unauthorized set implies deniability since the secret is information-theoretically hidden. (We stress, however, that even in this case the "faking" algorithm may not be efficient.) Deniable secret sharing explores an intermediate scenario where the adversary holds the shares of some unauthorized set, and *additionally obtains partial knowledge of the honest parties' shares.* This model is motivated by the fact that an attacker may attack servers in various ways, in some cases succeeding in extracting the entire share, while in other cases it may only be able to gather (through physical observations) some leakage on the share.

Formally defining deniable secret sharing requires some care, in particular in deciding what information is available to the faking algorithm. In the context of public-key deniable *encryption*, the faking algorithm knows the ciphertext it is trying to "explain", so a natural adaptation to *secret sharing* is to give the faking algorithm the secret shares it is trying to equivocate. This, however, is useless – it should be intuitively clear that the faking algorithm cannot successfully explain the leakage without knowing which parts of the secret shares had leaked. Therefore, we define deniability by giving the faking algorithm *the leakage*, rather than the entire secret sharing. Specifically, the faking algorithm is given the leakage queries and responses (but *not* the secret shares in their entirety), and using them "explains" the sharing

[8] We note that information-theoretic symmetric-key encryption schemes are easily deniable as every ciphertext can be explained with respect to every plaintext. However, symmetric-key encryption requires the parties to agree on the secret key, i.e., a-priori agree on common randomness, and it is unclear how to do so in a deniable way.

[9] We assume no a-priori common randomness between the dealer and the parties.

as a sharing of an arbitrary (possibly different) secret, by providing "fake" randomness for the sharing algorithm. This notion of "public leakage" arises naturally in many settings, since oftentimes leaked data becomes public (e.g., via social or other forms of media). For example, in the application scenario described above where a clinic A shares its database of patient medical records, this database might contain medical records of celebrities. A competing clinic B might be able to obtain leakage on the shares, allowing it to learn the identity and medical history of some celebrity patients. Clinic B might publish this information with the hopes of ruining the reputation of clinic A. Deniability allows clinic A to give an alternative explanation for the shares – in particular, replacing the identities of all celebrity patients – and publish it to dismiss the leaked information.

Our deniable secret-sharing scheme. We design deniable secret sharing schemes based on equivocal SSSs, where the faking algorithm emulates the equivocal SSS simulator. We note that to equivocate the shares, the equivocal SSS simulator needs to know the leakage queries and responses, and these are indeed available to the faking algorithm. Using the equivocated secret shares, the dealer can convincingly renounce the original secret, where the equivocated shares serve as a "proof" that validates an arbitrary secret, *even given the adversary's view*. Denying the secret provides a strong guarantee against coercing the dealer, especially since our focus is on *efficient* deniability. We further elaborate on this application in the full version [58].

1.2.2 MPC Resilient to Leakage of "Data at Rest"

Equivocal SSSs are useful towards designing Multiparty Computation (MPC) protocols which guarantee leakage-resilience of private inputs (alternatively, secret local states) when they are stored in memory. That is, MPC with leakage-resilience in the same "data at rest" leakage model described above. More specifically, we consider protocols in which following a leak-free *Sharing* phase, in which each party locally secret-shares its input (either sending them to other parties, or storing them locally until they will be communicated at a later point), the shares are stored in memory until a later leak-free *Computation* phase, in which parties run an MPC protocol on the secret shares to compute the outcome. We stress that the sharing phase is *function oblivious*, namely it can be executed before parties even know which function they wish to jointly compute on their inputs.

Similar to the other primitives considered in this work, this model is motivated by realistic application scenarios in which the (short) computation phase might be executed well after the (short) sharing phase. In particular, the leak-free assumption for the sharing phase is justified by situations in which parties obtain their secret inputs (e.g., bank account balance when the input is for the Millionaire's problem) while in their own homes, i.e., in locations which are less vulnerable to side-channel attacks; or because once they are given their private inputs, parties are likely to quickly "protect" them by performing a fast local computation (i.e., sharing, similar to how one would protect a sensitive word document by adding a password). The leak-free assumption for the computation phase is likewise justified because interactive protocols that require all parties to be online at the same time, and for the duration of the execution, are expected to be fast.

However, between the sharing and computation phases, the secret shares are stored in the internal state of the parties, e.g., on their laptops. Thus, the shares might be subjected to various side channel attacks (as laptop would most likely be carried around by their owners), especially since the computation phase might be executed long after the sharing phase had ended. Moreover, during this time leakage on one party's state might accumulate, up to the point that its internal state might leak entirely. Alternatively, an adversary might be able to infiltrate the party's laptop, thus obtaining its entire internal state.

Equivocal SSSs naturally give a leakage-resilient solution for MPC protocols in this leakage model, against adaptive leakage, by having parties share their inputs using an equivocal SSS. Specifically, leakage on the shares can be adaptively simulated, where if the actual amount of leakage on a party passes some a-priori bound then we consider the party as being (semi-honestly) "corrupted", which is supported by the equivocal SSS (that allows for adaptively leaking full shares). If, at the onset of the computation phase, a party is corrupted, we can equivocate the shares consistently with its actual input. (We note that if the MPC protocol run during the computation phase is adaptively-secure, then we can also handle adaptive corruptions *during* the computation phase.) Thus, equivocal SSSs can be used to protect MPC protocols from adaptive leakage and adaptive semi-honest party corruptions. We note that LR-SSSs which are *not* equivocal provides no guarantees against *adaptive* party corruptions (and sometimes not even against adaptive *leakage queries*).

1.3 Future Directions and Open Questions

We initiate the study of designing leakage-resilient cryptographic primitives by applying a light-weight procedure which respects additive reconstruction. We demonstrate the usefulness of our paradigm by applying it to various cryptographic primitives.

Our work still leaves several interesting research directions to explore. First, it would be interesting to explore which other primitives could be protected from probing leakage using our paradigm. Second, our constructions "respect" additive reconstruction, in the sense that applications relying on additive reconstruction can use our leakage-resilient primitives. As discussed above, this is helpful in the context of FSS and TES. Extending our paradigm to respect *general linear* reconstruction procedures will allow using our leakage-resilient constructions in an even wider array of applications. Moreover, devising an equivocal SSS with a multiplicity property (which enables multiplying shared secrets by operating locally on the shares, and then executing some simple "correction" protocol), would yield leakage-resilient MPC which resists leakage even during computation, thus extending our results beyond the "data at rest" model. Finally, it is natural to ask whether our results extend to more general leakage classes.

2 Techniques

In this section we describe our paradigm and the resultant constructions in more detail. We start by describing our compiler for SSSs.

Recall from Section 1.1 that our goal is to obtain simple SSSs with linear reconstruction and probing resilience against an adaptive adversary that can probe a constant fraction of bits from each share. More generally, we wish to design a general compiler from SSSs to LR-SSSs which preserves efficiency and respects additive reconstruction (namely, if the original scheme has additive reconstruction then the resultant LR scheme will have linear reconstruction). A natural starting point would be to use some sort of encoding, similar to how leakage-resilient compilers for general computations first encode the input for the computation. This encoding can either apply to the vector of secret shares, or to each share separately. To obtain leakage-resilience, it should posses some leakage-resilience guarantees. Moreover, to achieve linear reconstruction, this encoding should have linear reconstruction (i.e., be a linear code). However, to obtain *equivocation*, as needed by our main application of equivocal SSSs, the linear code should posses an additional equivocation property which, roughly, guarantees that given the leakage on the encoding of any message msg, and given

any arbitrary message msg', the leakage can be "explained" in retrospect as the leakage on an encoding of msg'. We note that such leakage-resilience and equivocation properties necessitate *randomized* encoding. Fortunately, such encodings exist.

2.0.1 Main building block: Reconstructible Probabilistic Encodings (RPEs)

RPEs [30, 6, 31, 5] are, roughly, linear codes with an equivocation property. More specifically, an RPE consists of a randomized encoding procedure Encode, a deterministic linear decoder Decode, and a randomized resampler algorithm Rec, where $\mathsf{Encode}(\mathsf{msg})$ outputs a random codeword from a set of possible codewords for msg. (For example, this set can consist of all codewords obtained by applying the generator matrix of a linear code to $\mathsf{msg} \circ s$, namely to the string obtained by concatenating an arbitrary suffix s to msg.) An RPE satisfies the following properties. First, it has error correction from a constant fraction of errors. Second, it has probing resilience for a constant fraction of probed bits, in the sense that for every $\mathsf{msg}, \mathsf{msg}'$, the leakage on random encodings of $\mathsf{msg}, \mathsf{msg}'$ are statistically close. Finally, it is equivocal in the sense that for any message msg, any subset $\mathcal{I}$ of probed bits leak from a random encoding $c \leftarrow \mathsf{Encode}(\mathsf{msg})$ of msg, and any message msg', $\mathsf{Rec}(\mathcal{I}, \mathsf{leak}, \mathsf{msg}')$ outputs an encoding c' of msg' which is statistically close to an encoding of msg' that is random subject to being consistent with the leakage.[10] We note that RPEs resist non-adaptive leakage (also in the equivocation property), but since all properties are statistical, this naturally extends to resisting adaptive leakage, as we show in the full version [58].

2.1 Equivocation from RPEs and Standard Secret Sharing

There is a natural connection between RPEs and equivocal secret sharing, since both offer a method of encoding a message in a way that can later be equivocated. This connection was recently used in [57], who used the equivocation of RPEs to construct zero-knowledge probabilistic proof systems. More specifically, Hazay et al. [57] interpreted RPEs as a degenerate form of equivocal secret sharing, in which there is a single share, which the adversary can probe. This, of course, does not provide any meaningful way of *distributing* (i.e., sharing) the data, which is the main purpose of a SSS (but it does give a meaningful notion of equivocal *encoding* of the data). Moreover, equivocal SSSs can provide a much stronger LR guarantee – the secret remains protected even if an (unauthorized) subset of shares are *leaked in full*. In particular, equivocal SSSs can potentially obtain a much larger leakage rate than RPEs.

Our main observation in this work is that instead of interpreting RPEs as equivocal SSSs, one can *use* them to *transform* essentially *any* standard SSS into an equivocal one. The transformation is conceptually simple: to share a secret, first share it using the underlying SSS, then encode each share separately using the RPE. To reconstruct the secret, first decode each RPE encoding, then recover the secret using the reconstruction procedure of the underlying SSS. The resultant scheme resists adaptive probing leakage, since leakage can be simulated by answering queries using encodings of 0 (or any arbitrary message). Moreover, a

[10] We note that RPEs are usually defined with perfect security, in the sense that leakage-resilience guarantees that the bits leaked from a random encoding of any message msg are uniformly distributed, and the output of Rec is distributed identically to a random encoding of msg' subject to being consistent with the leakage. We chose to present a relaxed version of RPEs because it suffices for our needs, and allows us to quantify exactly how errors in the RPE carry over to the resultant constructions.

combination of the equivocation of the RPE and the secrecy of the underlying SSS guarantees that the resultant SSS is equivocal. Indeed, a simulator can initially answer leakage queries according to an honestly-generated sharing of 0. At some point, the simulator is given an arbitrary secret msg'. At this point, secrecy of the underlying SSS guarantees the simulator can generate a secret sharing of msg' which is consistent with the subset of shares that were *revealed in full* to the adversary. Then, the resampler algorithm of the RPE can be used to equivocate encodings for the remaining shares, consistently with the leakage already provided on them. (We note that the actual analysis is more complex; see the full version [58] for details.) Finally, tampering-resilience follows from the error-correction of the RPE.

The equivocal SSS described in Corollary 2 is obtained by applying our compiler to Shamir's secret sharing and the RPE of [36, 37]. More specifically, Decatur et al. [36, 37] construct a linear code with constant rate, and leakage resilience against a constant fraction of leaked bits. Linearity of the code implies it is also equivocal due to its large dual distance (see [5, Lemma 2]).

Table 1 compares our equivocal SSS construction to known LR-SSSs. We elaborate on these schemes further in Section A.

Table 1 Comparison of Existing Leakage-Resilient Secret Sharing Schemes.
Here, "Equiv." denotes whether the scheme is equivocal, "adaptive queries" states whether the scheme is secure against adaptive leakage queries, "linear recon." describes whether the reconstruction procedure of the SSS is linear, m is the number of parties, and the "restricted joint" leakage model refers to the joint leakage model in which each share can be queried by a single leakage query. The first four rows construct specific LR-SS schemes while the last three rows give a general compiler that transforms any SS to a LR one.

	Scheme	Equiv.	Adaptive Queries	Linear Recon.	Rate	Leakage Rate	Leakage Model
Specialized	[53, 54]	No	No	No	$O(\frac{1}{m})$	$O(\frac{1}{m})$	Local
	[8]	No	No	Yes	$O(1)$	$O(1)$	Local
	[67]	No	Yes	No	$O(\frac{1}{\mathsf{poly}(m)})$	$O(\frac{1}{\mathsf{poly}(m)})$	Joint
	[28]	No	Yes	No	$O(1)$	$O(1)$	Restricted Joint
General	[81]	No	No	No	$O(1)$	$O(1)$	Local
	[29]	No	Yes	No	$O(\frac{1}{\mathsf{poly}(m)})$	$O(\frac{1}{\mathsf{poly}(m)})$	Joint
	This work	**Yes**	**Yes**	**Yes**	$O(1)$	$O(1)$	**Probing**

2.2 Share-then-Encode: A General Paradigm for Leakage- and Tampering-Resilience

We show that our technique is applicable to a wide range of distributed cryptographic primitives, constructing leakage-resilient primitives against probing leakage of data at rest, that are also tampering-resilient. For the primitives discussed below, we identify distinct sharing and reconstruction procedures, and protect the shares by RPE-encoding each share separately. These primitives include protocols and procedures during which the shares are computed on. To execute these protocols/procedures, parties first RPE-decode their share, perform the computation on it, RPE-encode the result, and finally erase the intermediate values of the computation. We note that several of the applications described below crucially rely on the linear reconstruction property of the underlying primitive.

In all our constructions, the blowup in share size (compared to the underlying non-leakage-resilient scheme) is constant, and the resultant scheme resists adaptive probing and tampering of a constant fraction of bits from each share separately. We stress that security is maintained even if the adversary simultaneously corrupts parties *and* obtains leakage on the shares of the honest parties. By default, we allow for adaptive corruptions as well as adaptive leakage queries. With the exception of verifiable secret sharing, the primitives we study below are usually defined with *static* security. Thus, to obtain security in the presence of leakage with adaptive party corruptions, we need to first define adaptive security for these primitives (*without* leakage), which we then extend to adaptive security in the presence of leakage. As we explain below, defining adaptive leakage-resilience is quite subtle, especially for primitives (such as threshold encryption and function secret sharing) with computational security. We note that if one is willing to settle for security against adaptive leakage queries but with *static party corruptions*, then we can simplify our definitions to only handle static corruptions (and rely on the standard static-security of the underlying primitives), however defining security in the presence of leakage remains quite intricate also in this case.

2.2.0.1 Verifiable Secret Sharing (VSS)

A VSS scheme strengthens standard SSSs to protect against a scenario in which the entity generating the shares (called the "dealer") is corrupted. More specifically, a VSS scheme consists of sharing and reconstruction procedures as in standard SSSs, but is also associated with a *Verification* protocol in which parties verify their shares are consistent.[11] VSS has secrecy and correctness as in standard secret sharing, and guarantees an additional *commitment* property, stating that even if a small subset of corrupted parties collude with a corrupted dealer, then following the verification phase the dealer is committed to *some* secret which will be reconstructed, regardless of the behavior of the corrupted parties during reconstruction.

We define leakage-resilient VSS as VSS schemes with a stronger secrecy guarantee that holds even in the presence of leakage on the shares. This is formalized through a distinguishing game between the adversary and a challenger, in which the adversary can adaptively corrupt parties throughout the sharing and verification phases (as in standard VSS), but once verification ends, it can additionally adaptively probe a constant fraction of bits from the share of every honest party. (Our construction also generalizes to protect against leakage between the sharing and verification phases.) We require that the adversary obtains only a negligible advantage in distinguishing between any two secrets shared by the dealer.

We obtain LR-VSS similarly to our equivocal SSS described above. Roughly, we transform a standard VSS scheme VSS into a Leakage-Resilient VSS (LR-VSS) VSS' by having the dealer generate the shares as in VSS, and then the parties verify the shares by running the verification protocol of VSS. Once verification ends, each party RPE-encodes its updated share and erases all information except for this (encoded) updated share.

Given that VSS is oftentimes used as a building block in constructing general multiparty computation protocols, as well as specific distributed tasks (e.g., coin tossing), we believe that our LR-VSS could be useful towards constructing future leakage-resilient protocols.

[11] We note that standard VSS schemes consider only a sharing protocol (which includes the verification protocol) and a reconstruction procedure. We chose to separate the sharing algorithm and the verification protocol, since this enables us to more clearly capture the leakage-resilience guarantees of our VSS protocols.

2.2.0.2 Distributed Zero-Knowledge (dZK) Proofs

Distributed ZK proofs [14] generalize the notion of standard ZK proofs to a setting in which the prover interacts with a set of verifiers, where the input statement to be verified is distributed between the verifiers, and the prover holds the input statement and the corresponding witness (in cases when such a witness exists). throughout the protocol execution, the prover distributes *proof shares* between the verifiers, which the verifiers then use to determine their output (either accept or reject). Our paradigm can be used to protect the proof shares, except when they are directly computed on. We note that in existing dZK proofs (e.g., [14]) there are extended time periods during which parties perform computations that are independent of the proof shares (e.g., running a coin-tossing sub-protocol). During such times, our paradigm protects the proof shares from leakage.

More specifically, we define leakage-resilient dZK by comparing a real-world execution, in which the adversary $\mathcal{A}$ interacts with a challenger $\mathcal{C}$, to an ideal execution in which $\mathcal{A}$ interacts with a simulator Sim. The adversary is allowed to *adaptively* corrupt verifiers, as well as adaptively probe proof shares of honest verifiers. In both executions, the challenger and simulator emulate the honest parties, whereas $\mathcal{A}$ assumes full control of the corrupted parties. We note that the standard security notion for dZK proofs is for *static* corruptions. Our constructions can be instantiated with such statically-secure dZK proofs, in which case LR holds for *static* party corruptions and *adaptive* leakage queries. To obtain fully-adaptive LR-dZK proofs – in which the adversary can also adaptively corrupted verifiers – in the full version [58] we extend the ZK definition of dZK to adaptive corruptions, and prove that a protocol of [13] is adaptively-secure. This could be of independent interest.

The high-level idea of our dZK proofs is similar to the VSS scheme described above: we employ a standard dZK proof Π as follows. The prover in our LR-dZK proof emulates the prover of Π, and RPE-encodes the proof shares before sending them to the verifiers. The verifiers store these encoded proof shares, except when Π requires that they compute on them. In this case each verifier locally decodes the proof shares needed for the computation, performs the computation, RPE-encodes the proof shares and the outcome (if needed) and erases all intermediate values generated during the computation.

2.2.0.3 Function Secret Sharing (FSS)

Function Secret Sharing (FFS) [17, 15] is a cryptographic building block that enables evaluation of a function f in a distributed manner. An FSS consists of two algorithms Gen and Eval. The former algorithm outputs secret function keys $(k_0, \ldots, k_m)$, one for each party . The latter algorithm is run locally by each party. Given a public input x, and the party's secret key k_i, it computes an output share $f_i(x)$ such that $f(x) = \sum_{i\in[m]} f_i(x)$. The security requirement ensures that the individual keys do not disclose (to a computationally-bounded adversary) any information about the function description.

Current FSS constructions are motivated by applications that involve private and distributive access to a large database, and crucially rely on the fact that FSS has linear reconstruction. (In fact, existing FSS schemes have *additive* reconstruction.) Two notable applications of FSS are Private Information Retrieval (PIR) or keyword search, and statistics collection. (See [17] for further discussion of these applications.) In both cases (a set of) clients generate the keys, and upload them to corresponding servers, where the keys are then used to repeatedly execute the desired operation (e.g., keyword search). Since multiple executions are performed, this also requires aggregating the output shares $f_i(x)$ which strongly relies on the linearity of the FSS reconstruction algorithm.

Defining leakage-resilience for FSS is subtle. This is because on the one hand, our goal is to obtain leakage-resilience against the strongest type of adversarial attacks – namely against computationally unbounded adversaries, while on the other hand, FSS is a computational primitive which only guarantees *computational* indistinguishability in the real/ideal paradigm. Thus, a main challenge which one faces when modeling security of leakage-resilient FSS is to decouple FSS-secrecy from leakage-resilience. We capture this separation by splitting the adversary into two algorithmic entities, one that attacks the scheme via queries to a leakage oracle, and another that receives the state of the former adversarial entity and attacks the secrecy of the underlying FSS. We stress that the former adversarial entity may be computationally unbounded. Another obstacle is that we need to maintain consistent answers (e.g., to repeated leakage queries on output shares generated for the same input x). That is, we need to ensure all responses are consistent with prior responses. This is achieved by maintaining a list which, for every input queried to the function, specifies the randomness used to generate the corresponding output shares in Eval. This randomness is used to answer all leakage queries related to this input. (We note that when an input is queried for the first time, a new entry is added to the list.)

At a high level, our construction modifies the key generation algorithm, where function keys are generated using the FSS generation algorithm, and then RPE-encoded. Similarly, to compute the output shares from the function key shares, each party locally RPE-decodes the key share, generates the output share using the evaluation algorithm Eval of the underlying FSS, and then RPE-encodes it. Since the original FSS has additive reconstruction, and the RPE has linear decoding, the resultant scheme has linear reconstruction. Thus, clients can remotely operate on the encoded output shares as in the original FSS scheme, and can decode the outcome $f(x)$ *after* it was reconstructed from the output shares. Thus, function and output shares can still be aggregated as required by FSS applications.

2.2.0.4 Threshold Encryption Schemes (TES) and Threshold Signatures

Lastly, we study the usefulness of our paradigm in the context of threshold cryptography [38, 70, 47, 56] where the secret key underlying some cryptographic task (e.g., encryption or signing), is used in a distributed manner. Namely, a set of parties mutually choose the secret key, where each party picks a random share and neither party (or a strict subset) can reconstruct the secret. Furthermore, each secret key share has a corresponding public key share. The latter shares are combined into the public key of the underlying object. Consequently, each party can encrypt or verify a signature as these operations are public, but decrypting (or, alternatively, signing) a message can be performed in a distributed manner by running a secure computation protocol. A notable example is an extension of the Diffie-Hellman key exchange protocol to the multiparty setting [38] that serves as a threshold public key encryption scheme for El Gamal [48].

Threshold encryption is a more involved computational object than FSS since it needs to preserve the secrecy of the secret key as well as the semantic security of the underlying encryption scheme. The scheme consists of a key generation protocol which generates the secret key shares in a distributed manner, an encryption algorithm which is similar to the encryption algorithm in standard encryption schemes, and a decryption protocol in which the parties use their secret key shares (from the key generation phase) to decrypt a ciphertext. Existing security definitions come in different flavours depending of whether security is game-based or simulation-based, and are typically specially-tailored to the specific underlying encryption scheme. We thus first provide a unified security definition that captures the security properties of any threshold encryption scheme. This definition – which might

be of independent interest – will later be used as the starting point for our definition of leakage-resilience for threshold encryption. Our security definition for threshold encryption is simulation-based against adaptive corruptions, but can be easily adapt to the static setting. It additionally captures the fact that the adversary can observe the decryption results of the honest parties – which is needed, for instance, to guarantee that security is preserved even when all parties learn the decryption for some plaintext – by giving the adversary access to a reveal oracle that discloses the decryption shares of a specific (valid) cipherext of the adversary's choice.[12]

Next, we extend this definition to obtain leakage-resilience by equipping the adversary with an additional leakage oracle. This oracle takes as input a leakage function, and returns its output on the key and decryption shares of the honest parties. Our leakage model protects the secret key shares of the honest parties (and of corrupted parties prior to their corruption), as well as the plaintext shares of honest parties prior to their decryption. This raises a similar challenge to the one described above in the context of FSS – we wish to protect the shares from leakage against *computationally unbounded adversaries*, while TES is computationally-secure. The assumption of separate leakage on ciphertext and key shares is motivated by the fact that these are physically separated in natural application scenarios. Indeed, different secret key shares are stored on different parties, and in many cases are also physically separated from ciphertexts. For example, a remote server (that *does not* know any secret key share) may generate a ciphertext c (this is possible because TES is a public key object). An adversary that is able to leak on the internal states of the parties will then obtain *separate* leakage on the ciphertext c, as well as on the key shares, and might also obtain several key shares in full. In this case, our definition guarantees that the plaintext, as well as the secret key shares that were not fully revealed, remain *information-theoretically* hidden.

To achieve information-theoretic security against leakage queries, we separate the secrecy of TES from the leakage-resilience property by splitting the adversary into three entities. The first entity is computationally-bounded and has access to an encryption oracle. The second entity is computationally-unbounded, and has access to a leakage oracle. We stress that these two entities must be kept separated, since the unbounded adversary can break the computational security of TES. Therefore, these adversarial entities do not share a state. Finally, the third adversarial entity takes the states of the former adversaries as input, and has access to corruption and reveal oracles. (The reveal oracle provides decryption shares of the honest parties on valid ciphertexts.) We note that the need to handle reveal queries makes the definition of leakage-resilient TES even more involved than that of leakage-resilient FSS.

Our definition captures both semi-honest and malicious adversaries, and can be adapted to signature schemes as well, where the property of indistinguishability of ciphertexts is replaced with unforgeability.

At a high-level, our TES construction works as follows. Key shares are generated by first generating TES key shares (by running the key generation protocol of the underlying TES), and then RPE-encoding each key share separately. Encryption is identical to the underlying TES. Finally, the decryption algorithm, given a key share and ciphertext, first RPE-decodes the key, then computes the decryption share using the decryption algorithm of the underlying TES, and then RPE-encodes the decryption share.

[12] We note that the actual formulation is more involved as it needs to eliminate chosen ciphertext attacks.

References

1 Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, João L. Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret sharing schemes for general access structures. In *CRYPTO*, pages 510–539, 2019.

2 Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC, Proceedings*, pages 474–495, 2009.

3 Marcin Andrychowicz, Stefan Dziembowski, and Sebastian Faust. Circuit compilers with $o(1/\log(n))$ leakage rate. In *EUROCRYPT, Proceedings, Part II*, pages 586–615, 2016.

4 Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. In *EUROCRYPT*, pages 593–622, 2019.

5 Marshall Ball, Dana Dachman-Soled, Siyao Guo, Tal Malkin, and Li-Yang Tan. Non-malleable codes for small-depth circuits. In *FOCS*, pages 826–837, 2018.

6 Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In *EUROCRYPT*, pages 881–908, 2016.

7 Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *CRYPTO*, volume 576, pages 420–432. Springer, 1991.

8 Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In *CRYPTO, Proceedings*, pages 531–561, 2018.

9 Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N. Rothblum. Program obfuscation with leaky hardware. In *ASIACRYPT, Proceedings*, pages 722–739, 2011.

10 Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage-tolerant interactive protocols. In *TCC, Proceedings*, pages 266–284, 2012.

11 Nir Bitansky, Dana Dachman-Soled, and Huijia Lin. Leakage-tolerant computation with input-independent preprocessing. In *CRYPTO, Proceedings, Part II*, pages 146–163, 2014.

12 Andrej Bogdanov, Yuval Ishai, Emanuele Viola, and Christopher Williamson. Bounded indistinguishability and the complexity of recovering secrets. In *CRYPTO*, pages 593–618, 2016.

13 Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. How to prove a secret: Zero-knowledge proofs on distributed data via fully linear PCPs. *IACR Cryptol. ePrint Arch.*, 2019:188, 2019. URL: `https://eprint.iacr.org/2019/188`.

14 Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In *CRYPTO, Proceedings, Part III*, pages 67–97, 2019.

15 Elette Boyle, Nishanth Chandran, Niv Gilboa, Divya Gupta, Yuval Ishai, Nishant Kumar, and Mayank Rathee. Function secret sharing for mixed-mode and fixed-point secure computation. In *EUROCRYPT*, pages 871–900, 2021.

16 Elette Boyle, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, and Amit Sahai. Secure computation against adaptive auxiliary information. In *CRYPTO*, pages 316–334, 2013.

17 Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *CCS*, pages 1292–1303, 2016.

18 Elette Boyle, Shafi Goldwasser, Abhishek Jain, and Yael Tauman Kalai. Multiparty computation secure against continual memory leakage. In *STOC, Proceedings*, pages 1235–1254, 2012.

19 Elette Boyle, Shafi Goldwasser, and Yael Tauman Kalai. Leakage-resilient coin tossing. In *DISC, Proceedings*, pages 181–196, 2011.

20 Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *EUROCRYPT, Proceedings*, pages 89–108, 2011.

21 Gabriel Bracha. An asynchronous $(n-1)/3$-resilient consensus protocol. In *PODC*, pages 154–162, 1984.

22 Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO, Proceedings*, pages 1–20, 2010.

23 Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510, 2010.

24 Zvika Brakerski and Gil Segev. Better security for deterministic public-key encryption: The auxiliary-input setting. In *CRYPTO, Proceedings*, pages 543–560, 2011.

25 Gianluca Brian, Antonio Faonio, and Daniele Venturi. Continuously non-malleable secret sharing for general access structures. In *TCC*, pages 211–232, 2019.

26 Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In *CRYPTO*, pages 90–104, 1997.

27 Ran Canetti, Sunoo Park, and Oxana Poburinnaya. Fully deniable interactive encryption. In *CRYPTO*, pages 807–835, 2020.

28 Nishanth Chandran, Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Adaptive extractors and their application to leakage resilient secret sharing. In *CRYPTO*, pages 595–624, 2021.

29 Eshan Chattopadhyay, Jesse Goodman, Vipul Goyal, Ashutosh Kumar, Xin Li, Raghu Meka, and David Zuckerman. Extractors and secret sharing against bounded collusion protocols. In *FOCS*, pages 1226–1242, 2020.

30 Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Black-box construction of a non-malleable encryption scheme from any semantically secure one. In *TCC*, pages 427–444, 2008.

31 Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. A black-box construction of non-malleable encryption from semantically secure encryption. *J. Cryptol.*, 31(1):172–201, 2018.

32 Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In *TCC, Proceedings*, pages 225–244, 2006.

33 Dana Dachman-Soled, Feng-Hao Liu, and Hong-Sheng Zhou. Leakage-resilient circuits revisited - optimal number of computing components without leak-free hardware. In *EUROCRYPT, Proceedings, Part II*, pages 131–158, 2015.

34 Ivan Damgård, Frédéric Dupuis, and Jesper Buus Nielsen. On the orthogonal vector problem and the feasibility of unconditionally secure leakage-resilient computation. In *ICITS, Proceedings*, pages 87–104, 2015.

35 Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In *SCN, Proceedings*, pages 121–137, 2010.

36 Scott E. Decatur, Oded Goldreich, and Dana Ron. A probabilistic error-correcting scheme. *IACR Cryptol. ePrint Arch.*, 1997:5, 1997.

37 Scott E. Decatur, Oded Goldreich, and Dana Ron. Computational sample complexity. *SIAM J. Comput.*, 29(3):854–879, 1999.

38 Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

39 Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *TCC, Proceedings*, pages 361–381, 2010.

40 Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520, 2010.

41 Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC, Proceedings*, pages 621–630, 2009.

42 Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In *TCC, Proceedings*, pages 207–224, 2006.

43 Stefan Dziembowski and Sebastian Faust. Leakage-resilient circuits without computational assumptions. In *TCC, Proceedings*, pages 230–247, 2012.

44 Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *FOCS, Proceedings*, pages 227–237, 2007.

45 Antonio Faonio and Daniele Venturi. Non-malleable secret sharing in the computational setting: Adaptive tampering, noisy-leakage resilience, and improved rate. In *CRYPTO*, pages 448–479, 2019.

46 Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In *EUROCRYPT, Proceedings*, pages 135–156, 2010.

47 Tore Kasper Frederiksen, Yehuda Lindell, Valery Osheter, and Benny Pinkas. Fast distributed RSA key generation for semi-honest and malicious adversaries. In *CRYPTO*, pages 331–361, 2018.

48 Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

49 Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In *CRYPTO, Proceedings*, pages 297–315, 2011.

50 Shafi Goldwasser and Guy N. Rothblum. Securing computation against continuous leakage. In *CRYPTO, Proceedings*, pages 59–79, 2010.

51 Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. In *FOCS*, pages 31–40, 2012.

52 Vipul Goyal, Yuval Ishai, Hemanta K. Maji, Amit Sahai, and Alexander A. Sherstov. Bounded-communication leakage resilience via parity-resilient circuits. In *FOCS*, pages 1–10, 2016.

53 Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In *STOC, Proceedings*, pages 685–698, 2018.

54 Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing for general access structures. In *CRYPTO, Proceedings, Part I*, pages 501–530, 2018.

55 Venkatesan Guruswami and Mary Wootters. Repairing Reed-Solomon codes. In *STOC, Proceedings*, pages 216–226, 2016.

56 Carmit Hazay, Gert Læssøe Mikkelsen, Tal Rabin, Tomas Toft, and Angelo Agatino Nicolosi. Efficient RSA key generation and threshold Paillier in the two-party setting. *J. Cryptol.*, 32(2):265–323, 2019.

57 Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, and Mor Weiss. ZK-PCPs from leakage-resilient secret sharing. In *ITC*, 2021.

58 Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, and Mor Weiss. Protecting distributed primitives against leakage: Equivocal secret sharing and more. *IACR Cryptol. ePrint Arch.*, 2022:497, 2022.

59 Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David A. Wagner. Private circuits II: keeping secrets in tamperable circuits. In *EUROCRYPT*, pages 308–327, 2006.

60 Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.

61 Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In *CRYPTO, Proceedings*, pages 41–58, 2010.

62 Yael Tauman Kalai and Leonid Reyzin. A survey of leakage-resilient cryptography. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 727–794. ACM, 2019.

63 Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, Sruthi Sekar, and Jenit Tomy. Locally reconstructable non-malleable secret sharing. *IACR Cryptol. ePrint Arch.*, 2021:657, 2021.

64 Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *SP*, pages 1–19, 2019.

65 Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO, Proceedings*, pages 104–113, 1996.

66 Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO, Proceedings*, pages 388–397, 1999.

67 Ashutosh Kumar, Raghu Meka, and Amit Sahai. Leakage-resilient secret sharing against colluding parties. In *FOCS*, pages 636–660, 2019.

68 Fuchun Lin, Mahdi Cheraghchi, Venkatesan Guruswami, Reihaneh Safavi-Naini, and Huaxiong Wang. Non-malleable secret sharing against affine tampering. *CoRR*, abs/1902.06195, 2019.

69 Fuchun Lin, Mahdi Cheraghchi, Venkatesan Guruswami, Reihaneh Safavi-Naini, and Huaxiong Wang. Leakage-resilient secret sharing in non-compartmentalized models. In *ITC*, pages 7:1–7:24, 2020.

70 Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In *CCS*, pages 1837–1854, 2018.

71 Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. Meltdown: Reading kernel memory from user space. In *USENIX Security*, pages 973–990, 2018.

72 Hemanta K. Maji, Hai H. Nguyen, Anat Paskin-Cherniavsky, Tom Suad, and Mingyuan Wang. Leakage-resilience of the shamir secret-sharing scheme against physical-bit leakages. In *EUROCRYPT*, pages 344–374, 2021.

73 Hemanta K. Maji, Anat Paskin-Cherniavsky, Tom Suad, and Mingyuan Wang. Constructing locally leakage-resilient linear secret-sharing schemes. In *CRYPTO*, pages 779–808, 2021.

74 Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.

75 Eric Miles. Iterated group products and leakage resilience against NC^1. In *ITCS*, pages 261–268, 2014.

76 Eric Miles and Emanuele Viola. Shielding circuits with groups. In *STOC*, pages 251–260, 2013.

77 Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO, Proceedings*, pages 18–35, 2009.

78 Jesper Buus Nielsen and Mark Simkin. Lower bounds for leakage-resilient secret sharing. In *EUROCRYPT, Proceedings, Part I*, pages 556–577, 2020.

79 Guy N. Rothblum. How to compute under $\mathcal{AC}^0$ leakage without secure hardware. In *CRYPTO, Proceedings*, pages 552–569, 2012.

80 Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014.

81 Akshayaram Srinivasan and Prashant Nalini Vasudevan. Leakage resilient secret sharing and applications. In *CRYPTO, Proceedings*, pages 480–509, 2019.

82 Ivan Tjuawinata and Chaoping Xing. Leakage-resilient secret sharing with constant share size. *CoRR*, abs/2105.03074, 2021. `arXiv:2105.03074`.

83 Kang Yang, Xiao Wang, and Jiang Zhang. More efficient MPC from improved triple generation and authenticated garbling. In *CCS*, pages 1627–1646. ACM, 2020.

A Related Work

There is a vast body of works on leakage-resilient cryptography. In this section we review the works which are most relevant to our model and results, in most cases only citing the first papers introducing the leakage model or LR construction. (See [62], and references therein, for a more detailed discussion of these models and various works in the field.) In the following, m denotes the number of parties, and t denotes the reconstruction threshold (in secret sharing schemes).

A.1 Leakage-Resilience for Data at Rest

Our leakage model is related to several leakage models considered in the past. In the context of secret sharing, the model most relevant to ours is that of [8], who also consider a computationally-unbounded adversary that can obtain the shares of an unauthorized set, as well as leakage computed separately on every other share. However, there are several notable differences. Specifically, in their model leakage is *non-adaptive* (whereas we protect against adaptive leakage), but can compute *any* function that is (1) sufficiently shrinking (i.e., the output is sufficiently shorter than the input) and (2) operates on each share separately, whereas our focus is on probing-resilience.

More generally, our model can be seen as a combination of the leakage model of [60] (that restricts the function class from which leakage functions can be chosen), and the "Only Computation Leaks" (OCL) model of [74] (which assumes that different memory components leak separately), though, as noted above, our constructions achieve the stronger OCL+ property [9]. We note that similar to standard OCL, our model is likewise motivated by scenarios in which physical separation of the shares, which are stored in different locations – e.g., on different remote servers – guarantees that leakage will indeed apply to each share separately. We note that the notion of protecting data at rest was studied in the past in the context of the bounded retrieval model [42, 32], bounded memory leakage [2], and several other models (see [62] and references therein). We elaborate more on these works, and their connection to our model, in Section A.3 below.

A.2 Leakage-resilient Secret Sharing

Most relevant to our work is the notion of Leakage-Resilient Secret-Sharing Schemes (LR-SSSs), which were first implicitly studied by Dziembowski and Pietrzak [44] (under the name "intrusion-resilient secret-sharing"), and explicitly (and independently) defined by [53, 8]. Following [53, 8], there has been a long line of works exploring various aspects of leakage-resilient secret-sharing [44, 35, 53, 54, 8, 81, 4, 67, 1, 81, 29, 69, 72, 73, 28]. These works consider different leakage models, as we now explain.

Local Leakage Model. The intrusion-resilient secret-sharing of [44] designed an n-out-of-n secret sharing scheme that resists adaptive leakage queries applied separately on each share, where each query is defined by an arbitrary polynomial-time computable function, and there are a-priori bounds on the total number of queries and leakage bits. Their scheme required interactive reconstruction. A subsequent work by Davi et al. [35] designed a 2-out-of-2 secret-sharing scheme with *non-interactive* reconstruction. (We note that [42] allowed leakage of full shares, but [35] do not.)

More recently, the works of [53, 8] independently (formally) introduced the notion of LR-SSSs, and studied leakage-resilience in the presence of non-adaptive local leakage, namely when the adversary can make a *single* leakage query that leaks from each share *independently* of the other shares. These works were motivated by different goals, and focus on different aspects of LR-SSSs. Specifically, Goyal and Kumar [53] used LR-SSSs as a stepping stone toward constructing non-malleable secret sharing, and designed m-party 2-out-of-m LR-SSSs. LR-SSSs for *arbitrary thresholds* were later constructed by Srinivasan and Vasudevan [81], who achieve constant rate and leakage rate, through a general compiler that employs an extractor and Shamir's secret sharing scheme as building blocks. In particular, their scheme does not have linear reconstruction. On the other hand, Benhamouda et al. [8] focused on analyzing leakage resilience of *standard* schemes – such as additive secret sharing, and Shamir

secret-sharing over large prime fields. Their motivation was the attacks of Guruswami and Wooters [55] on secret sharing over fields of characteristic 2.[13] Specifically, [8] proved that Shamir's scheme has constant leakage rate for thresholds $t = m - o(m)$. This was recently improved by Maji et al. [73] who extended the analysis of [8] to arbitrary thresholds (by analyzing random linear codes), with the caveat that they instantiate Shamir's scheme over *random* (as opposed to fixed) evaluation points.

Limitations of standard secret-sharing schemes. The aforementioned works show some limitations to the leakage-resilience properties of existing schemes. As noted above, Shamir's secret sharing scheme over fields of characteristic 2 does not resist leakage of even *a single bit* from each share [55], and similar insecurities hold more generally for fields of small characteristic (not necessarily 2) [8]. In terms of the reconstruction threshold t in m-party schemes, the analysis of Benhamouda et al. [8] holds only for certain parameter regimes (either $t \geq 0.85m$ when leaking a constant number of bits from each share, or $t = m - \sqrt[4]{m}$ when leaking a quarter of the bits). There are also known lower bounds on the values of t for which Shamir's scheme resists leakage (even of a single bit) [78], which can be circumvented by instantiating Shamir's scheme using random evaluation points [73].

Joint Leakage Model. Kumar et al. [67] extend the works of [53, 54, 8] in a different direction, focusing on a stronger leakage resilience guarantee, namely against *adaptive* leakage queries that depend *jointly* on shares of multiple parties. Specifically, they introduce the Bounded Collusion Protocol (BCP) leakage model, in which the adversary can adaptively leak an a-priori bounded number of bits, where each leakage bit is computed jointly from the shares of a subset of at most p parties, for an a-priori bound p. Roughly, BCP leakage models the leakage as a communication protocol run by the leaking adversary, where in each round the adversary chooses a set $\mathcal{P}$ of at most p parties that compute a "message" based on their shares and the previous protocol messages. The leakage consists of the transcript of this protocol. The schemes of [67] support any threshold, but can only withstand joint leakage of $p = O(\log m)$ shares.[14] This result was further refined recently by Chattopadhyay et al. [29], who design LR-SSSs withstanding larger collusion bounds of up to $p = t/\log t$ parties, They also consider a weaker leakage model in which the (adaptive) leakage queries are restricted to using *disjoint* sets of parties, namely, no share can be accessed by two leakage queries. In this restricted leakage model, they obtain leakage-resilience for any collusion bound $p \leq t$, with $O(1/m)$ rate and leakage rate. This was recently improved by Chandran et al. [28], who obtain constant rate and leakage rate in this restricted leakage model.

Global Leakage Model. Another line of works [12, 69] consider *global* leakage which is computed jointly on *all* shares, but restrict the class of permissible leakage functions. Bogdanov et al. [12] proved that Shamir's scheme over finite fields of characteristic 2 is leakage-resilient against constant-depth polynomial-sized circuits (i.e., AC^0 circuits), and sign polynomials of degree 2,[15] so long as the threshold $t = \omega(\mathsf{polylog}(m))$. The work of [69] designed LR-SSSs that are secure against affine leakage functions over $\mathbb{F}_2$. These works are

[13] We note that [8] extended this result by showing attacks on secret sharing over fields of small characteristic (not necessarily 2).

[14] We remark that the adversary can choose a different subset of $O(\log m)$ shares for each adaptive query, as long as the total number of leaked bits is bounded.

[15] A sign polynomial is a function $f(x) = \mathsf{sgn}(p(x))$ for some polynomial $p(\cdot)$, where sgn is the sign function.

somewhat orthogonal to ours since they allow for global leakage (i.e., on all shares) but obtain a very low leakage rate, which, in particular, does not even allow leaking a single bit (on average) from each share.

All the works mentioned above, except for [67, 29, 28, 69], only support non-adaptive queries, and the constructions secure against adaptive leakage queries do not admit efficient equivocation or linear reconstruction. In Table 1 we provide a curated list of works that are closest to our work, and identify the properties satisfied by the leakage-resilient schemes they construct.

A.3 Leakage-Resilient Memory and Storage

There is a vast body of works on protecting memory from leakage in different models, such as the bounded-retrieval model [42, 32], bounded memory leakage [2], auxiliary-input memory leakage [41], and continual memory leakage [23, 40]. These models can be seen as protecting against leakage of data at rest (e.g., protecting the secret key while stored in memory), but differ significantly from our model in the security guarantees (they permit information leakage on the secret state as long as the security of the scheme using the key is not compromised, whereas our goal is to hide the secret), the security quality (usually computational, whereas we protect against computationally-unbounded leaking adversaries), and the permitted leakage functions (usually arbitrary shrinking functions computable in polynomial time, whereas we focus on probing leakage). The model of leakage-resilient storage [35] – which aims to protect (properly encoded) storage – is similarly a model of leakage-resilience for data at rest. Similar to our model, the goal is to prevent *any* information leakage on the stored secret. Constructions in this model usually rely on complex primitives such as extractors, which do not result in linear reconstruction procedures.

A.4 Leakage-Resilient Distributed Primitives

Boyle et al. [19] introduce and study the notions of leakage-resilient coin-tossing and leakage-resilient VSS. Similar to our work, their constructions are secure against a computationally unbounded, malicious adversary that corrupts t parties, as well as obtains adaptive "local" leakage, namely the leakage function applies to each party separately. However, there are a few notable differences between our leakage model and results, and those of [19], which make them incomparable. First, their corruption model is *non-adaptive*, and this is inherent to their results since they delegate certain computations to committees (a-la [21]), whereas we allow for adaptive corruptions. Unlike our schemes, their constructions rely on (2-source and multi-source) randomness extractors with an additional robustness guarantee,[16] and therefore do not preserve algebraic properties of the original primitives (e.g., linear reconstruction). However, they allow for general leakage of up to ℓ information bits throughout the entire lifetime of the system, where ℓ is a constant fraction of the view size, whereas we focus on probing resilience for data at rest and can handle a constant fraction of leaked bits from each share. There are also works obtaining zero-knowledge protocols [49, 10] as well as MACs, commitment schemes, and OT [10] with a weaker leakage-resilience guarantee known as *leakage-tolerance*, in which the ideal-world adversary obtains leakage on the witness/secret. Thus, these works guarantee graceful degradation of security – instead of full security – against leakage.

[16] We note that they do have a VSS protocol that does not use extractors, but it only achieves a weaker LR guarantee in which the secret retains some min-entropy given the leakage.

A.5 Leakage-Resilient MPC and General Computations

Following the works of [60, 74], there has been a long line of works on protecting general computations against leakage (see [62] and references therein). These constructions usually consider a continuous leakage model in which the adversary observes repeated executions of the computation on inputs of its choice. Due to their generality, these constructions are less efficient (incurring polynomial blowups), complex, and in particular do not preserve the structure of the original computation (e.g., linear reconstruction). There are also works on protecting MPC protocols from leakage, either with full security against leakage [18, 8], or allowing some leakage in the ideal world [10, 16, 11]. The latter protocols achieve only a weaker leakage-resilience guarantee (since the ideal-world simulator also obtains leakage), whereas the former either achieve only computational security [18], or a poor leakage rate [8].

A.6 Tampering-Resilience

The notion of tampering-resilience for secret-sharing, also known as *non-malleable* secret-sharing [53], has received a lot of attention lately [53, 54, 81, 4, 45, 1, 25, 67, 68, 29, 28, 63]. Roughly speaking, non-malleable secret-sharing guarantees that even if an adversary tampers with all the shares, reconstruction either outputs the original secret, or a *completely independent secret* which is unrelated to the original secret. This should be contrasted with our notion of tampering-resilience, which guarantees that *the original secret will be reconstructed*. On the other hand, non-malleable secret-sharing can withstand higher tampering rates, in particular ones exceeding the error-correction bound for which unique recovery of the underlying secret is possible.

We note that another – very different – notion of tampering-resilience appeared in the literature in the context of protecting *secrecy* (e.g., of the input or internal state of a cryptographic scheme) in the presence of a tampering adversary. These works, originating from [59], are unrelated to our notion of tampering since they aim at protecting *secrecy*, whereas our goal is to maintain *correctness*.

Maliciously Circuit-Private FHE from Information-Theoretic Principles

Nico Döttling
Helmholtz Center for Information Security (CISPA), Saarbrücken, Germany

Jesko Dujmovic
Helmholtz Center for Information Security (CISPA), Saarbrücken, Germany
Universität des Saarlandes, Saarbrücken, Germany

Abstract

Fully homomorphic encryption (FHE) allows arbitrary computations on encrypted data. The standard security requirement, IND-CPA security, ensures that the encrypted data remain private. However, it does not guarantee privacy for the computation performed on the encrypted data. Statistical circuit privacy offers a strong privacy guarantee for the computation process, namely that a homomorphically evaluated ciphertext does not leak any information on how the result of the computation was obtained. Malicious statistical circuit privacy requires this to hold even for maliciously generated keys and ciphertexts. Ostrovsky, Paskin and Paskin (CRYPTO 2014) constructed an FHE scheme achieving malicious statistical circuit privacy.

Their construction, however, makes non-black-box use of a specific underlying FHE scheme, resulting in a circuit-private scheme with inherently high overhead.

This work presents a conceptually different construction of maliciously circuit-private FHE from simple information-theoretical principles. Furthermore, our construction only makes black-box use of the underlying FHE scheme, opening the possibility of achieving practically efficient schemes. Finally, in contrast to the OPP scheme in our scheme, pre- and post-homomorphic ciphertexts are syntactically the same, enabling new applications in multi-hop settings.

2012 ACM Subject Classification Security and privacy → Public key encryption; Security and privacy → Information-theoretic techniques

Keywords and phrases Fully Homomorphic Encryption, FHE, Homomorphic Encryption, Oblivious Transfer, Malicious Statistical Circuit Privacy, Multi-Hop, Information Theory, Cryptography

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.4

Related Version *Full Version*: `https://eprint.iacr.org/2022/495`

Funding *Nico Döttling*: This work is partially funded by the Helmholtz Association within the project "Trustworthy Federated Data Analytic" (TFDA) (funding number ZT-I-OO1 4).
Jesko Dujmovic: This work is partially funded by the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (funding number ZT-I-OO1 4).

1 Introduction

Fully Homomorphic Encryption

Fully homomorphic encryption (FHE) [18] has caused a paradigm shift in achieving round and communication efficient secure computation. FHE allows an untrusted server to publicly evaluate any function over encrypted data without the help of a secret key. FHE has become a tremendous success story in the last ten years, with constructions from increasingly weaker assumptions and achieving better efficiency [29, 11, 10, 21, 12, 2]. By now (levelled) FHE is even considered a standard cryptographic primitive, which can be based on the standard Learning with Errors (LWE) problem [27] with polynomial modulus-to-noise ratio. An

important feature of FHE is ciphertext compactness, which means that homomorphically evaluated ciphertexts do not grow with the size of the evaluated circuit. Furthermore, a recent line of work [16, 9, 19] has succeeded in achieving FHE with essentially optimal rate, i.e. for sufficiently long messages, the size of ciphertexts is only an additive amount larger than the encrypted plaintext. Thus, we say that these schemes achieve (or approach) plaintext-size to ciphertext-size ratio 1; we call this a rate-1 scheme for short.

Circuit-Private FHE

The standard security notion of FHE, IND-CPA security, guarantees the privacy of encrypted data. But it does not guarantee any concrete security for the evaluator beyond the guarantee that a ciphertext can convey only a limited amount of information about the computation from which it resulted due to compactness. In a *circuit-private* FHE scheme, an evaluator holding a circuit $\mathcal{C}$ has the following security guarantee. Assume that c is a ciphertext encrypting a message x, and assume the evaluator homomorphically evaluates $\mathcal{C}$ on c, resulting in a ciphertext d. The evaluator has the guarantee that d encrypts the output $\mathcal{C}(x)$ of the homomorphic computation but does not convey any further information about the circuit $\mathcal{C}$. We say that an FHE scheme satisfies semi-honest circuit privacy if this property holds for honestly generated keys and ciphertexts. Gentry [18] describes a simple *drowning-based* mechanism to achieve semi-honest circuit privacy (which typically leads to poor parameters for the underlying hardness assumption). Later works [17, 8] provided transformations adding semi-honest circuit privacy with very little overhead and without parameter deterioration.

In essence, circuit privacy can be seen as a property of a specific homomorphic evaluation algorithm. A circuit-private evaluation algorithm must be randomized, while non-circuit private evaluation algorithms can be deterministic.

Ostrovsky, Paskin and Paskin [26] provided the first *maliciously circuit-private FHE scheme*. This scheme was later generalized to the *multi-key setting* by Chongchitmate and Ostrovsky [13]. Malicious circuit privacy requires that the above property holds even for maliciously generated keys and ciphertexts. On a technical level, the notion of malicious statistical circuit privacy requires the existence of an (unbounded) ciphertext extractor, which extracts a plaintext from a given pair of public key and ciphertext, and a simulator which, given an output $\mathcal{C}(x)$ simulates a homomorphically evaluated ciphertext encrypting $\mathcal{C}(x)$. In the presence of a common reference string (CRS), the well-formedness of both keys and ciphertexts can be enforced by requiring keys and ciphertexts to include non-interactive zero-knowledge proofs of knowledge (NIZKPoK) of their well-formedness, such that plaintexts can be extracted using the knowledge extractor for the NIZKPoK.

However, [26] provide a maliciously circuit-private FHE scheme in the plain model (i.e. without CRS) and guarantee *statistical circuit privacy*. The main idea of their construction is to leverage a *conditional disclosure of secrets* protocol [1] instead of NIZK proofs. That is, an input ciphertext c contains additional *encrypted well-formedness information* γ, which they use in the maliciously circuit-private evaluation algorithm to enforce that the output ciphertext d is independent of the circuit $\mathcal{C}$ if c was not well-formed. This well-formedness information γ is *consumed* by the maliciously circuit-private evaluation algorithm, and the output ciphertext d contains no such well-formedness information. Hence, d cannot be used as input for the maliciously circuit-private evaluation algorithm but can still serve as input for standard (non-maliciously-circuit-private) homomorphic evaluation.

We outline the main ideas of [26] in the appendix of this paper's full version.

Multi-Hop FHE

We say that an FHE scheme is *single-hop* if ciphertexts resulting from a homomorphic evaluation cannot be used as input ciphertexts for further homomorphic evaluations. We refer to FHE schemes where homomorphically computed ciphertexts can again be used as input ciphertexts for further homomorphic computation as *multi-hop* (a notion introduced by [20]).

The basic scheme of [26] is only single-hop, but they show how to modify it to support multi-hop (non-maliciously-circuit-private) homomorphic evaluation. By the discussion in the last paragraph, this means that in the multi-hop setting, circuit privacy is only guaranteed if all evaluators are honest. Furthermore, it seems hard to establish that their techniques could yield a scheme that satisfies malicious circuit privacy even if some evaluators are malicious. That is, consider a scenario in the 2-hop setting, where we have a malicious key-generator and encryptor as well as a malicious first evaluator E_1 and an honest second evaluator E_2. The basic issue is that while the techniques of [26] enforce that both keys and ciphertexts produced by the encryptor are well-formed, they cannot provide a similar guarantee for ciphertexts produced by the first evaluator E_1. Consequently, E_1 may pass an arbitrarily malformed ciphertext to E_2. Then all circuit privacy guarantees for E_2 are lost.

1.1 Our Results

This work provides a conceptually simple construction of a fully homomorphic encryption scheme with malicious circuit privacy. As a bonus, ciphertexts generated by the encryption algorithm and ciphertexts produced by the homomorphic evaluation procedure are syntactically the same. This means our scheme supports malicious circuit privacy even if the input ciphertexts themselves are potentially the result of a homomorphic evaluation. Our construction significantly departs from the blueprint of [26]. On a technical level, our constructions build on and leverage rate-1 FHE schemes [19, 9], but also inherit the rate-1 property. As we will explain below, our construction equips a rate-1 FHE scheme with a novel evaluation algorithm but otherwise leave the underlying construction unmodified and is black-box in the underlying rate-1 FHE scheme. This means, in particular, that our maliciously circuit-private evaluation algorithm also supports input-ciphertexts which themselves are the result of homomorphic evaluations. We call such a scheme a *multi-hop-secure* maliciously circuit-private FHE scheme. Note that this property solely comes down to the type of input-ciphertext supported by the maliciously circuit-private homomorphic evaluation algorithm but otherwise leaves the definition of malicious statistical circuit-privacy unchanged.

Compared to the construction of [26], our construction can be considered a more direct way of achieving malicious circuit privacy.

1.2 Applications

We will briefly discuss two related applications we envision as use-cases for our multi-hop-secure MCP-FHE scheme.

- **Encrypted Databases with privacy for Write-Queries:** Consider a scenario where a cloud server holds a database encrypted under an FHE scheme. The owner of the database, who generated the FHE keys goes offline, but several mutually mistrusting workers perform homomorphic computations on the database, and these computations involve sensitive data held by the workers. While the IND-CPA security of the FHE

scheme protects the privacy of the database, the privacy of the workers' operations is ensured by the circuit privacy of the FHE scheme. However, if a malicious database owner and several malicious workers collude against a worker, then single-hop circuit privacy does not offer any guarantee to this worker. Consequently, to protect the privacy of this worker's operation, we need a multi-hop-secure MCP-FHE scheme.

- **Federated Learning with Model-Privacy:** In the machine-learing subfield of federated learning [25], the training data is distributed among several (physically) separated servers. A central server, coordinating a learning process sends partially-trained models to the training servers, who compute model-updates using their local training data and send the updates back to the central server. The purpose of this separation of the training data is two-fold. First, by ware-housing the training-data locally with the servers and only communicating (relatively small) model updates, an enormous amount of bandwidth can be saved which would otherwise be needed to transfer vast quantities of training data. Second, and maybe more importantly, each server is in control of the amount of outgoing data and therefore has the guarantee that his local data cannot be retrieved entirely by the central server.

  Now consider a scenario where a model-owner, in possession of a partially trained model, wants the training servers to compute updates on his model. However, the model may contain sensitive data which should not be leaked to the training servers. Consequently, encrypting the model under an FHE scheme protects the privacy of this model. To protect the privacy of the training servers' training data, we need to require circuit privacy. However, if the model owner colludes with some of the training servers, standard malicious circuit privacy is insufficient to protect the privacy of any of the training servers training-data. By using a multi-hop-secure MCP-FHE scheme, the training servers have the guarantee that even if the model owner colludes with other users, they will not learn more about this users data than they would have in a plain federated learning protocol (i.e. without the additional layer of homomorphic encryption).

1.3 Technical Outline of our Approach

Our construction significantly departs from the OPP approach [26]. On a very high level, our approach is to augment a given FHE scheme to *natively* support malicious function privacy for a very basic class of functions, namely affine functions, without resorting to tools which enforce the well-formedness of input ciphertexts. We will then be able to amplify this to the class of all functions by relying on the machinery of affine randomized encodings [22, 5], aka information-theoretically secure garbled circuits.

Statistically Sender-Private OT from High-Rate OT

We will first describe how a *high-rate* FHE scheme can be augmented to support malicious function privacy for affine functions. As described above, such high-rate FHE schemes were recently constructed by Gentry and Halevi [19] and Brakerski et al. [9].

Our starting point is a recent work of Badrinarayanan et al [6], who observed that high rate (sender-input to sender-message ratio) can be leveraged to achieve statistical sender privacy. This is similar in spirit to the work of [14], who build an OT protocol in the bounded-quantum-storage model. In more detail, [6] observed that any string-OT with *high rate* (i.e. greater than 1/2) yields a statistically sender private OT protocol (called weak OT in [6]) via a simple information-theoretic transformation. Specifically, the high-rate OT is used to transfer two random strings r_0 and r_1. But since the OT has high rate,

the OT-sender message ot_2 is shorter than the concatenation of the two random strings. Consequently, one can argue that one of the two strings r_0 and r_1 must have high conditional min-entropy given ot_2. Thus, using a suitable randomness extractor Ext, one can derive two masks $k_0 = \mathsf{Ext}(r_0, s_0)$ and $k_1 = \mathsf{Ext}(r_1, s_1)$ (for two seeds s_0 and s_1) and argue that either k_0 or k_1 must be statistically close to uniform conditioned on ot_2. The sender then also sends $(m_0 \oplus k_0, m_1 \oplus k_1)$, i.e. the actual messages blinded with the corresponding mask. An honest receiver will then be able to recover the m_b corresponding to his choice-bit b.

Note that this argument did not assume the well-formedness of the OT-sender message ot_1[1]. So consequently, no matter how malformed ot_1 is, the message ot_2 *must lose information* about either r_0 or r_1, and consequently one of the masks k_0, k_1 is uniformly random from the view of the receiver.

While the high-level idea of the proof and the statement of the corresponding theorem in [6] is true, there is a subtle loophole in their proof, which we will briefly explain here. To establish malicious statistical sender privacy, one needs to show the existence of an (unbounded) extractor which extracts the receiver's choice bit from the ot_1 message. In [6], this is achieved via the following argument: For a fixed ot_2 it holds that $H_\infty(r_0, r_1 | \mathsf{OT}_2(ot_1, r_0, r_1) = ot_2) \geq n$, thus it must either hold that $H_\infty(r_0 | \mathsf{OT}_2(ot_1, r_0, r_1) = ot_2) > n/2$ or $H_\infty(r_1 | \mathsf{OT}_2(ot_1, r_0, r_1) = ot_2) > n/2$. The unbounded extractor then computes both $h_b = H_\infty(r_b | \mathsf{OT}_2(ot_1, r_0, r_1) = ot_2)$ for $b \in \{0, 1\}$, and sets the extracted bit b^* to 0 if $h_0 < h_1$, otherwise to 1.

This reasoning assumes that conditional min-entropy obeys a chain-rule, i.e. the conditional min-entropy of (r_0, r_1) must split into the conditional min-entropies of r_0 and r_1. However, in general this is not the case. There are (contrived) choices of the "leakage function" $\mathsf{OT}_2(ot_1, \cdot, \cdot)$, for which even though $H_\infty(r_0, r_1 | \mathsf{OT}_2(ot_1, r_0, r_1) = ot_2) > n$, it holds that

$$H_\infty(r_0 | \mathsf{OT}_2(ot_1, r_0, r_1) = ot_2) = H_\infty(r_1 | \mathsf{OT}_2(ot_1, r_0, r_1) = ot_2) \approx 1,$$

i.e. even (r_0, r_1) have n bits of min-entropy, each of them individually only has a single bit of min-entropy[2].

Essentially, the problem is that it might depend on (r_0, r_1) which one of r_0 or r_1 is leaked by $\mathsf{OT}_2(ot_1, r_0, r_1)$, i.e. the choice of the bit b is not necessarily fixed by the function $\mathsf{OT}_2(ot_1, \cdot, \cdot)$ as implicitly assumed in the above argument. In other words, the function $\mathsf{OT}_2(ot_1, \cdot, \cdot)$ does not fix a choice bit b, but rather a *distribution of choice-bits* $b(r_0, r_1)$ which may depend on r_0, r_1 in arbitrary ways.

Consequently, a more involved extraction strategy is required to make the proof rigorous. This can indeed be achieved by resorting to the *min-entropy splitting lemma* of [14]. In essence, translated to our context, this lemma states that for every leakage function $\mathsf{OT}_2(ot_1, \cdot, \cdot)$ there does exist an explicit random variable $b = b(r_0, r_1)$ such that $H_\infty(r_b | \mathsf{OT}_2(ot_1, r_0, r_1) = ot_2, b) > n/2 - 1$[3].

Thus, we can adapt the extractor of [6] to extract based on the conditional min-entropies $H_\infty(r_0 | \mathsf{OT}_2(ot_1, r_0, r_1) = ot_2, b = 0)$ and
$H_\infty(r_1 | \mathsf{OT}_2(ot_1, r_0, r_1) = ot_2, b = 1)$ and make the proof strategy of [6] work.

---

[1] Indeed, we haven't even mentioned it yet.

[2] Example: If first bit of r_0 is 0, leak last $n - 1$ bits of r_0, otherwise leak last $n - 1$ bits of r_1. See also [24, 28].

[3] The actual statement holds for smooth min-entropy, but we omit this somewhat technical detail for the sake of this outline.

FHE with Statistical Function Privacy for Affine Functions

Our core-observation is that this very same approach also works if we replace the high-rate OT by a high-rate FHE scheme. As explained above, such FHE schemes with a rate approaching 1 were recently constructed in [19] and [9].

We remark that these schemes have two different ciphertext types. Type 1 ciphertexts are *decompressed* and allow for homomorphic operations, but these ciphertexts have a poor rate, as each ciphertext encrypts (say) just a single bit[4]. Type 2 ciphertexts are in a *compressed format*, and each ciphertext encrypts say ℓ bits, and these ciphertexts have a rate approaching 1, but do not support homomorphic computations. These have a public compression procedure, which takes a vector of ℓ type 1 ciphertexts and produces a single type 2 ciphertext. Likewise, there is a public decompression procedure which takes a single type 2 ciphertext and returns a vector of ℓ type 1 ciphertexts. We remark that compressing type 1 into type 2 ciphertexts is fairly efficient, but decompressing type 2 into type 1 ciphertexts involves a rather expensive bootstrapping operation in current schemes [19, 9].

In essence, we will harness the compress operation to *lose information* about strings which should remain private. Specifically, assume we have such a compressible FHE scheme Π. Now let $c = \mathsf{Enc}(\mathsf{pk}, b)$ be a ciphertext encrypting a bit b under Π. We obtain malicious statistical function privacy for affine functions via the following evaluation procedure, which mimics an oblivious transfer in Π. The evaluator chooses two uniformly random strings $r_0, r_1 \in \{0,1\}^\ell$ and evaluates the affine function $f(x) = x \cdot r_1 + (1-x) \cdot r_0$ on c, obtaining an encryption of $c' = \mathsf{Enc}(f(b))$. The ciphertext c' is of type 1 and has thus low rate. The evaluator now compresses c' into a high-rate type 2 ciphertext and immediately decompresses it into a type 1 ciphertext d, which is an encryption of r_b. As above, the evaluator now chooses two extractor seeds s_0 and s_1 and computes $v_0 = m_0 \oplus \mathsf{Ext}(k_0, s_0)$ and $v_1 = m_1 \oplus \mathsf{Ext}(k_1, s_1)$. Finally, It homomorphically evaluates the function $g(x,y) = (\mathsf{Ext}(y, s_1) \oplus v_1) \cdot x + (\mathsf{Ext}(y, s_0) \oplus v_0) \cdot (1-x)$ on the ciphertexts c and d, obtaining an encryption e of

$$\begin{aligned} g(b, r_b) &= (\mathsf{Ext}(r_b, s_1) \oplus \mathsf{Ext}(r_1, s_1) \oplus m_1) \cdot b \\ &\quad + (\mathsf{Ext}(r_b, s_0) \oplus \mathsf{Ext}(r_0, s_0) \oplus m_0)(1-b) \\ &= m_b, \end{aligned}$$

and the ciphertext e is the output of the homomorphic evaluation.

Thus, correctness follows from the derivation above. To argue statistical function privacy, we argue analogously as in the last paragraph. Namely, even if both the public key and the ciphertext c are arbitrarily malformed, we observe that when we compress c' into a type 2 ciphertext, call it $\hat{c}$, then since $\hat{c}$ is high-rate, it cannot fully determine both r_0 and r_1. Consequently, as in the argument above, either r_0 or r_1 must have high conditional min-entropy given $\hat{c}$[5]. Since d is computed from $\hat{c}$, the same holds for d, i.e. conditioned on d either r_0 or r_1 has high min-entropy. Consequently, by the extraction property of Ext either v_0 or v_1 is statistically close to uniform conditioned on d. Thus, e does not depend on both m_0 and m_1. To make the argument formal, we can argue as above that a bit b can be extracted from the ciphertext c (via an unbounded extractor) and that the output ciphertext e can be simulated given only m_b.

[4] In both [19] and [9] the ciphertexts in this mode are essentially GSW ciphertexts [21]

[5] Where the same caveat as above applies, i.e. we need to condition on an additional *spoiling bit* b.

Note that our construction makes no additional non-black-box of underlying cryptographic primitives beyond whatever the underlying FHE scheme does. That is, given the current high-rate FHE constructions [19, 9] the only operation in the above construction which needs to do any heavy lifting is the decompression step, which in these constructions involves a bootstrapping operation.

We remark, however, that even though bootstrapping involves making non-black-box use of the decryption circuit of the underlying FHE scheme. This non-black-box use typically comes to just performing a *rounding operation* homomorphically. Furthermore, it is conceivable that there might exist construction of high-rate FHE schemes which deviate from the blueprint of [19, 9] and do not rely on bootstrapping to achieve high rate.

Malicious Statistical Circuit Privacy for NC1 Circuits

We will now outline how malicious statistical circuit privacy for affine functions can be amplified to malicious statistical circuit privacy for NC1 circuits. The go-to tool to achieve this are decomposable affine randomized encodings (DARE), also known as garbled circuits. A garbling scheme allows us to encode a computation into an affine and a non-affine part. For any input it holds that the output of the affine part together with the non-affine part does not leak more than the result of this computation on this input. Information-theoretically DAREs are known for NC1 circuits (i.e. circuits of logarithmic depth) [23, 22, 5]. Randomized encodings have, e.g. been used to bootstrap KDM security for affine functions to KDM security for bounded-size circuits [3].

We make use of DAREs/GCs as follows, starting with an FHE scheme with malicious function privacy for affine functions as described in the previous paragraph. Assume that the evaluator wants to homomorphically evaluate an NC1 circuit $\mathcal{C}$ on a potentially maliciously generated input ciphertext c. First, the evaluator computes a randomized encoding of $\mathcal{C}$ consisting of an affine part T and a non-affine part $\tilde{\mathcal{C}}$. Then, it evaluates the affine function T on the ciphertext c using the maliciously function private evaluation procedure for affine functions, resulting in a ciphertext d. Finally, it evaluates the non-affine part $\tilde{\mathcal{C}}$ on d, resulting in an output ciphertext e. Correctness follows immediately from the correctness of the FHE scheme and the DARE. To argue malicious circuit privacy, first note that by the malicious function privacy for affine functions, the ciphertext d does not leak more than $T(x)$ (where x is the value which can be extracted from c) about T. Consequently, it holds that e does not leak more than $T(x)$ and $\tilde{\mathcal{C}}$ about $\mathcal{C}$, which by the security of the DARE scheme does not leak more than $\mathcal{C}(x)$.

We remark that in our construction the output ciphertext e potentially leaks the same information about the circuit $\mathcal{C}$ that $T(x)$ and $\tilde{C}$, i.e. essentially the size of $\mathcal{C}$. This is somewhat in contrast to the construction of [26], which ensures that no information about the evaluator's circuit is leaked. Whether leaking the size of the evaluator's circuit is inherent in multi-hop-secure MCP-FHE remains an (in our opinion interesting) open problem.

Malicious Statistical Circuit Privacy for all Circuits

We will briefly outline how the above techniques can be leveraged to handle arbitrary polynomial depth circuits. To achieve this, we will resort to an idea of Kilian [23]. Specifically, given a polynomial-depth circuit $\mathcal{C}$, we will slice $\mathcal{C}$ into layers $\mathcal{C}_1, \ldots, \mathcal{C}_k$ such that each $\mathcal{C}_i$ is an NC1 circuit and $\mathcal{C} = \mathcal{C}_k \circ \cdots \circ \mathcal{C}_1$ (i.e. we can evaluate $\mathcal{C}$ by sequentially evaluating the $\mathcal{C}_i$). The circuits $\mathcal{C}_i$ can now be evaluated using the techniques described in the previous section. However, this basic idea has an issue as the intermediate outputs of the $\mathcal{C}_i$ are not

protected and may therefore leak information about the $\mathcal{C}_i$ and therefore $\mathcal{C}$. To deal with this issue, we will replace the circuits $\mathcal{C}_i$ by circuits D_i which *encrypt their output wires* using a one-time pad. Specifically, the circuit D_1 first computes $\mathcal{C}_1$, but xors a one-time pad K_1 on the output, i.e. $D_1(x) = \mathcal{C}_1(x) \oplus K_1$. The circuit D_2 first decrypts its input using the key K_1 and encrypts its output using a key K_2, i.e. $D_2(x) = \mathcal{C}_2(x \oplus K_1) \oplus K_2$. We continue in the same fashion, until we reach D_k which computes $D_k(x) = \mathcal{C}_k(x \oplus K_{k-1})$. By the security of the one-time pad, the outputs of the D_i leak no information about the outputs of the $\mathcal{C}_i$.

We will further show that if one is willing to settle for computational rather than statistical circuit privacy, then the transformation described in the previous paragraph can be implemented using computational garbled circuits, which means that the most expensive step, the function private evaluation of the affine function, only needs to be performed once. In this setting, some care has to be taking in the security proof as our input-extractor is unbounded but security of the garbled circuits only holds computationally. However, this issue can be dealt with using a standard trick which moves the information obtained by the unbounded extractor into non-uniform advice, which is provided to the non-uniform reduction against the garbling scheme.

This concludes the overview.

Roadmap

In Section 2 we show how to turn any high-rate FHE into one, which allows for circuit private evaluation of affine functions. We use this in Section 3 to build a circuit private scheme for NC1, which we extend to arbitrary circuits in Section 4. We cover the preliminaries in Appendix A.

For more information see the full version of the paper.

2 OT from High-Rate LHE

Here we reiterate the statistical sender private OT of [6] with slight modifications in notation and sender-privacy proof. It transforms a high-rate linearly homomorphic encryption scheme (LHE) into a statistically sender private OT.

2.1 Construction of [6]

Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a high-rate LHE scheme where the messages are vectors over $\{0,1\}$. We will use the following circuit $\mathcal{C}$ where strings r_0 and r_1 are hard-wired into the circuit, and one of them is selected according to input bit b. Notice, this circuit is a linear function over $\{0,1\}$.

Circuit $\mathcal{C}[r_0, r_1](b)$:
- output r_b

Now follows the construction. In this construction n is the size of the messages m_0, m_1 and the parameter m is dependent on λ but can be chosen arbitrarily large.

$\mathsf{OT}_1(1^\lambda, b)$:
- Generate keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
- Let $c \leftarrow \mathsf{Enc}(\mathsf{pk}, b)$
- return (pk, c)

$\mathsf{OT}_2(1^\lambda, ot_1 = (\mathsf{pk}, c), m_0, m_1)$:

- Choose $s_0, s_1 \leftarrow_\$ \{0,1\}^m$ uniformly at random
- Choose $r_0, r_1 \leftarrow_\$ \{0,1\}^m$ uniformly at random
- Hardwire r_0, r_1 into $\mathcal{C}[r_0, r_1]$ to get circuit $\mathcal{C}'$
- return s_0, s_1, $\mathsf{Ext}(s_0, r_0) \oplus m_0$, $\mathsf{Ext}(s_1, r_1) \oplus m_1$, e, and $\mathsf{Eval}(\mathcal{C}', c)$

In the output, c is an encryption of b and $\mathsf{Eval}(\mathcal{C}', c)$ an encryption of r_b.

$\mathsf{OT}_3(\mathsf{sk}, ot_2)$:

- Let s_0, s_1, x_0, x_1, c, and e be the content of the message ot_2
- Let $b \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
- Let $r_b \leftarrow \mathsf{Dec}(\mathsf{sk}, e)$
- return $x_b \oplus \mathsf{Ext}(s_b, r_b)$

2.2 Correctness

Since $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is correct c is a correct encryption of b in that scheme. OT_2 then outputs s_0, s_1, $\mathsf{Ext}(s_0, r_0) \oplus m_0$, and $\mathsf{Ext}(s_1, r_1) \oplus m_1$ together with correct encryptions of b and r_b. In OT_3 we then decrypt b and r_b. Because Ext is deterministic (with a fixed seed s_b) we can reconstruct $m_b = m_b \oplus \mathsf{Ext}(s_b, r_b) \oplus \mathsf{Ext}(s_b, r_b)$.

2.3 Computational Receiver's Security

The sender only ever sees encryptions of the receivers input b and the public key of the LHE. Therefore, if the sender can learn anything about b he can also break the CPA security of the LHE.

2.4 Statistical Sender's Security

▶ **Theorem 1.** *Let* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *be an LHE with high rate, then* $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ *as detailed in Subsection 2.1 is a statistically sender private OT protocol.*

Proof. In the following, we show an unbounded simulator Sim that does not know m_0 or m_1 but has one-time access to an oracle for the function $f(b) = m_b$. With this oracle access, she produces an output which is statistically close to the output of OT_2, which has full access to r_0 and r_1.

$\mathsf{Sim}^f(ot_1 = (\mathsf{pk}, c))$:

- Choose $s_0, s_1 \leftarrow_\$ \{0,1\}^m$ uniformly at random
- Choose $r_0, r_1 \leftarrow_\$ \{0,1\}^m$ uniformly at random
- Hardwire r_0, r_1 into $\mathcal{C}[r_0, r_1]$ to get circuit $\mathcal{C}'$
- Let $e \leftarrow \mathsf{Eval}(\mathcal{C}', c)$
- Let C be the value such that $H_\infty(R_{1-C}|C, E)$ is minimal with C being chosen as in corollary 31.
- Query the oracle f for m_C
- Choose $S_{1-C} \leftarrow_\$ \{0,1\}^n$ uniformly at random
- If $C = 0$:
  - return s_0, s_1, $\mathsf{Ext}(s_0, r_0) \oplus m_0$, S_{1-C}, c, and e
- Else:
  - return s_0, s_1, S_{1-C}, $\mathsf{Ext}(s_1, r_1) \oplus m_1$, c, and e

We now use a hybrid argument to show that the above construction is statistically sender private. H_0 is the honest execution of the protocol.

$H_0(\mathsf{pk}, c, m_0, m_1)$:

- Choose $s_0, s_1 \leftarrow_\$ \{0,1\}^m$ uniformly at random
- Choose $r_0, r_1 \leftarrow_\$ \{0,1\}^m$ uniformly at random
- Hardwire r_0, r_1 into $\mathcal{C}[r_0, r_1]$ to get circuit $\mathcal{C}'$
- return s_0, s_1, $\mathsf{Ext}(s_0, r_0) \oplus m_0$, $\mathsf{Ext}(s_1, r_1) \oplus m_1$, c, and $\mathsf{Eval}(\mathcal{C}', c)$

In hybrid H_1 we replace $\mathsf{Ext}(s_{1-C}, r_{1-C})$ by a uniformly random S_0 of same size.

$H_1(\mathsf{pk}, c, m_0, m_1)$:

- Choose $s_0, s_1 \leftarrow_\$ \{0,1\}^m$ uniformly at random
- Choose $r_0, r_1 \leftarrow_\$ \{0,1\}^m$ uniformly at random
- Hardwire r_0, r_1 into $\mathcal{C}[r_0, r_1]$ to get circuit $\mathcal{C}'$
- Let $e \leftarrow \mathsf{Eval}(\mathcal{C}', c)$
- Let C be the value such that $H_\infty(R_{1-C}|C, E)$ is minimal with C being chosen as in corollary 31.
- Choose $S_{1-C} \leftarrow_\$ \{0,1\}^n$ uniformly at random
- If $C = 0$:
  - return s_0, s_1, $\mathsf{Ext}(s_0, r_0) \oplus m_0$, $S_{1-C} \oplus m_1$, c, and e
- Else:
  - return s_0, s_1, $S_{1-C} \oplus m_0$, $\mathsf{Ext}(s_1, r_1) \oplus m_1$, c, and e

---

In H_2 we remove the real sender inputs.

$H_2^f(\mathsf{pk}, c)$:

- Choose $s_0, s_1 \leftarrow_\$ \{0,1\}^m$ uniformly at random
- Choose $r_0, r_1 \leftarrow_\$ \{0,1\}^m$ uniformly at random
- Hardwire r_0, r_1 into $\mathcal{C}[r_0, r_1]$ to get circuit $\mathcal{C}'$
- Let $e \leftarrow \mathsf{Eval}(\mathcal{C}', c)$
- Let C be the value such that $H_\infty(R_{1-C}|C, E)$ is minimal with C being chosen as in corollary 31.
- Query the oracle f for m_C
- Choose $S_{1-C} \leftarrow_\$ \{0,1\}^n$ uniformly at random
- If $C = 0$:
  - return s_0, s_1, $\mathsf{Ext}(s_0, r_0) \oplus m_0$, S_{1-C}, c, and e
- Else:
  - return s_0, s_1, S_{1-C}, $\mathsf{Ext}(s_1, r_1) \oplus m_1$, c, and e

---

Now we argue why the hybrids are statistically close.

$H_0 \approx H_1$:

In H_1 we replace $\mathsf{Ext}(s_{1-C}, r_{1-C})$ by a uniformly random chosen S_{1-C}. Here we argue that the statistical distance between the two hybrids is negligible using 31.

Lemma 30 gives that

$$\begin{aligned} H_\infty(R_0, R_1|E = e) &> H_\infty(R_0, R_1) - log(1/\mathsf{Pr}\left[E = e\right]) \\ &\geq 2m - |e| \end{aligned}$$

Then corollary 31 gives that

$$H^{\varepsilon}_{\infty}(R_{1-C}|C, E = e) > (2m - |e|)/2 - 1 - log(1/\varepsilon)$$

for any ε. Then the smooth min-entropy conversion lemma 32 gives that

$$H_{\infty}(R_{1-C}|C, E = e) \geq -log(2^{-(2m-|e|)/2-1-log(1/\varepsilon)} + \varepsilon)$$

In the following, this number will be called α. Notice that α can only be positive if $2m - |e|$ is positive and e encrypts a message of size m. Therefore, the rate ρ need to be bigger than $1/2$ (i.e. $1/2 < \rho = m/|e|$).
Then we use the property of the extractor to ensure that $\mathsf{Ext}(s_{1-C}, r_{1-C})$ is statistically close to uniform (i.e. $SD(\mathsf{Ext}(s_{1-C}, r_{1-C}), S_{1-C}) \leq \varepsilon'$). Clearly, this can be reached if the rate $\rho > 1/2$. Therefore, the statistical distance between H_0 and H_1 is at most ε'.

$H_1 \approx H_2$:
In this hybrid, we altogether remove m_{1-C} which we can do because it is being XORed with a uniformly random string and therefore is perfectly hidden. Thus, H_1 and H_2 are identically distributed in this case. ◀

2.5 FHE with Circuit-Private OT Evaluation

Here, we show how to add a evaluation procedure $\mathsf{Eval}_{\mathsf{OT}}$ to a high-rate FHE, which can evaluate choice functions in a circuit private manner.

The construction is the same as for the OT above but the message reconstruction of OT_3 is done on the sender's side. Again, we use circuit $\mathcal{C}$

Circuit $\mathcal{C}[r_0, r_1](b)$:
- output r_b

But we also use circuit $\tilde{\mathcal{C}}$ which except for decrypting takes the role of OT_3

Circuit $\tilde{\mathcal{C}}[s_0, s_1, x_0, x_1](b, r_b)$:
- output $x_b \oplus \mathsf{Ext}(s_b, r_b)$

$\mathsf{Eval}_{\mathsf{OT}}(1^{\lambda}, \mathsf{pk}, m_0, m_1, c)$:
- Choose $s_0, s_1 \leftarrow_{\$} \{0,1\}^m$ uniformly at random
- Choose $r_0, r_1 \leftarrow_{\$} \{0,1\}^m$ uniformly at random
- Hardwire r_0, r_1 into $\mathcal{C}[r_0, r_1]$ to get circuit $\mathcal{C}'$
- Let $e \leftarrow \mathsf{Eval}(1^{\lambda}, \mathsf{pk}, \mathcal{C}', c)$
- Hardwire s_0, s_1, $x_0 = \mathsf{Ext}(s_0, r_0) \oplus m_0$, and $x_1 = \mathsf{Ext}(s_0, r_0) \oplus m_1$ into $\tilde{\mathcal{C}}[s_0, s_1, x_0, x_1]$ to get circuit $\tilde{\mathcal{C}}'$
- return $\mathsf{Eval}(1^{\lambda}, \mathsf{pk}, \tilde{\mathcal{C}}', (c, e))$

Correctness and receiver's security (in this case CPA security) stay the same as before. For circuit privacy (previously sender privacy) we now need to argue over the compression in e. The last step in $\mathsf{Eval}_{\mathsf{OT}}$ can be thought of as post-processing and does not change anything about the circuit privacy.

3 Circuit-Private NC1-HE from FHE with OT

An OT is similar to a circuit private HE for affine functions. We use Decomposeable Affine Randomized Encodings (DARE) to increase the set of function that we can evaluate with circuit privacy to all functions in NC1. We achieve this by letting the OT do the affine operations and then evaluate the DARE inside another layer of FHE.

3.1 Construction

Let (KeyGen', Enc', Eval', Dec') be an FHE with circuit private choice function evaluation procedure $\mathsf{Eval}'_{\mathsf{OT}}$ and (Garble, $\mathsf{GarbleInput}$, Ev) be a ϕ-private DARE. In this construction we use a circuit $\mathcal{C}$ with hardcoded garbled function F which simply evaluates the garbled function on the input.

$\mathcal{C}$ **[**F**]**$(d = (d_i)_{i\in[n]})$**:**

- return $\mathsf{Ev}(F, (d_i)_{i\in[n]})$

The construction then is:

KeyGen(1^λ)**:**

- return $\mathsf{KeyGen}'(1^\lambda)$

Enc(pk, m)**:**

- return $\mathsf{Enc}'(\mathsf{pk}, m)$

Eval$(1^\lambda, \mathsf{pk}, f, c = (c_i)_{i\in[n]})$**:**

- $(F, (r_{i,j})_{i\in[n],j\in\{0,1\}}) \leftarrow \mathsf{Garble}(f, 1^\lambda)$
- For each $i \in [n]$ let $z_i \leftarrow \mathsf{Eval}'_{\mathsf{OT}}(1^\lambda, \mathsf{pk}, r_{i,0}, r_{i,1}, c_i)$
- Hardwire F into $\mathcal{C}[F]$ to get the circuit $\mathcal{C}'$
- return $\mathsf{Eval}'(1^\lambda, \mathsf{pk}, \mathcal{C}', z = (z_i)_{i\in[n]})$

Dec(sk, c)**:**

- return $\mathsf{Dec}'(\mathsf{sk}, c)$

First Eval garbles f and then emulates the encoding mechanism $\mathsf{GarbleInput}$ inside of the FHE with the help of $\mathsf{Eval}_{\mathsf{OT}}$. This works because the $\mathsf{GarbleInput}$ is a choice function which is exactly what an OT calculates. With the encoded input and the garbled circuit F we run the Ev function inside the FHE and will only be able to leak as much information about the function as $(F, \mathsf{GarbleInput}(r, m))$ would have.

The correctness of (KeyGen, Enc, Dec, Eval) follows routinely from the correctness of (Garble, $\mathsf{GarbleInput}$, Ev), and (KeyGen', Enc', Eval', $\mathsf{Eval}'_{\mathsf{OT}}$, Dec'). Likewise, CPA security of (KeyGen, Enc, Dec, Eval) follows routinely from the CPA security of (KeyGen', Enc', Eval', $\mathsf{Eval}'_{\mathsf{OT}}$, Dec').

3.2 Malicious Statistical Circuit Privacy

▶ **Theorem 2.** *Let (*KeyGen'*,* Enc'*,* Eval'*) be an FHE with circuit private choice function evaluation procedure* $\mathsf{Eval}'_{\mathsf{OT}}$ *and (*Garble*,* $\mathsf{GarbleInput}$*,* Ev*) be a ϕ-private DARE (for some function ϕ) then the NC1-HE as detailed in Subsection 3.1 is ϕ-circuit-private.*

The proof of the theorem is in the full version of the paper.

3.3 Computational Circuit Privacy

If we use a computationally ϕ-private garbled circuit in this transformation instead of its information theoretical counterpart we instantly get an FHE which is ϕ-circuit-private against computational adversaries. Nothing about the construction needs to change; we only need to adjust the proof as detailed in the full version.

3.4 Multi-Hop-Security

Since evaluating does not change the structure of the ciphertexts the $NC1$-HE inherits the multi-hop-security property from the FHE (if the FHE is multi-hop then the $NC1$-HE is as well).

4 Circuit-Private FHE from Circuit-Private NC1-HE

To build a circuit-private FHE from a Circuit-Private NC1-HE, we go back to techniques from Kilian's classic paper [23]. On a high level, we split up the circuit into NC1 circuits and encrypt the connecting wires with the one-time pad.

Assume we want to evaluate a circuit $\mathcal{C}$ of polynomial depth. We show an example of this in Figure 1.

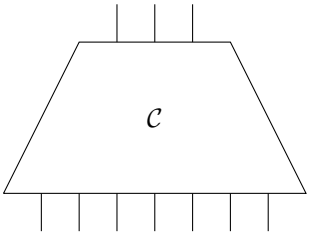

Figure 1 Circuit $\mathcal{C}$.

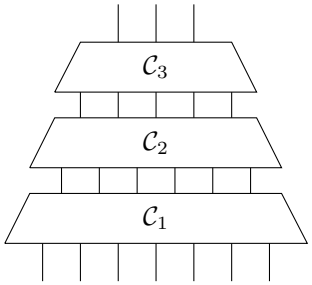

Figure 2 Circuit $\mathcal{C}$ split into subcircuits $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$. We chose three subcircuits for illustrative reasons. The amount of subcircuits depends on the depth of circuit $\mathcal{C}$.

We split up that circuit into subcircuits of depth $log(\lambda)$ such that they are $NC1$ circuit (as in Figure 2). If the circuit-private $NC1$-HE scheme is multi-hop, we can then evaluate each of these subcircuits sequentially in a circuit-private manner. This construction is an FHE scheme which leaks the depth of the circuit and the intermediate values.

We can, however, encrypt these intermediate values with a one-time pad and then decrypt it in the next subcircuit. We demonstrate this modification of the circuit in Figure 3.

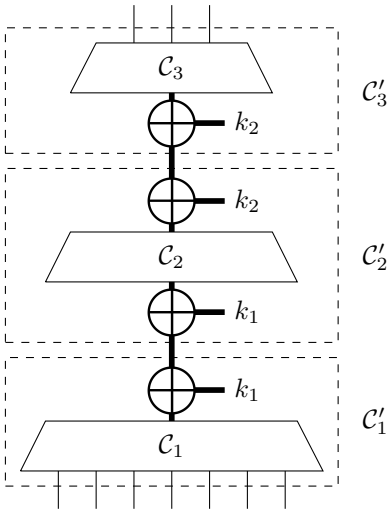

Figure 3 Subcircuits of $\mathcal{C}$ together with OTP encryption and decryption. Each thick wire represents a collection of wires. We use the circuits $\mathcal{C}'_1$, $\mathcal{C}'_2$, and $\mathcal{C}'_3$.

This is possible because encrypting and decrypting the one-time pad is incredibly (computationally) cheap. Therefore, the subcircuits combined with encryption and decryption are still in $NC1$. This way the intermediate values are statistically hidden.

The result is an FHE scheme, which is $\Phi_{depth,width}$ circuit private. $\Phi_{depth,width}$ leaks the depth of the circuit and the size of the intermediate values.

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 Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 297–314. Springer, 2014. doi:10.1007/978-3-662-44371-2_17.

3 Benny Applebaum. Key-dependent message security: Generic amplification and completeness. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 527–546. Springer, 2011. doi:10.1007/978-3-642-20465-4_29.

4 Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. *IACR Cryptol. ePrint Arch.*, page 385, 2017. URL: http://eprint.iacr.org/2017/385.

5 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 166–175. IEEE Computer Society, 2004. doi:10.1109/FOCS.2004.20.

6 Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III*, volume 10626 of *Lecture Notes in Computer Science*, pages 275–303. Springer, 2017. doi:10.1007/978-3-319-70700-6_10.

7 Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 784–796. ACM, 2012. doi:10.1145/2382196.2382279.

8 Florian Bourse, Rafaël Del Pino, Michele Minelli, and Hoeteck Wee. FHE circuit privacy almost for free. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 62–89. Springer, 2016. doi:10.1007/978-3-662-53008-5_3.

9 Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 407–437. Springer, 2019. doi:10.1007/978-3-030-36033-7_16.

10 Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 309–325. ACM, 2012. doi:10.1145/2090236.2090262.

11 Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.12.

12 Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 1–12. ACM, 2014. doi:10.1145/2554797.2554799.

13 Wutichai Chongchitmate and Rafail Ostrovsky. Circuit-private multi-key FHE. In Serge Fehr, editor, *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II*, volume 10175 of *Lecture Notes in Computer Science*, pages 241–270. Springer, 2017. doi:10.1007/978-3-662-54388-7_9.

14 Ivan Damgård, Serge Fehr, Renato Renner, Louis Salvail, and Christian Schaffner. A tight high-order entropic quantum uncertainty relation with applications. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 360–378. Springer, 2007. doi:10.1007/978-3-540-74143-5_20.

15 Yevgeniy Dodis, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540. Springer, 2004. doi:10.1007/978-3-540-24676-3_31.

16 Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2019. doi:10.1007/978-3-030-26954-8_1.

17 Léo Ducas and Damien Stehlé. Sanitization of FHE ciphertexts. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 294–310. Springer, 2016. doi:10.1007/978-3-662-49890-3_12.

18 Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009. doi:10.1145/1536414.1536440.

19 Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 438–464. Springer, 2019. doi:10.1007/978-3-030-36033-7_17.

20 Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-hop homomorphic encryption and rerandomizable yao circuits. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 155–172. Springer, 2010. doi:10.1007/978-3-642-14623-7_9.

21 Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013. doi:10.1007/978-3-642-40041-4_5.

22 Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 294–304. IEEE Computer Society, 2000. doi:10.1109/SFCS.2000.892118.

23 Joe Kilian. Founding cryptography on oblivious transfer. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 20–31. ACM, 1988. doi:10.1145/62212.62215.

24 Stephan Krenn, Krzysztof Pietrzak, and Akshay Wadia. A counterexample to the chain rule for conditional HILL entropy - and what deniable encryption has to do with it. In Amit Sahai, editor, *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, volume 7785 of *Lecture Notes in Computer Science*, pages 23–39. Springer, 2013. doi:10.1007/978-3-642-36594-2_2.

25 Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.*, 37(3):50–60, 2020. doi:10.1109/MSP.2020.2975749.

26 Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private FHE. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 536–553. Springer, 2014. doi:10.1007/978-3-662-44371-2_30.

27 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005. doi:10.1145/1060590.1060603.

28 Maciej Skórski. Strong chain rules for min-entropy under few bits spoiled. In *IEEE International Symposium on Information Theory, ISIT 2019, Paris, France, July 7-12, 2019*, pages 1122–1126. IEEE, 2019. doi:10.1109/ISIT.2019.8849240.

29 Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010. doi:10.1007/978-3-642-13190-5_2.

30 Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986. doi:10.1109/SFCS.1986.25.

A Appendix: Preliminaries

In this appendix, we define the concepts and notation that we use in the paper.

A.1 Notation

Assignments

Assignment of a value to a variable is denoted by $\leftarrow$ and $\leftarrow_{\$}$ is used for choosing a value from a set uniformly at random.

Negligible Functions

A function $f : \mathbb{N} \to \mathbb{R}$ is negligible in λ if there exists no positive polynomial p such that $f(\lambda) < \frac{1}{p(\lambda)}$ for all but finitely many λ.

Logarithms

The base of every logarithm in this document is 2.

Circuits

Typical implementations of FHE evaluate using circuit representation for functions. Therefore, we create circuits and then evaluate them. If $\mathcal{C}[a]$ is a circuit, a is a value which we hardwire into the circuit. The input size of a circuit $\mathcal{C}$ is called $in(\mathcal{C})$.

A.2 Public-Key Encryption Schemes

A public-key encryption scheme uses two keys, a public key pk and a secret key sk. We use the public key to encrypt messages, the result of which is called ciphertext. Without knowledge of the secret key, it is virtually impossible to recover the message from the ciphertext. The secret key, however, enables the holder to reliably retrieve the message from the ciphertext.

▶ **Definition 3** (Public-Key Encryption). *The following PPT algorithms describe a public-key encryption scheme:*

KeyGen(1^λ)*: The key-generation algorithm takes the security parameter λ as input and outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$.

Enc(pk, m)*: The encryption algorithm takes a public key* pk *and a message m as input and outputs a ciphertext c.*

Dec(sk, c)*: The decryption algorithm takes a secret key* sk *and a ciphertext c as input and outputs a message m. It rarely requires randomness.*

In the rest of the document, every encryption scheme will be public key.

▶ **Definition 4** (Correctness). *An encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is correct if for all message m and security parameters λ and* $(\mathsf{pk}, \mathsf{sk})$ *in the range of* $\mathsf{KeyGen}(1^\lambda)$ *we have* $m = \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m))$

The most popular notion of security for encryption schemes is CPA security (also known as IND-CPA security or semantic security).

▶ **Definition 5** (CPA Security). *An encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is cpa secure if for all PPT adversary pairs* $(\mathcal{A}_1, \mathcal{A}_2)$

$$\left| Pr\left[b = b' \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{pk}) \\ b \leftarrow_\$ \{0,1\} \\ b' \leftarrow \mathcal{A}_2(\mathsf{Enc}(\mathsf{pk}, m_b), \sigma) \end{array} \right] - \frac{1}{2} \right|$$

is negligible in λ.

A.3 Homomorphic Encryption

Certain changes on a ciphertext change the underlying plaintext in a structured way.

▶ **Definition 6** (Homomorphic Encryption). *These four PPT algorithms describe a homomorphic encryption scheme:* KeyGen,Enc, *and* Dec *as in pubilc-key encryption and*

Eval$(1^\lambda, \mathsf{pk}, f, c_1, ..., c_n)$*: The evaluation algorithm takes a security parameter λ, a public key* pk*, a string representation of a function f and n where n is the input size of f ciphertexts $c_1, \ldots, c_n$ as inputs and outputs a new ciphertext c.*

▶ **Definition 7** (Homomorphic Correctness). *Let $\mathcal{F}$ be a set of functions, f be an arbitrary element of $\mathcal{F}$, and $n = in(f)$. An $\mathcal{F}$-homomorphic encryption scheme* (KeyGen, Enc, Eval, Dec) *is correct if* (KeyGen, Enc, Dec) *is a correct encryption scheme, and for all messages $m_1, \ldots, m_n$, security parameters λ, and* (pk, sk) *from the support of* $\mathsf{KeyGen}(1^\lambda)$ *we have* $f(m_1, \ldots, m_n) = \mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(1^\lambda, \mathsf{pk}, f, \mathsf{Enc}(\mathsf{pk}, m_1), \ldots, \mathsf{Enc}(\mathsf{pk}, m_n)))$

▶ **Definition 8** (Linearly-Homomorphic Encryption). *A linearly-homomorphic encryption scheme (LHE) is an $\mathcal{F}$-homomorphic encryption scheme where $\mathcal{F}$ is the set of all multivariate linear functions.*

▶ **Definition 9** (Fully-Homomorphic Encryption). *A fully-homomorphic encryption scheme (FHE) is an $\mathcal{F}$-homomorphic encryption scheme where $\mathcal{F}$ is the set of all computable functions.*

CPA security is unchanged from public key encryption.

The ability to use a homomorphic evaluation on a ciphertext which has already gone through evaluation is called multi-hop. To define the correctness of a multi-hop HE we need to define a set $\mathcal{C}_{\mathsf{pk}}$ correctly generated ciphertexts. Each ciphertext comes from encryption or homomorphic evaluation on a correct plaintext.

▶ **Definition 10** (Multi-Hop Homomorphic Encryption). *Just like a $\mathcal{F}-$HE scheme, a multi-hop $\mathcal{F}-$HE scheme is a quadruple of PPT algorithms (*KeyGen,Enc,Eval,Dec*). Let λ be a security parameter,* (pk, sk) *be the output of* $\mathsf{KeyGen}(1^\lambda)$ *then*

$$\mathcal{C}_{\mathsf{pk}} = \left\{ c \;\middle|\; \begin{array}{l} m \in \mathcal{M} \wedge c = \mathsf{Enc}(\mathsf{pk}, m) \vee \\ f \in \mathcal{F} \wedge n = in(f) \wedge c_1, \ldots, c_n \in \mathcal{C}_{\mathsf{pk}} \wedge c = \mathsf{Eval}(1^\lambda, \mathsf{pk}, f, c_1, \ldots, c_n) \end{array} \right\}$$

is a set of correctly generated ciphertexts under public key pk*. Such a quadruple of algorithms is a multi-hop $\mathcal{F}-$HE scheme if it is a $\mathcal{F}-$HE and for all security parameters λ, outputs of the* $\mathsf{KeyGen}(1^\lambda)$ (pk, sk)*, functions $f \in \mathcal{F}$, $n = in(f)$, and ciphertexts $c_1, \ldots c_n \in \mathcal{C}_{\mathsf{pk}}$ we have* $f(\mathsf{Dec}(\mathsf{sk}, c_1), \ldots, \mathsf{Dec}(\mathsf{sk}, c_n)) = \mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(1^\lambda, \mathsf{pk}, f, c_1, \ldots, c_n))$

The rate captures how big a ciphertext is in comparison to its plaintext content.

▶ **Definition 11** (Rate). *An $\mathcal{F}-$HE scheme* (KeyGen, Enc, Eval, Dec) *has rate ρ if there exists a polynomial μ such that for all security parameters λ, possible outputs of* $\mathsf{KeyGen}(1^\lambda)$ (pk, sk)*, correctly generated ciphertexts $c \in \mathcal{C}_{\mathsf{pk}}$ of size $\geq \mu(\lambda)$ we have* $|\mathsf{Dec}(\mathsf{sk}, c)|/|c| \geq \rho(\lambda)$

We call an encryption scheme high rate if it has a rate greater than $1/2$.

Typically a HE is also defined with compactness. For compactness, we require the ciphertext to be independent in size from the functions evaluated to arrive at the ciphertext.

▶ **Definition 12** (Compactness). *An $\mathcal{F}-$HE scheme* (KeyGen, Enc, Eval, Dec) *is compact if there exists a rate ρ that only depends on λ.*

There is also a notion of malicious circuit privacy that guarantees that the ciphertext does not leak information about the function which was homomorphically evaluated on it beyond the result even if the public key and the ciphertexts are maliciously generated [26].

▶ **Definition 13** ((Malicious) Circuit Privacy). *We say an $\mathcal{F}-$HE scheme is maliciously, statistically circuit private if there exists an unbounded simulator* Sim *with one-time oracle access to f such that for all λ, and for all public keys* pk*, functions $f \in \mathcal{F}$, and ciphertexts $c = (c_1, \ldots, c_n)$ for $n = in(f)$ we have $SD(\mathsf{Sim}^f(1^\lambda, \mathsf{pk}, c), \mathsf{Eval}(1^\lambda, \mathsf{pk}, f, c))$ is negligible in λ.*

Our constructions do not quite achieve the malicious, statistically circuit privacy guarantee of [26]. However, we achieve a slightly weaker notion defined in the following.

▶ **Definition 14** (Φ-Circuit Privacy). *Let $\Phi : \mathcal{F} \to \{0,1\}^*$ be a (leakage) function. We say an $\mathcal{F}-HE$ scheme is Φ (maliciously) circuit private if there exists an unbounded simulator* Sim *with one-time oracle access to f such that for all λ, public keys* pk*, ciphertexts $c = c_1, \ldots, c_n$, functions $f \in \mathcal{F}$, and PPT adversaries $\mathcal{A}$ we have $|Pr[\mathcal{A}(\mathsf{Sim}^f(1^\lambda, \mathsf{pk}, c, \Phi(f)))] - Pr[\mathcal{A}(\mathsf{Eval}(1^\lambda, \mathsf{pk}, f, c))]|$ is negligible in λ.*

The only difference to the above notion of circuit privacy is that the simulator gets some leaked information Φ about the circuit. In most cases, Φ would leak some structural information such as the size of the circuit or its topology. This notion is adapted to expose some properties of the circuit from privacy definitions for garbled circuits.

A.4 Garbling Schemes

Garbling schemes were famously introduced by Yao in an oral presentation [30] about techniques for secure function evaluation. Our notation is adapted from [7] and also influenced the definition of Φ circuit privacy for HE. It allows to split up the evaluation of a function such that different parties can do parts of the computation. One party knows the input x to the function f and encodes it such that the other party can evaluate the function on the encoding (i.e. learn $f(x)$) without being able to compute the input.

▶ **Definition 15** (Garbling Schemes). *A garbling scheme is described by the following PPT algorithms:*

$\mathsf{Garble}(1^\lambda, f)$*: The circuit garbling algorithm takes a security parameter and the circuit representation of a function f as inputs and outputs a garbled circuit F and $2n$ bitstrings $X_1^0, X_1^1, \ldots, X_n^0, X_n^1$ where n is the input size of f.*

$\mathsf{GarbleInput}((X_1^0, X_1^1, \ldots, X_n^0, X_n^1), m)$*: The input garbling mechanism takes $2n$ bitstrings $X_1^0, X_1^1, \ldots, X_n^0, X_n^1$ and a message x as inputs and outputs the n bitstrings $X_1^{x_1}, \ldots, X_n^{x_n}$.*

$\mathsf{Ev}(F, (X_1, \ldots, X_n))$*: The evaluation algorithm takes a garbled function F and n bitstrings $X_1, \ldots X_n$ as inputs and outputs $f(x)$.*

▶ **Definition 16** (Correctness). *A garbling scheme* $(\mathsf{Garble}, \mathsf{GarbleInput}, \mathsf{Ev})$ *is correct if f is a function, x is an input to that function, λ is the security parameter, (F, e) is from the range of* $\mathsf{Garble}(1^\lambda, f)$ *then* $\mathsf{Ev}(F, \mathsf{GarbleInput}(e, x)) = f(x)$.

▶ **Definition 17** (Statistical Privacy). *A garbling scheme is Φ statistically private if there exists a unbounded algorithm* $\mathsf{Sim}(1^\lambda, y, \Phi)$ *such that,*

$$SD(\mathsf{Sim}(1^\lambda, y, \Phi(f)))|y = f(x)], \left[\mathcal{D}(F, X) \middle| \begin{array}{l} (F, e) \leftarrow \mathsf{Garble}(1^\lambda, f) \\ X \leftarrow \mathsf{GarbleInput}(e, x) \end{array}\right])$$

is negligible in λ.

Garbled circuits with statistical privacy are usually researched under the guise of Decomposable Affine Randmized Encodings (DARE) [22, 5, 4].

An example for this is [23]'s construction for branching programs.

A.5 Oblivious Transfer

String oblivious transfer (OT) is a protocol which allows two parties (sender and receiver) to interact in the following way: The sender has two strings m_0, m_1 and the receiver has a bit b. The goal is that the receiver learns m_b but the sender does not learn anything about b.

▶ **Definition 18** (Oblivious Transfer). *A (two-message) OT is described by the following PPT algorithms:*

$\mathsf{OT}_1(1^\lambda, b)$*: With the input of a security parameter* λ *and a bit* b*, the algorithm returns* ot_1 *and state.*

$\mathsf{OT}_2(1^\lambda, ot_1, m_0, m_1)$*: With the input of a security parameter* λ*, request* ot_1*, and two strings of same length* m_0, m_1*, the algorithm returns a response* ot_2

$\mathsf{OT}_3(ot_2, state)$*: With the input of a response* ot_2 *and a state state, the algorithm returns a string* m

▶ **Definition 19** (Correctness). *An OT* $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ *is correct if for all security parameters* λ*, bits* b*, messages* m_0, m_1*,* $(ot_1, state)$ *from the range of* $\mathsf{OT}_1(1^\lambda, b)$ *and* ot_2 *from the range of* $\mathsf{OT}_2(1^\lambda, ot_1, m_0, m_1)$ *we have* $m_b = \mathsf{OT}_3(ot_2, state)$

▶ **Definition 20** (Receiver's Security). *An OT* $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ *has (computational) receiver's security if for every PPT adversary* $\mathcal{A}$*, and security parameters* λ *we have* $\left|Pr[\mathcal{A}(\mathsf{OT}_1(1^\lambda, 0)] - Pr[\mathcal{A}(\mathsf{OT}_1(1^\lambda, 1)]\right|$ *is negligible in* λ*.*

▶ **Definition 21** (Statistical Sender's Security). *An OT* $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ *has statistical sender's security if there exists a deterministic unbounded simulator* Sim *such that for all security parameters* λ*, strings* ot_1*, strings* m_0, m_1 *of length* k *we have* $SD(\mathsf{OT}_2(1^\lambda, ot_1, m_0, m_1), \mathsf{Sim}^{m_{(\cdot)}}(1^\lambda, ot_1, k))$ *is negligible in* λ *with* Sim *having one time access to a* $m_{(\cdot)}$ *oracle.*

▶ **Definition 22** (Rate). *An OT* $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ *has rate* ρ *if there exists a polynomial* μ *such that for all security parameters* λ*, possible outputs* ot_1 *of* $\mathsf{OT}_1(1^\lambda, b)$*, and messages* m_0, m_1 *with* $|m_0| = |m_1| \geq \mu(\lambda)$ *we have* $|m_0|/|\mathsf{OT}(1^\lambda, ot_1, m_0, m_1)| \geq \rho(\lambda)$

For the purposes of this document every OT has computational receiver's security, and statistical sender's security.

A.6 Information Theory

The statistical distance is a metric on probability distributions. It is often used in cryptography because it is at the core of the definition of statistical indistinguishability. Statistical indistinguishability is a strictly stronger notion than computational indistinguishability, which is the most popular tool to define security notions in cryptography.

▶ **Definition 23** (Statistical Distance). *Let* X *and* Y *be two distributions with support in* $\{0,1\}^k$*. The statistical difference between* X *and* Y*,* $SD(X, Y)$ *is given by,*

$$SD(X, Y) = \frac{1}{2} \sum_{x \in \{0,1\}^k} |\mathsf{Pr}\left[X = x\right] - \mathsf{Pr}\left[Y = x\right]|$$

▶ **Lemma 24.** *The statistical distance has an equivalent definition*

$$SD(X, Y) = max_{f:\{0,1\}^k \to \{0,1\}} |\mathsf{Pr}\left[f(X) = 1\right] - \mathsf{Pr}\left[f(Y) = 1\right]|$$

Entropy measures a lack of knowledge about a system. The most famous entropy is the Shannon entropy H, which measures the lack of knowledge in a system that behaves randomly. Min-entropy, on the other hand, assumes a system which behaves maliciously.

▶ **Definition 25** (Min-Entropy). *Let* X *be a distribution. The min-entropy of* X *is*

$$H_\infty(X) = -log(max_x \Pr[X = x])$$

▶ **Definition 26** (Conditional (Smooth) Min-Entropy [14]). *The conditional smooth min-entropy* $H^{\varepsilon}_{\infty}(X|Y)$ *is defined as* $H^{\varepsilon}_{\infty}(X|Y) = max_{\mathcal{E}} min_y H_{\infty}(X\mathcal{E}|Y = y)$*, where the maximum is over all events* $\mathcal{E}$ *with* $Pr(\mathcal{E}) \geq 1 - \varepsilon$*.*

▶ **Corollary 27** (Corollary of Lemma 1 from [14]). *Let* X, Y *be distributions then* $H^{\varepsilon}_{\infty}(X|Y) > H_{\infty}(X, Y) - H_0(Y) - log(1/\varepsilon)$ *for all* ε*.*

Strong extractors make it possible to use one source of uniform randomness to convert a non-uniform distribution with some min-entropy into a uniform distribution.

▶ **Definition 28** (Strong Extractor). *A function* $\mathsf{Ext} : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^n$ *is a* (k, ϵ)*-strong extractor if for every distribution* X *with support in* $\{0,1\}^m$ *and* $H_{\infty}(X) = k$*, we have* $SD((\mathsf{Ext}(X, U_d), U_d), (U_n, U_d)) \leq \epsilon$ *where* U_d *is a uniform distribution over* $\{0,1\}^d$ *and* U_n *is one over* $\{0,1\}^n$*.*

Many of the useful rules like the chain rule for conditional Shannon entropy $H(X|Y) = H(X, Y) - H(Y)$ do not hold for min-entropy. Therefore we have to do hard work to handle claims about min-entropy.

The next lemma allows to lower bound the min-entropy using the average conditional min-entropy.

▶ **Lemma 29** (Weakened Lemma 2.2 of [15]). *For all random variables* X, Y*,* $\delta > 0$ *the conditional min-entropy we have* $H_{\infty}(X|Y = y) \geq \tilde{H}_{\infty}(X|Y) - log(1/\delta)$ *with probability* $1 - \delta$ *over the choice of* y

The leakage lemma for min-entropy helps with bounding the min-entropy of distributions that are conditioned on events.

▶ **Lemma 30** (Leakage Lemma for Min-Entropy of [28]). *For all random variables* X *and events* A, B *we have* $H_{\infty}(X|B, A) > H_{\infty}(X|B) - log(1/Pr(A|B))$

▶ **Corollary 31** (Corollary 4.3 of [14]). *Let* $\varepsilon \geq 0$*, and let* X_0,X_1 *and* Z *be random variables such that* $H^{\varepsilon}_{\infty}(X_0, X_1|Z) \geq \alpha$*. Then, there exists a binary random variable* C *over* $\{0,1\}$ *such that* $H^{\varepsilon+\varepsilon'}_{\infty}(X_{1-C}|Z, C) \geq \alpha/2 - 1 - log(1/\varepsilon')$ *for any* $\varepsilon' > 0$*.*

▶ **Lemma 32** (Smooth Min-Entropy Conversion). *If* $H^{\varepsilon}_{\infty}(X) \geq \alpha$ *then* $H_{\infty}(X) \geq -log(2^{-\alpha} + \varepsilon)$

Proof. Since $H^{\varepsilon}_{\infty}(X) \geq \alpha$ there exists a distribution Y such that $H_{\infty}(Y) \geq \alpha$ and $SD(X, Y)$. This means, for all y', $\mathsf{Pr}_{y \leftarrow Y}[y' = y] \leq 2^{-\alpha}$. Therefore, the biggest probability of X can only be bigger by ε. Then, for all x', $\mathsf{Pr}_{x \leftarrow X}[x' = x] \leq 2^{-\alpha} + \varepsilon$. ◀

From Privacy-Only to Simulatable OT: Black-Box, Round-Optimal, Information-Theoretic

Varun Madathil
North Carolina State University, Raleigh, NC, USA

Chris Orsini
North Carolina State University, Raleigh, NC, USA

Alessandra Scafuro
North Carolina State University, Raleigh, NC, USA

Daniele Venturi
Sapienza University of Rome, Italy

Abstract

We present an information-theoretic transformation from any 2-round OT protocol with only *game-based* security in the presence of malicious adversaries into a 4-round (which is known to be optimal) OT protocol with simulation-based security in the presence of malicious adversaries.

Our transform is the first satisfying all of the following properties at the same time:

- It is in the *plain model*, without requiring any setup assumption.
- It only makes *black-box* usage of the underlying OT protocol.
- It is *information-theoretic*, as it does not require any further cryptographic assumption (besides the existence of the underlying OT protocol).

Additionally, our transform yields a cubic improvement in communication complexity over the best previously known transformation.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols; Security and privacy → Information-theoretic techniques

Keywords and phrases Oblivious Transfer, Black-Box compiler, Malicious Security, Plain Model

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.5

Related Version *Full Version*: `https://eprint.iacr.org/2022/034.pdf`

Funding *Varun Madathil*: Protocol Labs, NSF grants #1718074,#1764025.
Alessandra Scafuro: Protocol Labs, NSF grants #1718074,#1764025.

1 Introduction

Oblivious Transfer (OT), introduced by Rabin [41], is a core primitive in cryptography. Intuitively, an OT protocol considers a sender with input a pair of strings (s_0, s_1) and a receiver with input a choice bit b. At the end of the protocol, the receiver should learn the string s_b (and nothing more), without the sender obtaining any information about b.

Privacy-only OT. Perhaps, the most basic way to define security of OT is to require that the receiver's messages are computationally indistinguishable when $b = 0$ and when $b = 1$, while the sender's messages computationally hide s_{1-b}. An OT protocol satisfying this property in the presence of *malicious* adversaries is sometimes referred to as *privacy-only* [22]. Privacy-only OT can be constructed from both generic assumptions such as the existence of trapdoor permutations, additively homomorphic encryption and public-key encryption with oblivious public key generation (see, e.g., [2, 15]), and concrete assumptions such as Decisional

Diffie-Hellman [36, 1], Quadratic Residuosity and Decisional Composite Residuosity [21], and Learning with Errors [5]. In particular, *2-round* privacy-only OT protocols against *malicious* adversaries are known under all such assumptions in the *plain model*.

Simulatable OT. Kilian [30] shows that an *ideal* "OT oracle" is sufficient to securely compute *any* cryptographic task. This seminal result has been extended in many different ways [27, 26, 4, 14], thus making OT a central tool in cryptography. Unfortunately, privacy-only OT is not sufficient to instantiate an ideal OT oracle, which instead requires a flavour of security known as *simulatability*. A simulatable OT protocol admits a polynomial-time algorithm, called *simulator*, that is able to fake transcripts of the real protocol without knowing the inputs of the honest parties, and by only having access to the ideal OT oracle.

To make an OT protocol simulatable, intuitively, one needs to augment the protocol with mechanisms that allow a simulator to extract the inputs of the malicious party from the protocol messages, as well as to correctly compute the messages from the honest parties without knowing their inputs. When a setup assumption, such as a Common Reference String (CRS) is available, it is possible to transform privacy-only OT to simulatable OT by embedding special trapdoors in the CRS: the simulator can use these trapdoors to "decrypt"/"equivocate" the protocol messages of the malicious/honest party. Indeed, a rich line of work shows that two rounds are sufficient in order to obtain simulatable OT in the CRS model [13, 28, 40, 9, 11]. Alternatively, 2-round simulatable OT can be obtained in the Random Oracle Model (ROM) [3, 7, 6] where parties have oracle access to a truly-random hash function.

Round-optimal simulatable OT in the plain model. While two rounds are sufficient to build simulatable OT in the CRS model and in the ROM, Katz and Ostrovsky [29] show that *four* rounds are *necessary* for simulatable OT in the plain model. The need for four rounds comes from the fact that, without assuming setup, there are no extraction trapdoors that the simulator can use. Hence, to extract the inputs of the malicious party, the simulator must use rewinding, which requires at least three rounds of communication.[1] In the same paper, Katz and Ostrovsky show that four rounds are also *sufficient*, by providing a simulatable OT protocol from certified trapdoor permutations. Their construction leverages 3-round witness indistinguishable (WI) arguments of knowledge (AoK) and 4-round zero-knowledge (ZK) AoK, to force parties to behave honestly and consistently with the OT protocol.

It is folklore[2] that a similar approach, based on adding general WI/ZK-AoK, could be used to transform any 2-round privacy-only OT to round-optimal simulatable OT. However, the latter would entail unrolling the computation of the underlying OT into a Boolean (or arithmetic) circuit, and using the OT circuit and the transcript as a statement for the WI/ZK proof of consistency; the secret inputs used to compute the transcript would instead be the witnesses for the proof. The AoK property of WI/ZK proofs enables a simulator to extract the witness for the proof by rewinding, whereas the soundness property ensures that the witnesses extracted by the simulator are consistent with the transcript of the underlying privacy-only OT protocol. While very general, such a non-black-box approach requires to unroll the circuit of the OT, hence the complexity of the compiler depends on the circuit complexity of the underlying OT (and not on the security parameter and inputs/outputs of the protocol).

[1] The lower bound of [29] works only for black-box simulators. However, note that even assuming a non-black-box simulator we are not aware of any technique allowing to extract inputs in less than three rounds (unless one assumes non-falsifiable assumptions, which are not considered plain model).

[2] Though we are not aware of any paper formally proving this.

Table 1 Comparing our work to existing compilers to fully simulatable oblivious transfer.

Work	OT from	Black-box Black-box	Plain Model	Optimal Optimal	Information theoretic
[29]	Certified TDPs	✗	✓	✓	✗
[20]	Semi-honest OT	✓	✓	✗	✗
[37]	Certified TDPs	✓	✓	✓	✗
[12]	Strongly-uniform OT	✓	✓	✓	✗
[10]	TDPs	✓	✓	✓	✗
This work	Defensible OT	✓	✓	✓	✓
This work	Semi-honest OT	✓	✗	✓	✓

A central question in cryptography is to understand whether a given task can be performed having only *black-box* access[3] to other cryptographic primitives. Thus, it is natural to ask whether one can provide a general black-box compiler from privacy-only OT to (possibly round-optimal) simulatable OT.

Black-box round-optimal simulatable OT in the plain model. When treating the underlying OT as a black-box the main challenge is to add an extraction/rewinding mechanism, from which the simulator can extract values that are *consistent* with the actual secret inputs played in the OT protocol, *and* such that the output of the honest parties is distributed identically in the real and ideal world.[4] In fact, Lindell and Pinkas [34] show that there are input-dependent attacks that emerge uniquely in the black-box approach. Haitner, Ishai, Kushilevitz, Lindell, and Petrank [19, 20] showed that input-extraction and input-consistency are possible to achieve via cut-and-choose techniques, but unfortunately their compiler requires at least 12 rounds (assuming some steps can be parallelized).

In a different work, Ostrovsky, Richelson and Scafuro [37] provided a 4-round simulatable OT from black-box use of certified trapdoor permutations. Very recently, Friolo, Masny and Venturi [12] greatly generalized the approach of [37] by exhibiting a compiler that transforms any *strongly-uniform* 2-round OT protocol into a 4-round simulatable OT. Strong uniformity means that the messages sent by the receiver appear computationally indistinguishable from random to a malicious sender.[5] Importantly, as shown in [12], strongly-uniform OT can be instantiated from the most common number-theoretic assumptions (e.g., DDH, CDH, LPN, Subset Sum, and LWE).

Yet, the compiler of [12] inherits the complexity and significant overhead of its predecessor. Indeed, as in [37], it entails two intermediate transformations: one for achieving simulatability against malicious receivers, and one for achieving simulatability against malicious senders. Each transformation is somewhat complex and requires the use of extractable commitments (to extract the inputs) and black-box commit-and-prove of equality (in particular, a modification of the one built by Kilian [30]) to enforce input consistency w.r.t. the underlying OT. In particular, the resulting communication complexity of this compiler is *quartic* in the security parameter (i.e., $O(|\mu_R|\lambda^4 + 2|\mu_S|\lambda)$ where μ_R is the message sent by the receiver and μ_S is the message sent by the sender in the underlying 2-round OT protocol).

[3] Black-box means that the underlying OT is treated as an oracle.
[4] Note that this consistency property comes for free when using the underlying OT protocol in a non-black-box way, since the protocol transcript is part of the statement of the WI/ZK proof.
[5] Specifically, they require an OT protocol that is strongly-uniform against malicious senders and simulatable in presence of semi-honest receivers.

More recently, Choudhuri, Ciampi, Goyal, Jain and Ostrovsky [10] constructed 4-round simulatable OT from black-box use of trapdoor permutations. Their construction still relies on the inefficient transformation from [37] to go from one-sided simulatable to fully simulatable OT.

Hence, the question:

Does there exist an efficient, information-theoretic, black-box transform from privacy-only OT to round-optimal simulatable OT in the plain model?

1.1 Our Contribution

In this work, we answer the above question in the positive by providing a simple, black-box, information-theoretic, transformation turning any privacy-only 2-round OT protocol into a round-optimal simulatable OT protocol. We elaborate on our contributions in more details below.

An information-theoretic transform. Our transformation does not require any additional cryptographic assumption besides the existence of privacy-only OT. As a result, our approach is much simpler than previous work and, in fact, yields an improved communication complexity of $O(|\mu_R|\lambda + |\mu_S|\lambda + \lambda^2)$. In particular, we prove the following theorem:

▶ **Theorem 1** (Main Theorem, informal)**.** *There is a black-box information-theoretic transformation from any 2-round privacy-only OT protocol to a 4-round simulatable OT protocol in the plain model.*

Towards assuming semi-honest privacy only. From a theoretical perspective, the holy grail in this line of research would be to build a round-optimal simulatable OT protocol that makes black-box usage of any *semi-honest* privacy-only 2-round OT protocol. While we do not settle this question in the plain model, we do give a positive answer in the ROM as explained below.

First, we observe that our compiler only requires privacy against *defensible* receivers [20], which is a weaker flavour of privacy than privacy only OT, which is private against malicious receivers. Defensible here refers to the fact that, while a malicious receiver can cheat in the protocol and learn both inputs of the sender without being detected, it should be hard to later convince the sender that it behaved honestly. Second, we prove that any 2-round semi-honest privacy-only OT is necessarily private against malicious senders.

Putting together the above two observations, we can plug into our transform any 2-round OT protocol that is both: (i) private against defensible receivers, and (ii) private against semi-honest senders. Next, we show that in the ROM we can relax the security requirements for the underlying OT even further to just requiring semi-honest privacy against both the sender and the receiver. More in details, we exhibit a transformation turning any 2-round semi-honest privacy-only OT into one satisfying properties (i) and (ii) above. Our transform is round-preserving, and simply requires the receiver to use randomness derived from the output of the random oracle. The programmability of the random oracle is used only in the reduction.

As mentioned earlier, it is well known that if the simulator is allowed to both observe and program the random oracle, it is fairly easy to build a simulatable OT protocol. Yet, since in our transform the random oracle is used only to lift (game-based) privacy against semi-honest receivers to (game-based) privacy against defensible receivers, we believe the

gap to fill towards a result in the plain model is much narrower. In other words, future work must only focus on finding a round-preserving transformation from OT with privacy against semi-honest receivers to OT with privacy against defensible receivers in the plain model.

1.2 Our Techniques

As mentioned above, in the black-box setting, the main challenge towards obtaining simulatable OT is to design extraction mechanisms which allow the simulator to extract inputs that are consistent with the ones played by the parties in the real world. The latter is particularly challenging when only 4 rounds of communication are available.

The compiler of Friolo, Masny and Venturi [12] achieves extractability and input consistency by adding black-box commit-and-prove proofs of consistency. In particular, they rely on such proofs for two reasons: (1) to force the receiver to sample one of the messages for the underlying OT protocol uniformly, and (2) to force the sender to create a valid secret sharing of its inputs (without opening the way to input-dependent abort attacks against the receiver), while allowing the simulator to successfully reconstruct both inputs.

In this work, we take a completely different approach. In particular, instead of adding mechanisms to *force* good behaviour, we only add *publicly-verifiable* checkpoints to *assess* good behavior. Public verifiability here means that the checkpoints are verifiable by looking only at the protocol transcript (without requiring access to secret inputs), which avoids attacks based on input-dependent aborts. As a result, we can enforce both extractability and indistinguishability of the simulation, by simply having parties justify some of their actions (when challenged). Thanks to this feature, our transform does not require any additional cryptographic primitives (e.g., commitments), and can be based just on privacy-only OT and threshold secret sharing.

Overview of our compiler. The sender and the receiver engage in m parallel *sessions* $(\mu_R^{(i)}, \mu_S^{(i)})$ of the underlying 2-round privacy-only OT protocol, using uniformly random inputs: the receiver uses random choice bits $b^{(1)}, \ldots, b^{(m)}$, whereas the sender uses pairs of random keys $(\kappa_0^{(1)}, \kappa_1^{(1)}), \ldots, (\kappa_0^{(m)}, \kappa_1^{(m)})$. This results in messages $\mu_R^{(i)}$ which are sent from the receiver to the sender in the first round of the compiled protocol.

In the second round, the sender responds to the messages received from the receiver with its own messages $\mu_S^{(i)}$ (computed via the underlying 2-round OT protocol), but also selects a random subset $\mathcal{A}$ of t_R indices in $[m]$ for cut-and-choose: In the third round, for each $i \in \mathcal{A}$, the receiver is asked to provide the randomness $\rho_R^{(i)}$ and the input $b^{(i)}$ (we call these a *defense*) that explain the message $\mu_R^{(i)}$ sent in the i-th session. Additionally, the receiver selects a random subset $\mathcal{B}$ of t_S indices in $[m] \setminus \mathcal{A}$ and forwards $\mathcal{B}$ and the defenses $(\delta_R^{(i)})_{i \in \mathcal{A}}$ along with a bit $d^{(i)} = b \oplus b^{(i)}$ for each of the $n = m - t_R - t_S$ sessions that were not selected for cut and choose (we call those the *alive sessions*). The bit $d^{(i)}$ allows to adjust the bit $b^{(i)}$ in the i-th session to the choice bit b of the receiver.

We note that, since the underlying OT satisfies privacy against malicious senders, the random bits $b^{(i)}$ used by the receiver are computationally hidden, which implies the adjusting bits $d^{(i)}$ are computationally close to uniform and hide the receiver's choice bit b to the eyes of a computationally-bounded malicious sender. Moreover, observe that at the end of the execution of the underlying 2-round OT sessions, the receiver might have already noticed that some of the messages $\mu_S^{(i)}$ played by the sender are "bad", in the sense that they yield an invalid output (we do not make any assumption about how the underlying OT protocol deals with bogus inputs). Hence, in such a case, it seems the receiver should

just abort (and righteously so) instead of continuing with the protocol. Doing so, however, opens the door to attacks based on input-dependent abort. For instance, the sender could plant a single $\perp$ in, say, the j-th session, by playing with the inputs $(\kappa_0^{(j)}, \perp)$, and thus learning that $b^{(j)} = 0$ in case the receiver did not abort. Later, after observing $d^{(j)}$, the sender can compute $b = d^{(j)} \oplus 0$ which is a clear security breach. To prevent this type of attack, in our compiler we never let parties abort depending on their local view. (In fact, up to this point, in our protocol the parties eventually abort only after checking the responses to cut-and-choose challenges, which do not involve secret inputs.)

In the fourth (and last) round, for each $i \in \mathcal{B}$, the sender reveals the randomness $\rho_S^{(i)}$ and the inputs $(\kappa_0^{(i)}, \kappa_1^{(i)})$ that explain the message $\mu_S^{(i)}$ sent in the i-th session. Moreover, it uses the n pairs of keys $(\kappa_0^{(i)}, \kappa_1^{(i)})$ corresponding to each alive session to mask n shares $s_0^{(i)}$ and $s_1^{(i)}$ of the actual secret inputs s_0 and s_1, yielding ciphertexts $(\gamma_0^{(i)}, \gamma_1^{(i)})$; here, we make use of the bits $d^{(i)}$ sent by the receiver to align the shares $s_0^{(i)}$ and $s_1^{(i)}$ with the keys obtained by the receiver in the i-th session. Namely, the share $s_0^{(i)}$ (resp. $s_1^{(i)}$) is encrypted using key $\kappa_{d^{(i)}}^{(i)}$ (resp. $\kappa_{1\oplus d^{(i)}}^{(i)}$) to create the ciphertext $\gamma_{d^{(i)}}^{(i)}$ (resp. $\gamma_{1\oplus d^{(i)}}^{(i)}$).

After checking the defenses from the sender, for each alive session, the receiver decrypts the ciphertext $\gamma_{b^{(i)}}^{(i)}$ using the key $\kappa_{b^{(i)}}^{(i)}$ previously obtained as output in the i-th session, which yields the i-th share $s_b^{(i)}$ of s_b and thus allows to reconstruct s_b using any subset of t shares (where t is the minimum number of shares required for reconstruction in the underlying secret sharing scheme).

We will show how to choose the parameters m, t_R, t_S and t when discussing the simulator below. Note that the receiver does not check if different subsets of shares lead to different secrets, neither it aborts if in some of the alive sessions it retrieves a bogus string. The only other aborting case in the real world would be when the receiver gets less than t "valid keys", and thus shares, which however we bound to happen with negligible probability if the number of parallel sessions and the cut-and-choose parameters are set appropriately.

Simulator for malicious receivers. Let R^* be a malicious receiver. The simulator Sim starts by running R^* and thus receiving the messages $(\mu_R^{(i)})_{i\in[m]}$ that the receiver sends in the first round. Hence, it can perfectly simulate the second round $((\mu_S^{(i)})_{i\in\mathcal{A}}, \mathcal{A})$ of the protocol using pairs of random keys $(\kappa_0^{(i)}, \kappa_1^{(i)})$ for each of the sessions $i \in [m] \setminus \mathcal{A}$ (as the honest sender would do).

Next, R^* replies with $((\delta_R^{(i)})_{i\in\mathcal{A}}, d_{i\in\mathsf{Alive}}^{(i)}, \mathcal{B})$ where $\mathsf{Alive} = [m] \setminus (\mathcal{A} \cup \mathcal{B})$ contains all the indices corresponding to alive sessions. We call the execution up to this point the *main thread*. Now, after checking the defenses from the receiver are good, Sim rewinds R^* and forwards it a freshly sampled second round $((\mu_S^{(i)})_{i\in\mathcal{A}'}, \mathcal{A}')$. This process is repeated until the cut-and-choose sets $\mathcal{A}'$ allows the simulator to obtain good defenses for at least $2/3$ of the the alive sessions in the main thread. The Sim aborts whenever there are more than $m/9$ bad defenses.

A combinatorial analysis shows that setting $m = O(\lambda)$ and $t_R = t_S = m/3$ suffices in order to ensure that: (i) the simulator runs in expected polynomial time, and (ii) the simulator aborts with probability that is negligibly close to the probability that the honest party would have aborted in the real world.

At this point, Sim can extract a bit $b^{(i)}$ for $2/3$ of the alive sessions. This in turn allows to define b as the value $\hat{b}^{(i)} = b^{(i)} \oplus d^{(i)}$ that appears at least $t/2$ times, where $t = 2/3|\mathsf{Alive}| = 2m/9$. Note that at this point the simulator will have received good defenses

for at least 2/3 of the alive sessions, otherwise the simulator will have aborted. The simulator forwards b to the OT ideal functionality and completes the simulation with R^* by using the value s_b returned from the functionality, along with a uniformly random string s'_{1-b}.

The rationale behind the above simulation strategy is that whenever a bit b appears more than $t/2$ times the adversary can only learn the value s_b or nothing at all. Note that R^* requires at least t shares to compute s_{1-b}. Out of the $m/3$ sessions in Alive, assume R^* is able to learn both strings for $|\mathsf{Alive}| - t = m/9$ shares. Now if there exist greater than $t/2 = m/9$ shares for the bit b, then there exist at most $m/9 - 1$ shares for s_{1-b}. Thus the adversary is only able to learn at most $m/9 + (m/9 - 1)$ shares of s_{1-b}. This allows the simulator to randomly sample s_{1-b} in the simulation. On the other hand if the adversary plays honestly it learns s_b as in the real-world protocol.

Simulator for malicious senders. The simulator Sim for the case of malicious senders is based on similar ideas. Let S^* be a malicious sender. This time, Sim starts by sampling the first round $(\mu_R^{(i)})_{i\in[m]}$ exactly as the honest receiver would do, upon which S^* replies with $((\mu_S^{(i)})_{i\in[m]\setminus\mathcal{A}}, \mathcal{A})$. Hence, the simulator generates the third round $((\delta_R^{(i)})_{i\in\mathcal{A}}, (d^{(i)})_{i\in\mathsf{Alive}}, \mathcal{B})$ as the honest receiver would do, except that the bits $d^{(i)}$ are picked uniformly at random (as Sim does not know b).

Next, the malicious sender sends the final round $((\delta_S^{(i)})_{i\in\mathcal{B}}, (\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}})$. Now, after checking the defenses from the sender are good, Sim rewinds S^* and forwards it a freshly sampled third round $((\delta_R^{(i)})_{i\in\mathcal{A}}, (d^{(i)})_{i\in\mathsf{Alive}'}, \mathcal{B}')$ where $\mathsf{Alive}' = [m] \setminus (\mathcal{A} \cup \mathcal{B}')$. This process is repeated until the cut-and-choose sets $\mathcal{B}'$ allows the Sim to obtain defenses $\delta_S^{(i)} = ((\kappa_0^{(i)}, \kappa_1^{(i)}), \rho_S^{(i)})$ for 2/3 of the alive sessions in the main thread. An analysis similar to the case of malicious receivers shows that Sim runs in expected polynomial time and aborts only with negligible probability.

At this point, the simulator can use the keys $(\kappa_0^{(i)}, \kappa_1^{(i)})$ for each index corresponding to an alive session for which S^* showed a good defense, in order to decrypt both ciphertexts $(\gamma_0^{(i)}, \gamma_1^{(i)})$. This allows Sim to extract both s_0 and s_1, after re-aligning the index of the shares consistently with the values $d^{(i)}$ used in the simulation of the main thread. Moreover, the fact that the underlying OT protocol satisfies privacy against malicious senders ensures that the bits $b^{(i)}$ used by the simulator in the alive session of the main thread are indistinguishable from random, and thus so are the values $d^{(i)}$ (as in the simulation).

Defensible privacy and the ROM. It is not hard to see that the proof for the case of malicious receivers actually only requires the underlying OT protocol to satisfy privacy against defensible (rather than malicious) receivers. Intuitively, this is because the sender can check the defenses of the receiver *before* sending the messages that contain its actual input.[6]

While many known constructions of 2-round OT satisfy the stronger property of privacy against malicious receivers, we observe that in the ROM one can obtain privacy against defensible receivers from any OT protocol with semi-honest privacy. The idea is to simply force the receiver to use good randomness by hashing a random string along with the choice bit.

[6] Unfortunately, the opposite is not true as the receiver obtains the defenses from the sender *after* it has already sent the bits $d^{(i)}$. This is the reason why we need to assume privacy against malicious (rather than defensible) senders for the underlying OT protocol. Nevertheless, recall that we also show the latter property comes for free in the case of 2-round protocols.

1.3 Comparison with Friolo et al. [12]

Besides using a very different approach, which leads to a significantly more efficient compiler, the main difference between our compiler and the one by Friolo et al. [12] is in terms of starting assumptions. Our compiler starts from any 2-round privacy-only OT, whereas [12] starts from any 2-round strongly-uniform OT. While is tempting to consider the latter as a much weaker assumption than the former and hence the resulting compiler more general, we notice that the two assumptions are somewhat incomparable. To appreciate the difference, first, consider an edge case where we try to instantiate the two compilers using any 2-round *universally composable* OT protocol. Since universal composability does not necessarily[7] imply strong uniformity, the compiler of [12] would not work. In contrast, our compiler would work as universal composability implies privacy-only.

Also, it is not hard to see that 2-round strongly-uniform OT is not implied by a 2-round semi-honest OT. Indeed, it is easy to come up with a 2-round semi-honest OT that is not strongly uniform. For example, consider the classical 2-round OT construction based on PKE with oblivious key generation instantiated with a contrived PKE where each public key is concatenated with a dummy bit that always equals zero. This PKE scheme still implies a 2-round semi-honest OT, yet the distribution of public keys is far from uniform (and thus the OT protocol is not strongly uniform).

We do acknowledge however that building a 2-round strong uniform semi-honest OT protocol appears to be an easier task than building a 2-round privacy-only OT protocol.

One may wonder whether the compiler of [12] can be simplified if starting with a 2-round privacy-only OT, instead of a strong-uniform OT. For instance, by removing the burden of the commit-and-open protocol on the receiver side. We note that the commit-and-open protocol in [12] is essential to achieve input extraction for both sides, which is required in order to prove simulation-based security. Hence, even when starting with a privacy-only OT protocol, the compiler of [12] cannot forgo the use of commit-and-open (or some other mechanism) for the simulator to extract the inputs.

We refer the reader to the full version[35] for a more detailed comparison between the efficiency of our compiler and the one of [12] in terms of communication complexity. For the sake of concreteness, we also discuss there an explicit instantiation of both compilers in the plain model, based on the hardness of LWE. In this case, we can instantiate our compiler using the 2-round OT protocol by Brakerski and Döttling [5] (which satisfies privacy against malicious, and hence defensible, receivers and semi-honest senders, and thus suffices for our compiler in Theorem 1). For the compiler in [12], instead, we can use the 2-round strongly-uniform OT protocol based on the PKE scheme by Peikert et al. [40]. As we show, this results in a communication complexity of $\tilde{O}(\lambda^3)$ for our compiler, against $\tilde{O}(\lambda^6)$ for the compiler of [12] (where λ is the security parameter).

1.4 Related Work

The compilers in [37, 12, 10] yield the only known round-optimal black-box constructions of simulatable OT in the plain model. In this section, we survey other relevant work on black-box constructions for two-party functionalities. Lindell, Oxman and Pinkas [33], relying on ideas from Ishai, Prabhakaran and Sahai [27], provide a significantly more efficient black-box compiler from semi-honest OT to malicious OT, which however takes at least 8 rounds

[7] See, e.g., [11] for a concrete example of a 2-round universally composable OT protocol that is not strongly uniform.

(and thus is not round-optimal). Pass and Wee [38] build commitment and zero-knowledge protocols from black-box access to one-way functions. Hazay and Venkitasubramaniam [24] improve this result by giving a round-optimal construction. More recently, a rich body of work [44, 39, 16] culminated in round-optimal black-box constructions for non-malleable commitments [17, 18], and commit-and-prove [31].

In the CRS model, Choi, Dachman-Soled, Malkin and Wee [8] show a black-box compiler from adaptive semi-honest OT into constant-round adaptive UC-secure two-party computation. Kiyoshima, Lin and Venkitasubramaniam in [32] provide a unified approach to build black-box protocols under trusted setup assumptions, improving on a previous approach by Hazay and Venkitasubramaniam [25]. These works are not round-optimal and are not in the plain model.

2 Preliminaries

Notation. We denote with $\lambda \in \mathbb{N}$ the security parameter. For $n \in \mathbb{N}$, we let $[n] = \{1, \ldots, n\}$. A negligible function, denoted $\mathsf{negl}(\lambda)$, is a function that vanishes faster than the reciprocal of any polynomial $\mathsf{poly}(\lambda)$. We use standard notation for computational/statistical indistinguishability of distribution ensembles.

For an interactive protocol Π between parties A, B holding inputs x, y respectively, we denote the transcript of a protocol execution as $\langle A(x), B(y)\rangle$. Additionally, we let $\mathbf{View}^B_{\Pi,A}(\lambda, x, y)$ be the random variable corresponding to the view of A in a run of Π with input x when interacting with B with input y. This view consists of A's input, randomness, and messages received.

Oblivious Transfer. Oblivious transfer (OT) is a two-party protocol Π in which a sender S has two input strings $s_0, s_1 \in \{0,1\}^\lambda$, and a receiver R has a choice bit $b \in \{0,1\}$. An OT protocol is called *non-trivial* if for any pair of strings $s_0, s_1 \in \{0,1\}^\lambda$, and for any $b \in \{0,1\}$, after participating in the interactive protocol, S outputs nothing and R learns s_b. Below, we recall relevant security notions for OT protocols.

Simulatable OT. The standard security definition for OT compares an execution of Π in the real world, where an attacker can corrupt either the sender S or the receiver R, with an execution in the ideal world where a trusted-third party knows all inputs and computes the output on behalf of the players. The corresponding ideal functionality is depicted in Figure 1. In what follows, we denote by $\mathbf{Real}_{\Pi,R^*(z)}(\lambda, s_0, s_1, b)$ (resp., $\mathbf{Real}_{\Pi,S^*(z)}(\lambda, s_0, s_1, b)$) the output of the malicious receiver R^* (resp., sender S^*) during a real execution of the protocol Π (with s_0, s_1 as inputs of the sender, b as choice bit of the receiver, and z as auxiliary input for the adversary), and by $\mathbf{Ideal}_{\mathcal{F}_{\mathsf{OT}},\mathsf{Sim}^{R^*(z)}}(\lambda, s_0, s_1, b)$ (resp., $\mathbf{Ideal}_{\mathcal{F}_{\mathsf{OT}},\mathsf{Sim}^{S^*(z)}}(\lambda, s_0, s_1, b)$) the output of the malicious receiver R^* (resp., sender S^*) in an ideal execution where the parties (with analogous inputs) interact with $\mathcal{F}_{\mathsf{OT}}$, and where the simulator is given black-box access to the adversary.

▶ **Definition 2** (Simulatable OT). *We say that* $\Pi = (S, R)$ securely computes $\mathcal{F}_{\mathsf{OT}}$ *if the following holds:*

- *For every non-uniform PPT malicious receiver* R^**, there exists a non-uniform PPT simulator* Sim *such that*

$$\left\{\mathbf{Real}_{\Pi,R^*(z)}(\lambda, s_0, s_1, b)\right\}_{\lambda,s_0,s_1,b,z} \overset{c}{\approx} \left\{\mathbf{Ideal}_{\mathcal{F}_{\mathsf{OT}},\mathsf{Sim}^{R^*(z)}}(\lambda, s_0, s_1, b)\right\}_{\lambda,s_0,s_1,b,z},$$

where $\lambda \in \mathbb{N}, s_0, s_1 \in \{0,1\}^\lambda, b \in \{0,1\}$*, and* $z \in \{0,1\}^*$.

Ideal Functionality $\mathcal{F}_{\mathsf{OT}}$:

- Upon receiving message $(\texttt{send}, s_0, s_1, S, R)$ from S, where $s_0, s_1 \in \{0,1\}^\lambda$, store s_0, s_1 and answer `send` to R and Sim.
- Upon receiving message $(\texttt{receive}, b)$ from R, where $b \in \{0,1\}$, send s_b to R and `receive` to S and Sim, and halt. If no message $(\texttt{send}, \cdot)$ was previously sent, do nothing.

Figure 1 Ideal functionality for oblivious transfer.

- *For every non-uniform PPT malicious sender S^*, there exists a non-uniform PPT simulator* Sim *such that*

$$\left\{\mathbf{Real}_{\Pi,S^*(z)}(\lambda, s_0, s_1, b)\right\}_{\lambda,s_0,s_1,b,z} \overset{c}{\approx} \left\{\mathbf{Ideal}_{\mathcal{F}_{\mathsf{OT}},\mathsf{Sim}^{S^*(z)}}(\lambda, s_0, s_1, b)\right\}_{\lambda,s_0,s_1,b,z},$$

where $\lambda \in \mathbb{N}, s_0, s_1 \in \{0,1\}^\lambda, b \in \{0,1\}$, and $z \in \{0,1\}^*$.

Privacy-Only OT. A weaker guarantee than simulatable security is the so-called *privacy* property, which does not require the existence of a simulator. Roughly, privacy for the receiver means that the choice bit is computationally hidden, whereas privacy for the sender means that the receiver can obtain at most one input from the sender. Below, we formalize this notion for different adversarial behaviours and assuming that the input of the players are uniformly random (which will suffice for our purpose).

Semi-honest privacy. The definition below formalizes privacy in the semi-honest setting (i.e., when corrupted parties do not deviate from the protocol).

▶ **Definition 3** (Semi-honest privacy for random inputs)**.** *Let $\Pi = (S, R)$ be a non-trivial OT protocol. We say that $\Pi = (S, R)$ is* private for random inputs against semi-honest receivers *if the following holds:*

$$\left\{\mathbf{View}^S_{\Pi,R}(\lambda, s_0, s_1, b), s_{1-b}\right\}_{\lambda,b} \overset{c}{\approx} \left\{\mathbf{View}^S_{\Pi,R}(\lambda, s_0, s_1, b), s'\right\}_{\lambda,b}$$

where $\lambda \in \mathbb{N}$, $b \in \{0,1\}$, and $s_0, s_1, s' \leftarrow\!\!\$\ \{0,1\}^\lambda$. Similarly, Π is private for random inputs against semi-honest senders *if the following holds:*

$$\left\{\mathbf{View}^R_{\Pi,S}(\lambda, s_0, s_1, b), b\right\}_{\lambda,s_0,s_1} \overset{c}{\approx} \left\{\mathbf{View}^R_{\Pi,S}(\lambda, s_0, s_1, b), b'\right\}_{\lambda,s_0,s_1}$$

where $\lambda \in \mathbb{N}$, $s_0, s_1 \in \{0,1\}^\lambda$, and $b, b' \leftarrow\!\!\$\ \{0,1\}$.

Malicious privacy. The definition below (adapted from [23]) formalizes privacy in the malicious setting (i.e., when corrupted parties can arbitrarily deviate from the protocol).

▶ **Definition 4** (Malicious privacy)**.** *Let $\Pi = (S, R)$ be a non-trivial OT protocol.*

Π is private for random inputs against malicious senders *if for every non-uniform PPT malicious sender S^*:*

$$\left\{\mathbf{View}^R_{\Pi,S^*(z)}(\lambda, s_0, s_1, b), b\right\}_{\lambda,s_0,s_1,z} \approx_c \left\{\mathbf{View}^R_{\Pi,S^*(z)}(\lambda, s_0, s_1, b), b'\right\}_{\lambda,s_0,s_1,z}$$

where $\lambda \in \mathbb{N}$, $s_0, s_1 \in \{0,1\}^\lambda$, $z \in \{0,1\}^*$, and $b, b' \leftarrow\!\!\$\ \{0,1\}$.

Defensible privacy. Following Haitner et al. [19, 20], we call *defense* by R^* an input $b \in \{0,1\}$ and a random tape $\rho_R \in \{0,1\}^*$ provided by the receiver at the end of the protocol. Intuitively, a defense is *good* if the honest receiver using this very input and randomness would have sent the exact same messages as the malicious receiver sent. A similar notion can be considered for the sender, where defenses are of the form (s_0, s_1, ρ_S) with $s_0, s_1 \in \{0,1\}^\lambda$ and $\rho_S \in \{0,1\}^*$.

The definition below formalizes the concept of good defense in the special case of 2-round OT protocols, which we model as follows. Let OTR a PPT algorithm taking as input the choice bit $b \in \{0,1\}$ and the random tape $\rho_R \in \{0,1\}^*$ for the receiver, and outputting a message $\mu_R \in \{0,1\}^*$ for the sender; similarly, let OTS be a PPT algorithm taking as input the strings $s_0, s_1 \in \{0,1\}^\lambda$ and the random tape $\rho_S \in \{0,1\}^*$ for the sender, as well as a message $\mu_R \in \{0,1\}^*$ from the receiver, and outputting a message $\mu_S \in \{0,1\}^*$ for the receiver. Finally, let OTD be a PPT algorithm taking as input the choice bit $b \in \{0,1\}$ and the random tape $\rho_R \in \{0,1\}^*$ for the receiver, as well as message μ_S from the sender, and outputting a value s in $\{0,1\}^\lambda$.

▶ **Definition 5** (Good defense for 2-round OT)**.** *Let $\Pi = (\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ be a 2-round OT protocol. Fix any transcript $\tau = (\mu_R, \mu_S) \in (\{0,1\}^*)^2$ for Π. We say that the pair $\delta_R = (b, \rho_R)$ (resp. $\delta_S = (s_0, s_1, \rho_S)$), where $b \in \{0,1\}$, constitutes a* good defense *by the receiver (resp. by the sender) for τ in Π if it holds that $\mu_R = \mathsf{OTR}(b; \rho_R)$ (resp. $\mu_S = \mathsf{OTS}(s_0, s_1, \mu_R; \rho_S)$).*

Loosely speaking, an OT protocol has defensible privacy if the privacy property holds against malicious adversaries that can provide a good defense.

▶ **Definition 6** (Defensible privacy for random inputs)**.** *Let $\Pi = (S, R)$ be a non-trivial OT protocol. We say that Π is* private for random inputs against defensible receivers *if for every non-uniform PPT malicious receiver R^*:*

$$\left\{\Gamma\left(\mathbf{View}^{S}_{\Pi,R^*(z)}(\lambda,(s_0,s_1),b),s_{1-b}\right)\right\}_{\lambda,b,z} \approx_c \left\{\Gamma\left(\mathbf{View}^{S}_{\Pi,R^*(z)}(\lambda,s_0,s_1,b),s'\right)\right\}_{\lambda,b,z}$$

where $\lambda \in \mathbb{N}$, $b \in \{0,1\}$, $z \in \{0,1\}^*$, $s_0, s_1, s' \leftarrow\!\!\$\ \{0,1\}^\lambda$, and $\Gamma(v,*)$ is set to $(v,*)$ if following the execution R^* outputs a good defense (and to $\perp$ otherwise). Similarly, Π is private for random inputs against defensible senders *if for every non-uniform PPT malicious sender S^*:*

$$\left\{\Gamma\left(\mathbf{View}^{R}_{\Pi,S^*(z)}(\lambda,(s_0,s_1),b),b\right)\right\}_{\lambda,s_0,s_1,z} \approx_c \left\{\Gamma\left(\mathbf{View}^{R}_{\Pi,S^*(z)}(\lambda,s_0,s_1,b),b'\right)\right\}_{\lambda,s_0,s_1,z}$$

where $\lambda \in \mathbb{N}$, $s_0, s_1 \in \{0,1\}^\lambda$, $z \in \{0,1\}^*$, $b, b' \leftarrow\!\!\$\ \{0,1\}$, and $\Gamma(v,*)$ is set to $(v,*)$ if following the execution S^* outputs a good defense (and to $\perp$ otherwise).

Secret Sharing. A threshold secret sharing scheme allows to share an input string $s \in \{0,1\}^\lambda$ into n shares $s_1, \ldots, s_n \in \{0,1\}^\lambda$ in such a way that it is possible to efficiently recover s from any subset of at least t shares, while at the same time an attacker corrupting up to $t-1$ share holders obtains no information about the secret.

▶ **Definition 7** (Secret sharing)**.** *An (n,t)-*secret sharing scheme *over $\{0,1\}^\lambda$ is defined by a pair of algorithms $(\mathsf{Share}, \mathsf{Recon})$, where Share is a randomized mapping of an input $s \in \{0,1\}^\lambda$ to shares $s = (s_1, s_2, \ldots, s_n) \in (\{0,1\}^\lambda)^n$, and Recon is a function mapping a subset $\mathcal{I}$ of $[n]$, along with the corresponding shares $s_\mathcal{I} = (s_i)_{i\in\mathcal{I}}$, to a value in $\{0,1\}^\lambda$, such that the following holds:*

1. **Reconstruction.** *For all $s \in \{0,1\}^\lambda$, and for all sets $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| \geq t$, the output of $\mathsf{Recon}(\mathcal{I}, s_\mathcal{I})$ such that $(s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(s)$ is equal to s.*

2. **Security.** *For all $s \in \{0,1\}^\lambda$, and for all sets $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| < t$, the joint distribution $s_\mathcal{I} = (s_i)_{i\in\mathcal{I}}$ of shares received by the subset of parties $\mathcal{I}$, where $(s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(s)$, is independent of the secret s.*

We call a share *valid* if it is a λ-bit string. For our construction, we will implicitly assume that running algorithm Recon upon input any sequence of t valid shares (possibly outside the support of Share) still yields a λ-bit message. Note that, e.g., Shamir's secret sharing [42] satisfies this property.

3 Observations on Two-Round OT

Here, we collect a few observations on 2-round OT protocols. First, in Section 3.1, we show that any 2-round OT protocol with privacy against semi-honest senders is already private against malicious senders. Next, in Section 3.2, we overview existing 2-round OT protocols that satisfy the notion of privacy against defensible receivers. Furthermore, we show a simple compiler in the ROM that adds the latter property to any 2-round OT protocol with semi-honest privacy only.

3.1 Privacy against Malicious Senders

The lemma below states that any 2-round OT protocol with privacy against semi-honest senders is already private against malicious senders.[8] Intuitively, this is the case since in a 2-round OT protocol the only thing a sender can do is to respond to the first message sent by the receiver, however, this does not help to learn the choice bit of the receiver. While we prove the lemma for the case of privacy with random inputs, a similar statement holds for any distribution of the receiver's choice bit.

▶ **Lemma 8.** *Any 2-round OT protocol that is private (for random inputs) against semi-honest senders is also private (for random inputs) against malicious senders.*

We defer the proof of this lemma to the full version [35].

3.2 Privacy against Defensible Receivers

Two-round OT protocols (for random inputs) with privacy against defensible receivers exist under standard number-theoretic assumptions, including DDH [36, 1], QR and DCR [21], and LWE [5]. In fact, these protocols satisfy the stronger property of privacy against *malicious* receivers.

In this section, we present a round-preserving transform turning any 2-round OT protocol with semi-honest privacy into one with privacy against defensible receivers in the ROM. Intuitively, our transform ensures that the receiver cannot influence the randomness used to generate the first OT message. This is achieved by hashing the choice bit b along with a random string ρ_R chosen by the receiver. We refer the reader to the full version [35] for a formal proof of the statement below.

▶ **Lemma 9.** *If there exists a 2-round OT protocol with privacy (for random inputs) against semi-honest senders and receivers, then there exists a 2-round OT protocol with privacy (for random inputs) against defensible receivers and semi-honest senders in the ROM.*

[8] A similar observation appears in [12, p. 12].

Protocol $\Pi = (S, R)$

Sender's Input: $s_0, s_1 \in \{0,1\}^\lambda$ Receiver's Input: $b \in \{0,1\}$

1. $(R \to S)$: For each $i \in [m]$, the receiver picks $b^{(i)} \leftarrow\$ \{0,1\}$ and $\rho_R^{(i)} \leftarrow\$ \{0,1\}^*$, lets $\mu_R^{(i)} = \mathsf{OTR}(b^{(i)}; \rho_R^{(i)})$, and sends $(\mu_R^{(i)})_{i\in[m]}$ to the sender.
2. $(S \to R)$: The sender's first message is computed as follows.
   a. Sample $\alpha_1, \ldots, \alpha_{t_R} \leftarrow\$ [m]$ and let $\mathcal{A} = \{\alpha_1, \ldots, \alpha_{t_R}\}$.
   b. For each $i \in [m] \setminus \mathcal{A}$, pick $\kappa_0^{(i)}, \kappa_1^{(i)} \in \{0,1\}^\lambda$ and $\rho_S^{(i)} \leftarrow\$ \{0,1\}^*$, and let $\mu_S^{(i)} = \mathsf{OTS}(\kappa_0^{(i)}, \kappa_1^{(i)}, \mu_R^{(i)}; \rho_S^{(i)})$.

   Send $(\mu_S^{(i)})_{i\in[m]\setminus\mathcal{A}}$ and $\mathcal{A}$ to the receiver.
3. $(R \to S)$: The receiver's second message is computed as follows.
   a. For each $i \in \mathcal{A}$, let $\delta_R^{(i)} = (\rho_R^{(i)}, b^{(i)})$.
   b. Sample $\beta_1, \ldots, \beta_{t_S} \leftarrow\$ [m] \setminus \mathcal{A}$ and let $\mathcal{B} = \{\beta_1, \ldots, \beta_{t_S}\}$.
   c. Let $\mathsf{Alive} = [m] \setminus (\mathcal{A} \cup \mathcal{B})$. For each $i \in \mathsf{Alive}$, let $\kappa_{b^{(i)}}^{(i)} = \mathsf{OTD}(b^{(i)}, \mu_S^{(i)}; \rho_R^{(i)})$ and define $d^{(i)} = b^{(i)} \oplus b$.

   Send $(\delta_R^{(i)})_{i\in\mathcal{A}}$ and $(d^{(i)})_{i\in\mathsf{Alive}}$, along with the set $\mathcal{B}$, to the sender.
4. $(S \to R)$: The sender's last message is computed as follows.
   a. For each $i \in \mathcal{A}$, check that $\delta_R^{(i)}$ is good. If not, abort.
   b. For each $i \in \mathcal{B}$, let $\delta_S^{(i)} = (\kappa_0^{(i)}, \kappa_1^{(i)}, \rho_S^{(i)})$.
   c. Let $|\mathsf{Alive}| = n = m - t_R - t_S$. Run $(s_0^{(i)})_{i\in\mathsf{Alive}} \leftarrow\$ \mathsf{Share}(s_0)$ and $(s_1^{(i)})_{i\in\mathsf{Alive}} \leftarrow\$ \mathsf{Share}(s_1)$ (i.e., secret share s_0 and s_1 and assign one share to each of the alive sessions).
   d. For each $i \in \mathsf{Alive}$, let $\gamma_{0\oplus d^{(i)}}^{(i)} = \kappa_{d^{(i)}}^{(i)} \oplus s_0^{(i)}$ and $\gamma_{1\oplus d^{(i)}}^{(i)} = \kappa_{1\oplus d^{(i)}}^{(i)} \oplus s_1^{(i)}$. (Note that this implies $\gamma_{b\oplus d^{(i)}}^{(i)} = \kappa_{b\oplus d^{(i)}}^{(i)} \oplus s_b^{(i)}$ for all $b \in \{0,1\}$.)

   Send $(\delta_S^{(i)})_{i\in\mathcal{B}}$ and $(\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}}$ to the receiver.
5. (Receiver's Output:) The receiver determines the final output as follows. For each $i \in \mathcal{B}$, check that $\delta_S^{(i)}$ is good. If not, abort. Else, for each $i \in \mathsf{Alive}$, let $s_b^{(i)} = \gamma_{b^{(i)}}^{(i)} \oplus \kappa_{b^{(i)}}^{(i)}$, pick any subset $\mathcal{I}$ of $(s_b^{(i)})_{i\in\mathsf{Alive}}$ containing at least t *valid* shares, and return $s_b = \mathsf{Recon}((s_b^{(i)})_{i\in\mathcal{I}})$.

Figure 2 Formal description of our black-box compiler.

4 Our Compiler

In this section, we show a black-box compiler for obtaining round-optimal simulatable OT starting from any 2-round OT satisfying privacy for random inputs against defensible receivers (cf. Definition 6) and malicious senders (cf. Definition 4). Recall that, by Lemma 8, the latter property follows from privacy for random inputs against semi-honest senders.

4.1 Protocol Description

Let $\Pi' = (\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ be a 2-round OT protocol. We transform Π' into a 4-round OT protocol Π as depicted in Figure 2. Intuitively, the protocol Π proceeds as follows:

Round 1: The receiver starts m parallel sessions of the underlying OT protocol Π'. In each session $i \in [m]$, it uses a uniformly random choice bit $b^{(i)}$, which yields a message $\mu_R^{(i)}$. Hence, it forwards $\mu_R^{(1)}, \ldots, \mu_R^{(m)}$ to the sender.

Round 2: The sender picks random indices $\alpha_1, \ldots, \alpha_{t_R} \in [m]$ for cut-and-choose, where $t_R = m/3$. For the remaining sessions, it picks two random strings $(\kappa_0^{(i)}, \kappa_1^{(i)})$ and computes $\mu_S^{(i)}$ as a response to the message $\mu_R^{(i)}$ using the underlying OT protocol. Looking ahead, these random strings will serve as masks to hide the input messages in the last round of the protocol. Hence, it forwards the indices $\alpha_1, \ldots, \alpha_{t_R}$ and the messages $\mu_S^{(i)}$ (for all sessions but the ones selected for cut-and-choose).

Round 3: For each session that the sender asked to open, the receiver prepares a defense $\delta_R^{(i)} = (b^{(i)}, \rho_R^{(i)})$ which explains the message $\mu_R^{(i)}$ that was sent in the first round. Then, it picks random indices $\beta_1, \ldots, \beta_{t_S} \in [m] \setminus \{\alpha_1, \ldots, \alpha_{t_R}\}$ for cut-and-choose, where $t_S = m/3$; looking ahead, in the next round the sender will have to provide defenses for those indices, which allows to extract the inputs of the sender in the simulation proof. Denote by Alive the set of indices corresponding to sessions which are still alive (i.e., that were not selected for cut-and-choose). Using the underlying OT protocol, the receiver obtains the mask $\kappa_{b^{(i)}}^{(i)}$ and forwards to the sender an adjusting bit $d^{(i)} = b^{(i)} \oplus b$ for each alive session; intuitively, the bit $d^{(i)}$ tells the sender how to encrypt the input strings in the last round.

Round 4: The sender first checks that each of the defenses $\delta_R^{(i)}$ are good (and aborts if not). The privacy property (against defensible receivers) of the underlying OT protocol guarantees that the receiver does not learn the masks $\kappa_{1-b^{(i)}}^{(i)}$ corresponding to these sessions. Additionally, the sender prepares its own defenses $\delta_S^{(i)} = (\kappa_0^{(i)}, \kappa_1^{(i)}, \rho_S^{(i)})$ for each index $i \in \mathcal{B}$. Hence, it secret shares the input strings s_0 and s_1, obtaining $n = m - t_R - t_S$ shares $(s_0^{(i)})_{i \in \mathsf{Alive}}$ and $(s_1^{(i)})_{i \in \mathsf{Alive}}$, so that each pair of shares can be associated to a single alive session. Finally, the sender uses the key $\kappa_{0 \oplus d^{(i)}}^{(i)}$ (resp. $\kappa_{1 \oplus d^{(i)}}^{(i)}$) in order to encrypt the share $s_0^{(i)}$ (resp. $s_1^{(i)}$) in the i-th session, and forwards the resulting pairs of ciphertexts $(\gamma_0^{(i)}, \gamma_1^{(i)})_{i \in \mathsf{Alive}}$ to the receiver.

Output Computation: Since $b^{(i)} = b \oplus d^{(i)}$, the receiver can use the keys $\kappa_{b^{(i)}}^{(i)}$ in order to obtain all the shares $(s_b^{(i)})_{i \in \mathsf{Alive}}$ and thus reconstruct s_b using any subset of t *valid* shares[9] (and abort if there are less than t valid shares).

The theorem below states the security of our compiler.

▶ **Theorem 10.** *Let* $\Pi' = (\mathsf{OTR}, \mathsf{OTS}, \mathsf{OTD})$ *be a 2-round OT protocol with privacy for random inputs against defensible receivers and against malicious senders, and let* $(\mathsf{Share}, \mathsf{Recon})$ *be a* t*-out-of-*n *secret sharing scheme. Then, for parameters* m, t_R, t_S, t, n *such that* $m = O(\lambda)$, $t_R = t_S = n = m/3$ *and* $t = 2n/3$, *the protocol* $\Pi = (S, R)$ *from Figure 2 securely realizes* $\mathcal{F}_{\mathsf{OT}}$.

We defer an overview of the proofs to the appendix and a more formal treatment to the full version[35].

5 Conclusions

We have shown a compiler for turning any 2-round *privacy-only* (i.e., game-based) OT protocol against *malicious* adversaries into a *round-optimal simulatable* OT protocol against *malicious* adversaries. Our transform is *black-box* (in that it only uses the underlying 2-round OT protocol as an oracle), *information-theoretic* (in that it does not rely on cryptographic assumptions), and in the *plain model* (in that it does not require any form of trusted setup). In fact, our compiler works even assuming the underlying 2-round OT protocol satisfies privacy against *semi-honest* senders and *defensible* receivers.

[9] Recall that a share is called valid if it is a λ-bit string.

It remains an open problem to find a similar transform (with the same properties) starting with any 2-round OT protocol satisfying only privacy against *semi-honest* (rather than *defensible*) receivers. Also, it would be interesting to find simple constructions of 2-round OT protocols in the plain model that are private against *defensible* receivers, and that rely on hardness assumptions from which we do not know how to obtain 2-round OT protocols that are private against *malicious* receivers (e.g., CDH, LPN, and Subset Sum).

References

1 William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 119–135. Springer, Heidelberg, May 2001. doi:10.1007/3-540-44987-6_8.

2 Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 547–557. Springer, Heidelberg, August 1990. doi:10.1007/0-387-34805-0_48.

3 Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. doi:10.1145/168588.168596.

4 Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78375-8_17.

5 Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 370–390. Springer, Heidelberg, November 2018. doi:10.1007/978-3-030-03810-6_14.

6 Megha Byali, Arpita Patra, Divya Ravi, and Pratik Sarkar. Fast and universally-composable oblivious transfer and commitment scheme with adaptive security. Cryptology ePrint Archive, Report 2017/1165, 2017. URL: https://eprint.iacr.org/2017/1165.

7 Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 597–608. ACM Press, November 2014. doi:10.1145/2660267.2660374.

8 Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 387–402. Springer, Heidelberg, March 2009. doi:10.1007/978-3-642-00457-5_23.

9 Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 73–88. Springer, Heidelberg, February / March 2013. doi:10.1007/978-3-642-36362-7_6.

10 Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Oblivious transfer from trapdoor permutations in minimal rounds. In *Theory of Cryptography Conference*, pages 518–549. Springer, 2021.

11 Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, Daniel Masny, and Daniel Wichs. Two-round oblivious transfer from CDH or LPN. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 768–797. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45724-2_26.

12 Daniele Friolo, Daniel Masny, and Daniele Venturi. A black-box construction of fully-simulatable, round-optimal oblivious transfer from strongly uniform key agreement. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 111–130. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-36030-6_5.

13 Juan A. Garay. Efficient and universally composable committed oblivious transfer and applications. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 297–316. Springer, Heidelberg, February 2004. doi:10.1007/978-3-540-24638-1_17.

14 Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78375-8_16.

15 Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st FOCS*, pages 325–335. IEEE Computer Society Press, November 2000. doi:10.1109/SFCS.2000.892121.

16 Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd FOCS*, pages 51–60. IEEE Computer Society Press, October 2012. doi:10.1109/FOCS.2012.47.

17 Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1128–1141. ACM Press, June 2016. doi:10.1145/2897518.2897657.

18 Vipul Goyal and Silas Richelson. Non-malleable commitments using Goldreich-Levin list decoding. In David Zuckerman, editor, *60th FOCS*, pages 686–699. IEEE Computer Society Press, November 2019. doi:10.1109/FOCS.2019.00047.

19 Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 412–426. Springer, Heidelberg, March 2008. doi:10.1007/978-3-540-78524-8_23.

20 Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions of protocols for secure computation. *SIAM Journal on Computing*, 40(2):225–266, 2011.

21 Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *Journal of Cryptology*, 25(1):158–193, January 2012. doi:10.1007/s00145-010-9092-8.

22 Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. ISC. Springer, Heidelberg, 2010. doi:10.1007/978-3-642-14303-8.

23 Carmit Hazay and Yehuda Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010.

24 Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. Round-optimal fully black-box zero-knowledge arguments from one-way permutations. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 263–285. Springer, Heidelberg, November 2018. doi:10.1007/978-3-030-03807-6_10.

25 Carmit Hazay and Muthuramakrishnan Venkitasubramaniam. On black-box complexity of universally composable security in the CRS model. *Journal of Cryptology*, 32(3):635–689, July 2019. doi:10.1007/s00145-019-09326-y.

26 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 406–425. Springer, Heidelberg, May 2011. doi:10.1007/978-3-642-20465-4_23.

27 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008. doi:10.1007/978-3-540-85174-5_32.

28 Stanislaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 97–114. Springer, Heidelberg, May 2007. doi:10.1007/978-3-540-72540-4_6.

29 Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, Heidelberg, August 2004. doi:10.1007/978-3-540-28628-8_21.

30 Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732. ACM Press, May 1992. doi:10.1145/129712.129782.

31 Susumu Kiyoshima. Round-optimal black-box commit-and-prove with succinct communication. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 533–561. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56880-1_19.

32 Susumu Kiyoshima, Huijia Lin, and Muthuramakrishnan Venkitasubramaniam. A unified approach to constructing black-box UC protocols in trusted setup models. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 776–809. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70500-2_26.

33 Yehuda Lindell, Eli Oxman, and Benny Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 259–276. Springer, Heidelberg, August 2011. doi:10.1007/978-3-642-22792-9_15.

34 Yehuda Lindell and Benny Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. *Journal of Cryptology*, 25(4):680–722, October 2012. doi:10.1007/s00145-011-9107-0.

35 Varun Madathil, Chris Orsini, Alessandra Scafuro, and Daniele Venturi. From privacy-only to simulatable ot: Black-box, round-optimal, information-theoretic. *Cryptology ePrint Archive*, 2022.

36 Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001.

37 Rafail Ostrovsky, Silas Richelson, and Alessandra Scafuro. Round-optimal black-box two-party computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 339–358. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7_17.

38 Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 403–418. Springer, Heidelberg, March 2009. doi:10.1007/978-3-642-00457-5_24.

39 Rafael Pass and Hoeteck Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 638–655. Springer, Heidelberg, May / June 2010. doi:10.1007/978-3-642-13190-5_32.

40 Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008. doi:10.1007/978-3-540-85174-5_31.

41 Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. URL: http://eprint.iacr.org/2005/187.

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

43 Wolfgang Stadje. The collector's problem with group drawings. *Advances in Applied Probability*, 22(4):866–882, 1990. doi:10.2307/1427566.

44 Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st FOCS*, pages 531–540. IEEE Computer Society Press, October 2010. doi:10.1109/FOCS.2010.87.

A Simulator for Malicious Receivers

The simulator Sim, with oracle access to R^*, is defined in Fig 3. The definition below is useful to reason about the simulator.

▶ **Definition 11.** *Let* Bad $\subset$ Alive *be the set of indices for which the simulator does not get a defense.*

Main thread:

1. Upon receiving $\mu_R^{(1)}, \ldots, \mu_R^{(m)}$ from R^*:
   - **a.** Sample indices $\alpha_1, \ldots, \alpha_{t_R} \leftarrow\$ [m]$ and let $\mathcal{A} = \{\alpha_1, \ldots, \alpha_{t_R}\}$.
   - **b.** For each $i \in [m] \setminus \mathcal{A}$, pick $\kappa_0^{(i)}, \kappa_1^{(i)} \in \{0,1\}^\lambda$ and $\rho_S^{(i)} \leftarrow\$ \{0,1\}^*$, and let $\mu_S^{(i)} = \mathsf{OTS}(\kappa_0^{(i)}, \kappa_1^{(i)}, \mu_R^{(i)}; \rho_S^{(i)})$.
   - **c.** Send $\mu_S^{(i)}$ (for each $i \in [m] \setminus \mathcal{A}$) and $\mathcal{A}$ to R^*.
2. Upon receiving $((\delta_R^{(i)})_{i\in\mathcal{A}}, (d^{(i)})_{i\in\mathsf{Alive}}, \mathcal{B})$ from R^* (where $\mathsf{Alive} = [m] \setminus \mathcal{A} \cup \mathcal{B}$), check that all defenses $\delta_R^{(i)} = (b^{(i)}, \rho_R^{(i)})$ are good. If not, simulate the sender aborting. Else initialize $\mathsf{Bits} = \emptyset$ and $\mathsf{ctr} = 0$

   Rewind thread:

   - **a.** Sample $\mathcal{A}' = \{\alpha'_1, \ldots, \alpha'_{t_R}\}$ randomly as in Item 1a.
   - **b.** Recompute $\mu_S^{(i)}$ using fresh randomness (for each $i \in [m]\setminus\mathcal{A}'$), send these messages along with $\mathcal{A}'$ to R^* and receive $((\delta_R^{(i)})_{i\in\mathcal{A}'}, \mathcal{B}', (d^{(i)})_{i\in\mathsf{Alive}'})$ in response, where $\mathsf{Alive}' = [m] \setminus (\mathcal{A}' \cup \mathcal{B}')$.
   - **c.** For every defense $\delta_R^{(i)} = (b^{(i)}, \rho_R^{(i)})$ corresponding to an index $i \in \mathsf{Alive}$ (from the main thread) that was not observed in a previous rewind: If the defense is good add the bit $b^{(i)}$ to the set Bits.
   - **d.** Increment $\mathsf{ctr} = \mathsf{ctr} + 1$. If $\mathsf{ctr} = 2^\lambda$, abort.
   - **e.** If $|\mathsf{Bits}| + m/9 < |\mathsf{Alive}|$ go to Item 2a; else proceed to the next step.

3. Complete the simulation in the main thread as follows.
   - **a.** For each $b^{(i)} \in \mathsf{Bits}$, compute $\hat{b}^{(i)} = b^{(i)} \oplus d^{(i)}$.
   - **b.** Let b be a bit that appears among the $\hat{b}^{(i)}$ more than $t/2$ times
   - **c.** Forward b to $\mathcal{F}_{\mathsf{OT}}$ obtaining $s_b \in \{0,1\}^\lambda$, sample $s'_{1-b} \leftarrow\$ \{0,1\}^\lambda$ and simulate the final message $((\delta_S^{(i)})_{i\in\mathcal{B}}, (\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}})$ as the honest sender would do.

Figure 3 Simulator against a malicious receiver.

▶ **Lemma 12.** *Conditioned on the fact that* Sim *did not abort in the main thread, the cardinality of* Bad *(Definition 11) is strictly less than $m/9$ with overwhelming probability.*

Proof. We defer this proof to the full version [35]. ◀

▶ **Lemma 13.** *The above simulator* Sim *runs in expected time that is polynomial in m and λ except with negligible probability.*

Proof. The number of rewind attempts needed to cover all indices of $\mathsf{Alive} \setminus \mathsf{Bad}$ is a variation of the coupon collector's problem [43] which has a polynomial time solution. We defer proofs to the full version [35]. ◀

Proof by hybrids. We next prove by a sequence of hybrids that the distribution of the output of R^* in the real world is computationally close to that in the ideal world with the above defined simulator. The hybrids are described[10] below:

Hybrid $\mathbf{Hyb}_0(\lambda, s_0, s_1, b)$**:** This is identical to $\mathbf{Real}_{\Pi,R^*(z)}(\lambda, s_0, s_1, b)$.

Hybrid $\mathbf{Hyb}_1(\lambda, s_0, s_1, b)$**:** Identical to the previous experiment, except that we now perform the rewinding as done by the simulator before concluding the protocol.

Hybrid $\mathbf{Hyb}_2(\lambda, s_0, s_1, b)$**:** Identical to the previous experiment except when generating the message $((\delta_S^{(i)})_{i\in\mathcal{B}}, (\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}})$ redefine $\gamma_{1-b^{(i)}}^{(i)} = \hat{\kappa}_{1-b^{(i)}} \oplus s_{1-\hat{b}^{(i)}}^{(i)}$ using an independent $\hat{\kappa}_{1-b^{(i)}} \leftarrow\$ \{0,1\}^\lambda$ (instead of $\kappa_{1-b^{(i)}}^{(i)}$) for each bit $b^{(i)} \in \mathsf{Bits}$ (recall that $\hat{b}^{(i)} = b^{(i)} \oplus d^{(i)}$).

Hybrid $\mathbf{Hyb}_3(\lambda, s_0, s_1, b)$**:** Identical to the previous experiment except for the following difference. After successfully completing the rewinding proceed as follows: For each $b^{(i)} \in \mathsf{Bits}$, compute $\hat{b}^{(i)} = b^{(i)} \oplus d^{(i)}$. Let b be a bit that appears among the $\hat{b}^{(i)}$ more than $t/2$ times. Forward b to $\mathcal{F}_{\mathsf{OT}}$ obtaining $s_b \in \{0,1\}^\lambda$, sample $s'_{1-b} \leftarrow\$ \{0,1\}^\lambda$ and simulate the final message $((\delta_S^{(i)})_{i\in\mathcal{B}}, (\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}})$ as the honest sender would do.

Since the final hybrid is identically distributed to the ideal world with the above defined simulator Sim, it remains to show that the above hybrids are all computationally indistinguishable. In the full version[35] we show that the hybrids are indistinguishable.

B Simulator for Malicious Senders

The simulator Sim, with oracle access to S^*, proceeds as follows in Figure 4.

▶ **Lemma 14.** *The above simulator* Sim *runs in expected time that is polynomial in* (m, λ).

Proof. The analysis is similar to that of Lemma 13 and we defer a more formal treatment to the full version [35]. ◀

Proof by hybrids. We next prove by a sequence of hybrids that the distribution of the output of S^* in the real world is computationally close to that in the ideal world with the above defined simulator. The hybrids are described below:

Hybrid $\mathbf{Hyb}_0(\lambda, s_0, s_1, b)$**:** This is identical to $\mathbf{Real}_{\Pi,S^*(z)}(\lambda, s_0, s_1, b)$.

Hybrid $\mathbf{Hyb}_1(\lambda, s_0, s_1, b)$**:** Identical to the previous experiment, except that we now perform the rewinding as done by the simulator before concluding the protocol.

Hybrid $\mathbf{Hyb}_2(\lambda, s_0, s_1, b)$**:** Identical to the previous experiment except that the $d^{(i)}$ are sampled randomly and for each$i \in [m] \setminus (\mathcal{A} \cup \mathcal{B})$ compute $\kappa_{b^{(i)}}^{(i)} = \mathsf{OTD}(b^{(i)}, \mu_S^{(i)}; \rho_R^{(i)})$. And the $s_b^{(i)}$ is computed as $\gamma_{b\oplus d^{(i)}}^{(i)} \oplus \kappa_{b\oplus d^{(i)}}^{(i)}$

Since the final hybrid is identically distributed to the ideal world with the above defined simulator Sim, it remains to show that the above hybrids are all computationally indistinguishable. We defer the formal proofs to the full version[35] where we show that the hybrids are indistinguishable.

[10] Note that the hybrids further depend on the malicious receiver R^* and on the OT protocol Π, but we omit those to simplify notation.

Main thread:

1. For each $i \in [m]$, pick $b^{(i)} \leftarrow\$ \{0,1\}$ and $\rho_R^{(i)} \leftarrow\$ \{0,1\}^*$, compute $\mu_R^{(i)} = \mathsf{OTR}(b^{(i)}; \rho_R^{(i)})$ and send $(\mu_R^{(i)})_{i\in[m]}$ to S^*.
2. Upon receiving $((\mu_S^{(i)})_{i\in[m]\setminus\mathcal{A}}, \mathcal{A})$ from S^*:
   a. For each $i \in \mathcal{A}$, let $\delta_R^{(i)} = (b^{(i)}, \rho_R^{(i)})$.
   b. Sample $\beta_1, \ldots, \beta_{t_S} \leftarrow\$ [m] \setminus \mathcal{A}$ and let $\mathcal{B} = \{\beta_1, \ldots, \beta_{t_S}\}$.
   c. Let $\mathsf{Alive} = [m] \setminus (\mathcal{A} \cup \mathcal{B})$. For each $i \in \mathsf{Alive}$, compute $\kappa_{b^{(i)}}^{(i)} = \mathsf{OTD}(b^{(i)}, \mu_S^{(i)}; \rho_R^{(i)})$ and sample $d^{(i)} \leftarrow\$ \{0,1\}$.
   d. Send $((\delta_R^{(i)})_{i\in\mathcal{A}}, (d^{(i)})_{i\in\mathsf{Alive}}, \mathcal{B})$ to S^*.
3. Upon receiving $((\delta_S^{(i)})_{i\in\mathcal{B}}, (\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}})$ from S^*, check that all defenses $\delta_S^{(i)} = (\kappa_0^{(i)}, \kappa_1^{(i)}, \rho_S^{(i)})$ are good. If not, simulate the receiver aborting. Else initialize $\mathsf{Keys} = \emptyset$ and $\mathsf{ctr} = 0$.

   Rewind thread:

   a. Sample randomly $\mathcal{B}' = \{\beta_1, \ldots, \beta_{t_S}\}$ as in Item 2b.
   b. Let $\mathsf{Alive}' = [m] \setminus (\mathcal{A} \cup \mathcal{B}')$. For $i \in \mathsf{Alive}'$, sample $d^{(i)} \leftarrow\$ \{0,1\}$ using fresh randomness, send these values along with $\mathcal{B}'$ and $(\delta_R^{(i)})_{i\in\mathcal{A}}$ to S^*, and receive $((\delta_S^{(i)})_{i\in\mathcal{B}}), (\gamma_0^{(i)}, \gamma_1^{(i)})_{i\in\mathsf{Alive}'})$ in response.
   c. For every defense $\delta_S^{(i)} = (\kappa_0^{(i)}, \kappa_1^{(i)}, \rho_S^{(i)})$ corresponding to an index $i \in \mathsf{Alive}$ (from the main thread) that was not observed in a previous rewind: If the defense is good, add[a] the keys $(\kappa_0^{(i)}, \kappa_1^{(i)})$ to Keys.
   d. Increment $\mathsf{ctr} = \mathsf{ctr} + 1$ and if $\mathsf{ctr} > 2^\lambda$ abort.
   e. If $|\mathsf{Keys}| < |\mathsf{Alive}| - m/9$ go to Item 3a; else proceed to the next step.

   [a] In case either of the two keys is not a λ-bit string, we assume the defense is bad.

4. For each pair of keys $(\kappa_0^{(i)}, \kappa_1^{(i)})$ in Keys, compute $s_0^{(i)} = \gamma_{0\oplus d^{(i)}}^{(i)} \oplus \kappa_{0\oplus d^{(i)}}^{(i)}$ and $s_1^{(i)} = \gamma_{1\oplus d^{(i)}}^{(i)} \oplus \kappa_{1\oplus d^{(i)}}^{(i)}$ using the adjusting bits $d^{(i)}$ sampled in the main thread. Finally, use these shares to reconstruct s_0 and s_1, forward (s_0, s_1) to $\mathcal{F}_{\mathsf{OT}}$ and output whatever S^* outputs.

Figure 4 Simulator for malicious sender.

On the Distributed Discrete Logarithm Problem with Preprocessing

Pavel Hubáček
Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic

Ľubica Jančová
Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic

Veronika Králová
Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic

Abstract

Protocols solving the Distributed Discrete Logarithm (DDLog) problem are a core component of many recent constructions of group-based homomorphic secret sharing schemes. On a high-level, these protocols enable two parties to transform multiplicative shares of a secret into additive share *locally without any communication*. Due to their important applications, various generic optimized DDLog protocols were proposed in the literature, culminating in the asymptotically optimal generic protocol of Dinur, Keller, and Klein (J. Cryptol. 2020) solving DDLog in time T with error probability $O(W/T^2)$ when the magnitude of the secret is bounded by W.

Given that DDLog is solved repeatedly with respect to a fixed group in its applications, a natural approach for improving the efficiency of DDLog protocols could be via leveraging some precomputed group-specific advice. To understand the limitations of this approach, we revisit the distributed discrete logarithm problem in the preprocessing model and study the possible time-space trade-offs for DDLog in the generic group model. As our main result, we show that, in a group of size N, any generic DDLog protocol for secrets of magnitude W with parties running in time T using precomputed group-specific advice of size S has success probability $\varepsilon = O\left(T^2/W + \max\{S, \log W\} \cdot T^2/N\right)$. Thus, assuming $N \geq W \log W$, we get a lower bound $ST^2 = \Omega(\varepsilon N)$ on the time-space trade-off for DDLog protocols using *large advice* of size $S = \Omega(N/W)$. Interestingly, for DDLog protocols using *small advice* of size $S = O(N/W)$, we get a lower bound $T^2 = \Omega(\varepsilon W)$ on the running time, which, in the constant-error regime, asymptotically matches the running time of the DDLog protocol *without any advice* of Dinur et al. (J. Cryptol. 2020). In other words, we show that generic DDLog protocols achieving constant success probability do not benefit from any advice of size $S = O(N/W)$ in the online phase of the DDLog problem.

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

Keywords and phrases Distributed discrete logarithm problem, preprocessing, generic group model

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.6

Related Version *Full Version*: `https://eprint.iacr.org/2022/521`

Funding Supported by the Grant Agency of the Czech Republic under the grant agreement no. 19-27871X and by the Charles University projects PRIMUS/17/SCI/9 and UNCE/SCI/004.

Acknowledgements We are thankful to Siyao Guo for clarifications regarding the known techniques for establishing lower bounds for problems in the generic group model with preprocessing. We also wish to thank the anonymous ITC 2022 reviewers for helpful detailed comments on our results and their presentation.

1 Introduction

The *Distributed* Discrete Logarithm problem (DDLog) serves as an abstraction for a non-interactive multiplicative-to-additive share conversion procedure central to many recent constructions of homomorphic secret sharing schemes [4, 2, 11]. On a high-level, during a homomorphic evaluation of a restricted multiplication straight-line program using a homomorphic secret sharing scheme, two parties hold encryptions of the secret inputs under some homomorphic public-key encryption scheme (e.g., ElGamal encryption in [4] or Paillier encryption in [11]), as well as the additive shares of the inputs. After performing the multiplication operation "multiply an input value by a memory value", the parties naturally obtain multiplicative shares of the result. Though, additive shares are needed for further evaluation, and this is where the DDLog problem plays its crucial role.

Distributed discrete logarithm problem for prime-order groups

Let $(\mathbb{G}, \cdot)$ be a group in the multiplicative notation. Two parties P_0 and P_1 are holding $h_0 \in \mathbb{G}$ and $h_1 \in \mathbb{G}$ (respectively) such that $\frac{h_1}{h_0} = \omega^x$ for some $\omega \in \mathbb{G}$ and secret $x \in \mathbb{Z}$. *Without communicating*, the parties need to convert their *multiplicative* shares h_0 and h_1 to *additive* shares $a_0 \in \mathbb{Z}$ and $a_1 \in \mathbb{Z}$ such that $a_1 - a_0 = x$. For example, in the case of [4], the underlying group is a prime-order group and the element ω is its generator. Next, we consider the DDLog problem in this case in more detail.

Let $(\mathbb{G}, \cdot)$ be a cyclic group of prime-order N with a generator g. A natural parameter of the DDLog problem is the magnitude of the secret x. In the distributed discrete logarithm problem in $\mathbb{G}$ *on interval of size* $W \in \mathbb{N}$ such that $W \leq N$, the parties P_0 and P_1 get as input g^b and g^{b+x} respectively, where $x \leftarrow \{x \in \mathbb{Z} \cap [-W/2, W/2]\}$ and $b \leftarrow \mathbb{Z}_N$. The representation of the group $\mathbb{G}$ and the generator g is known to both the parties. At the end of their local executions, the parties output elements $P_0(g^b), P_1(g^{b+x}) \in \mathbb{Z}_N$ respectively and they succeed in solving the distributed discrete logarithm problem instance if $P_1(g^{b+x}) - P_0(g^b) = x$.

[4] gave a protocol solving the DDLog problem achieving error probability at most ε with time complexity $O(W\varepsilon^{-1}\log(\varepsilon^{-1}))$ group operations. Using a pseudorandom function shared among the parties, their protocol selects a set of "special" group elements and each of the parties searches through T points closest to her input, in sense of multiplication by the group generator g. When a party finds a "special" point, the output of the DDLog procedure on her input is the number of steps (i.e., multiplications by g) she made until finding the "special" point. It is clear that if both parties find the same "special" point, $P_0(g^b) - P_1(g^{b+x}) = x$ holds. Therefore, the success probability of the above DDLog protocol is exactly the probability of the parties synchronizing on the same "special" point. In the application towards a construction of a homomorphic secret sharing scheme, the error probability introduced by the DDLog protocol propagates throughout the whole homomorphic evaluation process and, subsequently, significantly affects its efficiency.

The DDLog problem in prime-order groups was further explored in [7]. First, the authors proposed a more sophisticated DDLog protocol resulting in error probability $O(W/T^2)$ in T group operations. Their procedure consists of iterating random walks with carefully chosen parameters of maximal step-length and number of steps in each stage in a way that they continuously reduces the probability of the two parties not synchronizing on their path.

Second, the authors also analyzed the limitations of protocols solving the DDLog problem. By a reduction of the *Discrete Logarithm in Interval* (DLI) problem to the DDLog problem, they proved that the error probability of DDLog protocols running in time T is $\Omega(W/T^2)$ in specific families of groups where the DLI problem is hard. Additionally, [7] also analyzed the

DDLog problem in the generic group model [12, 9], i.e., in a model inhibiting the attacker from exploiting a particular structure and properties of the underlying group. Specifically, they showed that the error probability of any generic DDLog protocol is $\Omega(W/T^2)$.

Due to the matching negative results, the DDLog protocol can be considered as optimal. The error probability $\Omega(W/T^2)$ also implies a non-negligible correctness error in the HSS construction from [4].

DDLog as a synchronization problem

Note that, at its core, the *distributed* DLog problem is different than the standard DLog. In order to succeed, the parties can "merely" synchronize on some special element with high probability. In particular, they certainly do not need to find the discrete logarithms of their respective inputs to synchronize.[1] One could thus attempt to construct more efficient DDLog protocols by leveraging some precomputed group-specific advice that, on one hand, would allow to synchronize with high probability faster while, on the other hand, would not compromise the security of the DLog problem in the underlying group. This approach is particularly well motivated, for example, from the perspective of trying to amortize the cost of the many DDLog instances generated during a homomorphic evaluation in a homomorphic secret sharing scheme.

1.1 Our Results

In this work, we focus on the distributed discrete logarithm problem for prime-order groups in the preprocessing model and examine to what extent can preprocessing help to solve the distributed discrete logarithm problem. In other words, we allow an *offline phase* of the protocol for precomputing a bit-string of length S before receiving the challenge pair (g^b, g^{b+x}) without any restriction on the time of its computation. Then, the precomputed *advice* bit-string is passed as an additional input to the two parties running in the *online phase*, i.e., after receiving the respective challenges. Regarding the time complexity of the protocol, we are interested in the performance of the online phase. To examine whether the additional *advice* computed during the offline phase can help to reduce the error probability in the DDLog problem, we study this question in the generic group model.

▶ **Theorem 1** (main – informal). *Let $N, W \in \mathbb{N}$, such that $N > W$, N be a prime. Then any generic protocol solving DDLog in a group of order N in time T using precomputed advice of size S has success probability*

$$\varepsilon = O\left(\frac{T^2}{W} + \frac{\max\{S, \log W\} \cdot T^2}{N}\right),$$

where W denotes the length of the interval in the DDLog problem. Furthermore, if $N \geq W \cdot \log W$ then

$$\varepsilon = O\left(\frac{T^2}{W} + \frac{S \cdot T^2}{N}\right).$$

Our main result summarized above is an upper bound on the success probability of protocols leveraging preprocessing. Assuming $N \geq W \log W$, our bound for a "large" preprocessing advice of size $S = \Omega(N/W)$ translates into a bound on time-space tradeoff

[1] Importantly, standard DLog is hard in the groups employed in the current applications of DDLog.

in the DDLog problem $ST^2 = \Omega(\varepsilon N)$. Note that this bound matches the lower bounds for standard discrete logarithm problem with preprocessing given by [5] and [6]. As the DDLog problem naturally reduces to two instances of the classical discrete logarithm problem, the DLog algorithms matching these lower bounds give an optimal algorithm for DDLog in this regime of parameters.

For any "small" preprocessing advice of size $S = O(N/W)$, our bound translates to a lower bound on the time complexity of the DDLog problem $T^2 = \Omega(\varepsilon W)$. Interestingly, for DDLog protocols with constant success probability in this regime of parameters, our lower bound on time-complexity of protocols *with preprocessing* is matched by the time complexity of the protocol *with no preprocessing* running in time $T^2 = O(\frac{W}{1-\varepsilon})$ from [7]. Hence, if we want to achieve a constant success probability in the DDLog problem, then allowing the attacker to precompute a preprocessing advice of size $S = O(N/W)$ does not asymptotically help to save computation time in the online phase of the DDLog problem.

Finally, let us consider the regime of parameters relevant to the application of DDLog in homomorphic secret sharing, i.e., the original motivation for DDLog. There, we expect W to be a polynomial function in the security parameter, i.e., $W = O(\text{polylog}(N))$. In this case, our assumption $N \geq W \log W$ is fulfilled. Then, in order to leverage the time-space trade-off matching our lower bound, we would need to allow advice of size $S = \Omega(N/\log N)$, which is larger than, e.g., $S = \sqrt[3]{N}$. Moreover, advice of this size allows to calculate discrete logarithms in polynomial time, which would ultimately break the security of the HSS scheme.

Non-generic DDLog protocols

It is important to stress that neither our results nor the results of Dinur et al. [7] rule out *non-generic* DDLog protocols exploiting particular structure of some groups in which DLI is not a difficult problem, or where the order of the underlying group is not prime. For example, this is the case of the DDLog protocol and a subsequent construction of homomorphic secret sharing in [11] from Paillier encryption scheme. Specifically, [11] exploits the fact that computing discrete logarithms is easy in a particular subgroup of the Paillier group, which allowed them to construct an efficient protocol with *perfect correctness*. Nevertheless, the problem of designing efficient generic DDLog protocols is of interest besides the applications to homomorphic secret sharing schemes. For example in the recent work of Boyle et al. [3], the DDLog protocol of Dinur et al. [7] was exploited towards constructions of a new flavour of locality-preserving hash function.

1.2 Our Techniques

On a high-level, our approach is similar to works that studied the *standard* discrete logarithm problem with preprocessing in the generic group model [6, 5]. Though, as explained above, there are significant differences to the standard DLog problem. In particular, the online adversary is distributed in our context and, thus, we need to introduce the corresponding formalism. Additionally, the DDLog problem is parameterized by the "magnitude" bound W corresponding to the width of the interval used for sampling the secret. We strive to derive quantitative bounds on the power of generic preprocessing adversaries that depend *both* on W and the order of the group N, which corresponds to analyzing the power of algorithms exploiting the knowledge of the parameter W.

Similarly to [5], we use the auxiliary-input generic group model in order to analyze DDLog protocols with preprocessing. Given that proving lower bounds in the auxiliary-input model directly is notoriously challenging, we first prove a lower bound in the more amenable

bit-fixing model, which captures the presampling technique of Unruh [13]. Then, in order to translate our lower bound from the bit-fixing model into the auxiliary-input model, we leverage an analogue of theorem from [5] relating the security of primitives in the bit-fixing and auxiliary-input model in the presence of distributed online attackers.[2]

First, we notice that the proof of the correspondence between bit-fixing and auxiliary-input models in the presence of *distributed online attackers* follows via minor adjustments from the analogous correspondence for *standard online attackers* in [5]. We give the formal statement (Theorem 17) and a proof sketch highlighting the important points in Section 3.3.

The most technical part of our work is the upper bound on the success probability of a distributed online attacker for DDLog in the bit-fixing generic group model (BF-GGM). The high-level approach for proving an upper bound on the success probability in BF-GGM is to consider an alternative experiment in which the adversary clearly has limited probability of succeeding and then bounding the power of the adversary to distinguish such alternative experiment from the real security experiment. In more detail, we can create a list of formal polynomials capturing the relations among the group elements in terms of the encoding of the challenge g^x induced by the queries of the adversary to the group oracle. In the context of standard DLog, these are univariate polynomials in $\mathbb{Z}_N[X]$. The core of the proof is then analysing the probability of a collision event when substituting an actual value x into the polynomials – such a collision event corresponds to an inconsistent answer in the alternative experiment. In the context of distributed DLog, it is natural to consider bivariate polynomials in $\mathbb{Z}_N[X, B]$ that capture relations in terms of the encodings of the DDLog challenge (g^x, g^{x+b}). Though, the most significant technical difference is induced by the distributed nature of the online attacker. Specifically, we need to additionally handle a collision event among the queries of the two distributed parts of the online attackers.

2 Preliminaries

Below, we review the Schwartz-Zippel lemma (Proposition 2) and the standard Boole's inequality, also known as the union bound (Proposition 3).

▶ **Proposition 2** (Schwartz-Zippel). *Let $\mathbb{F}$ be a field. Let $f \in \mathbb{F}[X_1, \ldots, X_k]$ be a non-zero polynomial of total degree $d \geq 0$. Let S be a finite subset of $\mathbb{F}$ and let $x_1, \ldots, x_k$ be chosen independently uniformly at random from S. Then it holds that*

$$\Pr[f(x_1, \ldots, x_k) = 0] \leq \frac{d}{|S|}.$$

▶ **Proposition 3** (Union Bound). *Let $n \in \mathbb{N}$ and consider events $E_1, \ldots, E_n$, then*

$$\Pr\left[\bigcup_{i=1}^{n} E_i\right] \leq \sum_{i=1}^{n} \Pr\left[E_i\right].$$

[2] We note that one could possibly obtain similar results to ours by adapting the more recent technique from [8] leveraging an alternative characterization of the bit-fixing model.

3 DDLog with Preprocessing in the Generic Group Model

3.1 The Generic Group Model

In this section, we describe a variant of the generic group model that we consider in this work, and we define the DDLog problem in this generic group model. We use a variant of the generic group model [12, 9] similar to the one in [5] with adjustments accommodating distributed attackers.

Let $\mathbb{G}$ be a cyclic group of prime-order N with a generator g. The generic group model allows to capture the power of algorithms that do not leverage additional knowledge about the structure of the group or the specific representation of group elements. In more detail, the structure of $\mathbb{G}$ is random in the sense that the access of the adversary to the group elements is given by a random injective function $\sigma : \mathbb{Z}_N \to [M]$ for some natural number $M \geq N$. Thus, the elements of $\mathbb{G}$ are represented by the elements from $\mathrm{Im}(\sigma)$, i.e., for $a \in \mathbb{Z}_N$, the representation of the group element g^a is $\sigma(a)$. We call the mapping σ an *encoding function*.

▶ **Definition 4** (Encoding function)**.** *Let $N, M \in \mathbb{N}$ be natural numbers such that $M \geq N$. Consider $\mathbb{Z}_N$, the additive group of integers modulo N, and the set $[M] = \{1, \ldots, M\}$. An* encoding function of $\mathbb{Z}_N$ on $[M]$ *is an injective mapping $\sigma \colon \mathbb{Z}_N \to [M]$. We denote $\mathcal{I}_{N,M}$ the set of all encoding functions of $\mathbb{Z}_N$ on $[M]$ and $\mathcal{Y}_\sigma$ the image of σ.*[3]

In order to perform group operations in the generic group model, the adversary $\mathcal{A}$ is granted access to a group-operation oracle $\mathcal{O}$, which chooses a random encoding function $\sigma \leftarrow \mathcal{I}_{N,M}$ at the beginning of the experiment and then answers adversary's queries of the following types:

Forward query: Any query $a \in \mathbb{Z}_N$ is answered by $\sigma(a) \in [M]$.

Group-operation query: Any query $(s_1, s_2) \in [M] \times [M]$ is answered by $\sigma(x_1 + x_2)$, where x_1, x_2 are elements of $\mathbb{Z}_N$ such that $\sigma(x_1) = s_1$ and $\sigma(x_2) = s_2$. If any of s_1, s_2 is not in $\mathcal{Y}_\sigma$, the query is answered by $\perp$.

Inverse query: Any query $s \in [M]$ is answered by $\sigma(-x)$, where $x \in \mathbb{Z}_N$ is the preimage of s, i.e., $\sigma(x) = s$. If $s \notin \mathcal{Y}_\sigma$, the query is answered by $\perp$.

In the generic group model, we measure the time complexity of the attacker by the number of oracle queries performed during its execution.

▶ Remark 5 (Explicit inverse queries). Unlike previous works studying the standard DLog with preprocessing [6, 5] and the DDLog without preprocessing [7] in the generic group model, we allow the adversary to make *explicit* inverse queries. This choice is important for our results in order to achieve quantitatively better bounds. Note that an inverse query can be implemented using $O(\log N)$ group-operation queries. Thus, a lower bound in a variant of the generic group model *without* explicit inverse queries yields an equivalent lower bound in our model up to a multiplicative logarithmic factor in N. However, in order to derive meaningful bounds for the DDLog problem, we cannot afford to neglect logarithmic factors in N and, therefore, we analyse the performance of protocols that can make explicit inverse queries. We note that other works previously considered explicit inverse queries in GGM. See, for example, Neven, Smart, and Warinschi [10] or Blocki and Lee [1].

[3] We omit the subscript σ in $\mathcal{Y}_\sigma$ when the encoding is clear from the context.

$\mathsf{Exp}_{\mathcal{A}}^{G^{\mathcal{O}}}(\lambda)$:

1. On input 1^λ, the challenger $\mathcal{C}$ generates a secret x from a space of secrets X and a challenge (c_0, c_1) from the challenge space $\mathcal{CH}_x$. It forwards $(1^\lambda, c_0)$ to $\mathcal{A}^{(0)}$ and $(1^\lambda, c_1)$ to $\mathcal{A}^{(1)}$.
2. For $i = 0, 1$:
   a. On input $(1^\lambda, c_i)$, the attacker $\mathcal{A}^{(i)}$ makes queries to oracle $\mathcal{O}$ and receives answers from the oracle.
   b. The attacker $\mathcal{A}^{(i)}$ chooses a guess $x_i \in X$ and sends it to $\mathcal{C}$.
3. The output of the experiment is 1 if $x_1 - x_0 = x$ and 0 otherwise.

Figure 1 Distributed unpredictability experiment.

Unpredictability application

The distributed discrete logarithm problem can also be seen as a problem of guessing secret information given a challenge from some challenge space dependent on the secret. In particular, the attacker tries to guess x given the challenge of the form g^b, g^{b+x}. Nevertheless, the unpredictability application does not provide a good representation for this problem, as it carries several differences compared to it. Especially, the attacker in the distributed discrete logarithm problem is composed of two algorithms who cannot communicate and each of these algorithms only gets to see half of the challenge, moreover, "to solve" the DDLog problem, is to get the secret x secret-shared between these two algorithms, not known explicitly by any of them. We call this special type of attacker a *distributed attacker* and we represent similar problems by *distributed unpredictability application*.

▶ **Definition 6** (Distributed attacker). *A* distributed attacker $(\mathcal{A}^{(0)}, \mathcal{A}^{(1)})$ *is a pair of independent PPT* $\mathcal{A}^{(0)}$ *and* $\mathcal{A}^{(1)}$*, who cannot communicate. We say the distributed attacker runs in the time* T *if each of* $\mathcal{A}^{(0)}$ *and* $\mathcal{A}^{(1)}$ *does not make more than* T *oracle queries during its execution.*

Below, we define a distributed unpredictability application, which captures the problem of guessing some secret information x given a challenge from a challenge space $\mathcal{CH}_x$ in the generic group model.

▶ **Definition 7** (Distributed unpredictability application). *Let* $\lambda \in \mathbb{N}$ *be a security parameter. A* distributed unpredictability application G *in the oracle* $\mathcal{O}$*-model is defined by a space of secrets* X*, a set of challenge spaces* $\{\mathcal{CH}_x \mid x \in X\}$*, where the elements of* $\mathcal{CH}_x$ *are of form* (c_0, c_1)*, a PPT challenger* $\mathcal{C}$ *with oracle access and an oracle* $\mathcal{O}$*. We define the advantage of the distributed attacker* $\mathcal{A} = (\mathcal{A}^{(0)}, \mathcal{A}^{(1)})$ *on* G*, denoted* $\mathsf{Adv}_{\mathcal{A}}^{G^{\mathcal{O}}}(\lambda)$*, as the probability of success of* $\mathcal{A}$ *in the distributed unpredictability experiment* $\mathsf{Exp}_{\mathcal{A}}^{G^{\mathcal{O}}}(1^\lambda)$ *defined in Figure 1, i.e.,*

$$\mathsf{Adv}_{\mathcal{A}}^{G^{\mathcal{O}}}(\lambda) = \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{G^{\mathcal{O}}}(\lambda) = 1\right].$$

Now, we define the distributed discrete logarithm problem in the generic group model.

▶ **Definition 8** (Distributed discrete logarithm application). *Let* $\lambda \in \mathbb{N}$ *be a security parameter, let* $N, W \in \mathbb{N}$*,* $N > W$ *and* $\mathcal{O}$ *be a group operation oracle. The* (N, W)-distributed discrete logarithm application $G_{\mathsf{DDLog}}^{\mathcal{O}}(N, W)$ *is a distributed unpredictability application in* $\mathcal{O}$*-model, where the space of secrets is* $X = \{x \mid x \in \mathbb{Z} \cap [-W/2, W/2]\}$*, the challenge space for* $x \in X$

is defined as $\mathcal{CH}_x = \{(\sigma(b), \sigma(b+x)) \mid \sigma \in \mathcal{I}_{N,M}, b \in \mathbb{Z}_N\}$*, and the challenger* C^{DDLog} *is a PPT that samples* $b \leftarrow \mathbb{Z}_N$ *and* x *from the set of integers in the interval* $[-W/2, W/2]$ *uniformly at random. Then* C^{DDLog} *makes two forward queries* $b, b+x$ *to the oracle and passes* $(1^\lambda, \sigma(b))$ *as a challenge to* $\mathcal{A}^{(0)}$ *and* $(1^\lambda, \sigma(b+x))$ *as a challenge to* $\mathcal{A}^{(1)}$*.*

3.2 Preprocessing in the Generic Group Model

In this section, we introduce the definitional framework of the generic group model for the algorithms with preprocessing. The framework is adapted from [5] with some small adjustments in order to capture the distributed attacker. First, we define the preprocessing oracle.

▶ **Definition 9** (Preprocessing oracle)**.** *The* preprocessing oracle $\mathcal{O}$ *is an oracle formed by a pair of oracles* $(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})$*, where* $\mathcal{O}_{\mathsf{pre}}$ *can only be queried during the offline phase of an experiment, i.e., before the challenge is generated and* $\mathcal{O}_{\mathsf{main}}$ *can only be queried in the online phase of an experiment.*

The attacker in the preprocessing model is composed of two algorithms, one of them running in the offline phase and having access to $\mathcal{O}_{\mathsf{pre}}$, the other one running in the online phase and having an access to $\mathcal{O}_{\mathsf{main}}$.

▶ **Definition 10** ((S,T)-attacker, [5, Definition 1])**.** *Let* $S, T \in \mathbb{N}$ *and* $\mathcal{O} = (\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})$ *be a preprocessing oracle. An* (S,T)-attacker $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ in the $\mathcal{O}$-model *consist of two probabilistic algorithms:*

1. *A* preprocessing algorithm $\mathcal{A}_0$*, which is computationally unbounded and which interacts with* $\mathcal{O}_{\mathsf{pre}}$ *and outputs a bit-string of length at most* S *bits.*
2. *An* online algorithm $\mathcal{A}_1$*, which takes as input an* S*-bit output of* $\mathcal{A}_0$ *and a challenge from the challenger, then makes at most* T *queries to* $\mathcal{O}_{\mathsf{main}}$*, and outputs a guess.*

Now, we define the notions of *unpredictability application* and *distributed unpredictability application* in the preprocessing model. Definitions 12 and 13 correspond to Definitions 6 and 7 extended to the preprocessing model. The (S,T)*-attacker* (Definition 10) and the *unpredictability application with preprocessing* (Definition 11) correspond to the model used by [5], whereas the *distributed* (S,T)*-attacker* (Definition 12) and the *distributed unpredictability application with preprocessing* (Definition 7) allow us to model the DDLog problem with preprocessing in the generic group model.

The following definition formalizes the problem of guessing secret information x given a challenge from a related challenge space $\mathcal{CH}_x$, while we allow the attacker to precompute advice string before receiving the challenge.

▶ **Definition 11** (Unpredictability application with preprocessing)**.** *Let* $\lambda \in \mathbb{N}$ *be a security parameter. An* unpredictability application with preprocessing G *in the oracle* $(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})$*-model is defined by a space of secrets* X*, a set of challenge spaces* $\{\mathcal{CH}_x \mid x \in X\}$*, a PPT challenger* $\mathcal{C}$ *with oracle access to* $\mathcal{O}_{\mathsf{main}}$ *and a preprocessing oracle* $(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})$*. We define the advantage of* $\mathcal{A}$ *on* G*, denoted* $\mathsf{Adv}_{\mathcal{A}}^{G^{(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})}}(\lambda)$*, as the probability of success of* $\mathcal{A}$ *in the unpredictability experiment with preprocessing* $\mathsf{Exp}_{\mathcal{A}}^{G^{(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})}}(\lambda)$ *defined in Figure 2, i.e.,*

$$\mathsf{Adv}_{\mathcal{A}}^{G^{(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})}}(\lambda) = \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{G^{(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})}}(\lambda) = 1\right].$$

We say that an unpredictability application with preprocessing G *is* (S,T,ε)*-secure in the* $(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})$*-model if, for every* (S,T)*-attacker* $\mathcal{A}$ *and every* $\lambda \in \mathbb{N}$*, it holds that*

$$\mathsf{Adv}_{\mathcal{A}}^{G^{(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})}}(\lambda) \leq \varepsilon(\lambda).$$

$\mathsf{Exp}_{\mathcal{A}}^{G^{(\mathcal{O}_{\mathsf{pre}},\mathcal{O}_{\mathsf{main}})}}(\lambda)$:

1. The attacker $\mathcal{A}_0$ makes queries to oracle $\mathcal{O}_{\mathsf{pre}}$ and receives answers from the oracle.
2. At the end of its execution, $\mathcal{A}_0$ outputs an advice bit-string adv of maximal length S bits and forwards adv to the online phase attacker $\mathcal{A}_1$.
3. The challenger $\mathcal{C}$ generates a secret x from the space X and a challenge c from the challenge space $\mathcal{CH}_x$. It forwards $(1^\lambda, c)$ to $\mathcal{A}_1$.
4. On input $(1^\lambda, \mathsf{adv}, c)$, the attacker $\mathcal{A}_1$ makes queries to the oracle $\mathcal{O}_{\mathsf{main}}$ and receives answers from the oracle.
5. The attacker $\mathcal{A}_1$ chooses a guess $x' \in X$ and sends it to $\mathcal{C}$.
6. The output of the experiment is 1 if $x = x'$ and 0 otherwise.

Figure 2 Unpredictability experiment with preprocessing.

As the online attacker in the DDLog problem is not a single algorithm, yet an attacker composed of two algorithms who cannot communicate, we need to introduce a definition of such attacker in the preprocessing model.

▶ **Definition 12** (Distributed (S,T)-attacker). *Let $S, T \in \mathbb{N}$ and $\mathcal{O} = (\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})$ be a preprocessing oracle. A* distributed (S,T)-attacker $\mathcal{A} = (\mathcal{A}_0, (\mathcal{A}_1^{(0)}, \mathcal{A}_1^{(1)}))$ in the $\mathcal{O}$-model *consist of three probabilistic algorithms:*

1. *A preprocessing algorithm $\mathcal{A}_0$, which is computationally unbounded and which interacts with $\mathcal{O}_{\mathsf{pre}}$ and outputs a bit-string of length S bits.*
2. *A pair of online algorithms $(\mathcal{A}_1^{(0)}, \mathcal{A}_1^{(1)})$, which cannot communicate and each of which takes as input an S-bit output of $\mathcal{A}_0$ and a challenge, then makes at most T queries to $\mathcal{O}_{\mathsf{main}}$, and outputs a guess.*

Now, we define the *distributed unpredictability application with preprocessing*, which formalizes the problem of the distributed (S,T)-attacker's online algorithms guessing additive shares of secret information x, given a challenge from the related challenge space $\mathcal{CH}_x$.

▶ **Definition 13** (Distributed unpredictability application with preprocessing). *Let $\lambda \in \mathbb{N}$ be a security parameter. A* distributed unpredictability application with preprocessing *G in the oracle $(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})$-model is defined by a space of secrets X, a set of challenge spaces $\{\mathcal{CH}_x \mid x \in X\}$, where the elements of $\mathcal{CH}_x$ are of the form (c_0, c_1), a PPT challenger $\mathcal{C}$ with oracle access to $\mathcal{O}_{\mathsf{main}}$ and a preprocessing oracle $(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})$. We define the advantage of a distributed (S,T)-attacker $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1^{(0)}, \mathcal{A}_1^{(1)})$ in G, denoted $\mathsf{Adv}_{\mathcal{A}}^{G^{(\mathcal{O}_{\mathsf{pre}},\mathcal{O}_{\mathsf{main}})}}(\lambda)$, as the probability of success of $\mathcal{A}$ in the distributed unpredictability experiment with preprocessing $\mathsf{DistExp}_{\mathcal{A}}^{G^{(\mathcal{O}_{\mathsf{pre}},\mathcal{O}_{\mathsf{main}})}}(\lambda)$ defined in Figure 3, i.e.,*

$$\mathsf{Adv}_{\mathcal{A}}^{G^{(\mathcal{O}_{\mathsf{pre}},\mathcal{O}_{\mathsf{main}})}}(\lambda) = \Pr\left[\mathsf{DistExp}_{\mathcal{A}}^{G^{(\mathcal{O}_{\mathsf{pre}},\mathcal{O}_{\mathsf{main}})}}(\lambda) = 1\right].$$

We say a distributed unpredictability application with preprocessing G is (S,T,ε)-secure in the $(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})$-model if, for every distributed (S,T)-attacker $\mathcal{A}$ and every $\lambda \in \mathbb{N}$, it holds that $\mathsf{Adv}_{\mathcal{A}}^{G^{(\mathcal{O}_{\mathsf{pre}},\mathcal{O}_{\mathsf{main}})}}(\lambda) \leq \varepsilon(\lambda)$.

Now, we define the DDLog problem with preprocessing in the generic group model.

$\mathsf{DistExp}_{\mathcal{A}}^{G^{(\mathcal{O}_{\mathsf{pre}},\mathcal{O}_{\mathsf{main}})}}(\lambda)$:

1. The attacker $\mathcal{A}_0$ makes queries to oracle $\mathcal{O}_{\mathsf{pre}}$ and receives answers from the oracle.
2. At the end of its execution $\mathcal{A}_0$ outputs an advice bit-string adv of maximal length S bits and forwards adv to the online phase attackers $\mathcal{A}_1^{(0)}, \mathcal{A}_1^{(1)}$.
3. The challenger $\mathcal{C}$ generates a secret x from the space X and a challenge (c_0, c_1) from the challenge space $\mathcal{CH}_x$. It forwards $(1^\lambda, c_0)$ to $\mathcal{A}_1^{(0)}$ and $(1^\lambda, c_1)$ to $\mathcal{A}_1^{(1)}$.
4. For $i = 0, 1$:
   a. On input $(\mathsf{adv}, 1^\lambda, c_i)$ the attacker $\mathcal{A}^{(i)}$ makes at most T queries to oracle $\mathcal{O}_{\mathsf{main}}$ and receives answers from the oracle.
   b. The attacker $\mathcal{A}^{(i)}$ chooses a guess $x_i \in X$ and sends it to $\mathcal{C}$.
5. The output of the experiment is 1 if $x_1 - x_0 = x$ and 0 otherwise.

Figure 3 Distributed unpredictability experiment with preprocessing.

▶ **Definition 14** (Distributed discrete logarithm application with preprocessing)**.** *Let $\lambda \in \mathbb{N}$ be a security parameter, let $N, W \in \mathbb{N}$, $N > W$. The (N, W)-*distributed discrete logarithm application with preprocessing $G_{\mathsf{DDLog}}^{(\mathcal{O}_{\mathsf{pre}},\mathcal{O}_{\mathsf{main}})}(N, W)$ *is a distributed unpredictability application with preprocessing in $(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})$-model, where $\mathcal{O}_{\mathsf{pre}}$ is an oracle that samples $\sigma \leftarrow \mathcal{I}_{N,M}$ at the beginning of the experiment and $\mathcal{O}_{\mathsf{main}}$ is a group operation oracle for the encoding function σ, the space of secrets is $X = \mathbb{Z} \cap [-W/2, W/2]$, the challenge space for $x \in X$ is defined as $\mathcal{CH}_x = \{(\sigma(b), \sigma(b + x)) \mid b \in \mathbb{Z}_N\}$. The challenger C^{DDLog} is a PPT algorithm that samples $x \leftarrow X$ and $b \leftarrow \mathbb{Z}_N$. Then C^{DDLog} makes two forward queries $b, b + x$ to the oracle $\mathcal{O}_{\mathsf{main}}$ and passes $(1^\lambda, \sigma(b))$ as a challenge to $\mathcal{A}_1^{(0)}$ and $(1^\lambda, \sigma(b + x))$ as a challenge to $\mathcal{A}_1^{(1)}$.*

3.3 The Auxiliary-Input and Bit-Fixing Models

Following the technique of [5], we define two different preprocessing oracles: *Auxiliary-input generic group oracle* and *Bit-fixing generic group oracle*. The auxiliary-input generic group oracle allows us to model the preprocessing experiment. Nevertheless, it seems difficult to perform an analysis of complexity directly in this model, while the bit-fixing generic group oracle offers a model that is easier to analyse. [5, Theorem 1] proved a relation between an attackers' success probabilities in these two models. We state this result in Proposition 16.

The auxiliary-input generic group oracle allows modelling the preprocessing experiments for generic groups, in the sense that the interface $\mathcal{O}_{\mathsf{pre}}$ allows the offline attacker to see the entire group structure, i.e., the mapping σ. Then, $\mathcal{A}_0$ can choose a bit-string of maximal length S and pass it to the online phase attacker $\mathcal{A}_1$ as an additional input.

On the other hand, the bit-fixing generic group oracle allows the offline attacker to fix P points $(a, s) \in \mathbb{Z}_N \times \mathcal{Y}$ and the mapping σ is chosen afterwards, in the way that it respects these fixed points.

▶ **Definition 15.** *We define:*

Auxiliary-input generic group oracle $\mathsf{AI\text{-}GG}(N, M)$ *is a pair $(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})$, where:*

- *$\mathcal{O}_{\mathsf{pre}}$: Samples $\sigma \leftarrow \mathcal{I}_{N,M}$ and outputs σ.*
- *$\mathcal{O}_{\mathsf{main}}$: Answers forward queries, group-operation queries, and inverse queries using σ sampled by $\mathcal{O}_{\mathsf{pre}}$.*

Bit-fixing generic group oracle $\mathsf{BF\text{-}GG}(P, N, M)$ *is a pair* $(\mathcal{O}_{\mathsf{pre}}, \mathcal{O}_{\mathsf{main}})$*, where:*

- $\mathcal{O}_{\mathsf{pre}}$*: Samples* $\mathcal{Y} \subset [M]$ *of size* N *uniformly at random, takes as input at most* $P \in \mathbb{N}$ *pairs of the form* (a, s)*,* $a \in \mathbb{Z}_N$*,* $s \in \mathcal{Y}$ *with no collisions, and samples* σ *uniformly at random from the subset of* $\mathcal{I}_{N,M}$ *containing all the injections consistent with the sampled image* $\mathcal{Y}$ *and the given fixed points.*
- $\mathcal{O}_{\mathsf{main}}$*: Answers forward queries, group-operation queries, and inverse queries using* σ *sampled by* $\mathcal{O}_{\mathsf{pre}}$*.*

▶ **Proposition 16** ([5, Theorem 1])**.** *Let* $P, N, M \in \mathbb{N}$*,* $N \geq 16$ *and* $\gamma > 0$*. Consider a* (S, T, ε')*-secure unpredictability application with preprocessing* G *in the* $\mathsf{BF\text{-}GG}(P, N, M)$*-model. If* $P \geq 6(S + \log \gamma^{-1}) \cdot T_G^{\mathsf{comb}}$*, then* G *is* (S, T, ε)*-secure in the* $\mathsf{AI\text{-}GG}(N, M)$*-model for* $\varepsilon \leq 2\varepsilon' + \gamma$*. Where* T_G^{comb} *denotes the combined number of queries of the attacker and the challenger.*

Proof. For the proof we refer to [5, Appendix A]. ◀

Importantly, Proposition 16 was formulated in [5] only for (S, T, ε)-secure unpredictability applications with preprocessing, while not considering *distributed* applications. However, we are interested in proving upper bounds on the success probability of an attacker in the *distributed* discrete logarithm application with preprocessing, which is a distributed unpredictability application with preprocessing. If we applied Proposition 16 to the DDLog problem directly, we would be forced to represent the distributed attacker as a non-distributed attacker in the $\mathsf{BF\text{-}GG}(P, N, M)$-model, i.e., an attacker with an online algorithm that gets both challenges and performs up to $2T$ oracle queries. Then, we would apply the theorem on this *stronger* attacker and we would obtain the bounds in the $\mathsf{AI\text{-}GG}(N, M)$-model. Thus, the clear disadvantage of the distributed attacker of having the challenge given to two separate algorithms that are not allowed to communicate cannot be exploited in the $\mathsf{BF\text{-}GG}(P, N, M)$-model before applying the Proposition 16. Overall, this approach leads to loose bounds on the success probability of the distributed attacker. Nevertheless, our central observation is that the theorem holds in the same way for a distributed attacker.

We also remark that the generic group model used in [5] does not allow the attacker to perform inverse queries. This approach is justified by the fact that the authors apply Proposition 16 to derive bounds precise up to polylogarithmic factors in N. The inverse of an element x in a group of order N is equal to x^{N-1}. Therefore, applying the standard square-and-multiply algorithm, we can simulate the inverse operation using $O(\log N)$ group-operations. Thus, the analysis in a version of the generic group model without the inverse queries translates to the result precise up to polylogarithmic factors in N in a model where these queries are allowed. However, we seek to get bounds for the DDLog problem without neglecting logarithmic factors in N, and, therefore, we explicitly include the inverse query in our generic group model. We note that Proposition 16 holds also in our version of the generic group model. We explain the siginificant differences in more detail in the proof sketch of Theorem 17 below.

▶ **Theorem 17.** *Let* $P, N, M \in \mathbb{N}$*,* $N \geq 16$ *and* $\gamma > 0$*. Consider a* (S, T, ε')*-secure distributed unpredictability application with preprocessing* G *in the* $\mathsf{BF\text{-}GG}(P, N, M)$*-model. If* $P \geq 6(S + \log \gamma^{-1}) \cdot T_G^{\mathsf{comb}}$*, then* G *is* (S, T, ε)*-secure in the* $\mathsf{AI\text{-}GG}(N, M)$*-model for* $\varepsilon \leq 2\varepsilon' + \gamma$*. Where* T_G^{comb} *denotes the combined number of queries of the attacker and the challenger.*

Proof sketch. The proof follows from the proof of Proposition 16 stated in [5, Appendix A] replacing the (S, T)-attacker with preprocessing by a distributed (S, T)-attacker with preprocessing.

In order to prove [5, Theorem 1], the authors first prove closeness of two distributions of encoding functions, in the sense that they bound the probability that a distinguisher which is allowed to make T forward (query $a \in \mathbb{Z}_N$ is answered by $\sigma(a)$) and backward (query $l \in [M]$ is answered by $\sigma^{-1}(l)$) oracle queries succeeds to guess from which distribution the encoding function σ of $\mathbb{Z}_N$ on $[M]$ was chosen. In fact, this part of their proof is general and can be applied in the same manner in the setting with distributed attackers.

Then, to prove Proposition 16, they construct an (S,T)-attacker $\mathcal{A}' = (\mathcal{A}'_0, \mathcal{A}'_1)$ for the $\mathsf{BF\text{-}GG}(P,N,M)$ oracle model from an (S,T)-attacker $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ for the $\mathsf{AI\text{-}GG}(N,M)$ oracle model. Their $\mathcal{A}'_1$ is defined as $\mathcal{A}_1$ and their $\mathcal{A}'_0$ first simulates $\mathcal{A}_0$ to get the advice string and, based on it, it fixes at most P points in the encoding function. Thus, $\mathcal{A}'_0$ forces the oracle to choose the encoding function from a convenient distribution. Then, they let the distinguisher $\mathcal{D}$ for the encoding function internally run the online algorithm $\mathcal{A}_1$ on the advice string calculated by $\mathcal{A}_0$ and, subsequently, the challenger $\mathcal{C}$ for the unpredictability experiment with preprocessing. The output of the distinguisher is defined as the output of the unpredictability experiment with preprocessing resulting from the interaction of $\mathcal{A}_1$ and $\mathcal{C}$. The probability of the distinguisher outputting 1 corresponds either to the probability of success of $\mathcal{A}$ in the $\mathsf{AI\text{-}GG}(N,M)$ oracle model or to the probability of success of $\mathcal{A}'$ in the $\mathsf{BF\text{-}GG}(N,M)$ oracle model, depending on which distribution was σ chosen from. Due to the bound on the distinguishing probability between the two aforementioned distributions, they get a relation between the success probabilities of $\mathcal{A}$ in the $\mathsf{AI\text{-}GG}(N,M)$-model and $\mathcal{A}'$ in the $\mathsf{BF\text{-}GG}(P,N,M)$-model stated in the theorem, which concludes their proof.

If we replace the attacker $\mathcal{A}$ by a distributed attacker and define $\mathcal{A}'$ in the same way as in the original proof, $\mathcal{A}'$ will also be a distributed attacker (as the online algorithms of $\mathcal{A}$ and $\mathcal{A}'$ are defined to be the same) and we can perform the proof in the very same fashion as in [5] and get the same results for the distributed attacker.

The proof of Proposition 16 can also be adapted to the version of generic group model which allows inverse queries. When the distinguisher $\mathcal{D}$ simulates $\mathcal{A}_1$ and $\mathcal{C}$, it must provide the answers to their oracle queries. In their proof, [5] only dealt with the forward query, which $\mathcal{D}$ passes as a forward query to its oracle and passes the answer back, and the group-operation query, which is a query of the form $(a_1, a_2) \in [M]^2$, answered by $\sigma(\sigma^{-1}(a_1) + \sigma^{-1}(a_2))$. The group-operation query is simulated by $\mathcal{D}$ by performing two backward queries $\sigma^{-1}(a_1), \sigma^{-1}(a_2)$ and one forward query $\sigma(\sigma^{-1}(a_1) + \sigma^{-1}(a_2))$ to its oracle. In our case, we have to deal also with the inverse query, which is a query of the form $a \in [M]$, answered by the element corresponding to the inverse of a. This query can be simulated as one backward query $\sigma^{-1}(a)$ and one forward query $\sigma(-\sigma^{-1}(a))$. As in the original proof, our distinguisher makes at most $3T^{\mathsf{comb}}$ queries, where T^{comb} is the combined number of queries of $\mathcal{A}_1$ and $\mathcal{C}$. The rest of the proof is the same, and the same results follow. ◀

4 Lower Bounds for DDLog with Preprocessing in the Generic Group Model

Now, we present our main theorem giving an upper bound on the success probability of a distributed attacker with preprocessing in the DDLog problem. Our theorem is based on [5, Theorem 10], which examines the discrete logarithm problem in the preprocessing model. The structure of our proof is similar to the one of [5, Theorem 10]. However, if we simply adjusted the proof of [5, Theorem 10] to our case, we would get the bound for the success probability of an attacker $\varepsilon = O\left(\frac{ST^2+T^2\cdot\log(W)}{W}\right)$.

Nevertheless, the distributed attacker allows us to make a more careful analysis and obtain quantitatively better results. As the distributed attacker is a *weaker* attacker than its non-distributed representation, we can obtain a tighter bound in the $\mathsf{BF\text{-}GG}(P,N,M)$ oracle model. Then, we translate this bound to the $\mathsf{AI\text{-}GG}(N,M)$ oracle model using Theorem 17.

▶ **Theorem 18.** *Let $N, W \in \mathbb{N}$, $N > W$, N be a prime. The (N,W)-distributed discrete logarithm application with preprocessing $G_{\mathsf{DDLog}}^{\mathsf{AI\text{-}GG}(N,M)}(N,W)$ is (S,T,ε)-secure in the $\mathsf{AI\text{-}GG}(N,M)$-model for any*

$$\varepsilon = O\left(\frac{T^2}{W} + \frac{\max\{S, \log W\} \cdot T^2}{N}\right),$$

where W denotes the length of the interval in the DDLog problem. Furthermore, if $N \geq W \cdot \log(W)$ the theorem holds for

$$\varepsilon = O\left(\frac{T^2}{W} + \frac{S \cdot T^2}{N}\right).$$

The proof of Theorem 18 is given in Appendix A.

References

1. Jeremiah Blocki and Seunghoon Lee. On the multi-user security of short Schnorr signatures. *IACR Cryptol. ePrint Arch.*, page 1105, 2019. URL: `https://eprint.iacr.org/2019/1105`.
2. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: Optimizations and applications. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 2105–2122. ACM, 2017. `doi:10.1145/3133956.3134107`.
3. Elette Boyle, Itai Dinur, Niv Gilboa, Yuval Ishai, Nathan Keller, and Ohad Klein. Locality-preserving hashing for shifts with connections to cryptography. In Mark Braverman, editor, *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*, volume 215 of *LIPIcs*, pages 27:1–27:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. `doi:10.4230/LIPIcs.ITCS.2022.27`.
4. Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 509–539. Springer, 2016. `doi:10.1007/978-3-662-53018-4_19`.
5. Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. 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 I*, volume 10991 of *Lecture Notes in Computer Science*, pages 693–721. Springer, 2018. `doi:10.1007/978-3-319-96884-1_23`.
6. Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. 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 II*, volume 10821 of *Lecture Notes in Computer Science*, pages 415–447. Springer, 2018. `doi:10.1007/978-3-319-78375-8_14`.
7. Itai Dinur, Nathan Keller, and Ohad Klein. An optimal distributed discrete log protocol with applications to homomorphic secret sharing. *J. Cryptol.*, 33(3):824–873, 2020. `doi:10.1007/s00145-019-09330-2`.

8 Siyao Guo, Qian Li, Qipeng Liu, and Jiapeng Zhang. Unifying presampling via concentration bounds. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part I*, volume 13042 of *Lecture Notes in Computer Science*, pages 177–208. Springer, 2021. doi:10.1007/978-3-030-90459-3_7.

9 Ueli M. Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005. doi:10.1007/11586821_1.

10 Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Hash function requirements for Schnorr signatures. *J. Math. Cryptol.*, 3(1):69–87, 2009. doi:10.1515/JMC.2009.004.

11 Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of Paillier: Homomorphic secret sharing and public-key silent OT. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 678–708. Springer, 2021. doi:10.1007/978-3-030-77870-5_24.

12 Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997. doi:10.1007/3-540-69053-0_18.

13 Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 205–223. Springer, 2007. doi:10.1007/978-3-540-74143-5_12.

A Proof of Theorem 18

We start with a short overview of the proof. In order to bound the advantage of a distributed attacker in the $\mathsf{AI\text{-}GG}(N, M)$-model, we analyse its advantage in the $\mathsf{BF\text{-}GG}(P, N, M)$-model and we use Theorem 17 to translate this bound to the $\mathsf{AI\text{-}GG}(N, M)$-model. To bound the attacker's advantage in the $\mathsf{BF\text{-}GG}(P, N, M)$-model, we construct an alternative experiment, where we answer the attacker's queries with randomly chosen elements, while we keep answers consistent in between themselves maintaining a table of already seen elements. The secrets $x \in \mathbb{Z} \cap [-W/2, W/2]$ and $b \in \mathbb{Z}_N$ are chosen uniformly at random from the respective sets at the end of the experiment, after the attacker outputs its guess. Then, we express the probability of success of the distributed attacker in this alternative experiment and we bound the probability that the distribution of query responses given in the alternative experiment differs from the one that would be given in an honest execution (Lemmas 20 and 21). This gives us the bound on the attacker's advantage in the $\mathsf{BF\text{-}GG}(P, N, M)$-model.

Now, we proceed with the formal proof of the theorem. First, consider the interaction of $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1^{(0)}, \mathcal{A}_1^{(1)})$ with C^{DDLog} in the $\mathsf{BF\text{-}GG}(P, N, M)$-model, during which the $\mathcal{O}_{\mathsf{pre}}$ outputs $\mathcal{Y}$, the image of σ. In the rest of the proof, we condition on the choice of $\mathcal{Y}$. Next, we define an *alternative experiment*, during which we construct a table of pairs $(v(X), s)$, where $s \in \mathcal{Y}$ and $v(X) \in \mathbb{Z}_N[X, B]$ is a formal polynomial corresponding to the preimage of s under σ. The experiment and, correspondingly, the construction of the table proceed as follows:

1. $\mathcal{A}_0$ fixes σ in at most P points (a, s), $a \in \mathbb{Z}_N$, $s \in \mathcal{Y}$. We add each such point to the table as a pair (a, s), where a is a constant polynomial.
2. To create the challenge $s_b \in \mathcal{Y}$ for $\mathcal{A}_1^{(0)}$, C^{DDLog} chooses s_b from all unused values in $\mathcal{Y}$ uniformly at random. The pair (B, s_b) is stored in the table.
3. The execution of $\mathcal{A}_1^{(0)}$:
   a. Upon a forward query $q \in \mathbb{Z}_N$ the table is checked for the occurrence of $q \in \mathbb{Z}_N[X, B]$ as a constant polynomial. If such occurrence is found, we respond by the corresponding $s_q \in \mathcal{Y}$ that occurs in the table in a pair with q. Otherwise, we sample s_q uniformly at random from all the unused values in $\mathcal{Y}$ and we store the pair (q, s_q) in the table.
   b. To a group-operation query (s_1, s_2) we respond by $\perp$ if s_1 or s_2 is not in $\mathcal{Y}$. Otherwise, if s_1 is not in the table, we sample a_1 uniformly at random from all unused values in $\mathbb{Z}_N$ and we store the pair (a_1, s_1) in the table. The same applies for s_2. Afterwards, both s_1, s_2 are already stored in the table in pairs with some polynomials $u_1, u_2 \in \mathbb{Z}_N[B]$. We check the table for an occurrence of the polynomial $u_1 + u_2$. If there is a record $(u_1 + u_2, s_3)$ for some $s_3 \in \mathcal{Y}$ in the table, we respond by s_3 to the query. Otherwise we sample s_3 uniformly at random from all unused values in $\mathcal{Y}$, we store $(u_1 + u_2, s_3)$ in the table, and we respond by s_3 to the query.
   c. To an inverse query s_1, we respond by $\perp$ if $s_1 \notin \mathcal{Y}$. If $s_1 \in \mathcal{Y}$ and s_1 is not in the table, we sample u uniformly at random from all unused values in $\mathbb{Z}_N$ and we append the pair (u, s_1) to the table. Now, s_1 is in the table in pair with some polynomial $u \in \mathbb{Z}_N[B]$. We check the table for an occurrence of $(N-1) \cdot u$, if such entry $((N-1) \cdot u, s_2)$ is found for some $s_2 \in \mathcal{Y}$, we answer the query by s_2. Otherwise, we sample s_2 uniformly at random from all unused labels from $\mathcal{Y}$, answer the query by s_2, and we add the pair $((N-1) \cdot u, s_2)$ to the table.
   d. At the end of its execution, $\mathcal{A}_1^{(0)}$ outputs x_0.
4. To create the challenge s_{b+x} for $\mathcal{A}_1^{(1)}$, C^{DDLog} chooses s_{b+x} from all unused values in $\mathcal{Y}$ uniformly at random. The pair $(B + X, s_{b+x})$ is stored in the table.
5. The execution of $\mathcal{A}_1^{(1)}$ is handled in the same way as the execution of $\mathcal{A}_1^{(0)}$. Except if
   - $\mathcal{A}_1^{(1)}$ queries a group-operation query (s_1, s_2) such that either $(\alpha \cdot B + \beta, s_1)$, or $(\alpha \cdot B + \beta, s_2)$, for some $\alpha, \beta \in \mathbb{Z}_N$, $\alpha \neq 0$ is already in the table, or
   - $\mathcal{A}_1^{(1)}$ queries an inverse query s_1 such that $(\alpha \cdot B + \beta, s_1)$ for some $\alpha, \beta \in \mathbb{Z}_N$, $\alpha \neq 0$ is already in the table.

   We denote the event when either one of the above bullet points happens as F. If the event F occurs, we answer the query by $\perp$ and we do not append anything to the table. At the end of its execution, $\mathcal{A}_1^{(1)}$ outputs x_1.
6. C^{DDLog} chooses x uniformly at random from all integers in $[-W/2, W/2]$ and b uniformly at random from $\mathbb{Z}_N$. C^{DDLog} outputs 1 if and only if $x_1 - x_0 = x$.

Note that all of the polynomials in the table at the end of the execution are distinct, as we always first check for an occurrence of a polynomial before adding it to the table. We define a collision event, denoted E, as the event when, after a substitution of the values x, b chosen by C^{DDLog} for the variables X, B (respectively) in the polynomials in the table, there exist two entries (a, s) and (a, s') in the table such that $s \neq s'$. Note that the event E corresponds to a discrepancy in the query responses to the attacker. In other words, there is no encoding function σ such that all of our query responses would be correct because we associated the image of a with two distinct elements s, s'. If the event E does not occur and the event F does not occur either then there exists an encoding function σ compatible with all of our query responses. Furthermore, as we always choose the elements uniformly

at random from the appropriate sets, also the distribution of responses in the alternative experiment is identical to the distribution of responses in the real unpredictability experiment with preprocessing.

Thus, the distribution of answers seen by $\mathcal{A}$ in the alternative experiment differs from the one in the honest experiment only if at least one of the events E, F occurs. We bound the probability that the execution differs from the honest execution by bounding the probability $\Pr[E \cup F] \leq \Pr[E \mid \neg F] + \Pr[F]$.

Below, we introduce a lemma that characterizes the structure of the contents of the table at the end of the alternative experiment.

▶ **Lemma 19.** *At most $2T + 2$ non-constant polynomials are in the table at the end of the execution of the alternative experiment. Moreover,*

1. *at most $T + 1$ of those are of the form $\alpha B + \beta$, for some $\alpha, \beta \in \mathbb{Z}_N$, $\alpha \neq 0$ (we say "polynomials of type 1") and they were added either as the challenge for $\mathcal{A}_1^{(0)}$ in the step 2 or as a polynomial corresponding to a group-operation query response during the execution of $\mathcal{A}_1^{(0)}$ in the step 3 b, or as a polynomial in pair with a response to an inverse query in the step 3 c. Furthermore, the value $s \in \mathcal{Y}$ in pair with such polynomial in the table is never returned as an answer to a query made by $\mathcal{A}_1^{(1)}$. Also,*
2. *at most $T + 1$ of non-constant polynomials in the table are of the form $\alpha(X + B) + \beta$, for some $\alpha, \beta \in \mathbb{Z}_N$, $\alpha \neq 0$ (we say "polynomials of type 2") and they were added either as the challenge for $\mathcal{A}_1^{(1)}$ in the step 4 or as a polynomial corresponding to a group-operation query response or inverse query response during the execution of $\mathcal{A}_1^{(1)}$ in step 5. Furthermore, the value $s \in \mathcal{Y}$ in pair with such polynomial in the table is never returned as an answer to a query made by $\mathcal{A}_1^{(0)}$.*

There are no non-constant polynomials in the table of forms different than the polynomials of type 1 and 2.

Proof of Lemma 19. First, notice that a non-constant polynomial can only be introduced in the table in a pair with a challenge s_b, s_{b+x}, or in a pair with a response to a group-operation or inverse query, where at least one of the queried elements s_1, s_2 or the queried element s_1 (for the inverse query) has already been in the table in pair with a non-constant polynomial. As there are two challenges and at most $2T$ queries performed by the distributed attacker, there is at most $2T + 2$ non-constant polynomials in the table at the end of the execution.

Next, we show that the second part of the lemma holds. Notice that the first polynomial with a non-zero coefficient next to the variable X is added to the table with the challenge s_{b+x} in the step 4. Therefore, the polynomials with a non-zero coefficient next to the variable X can be added to the table as a result of a group-operation query or an inverse query only after this moment. Since there are at most T queries performed after this moment, there cannot be more than $T + 1$ polynomials with a non-zero coefficient next to the variable X, and, thus, not more than $T + 1$ polynomials of type 2. Furthermore, when a new polynomial is added to the table, we sample its pair value s from the unused values in $\mathcal{Y}$. As the polynomials of type 2 appear in the table only after the end of execution of $\mathcal{A}_1^{(0)}$, none of the values s in pairs with such polynomials could have been returned as an answer to a query made by $\mathcal{A}_1^{(0)}$.

Next, we prove that, at the end of the alternative experiment, there are no non-constant polynomials of forms different than polynomials of type 1 and 2 in the table.

Recall that non-constant polynomials are only being added to the table in the challenge pair, during a group-operation query as a sum of two polynomials already present in the table, where at least one of them is non-constant, or during an inverse query as an $(N - 1)$

multiple of a polynomial in pair with the queried element if it is non-constant. Therefore, it is obvious that all polynomials in the table are at most of degree one in both B and X. Now, it is enough to show there is no polynomial of the form $\alpha_1 B + \alpha_2 X + \beta$, where $\alpha_1, \alpha_2, \beta \in \mathbb{Z}_N$, $\alpha_2 \neq 0$, $\alpha_1 \neq \alpha_2$ that we refer to as a "polynomial of type 3". We already know no polynomial of this form has been added to the table before the step 4 of the experiment because all of the polynomials until this step are of degree zero in the variable X. The polynomial added with the challenge in step 4 is $X + B$, which is a polynomial of type 2. Therefore, it is enough to look at the polynomials added to the table during step 5. We prove by induction that no non-constant polynomials of type different than type 2 are added to the table during step 5. Suppose no non-constant polynomials different than type 2 were added to the table during step 5 before the i-th query in step 5. For $i = 1$, the assumption holds trivially. We prove the inductive step for $i + 1$ next. By the inductive hypothesis, it follows that there are no polynomials different from the constant polynomials, polynomials of type 1, and polynomials of type 2 before the i-th query of step 5 in the table. In case i-th query is a forward query, either a constant polynomial or no polynomial is added to the table. In case the i-th query is a group operation query (s_1, s_2), the following cases can occur:

1. At least one of the pair s_1, s_2 is not in $\mathcal{Y}$. Then $\perp$ is returned and no polynomials are added to the table.
2. Both s_1, s_2 lie in $\mathcal{Y}$, and either none of them is in the table or one of them is in the table in pair with a constant polynomial and the other one is not in the table or both are in the table in pair with a constant polynomial. Then, only constant polynomials are added to the table.
3. Both s_1, s_2 lie in $\mathcal{Y}$, one of them is in the table in pair with a polynomial of type 2, and the other one is not in the table, is in the table in pair with a constant polynomial, or is in the table in pair with a polynomial of type 2. Then, the response will be in the table in pair with a polynomial of type 2 or a constant polynomial.
4. Both s_1, s_2 lie in $\mathcal{Y}$ and one of them is in the table in pair with a polynomial of type 1. Then, the query is answered by $\perp$ and no polynomials are added to the table.

In case the i-th query is an inverse query s_1, the following cases can occur:

1. The element s_1 does not lie in $\mathcal{Y}$. Then, $\perp$ is returned and no polynomials are added to the table.
2. The element s_1 lies in $\mathcal{Y}$ and s_1 is not in the table, or it is in the table in a pair with a constant polynomial. Then, only constant polynomials are added to the table.
3. The element s_1 lies in $\mathcal{Y}$ and s_1 is in the table in a pair with a polynomial of type 2. Then, the response is in the table in pair with a polynomial of type 2.
4. The element s_1 lies in $\mathcal{Y}$ and s_1 is in the table in pair with a polynomial of type 1. Then, the query is answered by $\perp$ and no polynomials are added to the table.

By the above exhaustive case-analysis, no non-constant polynomial of type different than type 2 was added during the i-th query. Therefore, no non-constant polynomial of type different than type 2 was added during step 5. Thus, no non-constant polynomials of forms different than the polynomials of type 1 and 2 are in the table at the end of the execution of the alternative experiment.

Part 1 of the lemma follows from the fact that no non-constant polynomial of type different than type 2 is being added to the table during the execution of $\mathcal{A}_1^{(1)}$. More precisely, the polynomials of type 1 can only be introduced in the table in a pair with the challenge s_b or in a pair with a response to a group-operation query or an inverse query by $\mathcal{A}_1^{(0)}$. Since $\mathcal{A}_1^{(0)}$ makes at most T queries, at most $T + 1$ polynomials of type 1 are in the table at the

end of the execution. By the analysis of the possible group-operation and inverse query responses during step 5, an s in a pair with a polynomial of type 1 is never returned as an answer to a query made by $\mathcal{A}_1^{(1)}$. This concludes the proof of Lemma 19. ◀

The following lemma bounds the probability $\Pr[E \mid \neg F]$ (Due to space restrictions, the proof of the lemma appears in the full version).

▶ **Lemma 20.** *The probability* $\Pr[E \mid \neg F]$ *can by bounded as follows*

$$\Pr[E \mid \neg F] \leq \frac{(T+1)^2}{W} + \frac{(2T+2)\cdot(P+6T+1)}{N}.$$

In the following lemma, we bound the probability $\Pr[F]$ (Due to space restrictions, the proof of the lemma appears in the full version).

▶ **Lemma 21.** *The probability* $\Pr[F]$ *can be bounded as follows*

$$\Pr[F] \leq \frac{2\cdot T\cdot(T+1)}{N-(P+5T)}.$$

By the above Lemmas 20 and 21, we get

$$\Pr[E \cup F] \leq \frac{(T+1)^2}{W} + \frac{(2T+2)\cdot(P+6T+1)}{N} + \frac{2\cdot T\cdot(T+1)}{N-(P+5T)}.$$

Since x is chosen at the end of the experiment uniformly at random from the integer values in the interval $[-W/2, W/2]$ in the alternative experiment, the success probability of any $\mathcal{A}$ in the alternative experiment is at most $1/W$. By the union bound, we can bound the success probability ε' of $\mathcal{A}$ in the standard experiment as

$$\varepsilon' \leq \frac{(T+1)^2+1}{W} + \frac{(2T+2)\cdot(P+6T+1)}{N} + \frac{2\cdot T\cdot(T+1)}{N-(P+5T)}.$$

In the rest of the proof, we assume that $N \geq 16$ (required by Proposition 16). This can be done without loss of generality since we are proving an asymptotic bound. Now, we apply Theorem 17 in order to bound the attacker's success probability ε in the $\mathsf{AI\text{-}GG}(N, M)$-model. It holds that $T^{\mathsf{comb}}_{G^{\mathsf{DDLog}}} = 2T+2$, and we set $\gamma := 1/W$ and $P := 6(S+\log(W))\cdot(2T+2)$. By Theorem 17, we get that

$$\begin{aligned}\varepsilon &\leq 2\cdot\varepsilon' + \gamma \\ &\leq \frac{2\cdot(T+1)^2+3}{W} + \frac{2\cdot(2T+2)\cdot(P+6T+1)}{N} + \frac{4\cdot T\cdot(T+1)}{N-(P+5T)}.\end{aligned}$$

In the rest of the proof, we assume $T \geq 72$. Note that, after proving our result for the attackers with $T \geq 72$, we can bound the advantage of an attacker making less queries as follows. Suppose an attacker $\mathcal{A}$ makes $0 < T < 72$ queries during its execution. Note that any attacker $\mathcal{B}$ making $\tilde{T} := 72\cdot T$ queries can simulate $\mathcal{A}$. However, by our result, we can bound the advantage of $\mathcal{B}$ as $\epsilon_{\mathcal{B}} = O\left(\frac{\tilde{T}^2}{W} + \frac{\max\{S, \log W\}\cdot\tilde{T}^2}{N}\right)$. Therefore, the advantage $\epsilon_{\mathcal{A}}$ of $\mathcal{A}$ is also bounded by this expression and, as $\tilde{T} = 72\cdot T$, it also holds that $\epsilon_{\mathcal{A}} = O\left(\frac{T^2}{W} + \frac{\max\{S, \log W\}\cdot T^2}{N}\right)$.

Furthermore, assume that $N \geq 72 \cdot \max\{S, \log(W), 1\} \cdot T$. Otherwise if $N < 72 \cdot \max\{S, \log(W), 1\} \cdot T \leq \max\{S, \log(W), 1\} \cdot T^2$, then $\frac{\max\{S, \log(W), 1\} \cdot T^2}{N} > 1$ and the theorem's bound is looser than $\varepsilon = O(1)$, which holds trivially. Thus, we get

$$\begin{aligned} N - (P + 5T) &= N - (6(S + \log(W)) \cdot (2T + 2) + 5T) \\ &\geq N - (12 \cdot \max\{S, \log(W), 1\} \cdot (2T + 2) + 5T) \\ &\geq N - 36 \cdot \max\{S, \log(W), 1\} \cdot T \\ &\geq N/2. \end{aligned}$$

Consequently, for the sum of the second and the third term, we have

$$\begin{aligned} &\frac{2 \cdot (2T + 2) \cdot (P + 1 + 6T)}{N} + \frac{4 \cdot T \cdot (T + 1)}{N - (P + 5T)} \\ &\qquad \leq \frac{2 \cdot (2T + 2) \cdot (P + 1 + 6T)}{N} + \frac{8 \cdot T \cdot (T + 1)}{N} \\ &\qquad = \frac{4 \cdot (T + 1)(P + 1 + 8T)}{N} \\ &\qquad = \frac{4 \cdot (T + 1)(6 \cdot (S + \log W)(2T + 2) + 1 + 8T)}{N} \\ &\qquad = \frac{48 \cdot (T + 1)^2 \cdot (S + \log W) + (4T + 4) \cdot (1 + 8T)}{N} \\ &\qquad \leq \frac{96 \cdot \max\{S, \log(W)\} \cdot (T + 1)^2 + (4T + 4) \cdot (1 + 8T)}{N} \\ &\qquad \leq \frac{192 \cdot \max\{S, \log(W), 1\} \cdot T^2}{N}, \end{aligned}$$

where the last inequality holds as we assume $T \geq 72$. Specifically, for all $T \geq 72$, it holds that

$$96T^2 \geq 96 \cdot (2T + 1) + (4T + 4) \cdot (1 + 8T) = 32T^2 + 228T + 100,$$

which we use to bound the sub-quadratic terms in T in the nominator.

Therefore, it holds that

$$\varepsilon \leq \frac{3T^2}{W} + \frac{192 \cdot \max\{S, \log(W), 1\} \cdot T^2}{N} = O\left(\frac{T^2}{W} + \frac{\max\{S, \log(W)\} \cdot T^2}{N}\right).$$

Furthermore, if $N \geq (W \cdot \log(W))$ then we get that

$$\begin{aligned} \varepsilon \leq& \frac{3T^2}{W} + \frac{192 \cdot \max\{S, 1\} \cdot T^2}{N} + \frac{192 \cdot \log(W) \cdot T^2}{N} \\ \leq& \frac{3T^2}{W} + \frac{192 \cdot \max\{S, 1\} \cdot T^2}{N} + \frac{192 \cdot T^2}{W} = O\left(\frac{T^2}{W} + \frac{S \cdot T^2}{N}\right). \end{aligned}$$

The above bounds establish Theorem 18.

A Note on the Complexity of Private Simultaneous Messages with Many Parties

Marshall Ball
Courant Institute of Mathematical Sciences, New York University, NY, USA

Tim Randolph
Columbia University, New York, NY, USA

Abstract

For $k = \omega(\log n)$, we prove a $\Omega(k^2 n / \log(kn))$ lower bound on private simultaneous messages (PSM) with k parties who receive n-bit inputs.

This extends the $\Omega(n)$ lower bound due to Appelbaum, Holenstein, Mishra and Shayevitz [Journal of Cryptology, 2019] to the many-party ($k = \omega(\log n)$) setting. It is the first PSM lower bound that increases quadratically with the number of parties, and moreover the first *unconditional, explicit* bound that grows with both k and n. This note extends the work of Ball, Holmgren, Ishai, Liu, and Malkin [ITCS 2020], who prove communication complexity lower bounds on decomposable randomized encodings (DREs), which correspond to the special case of k-party PSMs with $n = 1$. To give a concise and readable introduction to the method, we focus our presentation on perfect PSM schemes.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols; Theory of computation → Communication complexity; Security and privacy → Information-theoretic techniques

Keywords and phrases Secure computation, Private Simultaneous Messages

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.7

Funding *Marshall Ball*: Part of this work was performed while the author was a student at Columbia University and a postdoc at University of Washington. This material is based upon work supported by the National Science Foundation under Grant #2030859 to the Computing Research Association for the CIFellows Project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation nor the Computing Research Association.
Tim Randolph: This material is based upon work supported by the following grants: NSF CCF-1563155, NSF IIS-1838154, NSF CCF-1814873, NSF CCF-1703925, NSF CCF-2106429, and NSF CCF-2107187.

Acknowledgements The authors thank Tal Malkin for helpful discussion, and several anonymous reviewers for helpful comments on an earlier draft.

1 Introduction

In this note, we consider *private simultaneous message* (PSM) protocols. In this setting, we are given as public input a function $f : [N]^k \to \{0,1\}$.[1] There are k parties, each with private inputs $x_1, x_2, \ldots, x_k$ and access to a shared random string. Each party sends a single message to a referee in such a way that the referee can reconstruct $f(x_1, x_2, ..., x_k)$ but learns no other information about the private inputs. The communication complexity, or size, of a PSM protocol is the total number of bits sent by all parties.

[1] We write $f : \{\{0,1\}^n\}^k \to \{0,1\}$, where $n = \log_2(N)$, interchangeably.

The (2-party) PSM model was introduced as an elegant variant of secure computation by Feige, Kilian, and Naor in 1994 [8], and readily extended to the k-party case by Ishai and Kushilevitz [11]. The simple protocol structure makes the model an attractive proving ground for understanding the "cost of privacy" (i.e., communication complexity) in secure computation over distributed protocols with no privacy or security guarantees.

In addition to their theoretical importance, PSM protocols have practical import because they readily decompose into an expensive offline phase and an efficient single-message online phase.[2]

Significant progress on upper and lower bounds on the communication complexity of PSM has occurred since its introduction. In 2019, Assouline and Liu, building on the work of Beimel, Kushilevitz, and Nissim [5, 6], demonstrated a general PSM protocol with complexity $O_k(N^{(k-1)/2})$ for infinitely many k [2]. Also in 2019, Applebaum, Holenstein, Mishra, and Shayevitz [1] patched up a hole in the original PSM paper to establish a $3n - O(\log(n))$ lower bound for the two party case. This bound can be trivially extended to the k-party case by dividing the input of a boolean function on kn bits between k parties, in which case the bound becomes $3kn - O(\log(n))$.

The PSM setting can also be viewed as the addition of a security requirement to simultaneous multi-party communication complexity. Although the *number-on-forehead* model, in which players can see every input but their own, is more common in this setting, perfect multi-party PSM corresponds well to the deterministic *number-in-hand* model, in which k parties receive inputs $x_1, x_2, \cdots, x_k$ and send messages that allow a referee to compute $f(x_1, x_2, \cdots, x_k)$. $\Omega(kn)$ lower bounds exist in this setting (see, for example, [9]). Every multi-party PSM protocol with perfect correctness can be converted to a deterministic simultaneous number-in-hand communication protocol by fixing a shared random string, so these lower bounds apply to our setting.

A final related notion is that of *randomized encodings*, which originate from Yao's garbled circuits [15] and were elaborated by Ishai and Kushilevitz [12]. A randomized encoding of a function $f : \{0,1\}^k \to \{0,1\}$ is a function $\hat{f}$ that takes as input $x \in \{0,1\}^k$ and a random string r. For any r, given $\hat{f}(x, r)$ it must be possible to recover $f(x)$ but no other information about x. A randomized encoding is *decomposable* if $\hat{f}(x, r)$ can be written $(\hat{f}_1(x_1, r), \hat{f}_2(x_2, r), ..., \hat{f}_n(x_k, r))$; that is, if every bit of the output depends only on a single bit of x. Thus a DRE immediately implies a PSM protocol: parties compute the relevant portions of $\hat{f}$ using their shared randomness and send them to the referee. Moreover, a k-party PSM scheme in which every party receives exactly one bit is a DRE. A recent paper by Ball, Holmgren, Ishai, Liu, and Malkin establishes $\Omega(k^2/log(k))$ lower bounds on the communication complexity of DREs, implying corresponding lower bounds on PSM protocols [3]. Their result relies on a measure of function complexity first introduced by Nečiporuk in 1966 [13] and subsequently used to prove lower bounds on the sizes of formulas, branching programs and span programs [4, 7].

In this note, we introduce a new variant of Nečiporuk's measure of function complexity (as defined in [3]) and show it suffices to extend the approach of [3]: demonstrating that PSM complexity is lower-bounded by our modified Nečiporuk measure. We give two natural examples of explicit functions with modified Nečiporuk measure $\Omega(k^2 n/\log(kn))$ and additionally show that nearly all functions have such measure. Our main result is the following theorem:

[2] This paradigm has gained significant traction in recent years. However, typical use cases require stronger security requirements than that provided by PSM, such as resilience to the referee colluding with senders (Non-Interactive Secure Multiparty Computation). Note that lower bounds on the communication complexity of PSM protocols immediately extend to lower bounds on its stronger cousins.

▶ **Theorem 1.** *As long as $k = \omega(\log(n))$, there exist (explicit) functions $f : [N]^k \to \{0,1\}$ such that any perfect PSM $\mathcal{X}$ for f has size*

$$|\mathcal{X}| \geq \frac{k^2 n}{2\log_2(kn)} - O(kn).$$

To the best of our knowledge, this is the first unconditional, explicit[3] lower bound on the communication of PSM that increases with k and n, the first to grow quadratically with k, as well as the best in the many-party ($k = \omega(\log(n))$) setting.

In Appendix B, we argue that significantly better bounds (i.e., $\omega(k^2 n)$) will likely require very different techniques due a certain kind of natural proof barrier [14] which may be of independent interest.

2 Preliminaries

▶ **Definition 2** (PSM Protocol). *A PSM protocol $(M, R, (\Pi)_{i\in[k]}, \texttt{Ref})$ for a function $f : [N]^k \to \{0,1\}$ specifies a message space M, a randomness space R, a tuple of functions, $\Pi_i : [N] \times R \to M$ for $i \in [k]$, and a referee $\texttt{Ref} : [M]^k \to \{0,1\}$. It must satisfy that for every input $x \in [N]^k$ and every $r \in R$,*

$$\texttt{Ref}\big((\Pi_i(x_i, r))_{i\in[k]}\big) = f(x)$$

(perfect correctness), and that for all $x, y \in [N]^k$ such that $f(x) = f(y)$, when r is sampled uniformly from R,

$$\{(\Pi_i(x_i, r))_{i\in[k]}\}_{r\sim R} \equiv \{(\Pi_i(y_i, r))_{i\in[k]}\}_{r\sim R}$$

(perfect security).

Informally, each function Π_i captures the behavior of one of the k agents. Given an input in $[N]$ and a shared random string drawn from R, each agent produces a private message $m \in M$ that is sent to the referee. The referee evaluates f based on the messages she receives. Correctness and security requirements ensure that the referee always evaluates f correctly but never learns any other information about the input.

For this note, we consider perfect correctness and security. This choice keeps our presentation tidy while focusing on our contribution, which is a modification of Nečiporuk's measure and the extension of [3] to the multiparty PSM setting. The perfect security case highlights the core conceptual techniques of [3]. The result extends to multiparty PSMs with statistical or even (nonuniform) computational security, following the presentation of [3].

The traditional definition of the PSM setting is provided for completeness and comparison. For our investigation, it will be convenient to use the following equivalent definition of a PSM.

▶ **Definition 3** (Random Family Definition of PSM). *A PSM protocol for a function $f : [N]^k \to \{0,1\}$ is a family of random variables*

$$\mathcal{X} = (\mathcal{X}_j^i)_{j\in[N]}^{i\in[k]}$$

[3] Applebaum et al. [1] show a random function requires high communication unconditionally. They only construct an explicit function with high communication *under a hardness assumption on co-nondeterministic circuits*. In contrast, we give an unconditional explicit lower bound.

supported on M and a referee function $\texttt{Ref}$ *such that for all* $x \in [N]^k$,

$$\Pr[\texttt{Ref}((\mathcal{X}^i_{x_i})^{i\in[k]}) = f(x)] = 1$$

(correctness) and for all x, y *such that* $f(x) = f(y)$, *we have*

$$(\mathcal{X}^i_{x_i})^{i\in[k]} \equiv (\mathcal{X}^i_{y_i})^{i\in[k]}$$

(security).

▶ **Definition 4** (PSM size [3]). *The* size *of a PSM* $\mathcal{X}$ *is*

$$|\mathcal{X}| = \max_{x\in[N]^k} \sum_{i\in[k]} \lceil \log_2(|\texttt{Supp}(\mathcal{X}^i_{x_i})|)\rceil,$$

where $\texttt{Supp}(\mathcal{Y})$ *indicates the support of* $\mathcal{Y}$.

This coincides with the multiparty extension of Applebaum et al.'s definition of size as $\log|A| + \log|B|$, where A and B are the message spaces of the two parties, in the two party case. This notion of size corresponds to the sum of the channel widths required by the sending parties.

3 A Measure of the Complexity of a Function

To lower bound PSM size, we use Nečiporuk's measure of function complexity, which counts the maximum number of distinct nonzero restrictions of a boolean function over any partition of the input. For any subset $S \subseteq [k]$, we write $\overline{S}$ to denote $[k] \setminus S$, and $f_{S|z}$ to indicate the restriction of f to S in which the coordinates corresponding to $\overline{S}$ are fixed to the vector $z \in \{0,1\}^{\overline{S}}$.

▶ **Definition 5** (Nečiporuk's Measure [13, 3]). *Let* $f : \{0,1\}^k \to \{0,1\}$ *be any boolean function. For any subset* $S \subseteq [k]$, *define*

$$g_S(f) := \log_2(|\{f_{S|z} : z \in \{0,1\}^{\overline{S}}, f_{S|z} \not\equiv 0\}|).$$

For any positive integer $m \leq k$, *let* $V = (V_1, V_2, ..., V_m)$ *denote an* m*-partition of* $[k]$. *The measure is* $G(f) := \max_V \sum_{V_i\in V} g_{V_i}(f)$.

We are interested in functions that take boolean input in $\{\{0,1\}^n\}^k$. Unlike in the DRE setting, each message sent to the referee depends collectively on n bits of the input. Thus, when we restrict functions, we will need to carefully distinguish between inputs in $\{0,1\}^n$ in which all bits are restricted and inputs in $\{0,1\}^n$ in which only some bits are restricted. This requires a corresponding modification of Nečiporuk's measure.

▶ **Definition 6** (First-Bit Restriction). *For any function* $f : \{\{0,1\}^n\}^k \to \{0,1\}$ *and any set* $S \subseteq [k]$, *the* first-bit restriction of f to S using (α, β) *is the function*

$$f_{S|(\alpha,\beta)} : \{0,1\}^S \to \{0,1\}$$

defined by restricting the inputs as follows:

1. *Fix the* n*-bit inputs corresponding to* $\overline{S}$ *to* $\alpha \in \{\{0,1\}^n\}^{\overline{S}}$.
2. *Fix the last* $n-1$ *bits of each* n*-bit input corresponding to* S *to the values described by* $\beta \in \{\{0,1\}^{n-1}\}^S$.

Thus a first-bit restriction of f to the set S is a boolean function on $|S|$ bits.[4]

▶ **Definition 7** (Modified Nečiporuk's Measure). *Let $f : \{\{0,1\}^n\}^k \to \{0,1\}$ be a function. For any subset $S \subseteq [k]$, define*

$$g_S^*(f) := \max_{\beta \in \{\{0,1\}^{n-1}\}^S} \log_2(|\{f_{S|(\alpha,\beta)} : \alpha \in \{\{0,1\}^n\}^{\overline{S}}, f_{S|(\alpha,\beta)} \not\equiv 0\}|).$$

For any positive integer $m \leq k$, let $V = (V_1, V_2, ..., V_m)$ denote an m-partition of $[k]$. The measure is $G^*(f) := \max_V \sum_{V_i \in V} g_{V_i}^*(f)$.

We can easily bound the largest possible Modified Nečiporuk Measure for any boolean function.

▶ **Proposition 8.** *For any boolean function $f : \{\{0,1\}^n\}^k \to \{0,1\}$,*

$$G^*(f) \leq \frac{k^2 n}{\log_2(kn)}.$$

Proof. For any $S \subseteq [k]$, $g_S^*(f) \leq (k - |S|)n$ as α can take no more than $2^{(k-|S|)n}$ different values. Moreover, $g_S^*(f) \leq 2^{|S|}$, the log of the number of distinct $|S|$-bit boolean functions. Balancing these two restrictions gives the upper bound. ◀

In Section 4, we show examples of functions with large modified Nečiporuk measure. Then, in Section 5, we show that modified Nečiporuk measure is a lower bound on PSM complexity. Together, these results imply Theorem 1.

4 Functions with Large Modified Nečiporuk Measure

For $k = \omega(\log(n))$, many functions have modified Nečiporuk measure that is asymptotically optimal. For instance:

▶ **Proposition 9** (Modified Nečiporuk Measure of Random Functions). *As long as $k = \omega(\log(n))$, a random function $f : \{\{0,1\}^n\}^k \to \{0,1\}$ has*

$$G^*(f) \geq \frac{k^2 n}{\log_2(kn)} - kn - 1$$

with probability at least $1 - \exp(-N^{\log_2(kn)})$.

We prove Proposition 9 in Appendix A. In addition to most randomly sampled functions, we describe two explicit functions with optimal modified Nečiporuk measure. The first is a straightforward generalization of the distinct elements function.

[4] One can consider a generalization of first bit restrictions by allowing α to specify a sequence of pairs from some partitioning of the input domains $\{0,1\}^n$ into pairs, and allowing "free variables" in the restricted function to correspond to evaluating f on either the first or second string in the pair. Suitably modifying G^* with such restrictions should yield identical bounds on PSM complexity. However, the present definition is more intelligible and sufficient for our purposes.

▶ **Definition 10** (Set Disjointness). *The* (k, n) set disjointness function *is defined for* k *such that* $k = \omega(\log n)$ *as follows:*

1. $DISJ_{k,n}$ *takes as input* kn *bits, parsed as bit strings of length* n*. We fix* m *such that* $2m\log_2(mn) = k$,[5] *and divide the* k *input strings into* m *"blocks", each containing* $2\log_2(mn)$ n*-bit strings.*
2. *We interpret each block* $i \in [m]$ *as a multiset* $S_i \subset [(mn)^2]$ *as follows. Let* $x_i^1, \cdots, x_i^{2\log(mn)}$ *denote the* n*-bit strings in block* i*. We consider* $x_i^1, \ldots, x_i^{2\log(mn)}$ *as the columns of an* $n \times 2\log(mn)$*-matrix, and read the* $2\log(mn)$ *elements of* S_i *off the rows. (That is, we concatenate the first bits of each* n*-bit string to get the first element of* S_i*, the second bits to get the second element, etc.)*
3. $DISJ_{k,n}$ *outputs 1 if all sets are pairwise disjoint (*$S_i \cap S_j = \emptyset$ *for all* $i \neq j$*).*

▶ **Proposition 11.** $G^*(\mathrm{DISJ}_{k,n}) = \Omega(k^2 n/\log(kn))$.

Proof. For $i \in [m]$, let V_i denote the ith block of bit strings, which corresponds to the subset $S_i \subset [(mn)^2]$. We will show $g^*_{V_i}(\mathrm{DISJ}) = \Omega(mn\log(mn)) = \Omega(kn)$. As $k = 2m\log_2(mn)$ implies $m > \frac{k}{2\log_2(kn)}$, this implies the result by symmetry on the m other choices for V_i.

Consider building first-bit restrictions as follows. Set $\beta \in \{\{0,1\}^{n-1}\}^{V_i}$ so that all but the first element of S_i take the value $(mn)^2$. (By the definition of first-bit restrictions, the first element of S_i is unrestricted.)

Then consider values for α, which specifies each S_j, $j \neq i$, that select *disjoint* subsets of $[(mn)^2 - 1]$. Each such α yields a restriction with respect to β: namely, the function that evaluates to 1 whenever the free element of S_i is not a member of $\bigcup_{S_j, j\neq i} S_j$. Thus, every choice of α that specifies a distinct set $\bigcup_{S_j, j\neq i} S_j$ composed of disjoint members corresponds to a distinct restriction.

Finally, we can bound the choices of α via

$$\#\alpha = \binom{(mn)^2 - 1}{n(m-1)} \geq \left(\frac{(mn)^2 - 1}{n(m-1)}\right)^{n(m-1)} \geq (mn)^{n(m-1)}.$$

Thus, $g^*_{V_i}(\mathrm{DISJ}) \geq n(m-1)\log_2(mn) = \Omega(kn)$. ◀

We additionally show a variation on the Indirect Storage Access function has maximal modified Nečiporuk measure.

▶ **Definition 12** (First-bit Indirect Storage Access). *The* (k, n) *first-bit indirect storage access (FISA) function is defined for* k *such that* $k = \omega(\log n)$ *as follows:*

1. $FISA_{k,n}$ *takes as input* $\log_2(k) + kn$ *bits. The first* $\log_2(k)$ *bits are parsed as an integer* $y \in [k]$ *and the remainder as the matrix* $A \in \{0,1\}^{k\times n}$*.*
2. $FISA_{k,n}$ *first identifies the* n*-bit string* A^y *(i.e., the* y^{th} *row of* A*), and then identifies an index in* $[kn]$ *corresponding to the bit string* $x = A_0^y A_0^{y+1} ... A_0^{y+\log(kn)-1}$*. (If we overflow the boundaries of* A*, we continue from* A^0*.) It then returns the* x*th bit of* A *(i.e.,* $A^{\lfloor x/n \rfloor}_{x \pmod n}$*).*

▶ **Proposition 13.** $G^*(FISA_{k,n}) = \Omega(k^2 n/\log(kn))$.

[5] Strictly speaking, this limits us to pairs (k, n) such that some integer m satisfies our condition. In order to define the function for arbitrary k, we might choose the largest m such that $2m\log_2(mn) \leq k$, set $k' := 2m\log_2(mn)$, and compute the (k', n) set disjointness function on the first k' input strings.

Proof. For simplicity, we treat $FISA_{k,n}$ as a kn-bit boolean function when calculating $G^*(FISA_{k,n})$, as the additional $\log_2(k)$ bits of input are absorbed by Ω. Consider a partition $V = (y, V_1, V_2, ..., V_{k/\log_2(kn)})$, in which each V_i contains $\log_2(kn)$ contiguous n-bit strings in A.

Fix $i \in [k/\log_2(kn)]$ and consider $g^*_{V_i}(FISA_{k,n})$. In particular, consider the first-bit restrictions of $FISA_{k,n}$ to V_i in which y is set so that the bit string x is read from the first bit of each n-bit string in V_i. Fixing the remaining bits of A to every possible string results in $2^{kn-n\log(kn)} - 1$ distinct non-zero restrictions. Summing over each set in the partition of A gives $G^*(f_{k,n}) = \Omega(k^2 n/\log(kn))$. ◀

5 A New Lower Bound for Many-Party PSM

We conclude by showing that the modified Nečiporuk measure provides a lower bound on PSM size. To prove this fact, we use an "information squeezing" argument enabled by the following observation that follows from the perfect security requirement in Definition 3 and is further elaborated in the proof of Lemma 15 below.

▶ **Observation 14.** *Let $f : \{\{0,1\}^n\}^k \to \{0,1\}$ be a kn-bit boolean function and $S \subseteq [k]$ denote a subset of parties. If there exist two first-bit restrictions $f_{S|(\alpha,\beta)}$ and $f_{S|(\alpha',\beta)}$ to S such that for some $x \in \{0,1\}^S$, $f_{S|(\alpha,\beta)}(x) = f_{S|(\alpha',\beta)}(x)$, the distribution over messages sent by $\overline{S}$ must be identical on α and α' in any PSM with perfect security.*

In other words, if two first-bit restrictions agree on at least one input, the perfect security requirement implies that the distribution over messages sent by $\overline{S}$ must be identical. Thus, in order to ensure correctness, the messages sent by a subset of parties S must contain enough information to distinguish between the distinct first-bit restrictions of f to S created by setting the inputs $\overline{S}$ to different values. The precise amount of information required is captured by the modified Nečiporuk measure.

▶ **Lemma 15** (Nečiporuk Lower Bounds PSM Size)**.** *Let $\mathcal{X}$ be a PSM for any function $f : \{\{0,1\}^n\}^k \to \{0,1\}$. Then*

$$|\mathcal{X}| \geq G^*(f)/2.$$

Combining Lemma 15 with Proposition 9 yields Theorem 1 as an immediate consequence. In Appendix B, we argue that stronger lower bounds are beyond a certain kind of natural proof barrier.

5.1 Proof of Lemma 15: Nečiporuk Lower Bounds PSM Size

Proof. Fix $f : \{\{0,1\}^n\}^k \to \{0,1\}$, and let V be a partition of $[k]$ that maximizes our modified Nečiporuk measure for f; that is,

$$V \in \arg\max_U \sum_{U_i \in U} g^*_{U_i}(f)$$

Consider an arbitrary set $S \in V$. Select $\beta \in \{\{0,1\}^{n-1}\}^S$ to fix all but the first bits of S and maximize $|\{f_{S|(\alpha,\beta)} : \alpha \in \{\{0,1\}^n\}^{\overline{S}}, f_{S|(\alpha,\beta)} \not\equiv 0\}|$. Let $\mathcal{D}$ denote the uniform distribution over a minimal set T such that

$$\{f_{S|(\alpha,\beta)} | \alpha \in T\} = \{f_{S|(\alpha,\beta)} : \alpha \in \{\{0,1\}^n\}^{\overline{S}}, f_{S|(\alpha,\beta)} \not\equiv 0\}.$$

We observe that $H(\mathcal{D}) = g^*_S(f)$, where H denotes the entropy function. Given a PSM $\mathcal{X}$ for f, we proceed to define two subfamilies of random variables that will capture the information in $\mathcal{D}$.

$$\mathcal{A} := (\mathcal{X}^i_{\{0\}\times\beta_i}, \mathcal{X}^i_{\{1\}\times\beta_i})_{i\in S}$$

$$\mathcal{B} := (\mathcal{X}^i_{d_i})_{i\in\overline{S}, d\sim\mathcal{D}}.$$

Consider drawing $(a, b) \sim \mathcal{A} \times \mathcal{B}$, conditioned on a particular $d \in \mathtt{Supp}(\mathcal{D})$. Given any $y \in \{0,1\}^{|S|}$, the referee can use the appropriate members of a and b to compute $f_{S|(d,\beta)}(y)$. Computing $f_{S|(d,\beta)}(y)$ for each $y \in \{0,1\}^{|S|}$ uniquely identifies d because each member of T corresponds to a unique restriction; thus $H(\mathcal{D}|\mathcal{A}, \mathcal{B}) = 0$.

Furthermore, for any distinct $d, d' \in \mathtt{Supp}(\mathcal{D})$, there exist y and y' such that $f_{S|(d,\beta)}(y) = f_{S|(d',\beta)}(y') = 1$. (This follows by definition, as every $d \in \mathtt{Supp}(\mathcal{D})$ corresponds to a restriction that is not the zero function.) Thus, by the security property of PSM, we know that $(\mathcal{X}^i_{d_i})_{i\in\overline{S}} \equiv (\mathcal{X}^i_{d'_i})_{i\in\overline{S}}$. In other words, $\mathcal{B}$ contains no information about $\mathcal{D}$: $H(\mathcal{D}|\mathcal{B}) = H(\mathcal{D})$.

Using fundamental properties of the entropy function, we have that

$$H(\mathcal{A}) \geq H(\mathcal{A}|\mathcal{B}) = H(\mathcal{A}, \mathcal{D}|\mathcal{B}) - H(\mathcal{D}|\mathcal{A}, \mathcal{B}). \tag{1}$$

As $H(\mathcal{D}|\mathcal{A}, \mathcal{B}) = 0$ and $H(\mathcal{A}, \mathcal{D}|\mathcal{B}) \geq H(\mathcal{D}|\mathcal{B}) = H(\mathcal{D})$, we have

$$H(\mathcal{A}) \geq H(\mathcal{D}) = g^*_S(f). \tag{2}$$

Partition $\mathcal{A}$ into $\mathcal{A}_0 := (\mathcal{X}^i_{\beta_i\times\{0\}})_{i\in S}$ and $\mathcal{A}_1 := (\mathcal{X}^i_{\beta_i\times\{1\}})_{i\in S}$. As $H(\mathcal{A}) = H(\mathcal{A}_0, \mathcal{A}_1) \leq H(\mathcal{A}_0) + H(\mathcal{A}_1)$, $H(\mathcal{A}_b) \geq H(\mathcal{A})/2$ for some $b \in \{0,1\}$. Thus

$$2 \cdot \log_2(|\mathtt{Supp}(\mathcal{A}_b)|) \geq g^*_S(f).$$

Repeating this argument for each $S \in V$ demonstrates the existence of $x \in \{\{0,1\}^n\}^k$ for which $\sum_{i=1}^k \log_2(|\mathtt{Supp}(\mathcal{X}^i_{x_i})|) \geq G^*(f)/2$. ◀

References

1. Benny Applebaum, Thomas Holenstein, Manoj Mishra, and Ofer Shayevitz. The communication complexity of private simultaneous messages, revisited. *Journal of Cryptology*, pages 1–37, 2019.
2. Leonard Assouline and Tianren Liu. Multi-party PSM, revisited. Technical report, Cryptology ePrint Archive, Report 2019/657, 2019. URL: `https://eprint.iacr.org/2019/657`.
3. Marshall Ball, Justin Holmgren, Yuval Ishai, Tianren Liu, and Tal Malkin. On the complexity of decomposable randomized encodings, or: How friendly can a garbling-friendly PRF be? In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
4. Paul Beame, Nathan Grosshans, Pierre McKenzie, and Luc Segoufin. Nondeterminism and an abstract formulation of Nečiporuk's lower bound method. *ACM Transactions on Computation Theory (TOCT)*, 9(1):1–34, 2016.
5. Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In *Theory of Cryptography Conference*, pages 317–342. Springer, 2014.
6. Amos Beimel, Eyal Kushilevitz, and Pnina Nissim. The complexity of multiparty PSM protocols and related models. 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 II*, volume 10821 of *Lecture Notes in Computer Science*, pages 287–318. Springer, 2018. `doi:10.1007/978-3-319-78375-8_10`.

7 I Nečiporuk Eduard. On a boolean function. In *Soviet Math. Dokl*, volume 7, pages 999–1000, 1966.

8 Uri Feige, Joe Killian, and Moni Naor. A minimal model for secure computation. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 554–563, 1994.

9 Orr Fischer, Rotem Oshman, and Uri Zwick. Public vs. private randomness in simultaneous multi-party communication complexity. In *International Colloquium on Structural Information and Communication Complexity*, pages 60–74. Springer, 2016.

10 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.

11 Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, pages 174–183. IEEE, 1997.

12 Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 294–304. IEEE, 2000.

13 Edward I Nechiporuk. A boolean function. *Engl. transl. in Sov. Phys. Dokl.*, 10:591–593, 1966.

14 Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997. doi:10.1006/jcss.1997.1494.

15 Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE, 1986.

A Proof of Proposition 9: Random Functions have Large Nečiporuk Measure

With $k = \omega(\log(n))$, let $f : [N]^k \to \{0,1\}$ be a random function. Recall that

$$2^{g^*_S(f)} := \max_{\beta \in \{\{0,1\}^{n-1}\}^S} |\{f_{S|(\alpha,\beta)} : \alpha \in \{\{0,1\}^n\}^{\overline{S}}, f_{S|(\alpha,\beta)} \not\equiv 0\}|.$$

For a given S, we define the convenient function

$$z^*_S(f,\beta) := |\{f_{S|(\alpha,\beta)} : \alpha \in \{\{0,1\}^n\}^{\overline{S}}\}|.$$

Thus $2^{g^*_S(f)} \geq \max_\beta z^*_S(f,\beta) - 1$, where the -1 arises because $z^*_S(f,\beta)$ may also count the zero function.

Fix b, the parameter that will specify the fixed bits of the inputs in S, arbitrarily. For every function $\phi : \{0,1\}^{|S|} \to \{0,1\}$, define the indicator random variable $\mathcal{Y}_{\phi,b}$ as follows:

$$\mathcal{Y}_{\phi,b} = \begin{cases} 1, & \text{if } \exists a \in \{\{0,1\}^n\}^{\overline{S}} : f_{S|(a,b)} = \phi \\ 0, & \text{otherwise} \end{cases}$$

Thus we have

$$z^*_S(f,b) = \sum_\phi \mathcal{Y}_{\phi,b}.$$

For any fixed ϕ, we can lower bound $\mathbb{E}[\mathcal{Y}_{\phi,b}]$ as follows using the fact that $(1+x)^n \leq \frac{1}{1-nx}$ for $x \in [-1,0]$, $n \in \mathbb{N}$.

$$\mathbb{E}[\mathcal{Y}_{\phi,b}] = 1 - (1 - 2^{-2^{|S|}})^{2^{n|\overline{S}|}} \geq 1 - \frac{1}{1 + 2^{n|\overline{S}| - 2^{|S|}}} = \frac{2^{n|\overline{S}|}}{2^{2^{|S|}} + 2^{n|\overline{S}|}}$$

Rewriting using linearity of expectation, we get

$$\mathbb{E}[z^*_S(f,b)] = \mathbb{E}[\sum_{\phi} \mathcal{Y}_{\phi,b}] = \sum_{\phi} \mathbb{E}[\mathcal{Y}_{\phi,b}] \geq 2^{2^{|S|}} \cdot \frac{2^{n|\overline{S}|}}{2^{2^{|S|}} + 2^{n|\overline{S}|}}.$$

Furthermore, we note that we can upper bound $\mathbb{E}[z^*_S(f,b)]$ by $2^{n|\overline{S}|}$, the number of ways to fix values for $\overline{S}$.

We can consider $z^*_S(f,b)$ as a doob martingale on the independent random variables $f_{S|(a,b)}$ for $a \in \{\{0,1\}^n\}^{\overline{S}}$. As $z^*_S(f,b)$ counts the number of distinct restrictions, changing $f_{S|(a,b)}$ for a single value of a changes $z^*_S(f,b)$ by at most 1. As a result, we can apply McDiarmid's inequality to get

$$\Pr_f[\mathbb{E}[z^*_S(f,b)] - z^*_S(f,b) \geq t] \leq \exp(\frac{-2t^2}{\mathbb{E}[z^*_S(f,b)]}).$$

Substituting our lower and upper bounds for $\mathbb{E}[z^*_S(f,b)]$ on the left and right, and using the fact that $2^{g^*_S(f)} \geq z^*_S(f,b) - 1$ for any choice of b by definition, we have

$$\Pr_f[2^{g^*_S(f)} \leq \frac{2^{2^{|S|}+n|\overline{S}|}}{2^{2^{|S|}} + 2^{n|\overline{S}|}} - t - 1] \leq \exp(\frac{-2t^2}{2^{n|\overline{S}|}}).$$

Finally, setting $|S| = \log_2(kn)$ and $t = 2^{kn/2}$ yields

$$\Pr_f[2^{g^*_S(f)} \leq \frac{2^{kn}}{2^{n\log_2(kn)} + 1} - 2^{kn/2} - 1] \leq \exp(-2N^{\log_2(kn)}).$$

Taking a union bound over $k/\log_2(kn)$ choices of S, it follows that $G^*(f) > \frac{k^2 n}{\log_2(kn)} - kn - 1$ for all but an exponentially small fraction of functions.

B On the Possibility of $\omega(k^2 n)$ PSM Lower Bounds

We conclude by observing that the existence of strong pseudo-random functions[6] (PRFs) with efficient PSM schemes would rule out the possibility of proving $\omega(k^2 n)$ lower bounds using a natural class of arguments. Specifically, if there exists an exponentially strong PRF that admits a PSM of size $O(k^2 n)$, an argument due to Razborov and Rudich rules out the possibility that any natural proof with sublinear-time constructivity can prove an $\omega(k^2 n)$ lower bound. We conjecture that a candidate PRF presented by Ball et al. [3] meets these requirements.

Ball et al. conjecture that taking the quadratic residue of the sum of an input with the secret key in prime fields is exponentially secure, if the input domain is slightly restricted.

▶ **Conjecture 16** ([3]). *Let $p(m)$ be a sequence of primes such that $p > 2^{2m+1}$, for all $m \in \mathbb{N}$. Let $f : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ denote the function,*

$$f : (k,x) \to (k+x+1)^{\frac{p-1}{2}} \qquad \textit{(where } k,x \textit{ are interpreted as integers.)}$$

Then, for some $s(m) = 2^{\Omega(m)}$, any size-$s(m)$ circuit can distinguish oracle access to either f_k where k is sampled uniformly from $\{0,1\}^m$ or a truly random function with advantage at most $2^{-s(m)}$.

[6] For definitions and further discussion on pseudo-random functions we refer the reader to [10].

▶ **Proposition 17.** *The DRE for f (with $m = kn/2$ and $p(m) \in (2^{2m+1}, 2^{2m+2}]$ as defined in Conjecture 16) outlined in [3] can be extended to give a PSM with size $O(k^2 n)$.*

Proof. The ith component of the DRE for f is

$$\hat{f}_i(x_i; s, r_1, \ldots, r_k) = s^2 \cdot (x_i \cdot 2^{i-1}) + r_i,$$

where s is sampled uniformly from $\mathbb{Z}_p^*$ and $r_1, \ldots, r_k$ are sampled uniformly from $\mathbb{Z}_p$ conditioned on the fact that $\sum r_i = 0$.

Note that $\sum \hat{f}_i(x_i; s, r) = s^2 \cdot x$. Consequently, if $f(x) = 1$ the sum is uniformly distributed over residues and if $f(x) = -1$ the sum is distributed over non-residues. Moreover, any sum of an incomplete subset of DRE components is uniform over $\mathbb{Z}_p$. Thus, in the PSM setting, k parties can consolidate the $\hat{f}_i$'s corresponding to their input into an $O(kn)$-bit message to create a PSM with size $O(k^2 n)$. ◀

We recall Razborov and Rudich's concept of natural proofs of explicit circuit lower bounds [14]. They observed that all known explicit circuit lower bounds proceeded by defining a natural combinatorial property, showing a certain complexity class cannot compute functions with such a property, and then exhibiting an explicit function that does have the property. Their ingenious contribution was to *formalize* the notion of a natural property P as a subset of all boolean functions with following properties:

1. (Largeness) A random function is in P with high probability.
2. (Constructivity) Given its truth table, it is possible to decide if a function is in P in polynomial time.
3. (Useful) P does not contain functions from the class one wishes to prove a lower bound against. For us, this means P does not contain functions with DREs of size $ck^2 n$ for any constant c (and large enough k, n).

The definition can be naturally extended to capture *sublinear-time*[7] natural properties. We note all examples of natural properties that we are aware of can be made to admit sublinear-time constructivity, including that of Applebaum et al. [1].

▶ **Definition 18.** *$\Pi = (\Pi_Y, \Pi_N)$ is a* natural property with sublinear-time constructivity, useful against a class C *if Π_Y, Π_N are disjoint subsets of boolean functions such that*

1. *(Largeness) A random n-bit function is in Π_Y with probability 2/3.*
2. *(Sublinear-Time Constructivity) The promise problem Π admits a randomized oracle machine, $A^{(\cdot)}$ that runs in time $o(2^n)$ such that*
   - *$f \in \Pi_Y \implies \Pr[A^f = 1] > 2/3$.*
   - *$f \in \Pi_N \implies \Pr[A^f = 0] > 2/3$.*
3. *(C-useful) $C \subseteq \Pi_N$.*

We now conjecture that the PRF candidate of Ball et al. is resilient to any *uniform* attack that runs in slightly less than exponential time.

▶ **Conjecture 19.** *Let $p(m)$ be a sequence of primes such that $p > 2^{2m+1}$, for all $m \in \mathbb{N}$. Let $f : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ denote the function,*

$$f : (k, x) \to (k + x + 1)^{\frac{p-1}{2}} \qquad \textit{(where } k, x \textit{ are interpreted as integers).}$$

[7] In this case "sublinear" refers to the size of the truth table of the function.

Then, any randomized oracle-algorithm A that runs in time $o(2^m)$ on input parameter 1^m cannot distinguish between f_k where $k \xleftarrow{u} \{0,1\}^m$ and a truly random function with non-negligible probability.

Following Razborov and Rudich's argument in this new setting yields the following proposition.

▶ **Proposition 20.** *If Conjecture 19 is true, then sublinear-time-natural proofs of size $\omega(k^2 n)$ PSM lower bounds do not exist.*

Proof. Suppose for contradiction the existence of a natural property Π with sublinear-time constructivity, useful against some class C containing functions that admit PSMs of size $O(k^2 n)$. As most random functions are contained in Π_Y, and our candidate PRF is contained in Π_N, we can distinguish our PRF candidate from random in time $o(2^n)$ using constructivity. This violates Conjecture 19. ◀

Multiparty Computation with Covert Security and Public Verifiability

Peter Scholl
Aarhus University, Denmark

Mark Simkin
Ethereum Foundation, Aarhus, Denmark

Luisa Siniscalchi
Aarhus University, Denmark
Concordium Blockchain Research Center, Aarhus, Denmark

Abstract

Multiparty computation protocols (MPC) are said to be *secure against covert adversaries* if the honest parties are guaranteed to detect any misbehavior by the malicious parties with a constant probability. Protocols that, upon detecting a cheating attempt, additionally allow the honest parties to compute certificates, which enable third parties to be convinced of the malicious behavior of the accused parties, are called *publicly verifiable*. In this work, we make several contributions to the domain of MPC with security against covert adversaries.

We identify a subtle flaw in a protocol of Goyal, Mohassel, and Smith (Eurocrypt 2008), meaning that their protocol does not allow to identify a cheating party, and show how to fix their original construction to obtain security against covert adversaries.

We present generic compilers that transform arbitrary passively secure preprocessing protocols, i.e. protocols where the parties have no private inputs, into protocols that are secure against covert adversaries and publicly verifiable. Using our compiler, we construct the first efficient variants of the BMR and the SPDZ protocols that are secure and publicly verifiable against a covert adversary that corrupts all but one party, and also construct variants with covert security and identifiable abort.

We observe that an existing impossibility result by Ishai, Ostrovsky, and Seyalioglu (TCC 2012) can be used to show that there exist certain functionalities that cannot be realized by parties, that have oracle-access to broadcast and arbitrary two-party functionalities, with information-theoretic security against a covert adversary.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases Multi-party computation, covert security, public verifiability

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.8

Related Version *Full Version*: `https://eprint.iacr.org/2021/366.pdf`

Funding *Peter Scholl*: Supported by the Independent Research Fund Denmark (DFF) under project number 0165-00107B (C3PO) and Aarhus University Research Foundation.
Luisa Siniscalchi: Supported by the Concordium Blockchain Research Center.

1 Introduction

Secure multiparty computation (MPC) protocols allow collections of parties, where each party holds some private input, to compute arbitrary functions of those inputs in a way that reveals the result of the computation, but nothing else beyond that. Ideally, we would like our MPC protocols to be as fast and as secure as possible, but in reality we often have to choose one over the other. Protocols that are secure against passive adversaries who follow

the protocol specification honestly, but try to learn more about the other parties' inputs from what they see, are typically quite fast, whereas protocols that are secure against actively misbehaving adversaries tend to be significantly slower.

To overcome the dilemma of needing to choose between good efficiency and good security, Aumann and Lindell [2] introduced an intermediate security notion which they call *security against covert adversaries*[1]. In a covertly secure protocol, corrupt parties may try to cheat and learn information about the honest parties' inputs, however, it is guaranteed that any such misbehavior will be detected with some constant probability ϵ, known as the *deterrence factor*. If cheating is detected then the honest parties will agree on at least one misbehaving party. The authors motivate covert security by arguing that, in certain scenarios, the repercussions of being caught misbehaving outweigh the gains that come from successfully cheating, so an adversary should be incentivized to behave honestly.

Subsequently, Asharov and Orlandi [1] proposed a strengthening of covert security, which requires the honest parties to not only detect misbehavior with a constant probability, but also prove it to third parties in a *publicly verifiable* manner. That is, the honest parties, upon detecting misbehavior during a protocol execution, should be able to compute a certificate that provably shows that at least one of the corrupted parties attempted to cheat.

Goyal, Mohassel and Smith [17] showed how to construct efficient two- and multiparty computation protocols with security against covert adversaries based on Yao's Garbled Circuits and its multiparty equivalent, the BMR protocol [6]. Damgård et al. [11] present a protocol in the preprocessing model, i.e. where the overall execution is split into an input-independent preprocessing and a input-dependent online phase, with a weaker notion of security against covert adversaries, where the misbehaving party is not necessarily identified, based on the SPDZ framework [14]. Damgård, Geisler, and Nielsen [10] present a compiler that transforms certain passively secure protocols based on secret sharing into protocols with security against covert adversaries. Their compiler only applies to protocols that assume an honest majority among the parties. None of the works above provide public verifiability.

The first two-party protocol with public verifiability was presented in the work of Asharov and Orlandi [1]. More efficient publicly verifiable two-party protocols with the same security guarantees have subsequently been proposed by Kolesnikov and Malozemoff [22] and Hong et al. [19]. In a recent work by Damgård, Orlandi, Simkin [13], the authors propose a generic compiler that transforms arbitrary two-party protocols with passive security into protocols with security against covert adversaries and public verifiability. The authors also sketch how to extend their compiler to the multiparty setting in the presence of an adversary that corrupts a constant fraction of the parties. Their multiparty protocols, however, have a deterrence factor that is inversely proportional to the fraction of corrupted parties and the resulting protocols are unlikely to be faster, in terms of concrete efficiency, than existing multiparty computation protocols with active security.

Given this state of the art, it is natural to ask whether we can construct MPC protocols, which provide security and public verifiability against an adversary that can corrupt all but one party, and are concretely more efficient than the fastest actively secure protocols.

[1] In the remainder of this paper, we will use "security against covert adversaries" and "covertly secure protocols" interchangeably.

1.1 Our Contribution

In this work, we make several contributions to the domain of MPC with security against covert adversaries with and without public verifiability.

1.1.1 On the Relation Between Covert and Active Adversaries

Firstly, we observe that in the multi-party setting (in contrast to two parties) there is a subtle but important difference between the standard definitions of covert security and active security used in the literature. The standard definition of covert security [2] explicitly requires that the honest parties agree upon the identity of a party who is caught cheating, a property we call *identifiable cheating*. Moreover, they require *identifiable abort*, meaning that a corrupt party who aborts the computation (without trying to learn additional information) is also identified. On the other hand, in the setting of dishonest majority, where more than half of the parties may be corrupted, actively secure protocols typically settle for security with abort, without identifiability (which is more expensive to realize). Hence, a covert secure protocol with identifiable abort is not strictly weaker than a standard actively secure protocol, but rather the two notions are incomparable. A more appropriate point of comparison is with an actively secure protocol with identifiable abort, which typically has a much higher cost [21]. Note that one can also consider a weaker notion of covert security with abort, where aborting parties are not identified, as has been done, for instance, in a covertly secure variant of the SPDZ protocol [11].

1.1.2 MPC with Security Against Covert Adversaries and Identifiable Abort

We identify a subtle flaw in the work of Goyal, Mohassel and Smith [17], which renders their multiparty protocol potentially insecure. More concretely, we show that while their solution correctly detects misbehavior with a constant probability, it does not necessarily allow the honest parties to unanimously agree on one of the misbehaving parties, so it does not satisfy the basic requirement of identifiable cheating. To fix their construction, we present a generic compiler for upgrading any passively secure preprocessing protocol, i.e. where the parties have no private inputs, into one with covert security and identifiable abort. This suffices to obtain a modified version of their construction with the desired security guarantees, namely, both identifiable cheating and identifiable abort. Furthermore, our compiler can be used to obtain a covertly secure variant of the SPDZ protocol with identifiable abort, improving upon the security of the covert protocol from [11], which does not have identifiable abort.

1.1.3 Preprocessing with Security and Public Verifiability Against Covert Adversaries

Our second compiler transforms any passively secure preprocessing protocol into a protocol with publicly verifiable covert (PVC) security (with the same corruption threshold). In the PVC setting, here we settle for security-with-abort and not identifiable abort. This is because publicly verifying that a party aborted may require proving to a third party that some corrupt party did not send a message; achieving this seems to require publishing the entire protocol transcript, which is a large extra cost.

Our compiler for PVC security leverages time-lock puzzles [23] in a novel fashion, to force a corrupt party to "commit" to opening some protocol executions before it learns whether or not a cheating attempt was successful. Importantly, the parties generate the puzzles locally, rather than inside MPC, and furthermore, the puzzles only have to be solved in case of a dispute, which allows us to construct concretely efficient protocols.

1.1.4 Applications

Our compilers for preprocessing apply to a large, general class of preprocessing functionalities for producing correlated randomness. To obtain a complete MPC protocol, we can apply one of our covert compilers to generate the preprocessing for an *actively secure* online phase. Even though we are then running an actively secure protocol, these are typically cheap and information-theoretic, with a much smaller overhead to achieve active security compared with the preprocessing phase. If the preprocessing has publicly verifiable covert security, then so does the combined protocol with the online phase, since any cheating in the online phase simply leads to abort. Similarly, if the preprocessing has covert security with identifiable abort, then so does the combined protocol as long as the online phase is secure with identifiable abort.

To illustrate this, we consider a few examples, namely the BMR [6], SPDZ [14] and TinyTable [12] families of protocols. These all perform MPC with up to $n-1$ out of n corruptions and security with abort, for the settings of binary circuits, arithmetic circuits, and circuits augmented with "truth-table" gates, respectively. By applying our compiler, we can obtain the first preprocessing protocols for these, which achieve publicly verifiable covert security and improved performance. For example, to obtain a deterrence factor of $\epsilon = 2/3$, i.e. any misbehavior will be detected with probability $2/3$, our compiled covert protocol will, roughly, be only three times slower than its passively secure counterpart. Our compiler particularly shines for the TinyTable protocol [12], where we can improve performance by an order of magnitude or more when relaxing to covert security. This is because we are able to replace its costly, actively secure preprocessing with a much cheaper passive protocol. Also, as a contribution of independent interest, we present an optimized preprocessing phase for the passively secure BMR protocol, which reduces bandwidth in the preprocessing phase by around 20%, compared with previous methods [7, 9].

As mentioned earlier, a particularly interesting use-case is the setting of identifiable abort. Here, existing actively secure protocols based on SPDZ [4] and BMR [5] have a lot more overhead compared with the non-identifiable case, in particular because they require a secure broadcast channel. Our compiler is much simpler, and only needs broadcast in case some dishonest behaviour occurs, so we expect them to be much more efficient in typical usage.

1.1.5 Information-Theoretic Impossibility

Finally, we also show that there exist certain functionalities that cannot be realized with information-theoretic security against a covert adversary by parties that have oracle-access to a broadcast and arbitrary two-party functionalities. Our proof strategy for proving this impossibility is essentially identical to a previous proof by Ishai, Ostrovsky, and Seyalioglu [20], which shows that the same result holds if one aims for active security with identifiable abort. Similary to the actively secure identifiable abort compiler of [21], this motivates our approach of our covert security compilers, which make black-box access to the next-message function of a passively secure protocol, instead of general two-party functionalities like oblivious transfer. We do not claim any particular technical novelty here, but we provide the full proof in the full version for the sake of completeness.

1.1.6 Concurrent and Subsequent Work

Concurrently to our work,[2] Faust et al. [16] presented a compiler for publicly verifiable covert security. Their work does not consider identifiable abort, but is similar to our second compiler for public verifiability using time-lock puzzles. We mention here a few differences. Firstly, our protocol is more lightweight in its use of generic actively secure MPC, since we do not need to generate a time-lock puzzle inside MPC. This means we only invoke an actively secure protocol on a circuit with $O(nk\lambda)$ AND gates, for n parties, covert repetition factor k and security parameter λ, compared with $O(nk\lambda + \log^2 N)$ gates, where N is an RSA modulus, for the optimized variant of [16]. Secondly, in our protocol, less data is sent over a broadcast channel, since our compiler only broadcasts hashes of the underlying semi-honest protocol instead of the entire transcript. On the other hand, while our compiler requires the parties and judge to solve n time-lock puzzles, [16] only requires solving a single time-lock puzzle, and also uses the notion of a verifiable TLP, which leads to a more efficient judge, with a fast way of verifying puzzle solutions. Also, the security notions are slightly different, since [16] realizes a relaxed form of covert security, where the adversary may choose to cheat after learning its output, while we use the standard definition, but only consider relaxed preprocessing functionalities where corrupt parties may choose their own outputs (for preprocessing functionalities, this notion is implied by that of [16]). Finally, we give concretely efficient instantiations of our protocols, while [16] mainly analyzes asymptotic efficiency.

Subsequently to our work, the recent work of [15] introduces a new notion of financially backed covert security (FBC) which extends the notion of publicly verifiable covert security, where the corrupted party gets financially punished (rather than only being publicly detected). They generically transform different types of PVC protocols into FBC protocols, and their work can also be applied to our PVC protocols.

1.2 Technical Overview

1.2.1 Covert Security via Cut-and-Choose

Most existing protocols [2, 17, 10, 1, 11, 22, 19, 13] (with the exception of [13]) for secure computation with covert security follow the same general blueprint. They all start by considering some passively secure protocol that is run k times in parallel, where $k-1$ randomly chosen executions will eventually be opened and used for checking the behaviour of the involved parties and the last remaining execution will be used for computing the desired output. From a technical point of view, the main challenge is to find the right moment for revealing which executions are checked and opening them. If the executions are opened too early, then cheating may be possible in subsequent phases of these protocols. If they are opened too late, then there is a risk of revealing information about the private inputs of the honest parties.

A well-suited class of protocols, that implement some function f, to consider in this setting, are those that can be split into two phases: (1) a passively secure, but input-independent, preprocessing phase which realizes a correlated randomness functionality; (2) an actively secure online phase which takes as input the correlated randomness from the preprocessing phase in order to implement f. For technical reasons, we focus on passively secure preprocessing protocols which realize a mildly relaxed form of functionality called a

[2] Both papers were submitted to Eurocrypt 2021.

corruptible correlated randomness functionality [8]. Here, the functionality is parameterized by some distribution $\mathcal{D}$, and gives the adversary the power to choose the outputs from $\mathcal{D}$ that are given to the malicious parties; the functionality then *reverse samples* the output for the honest parties based on the malicious parties' output and the distribution $\mathcal{D}$. This class of functionalities encompasses those in popular MPC protocols like the BMR protocol [6] or the SPDZ framework [14]. Given such a protocol, we can run the preprocessing phase k times in parallel and check $k-1$ executions at the very end of that phase. If the check passes, the protocol proceeds to the actively secure online phase, where misbehavior is no longer a concern. Looking ahead, we would like to stress that our final protocols all satisfy the standard notion of covert security, despite using a relaxed preprocessing functionality.

Given this high-level blueprint, the remaining task is to design an appropriate check protocol that enables the honest parties to agree on a misbehaving party. To see whether some party P_i has sent the correct message in round r during the protocol execution, one needs to know P_i's private random tape and all messages P_i has received so far.

Based on these observations, a first attempt towards designing a check protocol could be as follows: Initially all parties commit to k random tapes each, i.e. each P_i for $i \in [n]$ commits to random tapes $r_{(i,1)}, \ldots, r_{(i,k)}$. The parties run the preprocessing protocol k times, where P_i uses random tape $r_{(i,j)}$ in execution j. Once all k executions terminate, the parties jointly flip a coin $c \in [k]$ and open all commitments via broadcast belonging to executions $j \in [k]$ with $j \neq c$. If any party aborts at this stage, we accuse it of cheating. If none of the parties abort, then everybody will know the vectors $(r_{(1,j)}, \ldots, r_{(n,j)})$ of random tapes used in executions $j \neq c$. Each party P_i can use the vectors of random tapes to generate a full honest transcript for each execution and use it to check whether it received messages that were consistent with the honest transcript during the protocol execution. Unfortunately, this approach has several problems.

The first one is that two honest parties, who may receive malformed messages from different adversarially corrupt parties during the protocol executions, have no obvious way of agreeing on which party to accuse unanimously. Even worse, a malicious party could falsely accuse some honest party to further make everyone's life more difficult.

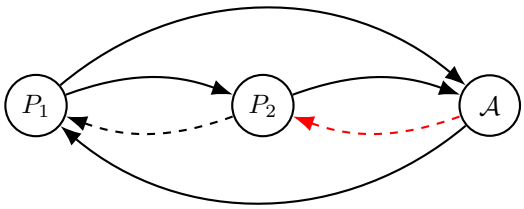

Figure 1 Illustration of inconsistent cheater identification.

The second, problem is that P_i receiving a message from P_j that is inconsistent with the honest transcript *does not* mean that P_j misbehaved. Consider the example in Figure 1, where all parties behave honestly (indicated by solid black lines) except for some corrupt party $\mathcal{A}$, which sends a malformed messages to P_2 (red dashed line).

Given that P_2's messages to P_1 may be a function of the messages it receives from $\mathcal{A}$, we potentially end up in a situation, where P_2 subsequently sends an incorrect, but honestly generated, message to P_1 (dashed black line). The takeaway from this discussion is that P_1 cannot simply compare the messages it received from other parties with the messages in an honest transcript to deduce who misbehaved in the protocol execution. This is exactly

what goes wrong in the compiler of Goyal et al. [17], which tries to recover from cheating by opening the randomness for the underlying semi-honest protocol π. When π is instantiated using the GMW protocol based on pairwise OT channels, as suggested in [17], although cheating may be detected, it is not possible to identify the cheating party.

If the semi-honest protocol in their approach was adjusted to send every message over a broadcast channel (using public-key encryption), then their approach would be sound. This, however, would introduce an overhead of $\mathcal{O}(n)$ broadcasts; even when all parties behave honestly. Looking ahead, our protocol will only make use of broadcasts when malicious parties actively misbehave. Even in the presence of an adversary who tries to trigger as many broadcasts as possible, our protocol is still more efficient than the approach of Goyal et al., since for the specific case of BMR, we use various optimizations which means that only one instead of k garbled circuits needs to be broadcast. Additionally, it is worth noting that in the case of public verifiability covert security, we only need broadcast with abort, which can be instantiated much more efficiently.

1.2.2 Achieving Identifiable Abort

To get around the issue described above, we define a new property for MPC protocols that we call *identifiable cheating from random tapes (IDC)*. We say that a protocol has the given property, if there exist two protocols Certify and Identify associated with the given protocol. Certify takes the random tapes of all parties in an execution and the local view of some party P_i as input and outputs a partial certificate cert_i. The algorithm Identify takes n partial certificates as input and either outputs the index of a malicious party that misbehaved in the protocol execution or outputs $\perp$ to indicate that nobody misbehaved. Importantly, we require Identify to function correctly, even if the corrupted parties output false partial certificates or do not send anything. We show how to transform any passively secure protocol into one that supports IDC. The formal details and results regarding this property can be found in the full version. We remark that the notion of $\mathcal{P}$-verifiability from [21] achieves a similar goal, but this transformation (and the variant from [5]) uses broadcast so is less efficient.

Now, we can follow the blueprint for constructing covertly secure protocols outlined before, but instead of each party comparing its view to the messages in an honest execution, we use Certify and Identify to check whether any party misbehaved and if so which index to output. The details of this construction can be found in Section 3.

1.2.3 Public Verifiability

To obtain not only covert security, but also public verifiability, we have to overcome several additional challenges. The first problem is that the IDC property sketched above is not sufficient for producing *publicly* verifiable certificates. More concretely, the IDC property does not provide any guarantees about the output of Identify, when the adversary is additionally allowed to replace some of the honest parties' partial certificates with some maliciously chosen values. This could potentially enable the adversary to produce a collection of false partial certificates, which accuse an honest party of misbehaving. Hence, we need a stronger flavor of this building block, which we call *publicly verifiable identifiable cheating from random tapes (PVIDC)*, where Identify correctly identifies a malicious party or outputs $\perp$, even if the adversary is allowed to tamper with the honest parties' partial certificates. Here again, we show how to transform arbitrary passively secure protocols into ones that support PVIDC. Our transformation for obtaining PVIDC is slightly less efficient than the transformation for just IDC, but still significantly more efficient than running a fully actively secure protocol. The formal definition of this property can be found in the full version.

The second problem is, that upon revealing which executions should be checked, the adversary can simply stop responding and thereby prevent the honest parties from checking the executions and obtaining a certificate of the adversary's misbehavior. In contrast to covert security without public verifiability, here we cannot simply accuse the aborting party of cheating, because the honest parties have no corresponding publicly verifiable evidence. At first glance, our goals at this step may even seem contradictory. On one hand, during the checking phase, we would like to ensure that the adversary cannot tell whether the information it is about to reveal is useful for incriminating it. On the other hand, we would like to ensure that any other party can use the revealed information for determining whether cheating has happened or not and who the cheating party was.

To get out of this predicament, we make use of a tool called time-lock puzzles [23]. Such puzzles allow a sender to publish a message that cannot be read before a certain time has passed, e.g. in our case before at least some number of rounds in a synchronized network have passed. Crucially, the message becomes visible eventually, *without any interaction from the sender*. Time-lock puzzles can be built from RSA-based timing assumptions, without any trusted setup and generating a puzzle requires just a single exponentiation. Recently, time-lock puzzles have also been used to build 2-PC with output-independent abort [3], with a construction using similar ideas to ours (except that we are in the multi-party setting, and also achieve public verifiablity).

On an intuitive level, we use time-lock puzzles as follows:[3] At the beginning of the checking phase, all parties jointly execute an actively secure MPC protocol, where each party P_i inputs all k commitment openings that belong to the random tapes the party used. The MPC protocol picks $k-1$ executions at random and outputs time-lock puzzles of all random tapes belonging to those. Additionally, the parties obtain a secret sharing of the index c of the execution that is not being checked. All parties sign the time-lock puzzles, the commitments and broadcast the computed signatures. Because the puzzles cannot be solved fast enough, the adversary needs to decide whether to abort this phase of the execution without seeing the contents of the puzzles and thus without knowing which executions are being checked. Once the honest parties have signatures of the corrupted parties on the time-lock puzzles, they are, roughly speaking, guaranteed to have some useful information that can be shown to an external party in case cheating will be detected. Once all parties signed the time-lock puzzles, they all publish their share of c and then publish the openings of the random tapes used in the executions $j \neq c$. Now if the adversary decides to abort, because it doesn't like which executions are being checked, then the honest parties can obtain its necessary random tapes from the signed time-lock puzzles.

In our final protocol, the time-lock puzzles are generated locally by the parties, outside of MPC, and incur an overhead that is independent of the size of the circuit to be evaluated. Considering the evaluation of larger circuits, these additional costs from using time-lock puzzles become minor and in executions, where all parties behave honestly, no time-lock puzzles need to be solved by any party. The details of this protocol can be found in full version.

[3] We are omitting several important details here that can be found in the technical parts, e.g. in the full version of the paper.

1.2.4 Instantiating the Compilers

Our compilers can be instantiated with any MPC protocol in the preprocessing model, as long as its preprocessing functionality implements the *corruptible* correlated randomness functionality explained above. For most MPC protocols based on secret-sharing, this requirement is already satisfied out-of-the-box. This includes protocols such as SPDZ [14, 11], TinyTable [12] and a version of SPDZ with identifiable abort [4]. We therefore easily obtain covertly secure variants of these protocols (with public verifiability or identifiable abort) by plugging in a semi-honest version of the preprocessing, which improves efficiency by avoiding e.g. expensive zero-knowledge proofs typically used in SPDZ.

The case of constant-round MPC, based on garbled circuits, is slightly more challenging. Here, if we want public verifiability, we can again directly apply our compiler to a semi-honest version of the BMR protocol similar to [7, 18], since we observe this works with a corruptible preprocessing functionality (we in fact give an optimized semi-honest preprocessing protocol, which reduces the number of OTs by 25%).

For identifiable abort, however, we need to modify the BMR functionality so that (1) we get a secure online phase with identifiable abort, and (2) the BMR functionality should be a corruptible preprocessing functionality. We observe that (1) is straightforward to achieve, by having each party send a commitment to its share of the garbled circuit in the preprocessing protocol; this ensures that any party who sends an incorrect share in the online phase can by identified, and is also cheap to implement, since our compiler only needs this to be done with passive security.

However, this is not compatible with the definition of a corruptible preprocessing functionality, indeed we would have to reverse-sample an honest party's message and decommitment information, *after* the corresponding commitment is provided by the adversary. This strong form of equivocation is not possible with standard commitments. Instead, we use *unanimously identifiable commitments* [20], which can be built information-theoretically in such a way that allows this. Setting up these commitments involves a little more work in the preprocessing, but this overhead is independent of the circuit size, since the parties only commit to a hash of their garbled circuit shares.

2 Preliminaries

Notation

Let λ be the computational and δ be the statistical security parameter. We write $[n]$ to denote the set $\{1, \ldots, n\}$. For all algorithms that follow, we will regularly omit the security parameter input and it is understood that this input is provided implicitly. We define the view of a party in the execution of the protocol Π as the messages she received during an execution of Π along with her input and random tapes. For a functionality $\mathcal{F}$, we write $[\mathcal{F}]^{\mathsf{ida}}$ to denote the corresponding ideal functionality with identifiable abort.

Secure Broadcast

We assume a broadcast channel, as well as a public-key infrastructure (PKI) which is implied by broadcast. In our protocols with public verifiability, it is enough to have broadcast with abort, which can be implemented with a cheap, 2-round echo protocol. For our protocols with identifiable abort, however the broadcast must be identifiable, which involves a more expensive protocol with $O(n)$ rounds and digital signatures.

Secure Multiparty Computation

All of our security definitions follow the ideal/real simulation paradigm in the standalone model. Throughout this paper we will consider protocols that are executed over a synchronous network with static, rushing adversaries and we assume the existence of secure authenticated point-to-point channels between the parties. The MPC definitions can be found in the full version, in particular the standard notion of covert security and covert security with publicly verifiability are defined; we remark that in the notion of covert security, we explicitly require *identifiable abort*, meaning that the honest parties agree upon the identity of the party who caused an abort. On the other hand, covert security with public verifiability is *not* identifiable; the adversary just sends the abort command without an index.

The other useful definitions, for instance time-lock puzzle, can be found in the full version.

2.1 Corruptible Correlated Randomness Functionality

For our work we will consider a mild relaxation of correlated randomness functionality $\mathcal{F}_{\mathsf{corr}}^{\mathcal{D}}$ called a *corruptible correlated randomness* functionality [8]. $\mathcal{F}_{\mathsf{corr}}^{\mathcal{D}}$ allows the parties in the corrupted set C to choose their correlated randomness $\{R'_i\}_{i\in C}$ and then $\mathcal{F}_{\mathsf{corr}}^{\mathcal{D}}$ has to reverse sample based on $\{R'_i\}_{i\in C}$ the correlated randomness for the honest parties consistently with the distribution $\mathcal{D}$. We model this equipping the functionality $\mathcal{F}_{\mathsf{corr}}^{\mathcal{D}}$ with an efficient reverse sample algorithm RS. We note that $\mathcal{F}_{\mathsf{corr}}^{\mathcal{D}}$ is nonetheless sufficient for all major overall protocols in the preprocessing model.

Functionality $\mathcal{F}_{\mathsf{corr}}^{\mathcal{D}}$

The functionality interacts with parties $P_1, \ldots, P_n$. Let $C \subset [n]$ be the set of parties corrupted by the ideal world adversary S.
Upon receiving message $(\texttt{CorrRand}, \{R'_i\}_{i\in C})$ from S, the functionality samples $\{R_i\}_{i\in[n]\setminus C} \leftarrow \mathsf{RS}(\{R'_i\}_{i\in C}, \mathcal{D})$ and sends R_i to each P_i with $i \in [n] \setminus C$.

Figure 2 Functionality for corruptible correlated randomness.

▶ Remark 1. We note that our ideal functionality implicitly requires the adversary S to provide adversarial correlated randomness that can be part of a valid output of the functionality. This is not a restriction, since we will later on prove that any real-world adversarial attack can be translated into such a restricted ideal-world adversary.

3 Preprocessing with Identifiable Abort

In this section we are presenting a protocol $[\Pi_{\mathsf{corr}}]^{\mathsf{cov}}$ which implements $\mathcal{F}_{\mathsf{corr}}^{\mathcal{D}}$ with security against covert adversaries, who corrupts $n-1$ parties, and deterrence factor $\epsilon = 1 - \frac{1}{k}$.

$[\Pi_{\mathsf{corr}}]^{\mathsf{cov}}$ makes use of a preprocessing protocol Π_{corr} that has identifiable cheating from random tapes (IDC), and consistent identifiable abort. Roughly speaking, $[\Pi_{\mathsf{corr}}]^{\mathsf{cov}}$ proceeds as follow. Initially each party commits to k random tapes that are a result of a coin-flip, i.e. each P_i for $i \in [n]$ commits to random tapes $r_{i,1}, \ldots, r_{i,k}$. The parties run the preprocessing protocol Π_{corr} k times, where P_i uses random tape $r_{i,\ell}$ in execution ℓ. Once all k executions terminate, the parties jointly flip a coin $c \in [k]$ and open all commitments via broadcast belonging to executions $\ell \in [k]$ with $\ell \neq c$. If any party aborts at this stage, we accuse it of cheating. If none of the parties abort, then everybody will know the vectors $(r_{1,\ell}, \ldots, r_{n,\ell})$

of random tapes used in executions $\ell \neq c$. Once each party P_i received vectors of random tapes she runs algorithms $\mathsf{Certify}$ and Ver of Π_{corr} in order to identify if a malicious party misbehaved. If no cheating is detected P_i outputs the output of the c-th execution of Π_{corr}.

The formal description of the protocol can be found in Figure 3. In the formal description to not overburden the notation we avoid to specify that when a party, say P_i, does not broadcast a message (or the parties receive an (abort, i) from the functionality) the parties terminates the computation sending $(\mathsf{corrupted}, i)$.

Protocol $[\Pi_{\mathsf{corr}}]^{\mathsf{cov}}$covprot

Let Π_{corr} be a protocol that computes $\mathcal{F}_{\mathsf{corr}}^{\mathcal{D}}$ with passive security and has identifiable cheating from random tapes and consistent identifiability abort.

1. Each party P_i receives a signing key sk_i and verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ from the PKI.
2. For $i, j \in [n]$, each P_j sends $(\mathsf{comFlip}, (i, 1)), \ldots, (\mathsf{comFlip}, (i, k))$ to $[\mathcal{F}_{\mathsf{FLIP}}]^{\mathsf{ida}}$.
3. For $i, j \in [n]$, each P_j sends $(\mathsf{openOne}, (i, 1), i), \ldots, (\mathsf{openOne}, (i, k), i)$ to $[\mathcal{F}_{\mathsf{FLIP}}]^{\mathsf{ida}}$.
4. For $i, j \in [n]$, each P_i receives $r_{i,1}, \ldots, r_{i,k}$ from the ideal functionality.
5. All parties jointly execute Π_{corr} in parallel k times, where party P_i uses random tape $r_{i,\ell}$ in the ℓ-th execution of the protocol. Let $R_{i,\ell}$ be the output of party P_i in execution ℓ.
6. All parties send $(\mathsf{coinFlip}, 0)$ to $[\mathcal{F}_{\mathsf{FLIP}}]^{\mathsf{ida}}$ and obtain a uniformly random value $\ell^* \in [k]$.
7. For $\ell \in [k] \setminus \{\ell^*\}$ each party P_i
   a. sends $(\mathsf{openAll}, (i, \ell))$ to $[\mathcal{F}_{\mathsf{FLIP}}]^{\mathsf{ida}}$.
   b. receives back $\vec{r}_\ell := (r_{1,\ell}, \ldots, r_{n,\ell})$ corresponding to the random tapes used in execution ℓ from $[\mathcal{F}_{\mathsf{FLIP}}]^{\mathsf{ida}}$.
   c. computes $\mathsf{cert}_{i,\ell} \leftarrow \mathsf{Certify}(\mathsf{vk}_1, \ldots, \mathsf{vk}_n, \vec{r}_\ell, \mathsf{view}_{i,\ell})$, where $\mathsf{view}_{i,\ell}$ is P_i's view in the ℓ-th execution of Π_{corr}.
   d. send $\mathsf{cert}_{(i,\ell)}$ to $[\mathcal{F}_{\mathsf{BC}}]^{\mathsf{ida}}$.
8. Each party P_i receives certificates $(\mathsf{cert}_{1,\ell}, \ldots, \mathsf{cert}_{n,\ell})$ for each execution $\ell \neq \ell^*$ and computes $v_\ell \leftarrow \mathsf{Ver}(\mathsf{vk}_1, \ldots, \mathsf{vk}_n, \vec{r}_\ell, \mathsf{cert}_{1,\ell}, \ldots, \mathsf{cert}_{n,\ell})$. Let J be the set of indices with $v_\ell \neq \perp$. If $J \neq \emptyset$, then P_i broadcast $(\mathsf{corrupted}, v_j)$ with the smallest j from J.
9. Each party P_i outputs R_{i,ℓ^*}.

Figure 3 Preprocessing protocol for MPC with covert security and identifiable abort.

▶ **Theorem 2.** *Suppose protocol Π_{corr} securely implements $\mathcal{F}_{\mathsf{corr}}^{\mathcal{D}}$ with passive security, has identifiable cheating from random tapes, and consistent identifiability abort. Let $[\mathcal{F}_{\mathsf{FLIP}}]^{\mathsf{ida}}$ be the ideal committed coin flip functionality with identifiable abort. Let $[\mathcal{F}_{\mathsf{BC}}]^{\mathsf{ida}}$ be the broadcast functionality. Then $[\Pi_{\mathsf{corr}}]^{\mathsf{cov}}$ implements $\mathcal{F}_{\mathsf{corr}}^{\mathcal{D}}$ with security against covert adversaries, who corrupts $n-1$ parties, and deterrence factor $\epsilon = 1 - \frac{1}{k}$.*

The ideal functionality used in the theorem statement, the proof of Theorem 2 and the definition of the property of cheating identifiability from random tapes can be also find in the full version of the paper.

References

1 Gilad Asharov and Claudio Orlandi. Calling out cheaters: Covert security with public verifiability. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 681–698. Springer, Heidelberg, December 2012. doi:10.1007/978-3-642-34961-4_41.

2 Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 137–156. Springer, Heidelberg, February 2007. doi:10.1007/978-3-540-70936-7_8.

3 Carsten Baum, Bernardo David, Rafael Dowsley, Jesper Buus Nielsen, and Sabine Oechsner. TARDIS: A foundation of time-lock puzzles in UC. In Anne Canteaut and Francois-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part III*, Lecture Notes in Computer Science, pages 429–459. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77883-5_15.

4 Carsten Baum, Emmanuela Orsini, and Peter Scholl. Efficient secure multiparty computation with identifiable abort. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 461–490. Springer, Heidelberg, October / November 2016. doi:10.1007/978-3-662-53641-4_18.

5 Carsten Baum, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. Efficient constant-round MPC with identifiable abort and public verifiability. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 562–592. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56880-1_20.

6 Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd Annual ACM Symposium on Theory of Computing*, pages 503–513. ACM Press, May 1990. doi:10.1145/100216.100287.

7 Aner Ben-Efraim, Yehuda Lindell, and Eran Omri. Optimizing semi-honest secure multiparty computation for the internet. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 578–590. ACM Press, October 2016. doi:10.1145/2976749.2978347.

8 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 489–518. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26954-8_16.

9 Lennart Braun, Daniel Demmler, Thomas Schneider, and Oleksandr Tkachenko. Motion – a framework for mixed-protocol multi-party computation. Cryptology ePrint Archive, Report 2020/1137, 2020. URL: https://eprint.iacr.org/2020/1137.

10 Ivan Damgård, Martin Geisler, and Jesper Buus Nielsen. From passive to covert security at low cost. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 128–145. Springer, Heidelberg, February 2010. doi:10.1007/978-3-642-11799-2_9.

11 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. Springer, Heidelberg, September 2013. doi:10.1007/978-3-642-40203-6_1.

12 Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The TinyTable protocol for 2-party secure computation, or: Gate-scrambling revisited. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 167–187. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63688-7_6.

13 Ivan Damgård, Claudio Orlandi, and Mark Simkin. Black-box transformations from passive to covert security with public verifiability. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 647–676. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56880-1_23.

14 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. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_38.

15 Sebastian Faust, Carmit Hazay, David Kretzler, and Benjamin Schlosser. Financially backed covert security. Cryptology ePrint Archive, Report 2021/1652, 2021. URL: https://ia.cr/2021/1652.

16 Sebastian Faust, Carmit Hazay, David Kretzler, and Benjamin Schlosser. Generic compiler for publicly verifiable covert multi-party computation. In Anne Canteaut and Francois-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, Lecture Notes in Computer Science, pages 782–811. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77886-6_27.

17 Vipul Goyal, Payman Mohassel, and Adam Smith. Efficient two party and multi party computation against covert adversaries. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 289–306. Springer, Heidelberg, April 2008. doi:10.1007/978-3-540-78967-3_17.

18 Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 598–628. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70694-8_21.

19 Cheng Hong, Jonathan Katz, Vladimir Kolesnikov, Wen-jie Lu, and Xiao Wang. Covert security with public verifiability: Faster, leaner, and simpler. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 97–121. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17659-4_4.

20 Yuval Ishai, Rafail Ostrovsky, and Hakan Seyalioglu. Identifying cheaters without an honest majority. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 21–38. Springer, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9_2.

21 Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 369–386. Springer, Heidelberg, August 2014. doi:10.1007/978-3-662-44381-1_21.

22 Vladimir Kolesnikov and Alex J. Malozemoff. Public verifiability in the covert model (almost) for free. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 210–235. Springer, Heidelberg, November / December 2015. doi:10.1007/978-3-662-48800-3_9.

23 Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto, 1996.

On Seedless PRNGs and Premature Next

Sandro Coretti
IOHK, Zürich, Switzerland

Yevgeniy Dodis
New York University, NY, USA

Harish Karthikeyan
New York University, NY, USA

Noah Stephens-Davidowitz
Cornell University, Ithaca, NY, USA

Stefano Tessaro
University of Washington, Seattle, WA, USA

Abstract

Pseudorandom number generators with input (PRNGs) are cryptographic algorithms that generate pseudorandom bits from accumulated entropic inputs (e.g., keystrokes, interrupt timings, etc.). This paper studies in particular PRNGs that are secure against *premature next* attacks (Kelsey et al., FSE '98), a class of attacks leveraging the fact that a PRNG may produce an output (which could be seen by an adversary!) *before* enough entropy has been accumulated. Practical designs adopt either unsound entropy-estimation methods to prevent such attacks (as in Linux's `/dev/random`) or sophisticated pool-based approaches as in Yarrow (MacOS/FreeBSD) and Fortuna (Windows).

The only prior theoretical study of premature next attacks (Dodis et al., Algorithmica '17) considers either a *seeded* setting or assumes constant entropy rate, and thus falls short of providing and validating practical designs. Assuming the availability of random seed is particularly problematic, first because this requires us to somehow generate a random seed without using our PRNG, but also because we must ensure that the entropy inputs to the PRNG remain independent of the seed. Indeed, all practical designs are seedless. However, prior works on seedless PRNGs (Coretti et al., CRYPTO '19; Dodis et al., ITC '21, CRYPTO'21) do not consider premature next attacks.

The main goal of this paper is to investigate the feasibility of theoretically sound seedless PRNGs that are secure against premature next attacks. To this end, we make the following contributions:

1. We prove that it is impossible to achieve seedless PRNGs that are secure against premature-next attacks, even in a rather weak model. Namely, the impossibility holds even when the entropic inputs to the PRNG are independent. In particular, our impossibility result holds in settings where seedless PRNGs are otherwise possible.
2. Given the above impossibility result, we investigate whether existing seedless pool-based approaches meant to overcome premature next attacks in practical designs provide meaningful guarantees in certain settings. Specifically, we show the following.
   - We introduce a natural condition on the entropic input and prove that it implies security of the round-robin entropy accumulation PRNG used by Windows 10, called Fortuna. Intuitively, our condition requires the input entropy "not to vary too wildly" within a given round-robin round.
   - We prove that the "root pool" approach (also used in Windows 10) is secure for general entropy inputs, provided that the system's state is not compromised *after* system startup.

2012 ACM Subject Classification Security and privacy → Mathematical foundations of cryptography; Security and privacy → Information-theoretic techniques; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases seedless PRNGs, pseudorandom number generators, PRNG, Fortuna, premature next

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.9

3rd Conference on Information-Theoretic Cryptography (ITC 2022).
Editor: Dana Dachman-Soled; Article No. 9; pp. 9:1–9:20

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

Funding *Yevgeniy Dodis*: Partially supported by gifts from VMware Labs and Google, and NSF grants 1815546 and 2055578.
Stefano Tessaro: Supported in part by NSF grants CNS-1930117 (CAREER), CNS-1926324, CNS-2026774, a Sloan Research Fellowship, and a JP Morgan Faculty Award.

1 Introduction

Pseudo-random number generators (PRNGs) are one of the most critical building blocks of secure systems. In particular, no meaningful cryptography is achievable without (pseudo) randomness. In practice, PRNGs' main functionality is to *accumulate* entropy (modeled by the function refresh in the syntax) into one or more pools from several sources (such as keystrokes, interrupt timings, etc.), and to then *extract* "clean" pseudorandom bits from these pools (modeled by the function next). In other words, refresh calls is used to accumulate entropy into the state of the PRNG while next is used to produce outputs from this PRNG state. While doing this, PRNGs must resist powerful attacks. On the one hand, the available entropy sources (i.e., the input to the PRNG) may be partially controlled by the adversary interacting with the system. On the other hand, the state of the PRNG may be compromised, and we want to protect both prior uses of the PRNG (i.e., we want forward security), as well as allow for recovery from such compromise. PRNGs, in particular, differ from "traditional" pseudorandom generators, which instead already assume a fully entropic input.

Several practical PRNG designs have been proposed, including those in operating systems such as `/dev/random` [20] for Linux, Yarrow [15] for MacOS/iOS/FreeBSD, and Fortuna [10] for Windows, and in standards like NIST's SP 800-90A [2]. Designing secure PRNGs remains however a complex task, and several flaws have been identified in existing designs (cf. e.g. [19, 21]).

Seedless PRNGs. One could hope that the best way forward is to develop provably secure PRNGs, following a line of work initiated by Barak and Halevi [1]. Yet, theoretical validation presents several technical challenges. In particular, we want such PRNGs to be *general*, in that they achieve security under the minimal assumption that the available sources have sufficient entropy. To address this, much of the prior work has considered a *seeded* setting, first proposed by Dodis et al. [7]. Here, the PRNG can rely on a random *seed* which is *independent* from the accumulated entropy (but known to the attacker), an approach inherited from the necessity of seed for extraction from general entropy sources [17]. Such a seeded approach was taken by several subsequent works [7, 8, 19, 11, 14, 12]. This seed serves as an input to both refresh and next.

However, the seeded setting is not necessarily practical. Indeed, since our end goal is that of generating randomness in the first place, one may question where a uniformly random and independent seed would come from! Moreover, and perhaps even more importantly, it is unreasonable to expect our input sources to be truly independent of the seed. (E.g., our future keystrokes can certainly depend on the prior output of our PRNG, which depends on the seed.) Unsurprisingly, all practical designs are seedless.

This issue has motivated recent work studying different ways in which the impossibility of deterministic extraction can be circumvented without the need for seed. Coretti et al. [4] consider constructions based on cryptographic hash functions modeled as random oracles and introduced corresponding meaningful notions of entropy in this setting. The formal definition is presented as Definition 1. Subsequent works by Dodis et al. [6, 5] consider the simple case in which the inputs are *independent* and without assuming ideal primitives.

This paper: Seedless PRNGs & Premature-next attacks. All prior theoretical work on *seedless* PRNGs relied heavily on the assumption that the PRNG is allowed sufficient time to accumulate entropy before having to provide any output, i.e., they do not handle so-called *premature-next* attacks [16]. In such an attack, the adversary requests output from the PRNG before it has accumulated enough entropy to guarantee security. Much prior work (including all prior work in the seedless setting) simply assumes that all accumulated entropy is lost upon such a premature-next call. With such a definition, a PRNG might fail to produce a single pseudorandom bit, regardless of how much entropy is provided!

Linux's `/dev/random` [20] attempts to overcome premature-next attacks by blocking the RNG as long as insufficient entropy has been accumulated, but this approach cannot be theoretically sound, as estimating entropy is impossible [18, 10]. Indeed, a concrete way to fool `/dev/random`'s entropy estimation was given in [7]. In contrast, Yarrow [15] and Fortuna [10] propose a clever solution to the problem. Abstractly, these constructions have a *register* as well as many *pools*. Only the register is used to provide output. Each time these PRNGs receive some input, they "add it to one pool" selected in a round-robin fashion. Then, at different rates, each pool is used to occasionally update the register. We call this "emptying the pool." Intuitively, while a premature-next call might completely leak the input to any pools that have been emptied recently, it will not reveal any information about inputs to pools that have not been emptied since they received this input.

A generalization of this pool-based approach was analyzed formally by Dodis et al. [8]. However, their analysis either assumes seeds (thus departing from the deterministic approach taken by Yarrow/Fortuna) *or* requires that the entropy rate is constant – i.e., that all inputs have the same (unknown, adversarially chosen) entropy. Both situations are undesirable, and in this paper, we aim to make progress on the following general question:

Can we have seedless PRNG designs which provably resist (in some meaningful way) premature-next attacks?

1.1 Our Results

Impossibility of seedless PRNGs. We first address the feasibility question of whether seedless PRNGs can, in principle, be secure against premature-next attacks. We would like in particular to assess whether recent positive results on seedless PRNGs, [4, 6, 5] can be extended to resist premature-next attacks.

Notice that, if the attacker can choose to vary the entropy of the inputs, then no "deterministic pool-based approach" can work. (As in [8], we formalize this below using the notion of a scheduler.) In particular, if we require γ bits of entropy to go into a single pool in order to recover from compromise and the attacker knows when pools will be filled and emptied, then the attacker can simply provide a bit less than γ bits to each pool before it is emptied. (This intuition is formalized in [8].) However, one can imagine much more complicated constructions. E.g., we might choose which pool to fill or empty based on the (entropic) input (perhaps even with some attempt at entropy estimation like `/dev/random`), or we might not use a pool-based approach at all.

Surprisingly, we show that *no* seedless PRNG can resist premature next attacks, even if the inputs are sampled independently. In particular, as deterministic extraction without the seed *is* possible for independent inputs [3], our impossibility is inherently due to the premature next problem. In more detail, following [8], we parameterize security by two values, γ^*, and β. The goal is to guarantee that *if* the PRNG has obtained γ^* bits of min-entropy within T^* steps after the last state compromise, then the PRNG will revert to producing pseudorandom bits within βT^* steps after the same state compromise. We prove that for any

choice of γ^*, β, there exists an efficient adversary providing $q \geq {\gamma^*}^2\beta^2$ PRNG inputs (each with one independent bit of entropy) which violate the PRNG security against premature next attacks. Since q is typically huge, this rules out any reasonable settings of γ^* and β.

In addition to being interesting in its own right, this shows a natural setting where meaningful PRNG security (e.g., entropy accumulation and extraction) is possible *without* premature-next attacks, but impossible *with* them.

Toward Positive Results. The above strong impossibility result, including the "separation" between randomness accumulation/extraction and premature-next security, motivates us to search for positive results even (optimistically) assuming *perfect entropy accumulation and extraction*. In fact, we already have two widely used solutions that appear to work in practice. First, we already mentioned the round-robin pool-based approach, called Fortuna, which is part of Windows 10 and macOS. Second, Windows 10 [9] uses a special "root pool" to solve the problem of initial entropy accumulation when the computer starts up. This single pool is emptied at exponentially increasing intervals (e.g., at time $1, \beta, \beta^2, ...$) to (heuristically) solve the problem that sometimes the computer might boot with no good source of randomness for an unknown period of time. Intuitively, if good entropy starts to come in at (unknown) time t, the root pool will allow the PRNG to produce good random bits by time at most βt. While this simple approach does not work when one is worried about state compromise at an unknown time (and this is why more than 1 pool is used for general purpose PRNGs like Fortuna), it appears quite effective for accumulating entropy at startup.

Given the existence of these two heuristics to accumulate entropy within pools, we ask whether we can find natural conditions where these approaches *provably* work, despite our strong impossibility result above. To make this question formal, we define a clean model of seedless (pool-based) *schedulers*, extending the corresponding notion of schedulers [8] to the seedless setting. Intuitively, if we have k pools, given each entropic sample X_i, the scheduler decides which pool $\mathsf{in}_i \in [k]$ will accumulate this entropy, and, which pool $\mathsf{out}_i \in [k]$ (if any) will contribute its accumulated entropy back to the register. Moreover, to model ideal entropy accumulation and extraction, we assume that the entropy that was thrown to pool i simply adds up without loss.

In fact, at this level of abstraction, we can completely forget about entropy and PRNGs and simply consider an abstract notion of a scheduler, whose goal is to distribute a sequence of weights $w_1, \ldots, w_q \in [0, 1]$ into pools, sometimes emptying one of the pools with the following guarantee. If there are t consecutive weights $w_{t_0+1}, \ldots, w_{t_0+t}$ whose sum is larger than some threshold α, then there should be a pool that accumulates at least weight 1 in this same time period (without being emptied) and is emptied shortly thereafter, say before time step $t_0 + \beta t$. We call this (α, β)-security. Here, a pool accumulating weight 1 in this abstract scheduler game corresponds to a pool accumulating sufficient entropy in a pool-based PRNG. [8] proved formally that a secure scheduler can be used to convert PRNGs that are secure in a model that does not allow for premature-next attacks (used for the individual entropy pools) into a PRNG that is secure in a model with premature-next attacks. In particular, given an (α, β)-secure scheduler together with a PRNG that recovers from compromise after receiving γ bits of entropy *without allowing for premature-next attacks*, we can construct a PRNG that recovers from compromise even in the presence of premature next in time βt, where t is the time needed to receive $\alpha\gamma$ bits of entropy.

Because of our general impossibility result above, we cannot achieve general (α, β)-security. (We also give direct proof of this fact in the setting of schedulers below.) We then show two positive results yielding proven security guarantees for the two schedulers used in the real world, by giving meaningful restrictions to the model.

- First, we show that the root-pool approach achieves nearly optimal (α, β)-security to accumulate entropy at start-up, where $\alpha \approx \log_\beta q$ (and we can take any integer $\beta \geq 2$). Plugging in known constructions yields a PRNG in the root-pool model (i.e., in which we assume that compromise only happens at time 0) that is exponentially better than our general-scheduler lower bound stating $\alpha\beta \geq \sqrt{q}$.
- Second, we show that the round-robin Fortuna construction with $k \geq \log_\beta q$ pools achieves (α, β)-security with $\alpha \approx \log_\beta q$, provided one uses a more conservative notion of entropy called k-smooth entropy.[1] For constant-rate entropy sources, this notion of entropy is identical to the traditional min-entropy, and our result indeed generalizes the earlier observation of [8] regarding constant-rate sources. More generally, our notion of k-smooth entropy essentially captures the idea that wildly fluctuating entropy should be penalized, which we believe is a practically relevant idea (and in particular seems to be behind the heuristics used in practice). In other words, despite simple attacks on the Fortuna scheduler in the unrestricted setting, we found a natural condition where this scheduler works.

We stress that our scheduler results only solve the premature next problem assuming ideal entropy accumulation and extraction, but we hope future work will extend them to full-blown PRNGs, which provably overcome our negative results under similar restrictions.

2 Preliminaries

We write $\mathbb{N} := \{0, 1, 2, \ldots\}$ for the set of natural numbers and for positive integers $k \geq 1$, we write $[k] := \{0, \ldots, k-1\}$ for the natural numbers up to $k-1$. When a value x is sampled uniformly from a distribution X, we will denote it by $x \leftarrow X$. By U_n, we will denote a uniform distribution over bit strings of length n.

We consider PPT adversaries, in some security parameter λ. All our variables in our security definitions will depend on this security parameter.

Min-Entropy. The *prediction probability* of a random variable X is $\mathsf{Pred}(X) := \max_x \mathsf{P}[X = x]$ and the *min-entropy* is $\mathrm{H}_\infty(X) = -\log(\mathsf{Pred}(X))$.

Security Games. All of the security properties considered in this paper are captured by considering a game between a challenger and an attacker $\mathcal{A}$, both of which may have access to an ideal primitive P. The goal of the attacker is to guess a random bit b chosen by the challenger, who offers a set of oracles to the attacker to aid with this task. The *advantage* of $\mathcal{A}$ is defined as

$$2 \cdot \left| \mathsf{P}[\mathcal{A} \text{ wins}] - 1/2 \right| ,$$

where the probability is over the randomness of $\mathcal{A}$, of the challenger, and of the ideal primitive. The cases where $b = 0$ and $b = 1$ are referred to as the *real world* and the *ideal world*, respectively. One may equivalently consider $\mathcal{A}$'s advantage at telling these two worlds apart, i.e.,

$$\left| \mathsf{P}[\mathcal{A} = 1 | b = 0] - \mathsf{P}[\mathcal{A} = 1 | b = 1] \right| .$$

[1] We have a general bound for all k, including a constant number of pools, where $\alpha = O(k)$ and $\beta = O(q^{1/k})$.

3 Impossibility of "Premature Next" Seedless PRNGs

This section considers the security of seedless PRNGs against *premature next attacks* [16]. The idea behind such an attack is that next – the algorithm extracting pseudorandom bits from the PRNG state – is called before the state has accumulated sufficient entropy. The resulting output will therefore not be fully random, and an adversary can potentially use the output of many such calls to recover the state. The notion of robustness against premature-next attacks was formalized by Dodis et al. [8]. Their work generalized and analyzed a key technique to mitigate such attacks that originated in the designs of the Yarrow [15] and Fortuna [10] PRNGs. Roughly, the key idea is that the entropic inputs to the PRNG are carefully distributed to several "smaller" PRNGs, which we refer to as *pools*, and, with different frequencies, these pools are used to randomize a *register* from which random bits are extracted. (We formalize this approach in detail below.) While both Yarrow and Fortuna use deterministic *scheduling* strategies to assign entropic inputs to a pool and to decide when each pool contributes to the register, the provable robustness against premature-next attacks is achieved in [8] by relying on a *random seed* (independent from the inputs) to ensure that the entropy received from the adversary is roughly evenly distributed among the pools.

It is not hard to see that the fixed pool assignment schedule adopted by Yarrow/Fortuna cannot be robust against premature next attacks without extreme restrictions on the adversaries (e.g., the constant rate restriction). However, other seedless strategies are possible (e.g., one could assign entropic inputs to pools chosen depending on the inputs themselves, or some previous inputs; or one might try to divide each input up into smaller pieces in some way; or one might not use pools at all), and the larger question remains on the feasibility of a *seedless* PRNG which is robust, even with premature next calls. One of course should exercise some care, a fully secure deterministic PRNG cannot exist (regardless of premature-next attacks) for the same reasons deterministic extraction is impossible. So, we must make some restrictions on the input distributions provided by the adversary. For this reason, in the following, we will focus on the case of *independent* inputs, for which deterministic extraction is–in principle–possible.

Even in this setting, the main result of this section is an impossibility result. (So, the fact that we restrict our attention to independent inputs simply makes our result stronger.) Specifically, we show that it is impossible to have such a seedless PRNG which is robust against premature next attacks, even in a setting where the entropic inputs are independent.

Before we present our result, which is stated below as Theorem 5, we introduce some more syntax and definitions.

3.1 Pseudorandom Number Generators with Input

In this section, we will briefly recall the syntax of this primitive. We will use the seedless definition for this paper. We refer the readers to the work of Coretti et al. [4] for a detailed exposition.

Syntax. A PRNG is a stateful cryptographic primitive that accumulates entropy by absorbing inputs which it then uses to produce pseudorandom bits when the entropy of its state is high. A PRNG consists of two algorithms as defined below:

▶ **Definition 1** (Syntax of PRNGs). *A* pseudorandom number generator with input (PRNG) *is a pair of algorithms* $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$ *sharing a* μ*-bit state* s*, where*

- refresh *takes a state* s *and an input* $x \in \{0,1\}^m$ *and produces a new state* $s' = \mathsf{refresh}(s, x)$, *and*
- next *takes a state* s *and produces a new state and an output* $y \in \{0,1\}^r$, *i.e,* $(s', y) = \mathsf{next}(s)$.

A PRNG processing m*-bit inputs and producing* r*-bit output is called a* (m, r)-PRNG.

For our impossibility result, we will focus on $(1, 1)$-PRNG. This is without loss of generality, as we could always buffer m such entropic inputs before applying a "bigger" refresh call on m such bits, and impossibility for 1 output bit implies that for $r \geq 1$ output bits.

Security. The work of Coretti et al. [4] dealt with robustness security game, *without* support for Premature Next (ROB). For purposes of this paper, we will focus on robustness security *with* Premature Next (NROB), as defined in Figure 1. While we adapt the original definition from [8] to the seedless setting, we note that we present a highly simplified security game that is enough to provide for our impossibility result. (We also leave out some functionality that is not necessary for the attacker in our impossibility result, which again simply makes our impossibility result much stronger.)

Most significantly, we assume that all of the samples provided by the attacker are *independent* from each other (which makes out impossibility much result stronger). Formally, attacker outputs a distribution X_i for the next entropic sample, and the security game independently samples a concrete value $x_i \leftarrow X_i$ from this distribution, without giving any side information back to the attacker. This allows for much simpler accounting for entropy, – by simply adding individual entropy of samples X_i produced by the attacker, – without worrying about (quite subtle) conditional entropy of such samples.

In more detail, NROB game allows adversary $\mathcal{A}$, whose state is represented by the variable σ, to access the following oracles:

- **get-next** allows the attacker to get pseudorandom outputs by calling the next procedure on the current state and returning the output y.
- **next-ror** creates a challenge, i.e., if $b = 1$, it outputs a uniform random value $y_1 \in \{0,1\}$ instead of the PRNG output y_0. Here, the PRNG output is second part of the output of next procedure.
- **get-state** models state compromises by revealing the value of the state of the adversary.

▶ **Definition 2** (Definition of an Attacker). *An attacker* $\mathcal{A}$ *is called a* (q, τ)*-attacker if it provides at most* q *input distributions for* refresh *and runs in time at most* τ.

For security, the game keeps track of the entropy counter c which counts the entropy the attacker injected into the system since the latest compromise. When c reaches a critical value γ^*, we would like our PRNG to recover. However, instead of demanding immediate recovery (like in the simpler robustness game ROB discussed in Section A), we allow a factor of β gap. Concretely, if entropy γ^* took T^* steps to accumulate, we demand recovery by time $T \leq \beta T^*$.

▶ **Definition 3.** *The advantage of a* (q, τ)*-attacker* $\mathcal{A}$ *in the* $\mathsf{NROB}(\gamma^*, \beta, q)$ *game is denoted by* $\mathrm{Adv}^{\mathrm{NROB}}_{\mathsf{PRNG}}(\mathcal{A})$. *Further, we say that* PRNG *is* $(\gamma^*, \beta, q, \epsilon, \tau)$*-secure if for any* (q, τ)*-attacker* $\mathcal{A}$,

$$\mathrm{Adv}^{\mathrm{NROB}}_{\mathsf{PRNG}}(\mathcal{A}) \leq \epsilon.$$

Game The PRNG Robustness* Game

NROB

$\sigma = \perp$; $s = 0$; $c = 0$
$b \leftarrow \{0,1\}$; corrupt = true
$T = 0; T^* = 0$
for $i = 1, \ldots, q$ **do**
  $(\sigma, X_i) \leftarrow \mathcal{A}^{\mathbf{get-next}, \mathbf{get-state}, \mathbf{next-ror}}(\sigma)$
  $x_i \leftarrow X_i$
  $s = \mathsf{refresh}(s, x_i)$
  $c = c + \mathrm{H}_\infty(X_i)$
  $T = T + 1$
  if $c \geq \gamma^*$ **then**
    if $T^* = 0$ **then**
      $T^* = T$
    if $T \geq \beta T^*$ **then**
      corrupt = false
$b' \leftarrow \mathcal{A}(\sigma)$

get − next

$(s, y) = \mathsf{next}(s)$
return y

next − ror

$(s, y_0) = \mathsf{next}(s)$
$y_1 \leftarrow \{0,1\}^r$
if corrupt = true **then**
  return y_0
return y_b

get − state

$c = 0$; corrupt = true
$T = 0; T^* = 0$
return s

Figure 1 The Robustness Game with Premature Next Calls $\mathsf{NROB}(\gamma^*, \beta, q)$. This is in contrast to the Robustness Game without Premature Next Calls which is presented in Figure 5.

We refer the readers to the works of Dodis et al. [8] and Coretti et al. [4] for discussions on different security models. For comparison, though, we provide the formal definition of the simpler ROB notion in Section A. The critical difference between ROB and NROB is that the former resets the entropy counter $c = 0$, if an adversary invokes $\mathbf{get - next}$ when $\mathsf{corrupt} = \mathsf{true}$. Additionally, ROB implicitly sets $\beta = 1$, meaning immediate recovery when enough entropy enters the system after the compromise (or any premature next call).

3.2 Impossibility Result

The idea of our attack is that the adversary provides bit inputs such that every n inputs has one bit of entropy. Further, the premature next call will reveal information about this bit. We will prove the result through a series of lemmas. As mentioned before, we will assume that the inputs and the outputs are merely bits.

In the remainder of this section, we will work with a function $f_{\mathsf{PRNG}} : \{0,1\}^\mu \times \{0,1\}^n \to \{0,1\}$, for $\mathsf{PRNG} = (\mathsf{refresh}, \mathsf{next})$. This function $f_{\mathsf{PRNG}}(s, x)$ represents the application of n iterated refresh calls, starting from an initial state s with input $x_1, \ldots, x_n \in \{0,1\}$, before finally applying next to produce an output bit y, or more formally:

$f_{\mathsf{PRNG}}(s, x_1 || \ldots || x_n)$:
  for $i = 1$ **to** n
    $s = \mathsf{refresh}(s, x_i)$
  $(s, y) = \mathsf{next}(s)$
  return y

This is equivalent to applying one "big-refresh" before one next, as indicated before. Further, we write x_{-i} for $x_1 || \ldots || x_{i-1} || x_{i+1} || \ldots || x_n$, i.e., the binary string x, except for the i-th bit. Then, we can define $x_{-i,\chi}$ to be the string where the i-th bit is set to χ, i.e. $x_{-i,\chi} := x_1, \ldots, x_{i-1}, \chi, x_{i+1}, \ldots, x_n$. For any function g and any i, we abuse notation and write $g(x_{-i,\chi})$ as a shorthand for $g(x_1 || \ldots || x_n)$ where i-th bit is χ. We will also use X to denote the random variable corresponding to $x_1 || \ldots || x_n$ and use X_{-i} to denote the random variable corresponding to x_{-i}.

▶ **Lemma 4.** *There exists a randomized $O(n^2)$ algorithm FIND^g with oracle access to any function $g : \{0,1\}^n \to \{0,1\}$, such that with probability at least $1 - 2^{-n}$ (over the coins FIND^g), FIND^g outputs (i, z) which satisfies precisely one of the following two (disjoint) properties:*

- *$i = 0$, $z \in \{0,1\}$, and $\mathsf{P}[g(U_n) = z] \geq 0.6$.*
- *$1 \leq i \leq n$, $z \in \{0,1\}^{n-1}$ and $g(x_{-i,0}) \neq g(x_{-i,1})$ where $x_{-i} = z$.*

(In other words, FIND^g either discovers that $g(U_n)$ is biased, or it identifies two n-bit strings that differ in a single bit such that g returns different values on these two strings.)

Proof. The algorithm FIND^g is defined in Figure 2. The FIND^g's output satisfies case 2 unless , after n^2 tries, the algorithm fails to find a value in the first loop. Further, in the second loop, the algorithm merely outputs the majority element.

Analysis of First for Loop. Let us look at trying to determine i, z such that it satisfies the second property. To this end, we will rely on results from graph theory. Specifically, we will use the edge isoperimetric inequality for a Hypercube graph [13, §4], which we recall (in our context) below.

For our setting, we have a Hypercube graph $Q_n = (V, E)$ where each vertex corresponds to a binary vector of length n, i.e., $|V| = 2^n$. Further E is the set of all edges that connects $(\mathbf{u}, \mathbf{v})$ if the Hamming distance between $\mathbf{u}$ and $\mathbf{v}$ is exactly 1. This gives us that: $|E| = n \cdot 2^{n-1}$. Now, we are interested in edges between a vertex $\mathbf{u}$ and $\mathbf{v}$ if $g(\mathbf{u}) \neq g(\mathbf{v})$. Now, for any set S of size $k \leq 2^{n-1}$, the number of "cut" edges C from the set to its compliment is bounded by the isoperimetric inequality [13, §4.2.1] as follows:

$$C \geq k \cdot (n - \log_2 k) \geq k$$

However, now we need to determine how many $\mathbf{u} \in V$ exists such that $g(\mathbf{u}) = 0$ (or 1). If, $0.4 \leq \mathbb{E}[g(U_n)] \leq 0.6$, then we know that there exists $0.4 \cdot 2^n$ vectors $\mathbf{u}$ with $g(\mathbf{u}) = 0$ and a similar number for $g(\mathbf{u}) = 1$.

Therefore, the probability of choosing the desired edge is at least:

$$\frac{k}{n \cdot 2^{n-1}} \geq \frac{0.4 \cdot 2^n}{n \cdot 2^{n-1}} = \frac{0.8}{n}$$

In other words, the probability that a randomly chosen edge is the desired edge occurs with probability $p \geq 0.8/n$. Therefore, one can simply pick an edge $e \in E$, uniformly at random, and then test to see if it is the desired edge. Now, if one were to do n^2 such tests, we get:

$$\mathsf{P}[g(x_{-i,0}) \neq g(x_{-i,1})] > 1 - 2^{-n}$$

This math follows from the fact that the probability of failure of algorithm is:

$$\left(1 - \frac{0.8}{n}\right)^{n^2} \leq e^{-0.8n} < 2^{-n}$$

Note that this result only follows if $0.4 \leq \mathbb{E}[g(U_n)] \leq 0.6$.

Analysis of Second for Loop. However, if $\mathbb{E}[g(U_n)] < 0.4$ or $\mathbb{E}[g(U_n)] > 0.6$, then we know that the distribution, is biased either in favor of 0 or 1. If it is biased in favor of 1 (i.e., $\mathbb{E}[g(U_n)] > 0.6$), then we know that $> 0.6 \cdot 2^n$ inputs $\mathbf{x}$ will evaluated to 1 or $< 0.4 \cdot 2^n$. In other words, the probability of success $p > 0.6$. Therefore, one can apply Chernoff bounds, to get that $\mathsf{P}[g(U_n) = z] \geq 0.6$ with probability $1 - 2^{-n}$.

The correctness of FIND^g follows from our earlier discussion. It is easy to see that FIND^g runs in time $O(n^2)$ as the lines inside the first for loop take constant time if one were to sample the edge by picking i and x_{-i}. ◀

Algorithm $\textsc{Find}^g$

for $i = 1$ **to** n^2:
  Pick an edge $(\mathbf{u}, \mathbf{v}) \in E$, uniformly at random.
  Use oracle access to g to compute $g(\mathbf{u})$ and $g(\mathbf{v})$.
  if $g(\mathbf{u}) \neq g(\mathbf{v})$ **then**
    Find i such that $u_i \neq v_i$.
    By definition, there exists a unique i that satisfies this condition.
    return (i, u_{-i})
    break
for $i = 1$ **to** $120 \cdot n$:
  $count = 0$
  Sample $\mathbf{x} \leftarrow \{0,1\}^n$
  Compute $count = count + g(\mathbf{x})$
if $count > n/2$ **then** $z = 1$
else $z = 0$
return $(0, z)$

Figure 2 Description of $\textsc{Find}^g$.

▶ **Theorem 5.** *There is no* $(\gamma^*, \beta, q, 0.1, \tau)$*-secure PRNG for* $\gamma^*\beta < \sqrt{q}$ *and* $\tau \geq \Omega((t_{\mathsf{next}} + t_{\mathsf{refresh}}) \cdot n^3)$ *where* $n = \gamma^* \cdot \beta$ *and* t_{next} *and* t_{refresh} *are the time required to compute* next *and* refresh *respectively.*

Proof. We will use the $\textsc{Find}^g$ algorithm defined in Lemma 4 to create an adversary $\mathcal{A}$ that wins the $\mathsf{NROB}(\gamma^*, \beta)$ security game. The pseudocode for the adversary is provided in Figure 3. Here, the definition of the function g is as follows: $g(\mathbf{x}) = f(s, x_1 || \ldots || x_n)$ where s is the current state s and $n = \gamma^* \cdot \beta$. $\mathcal{A}$ is aware of the very first state s. The attacker then runs $\textsc{Find}^g$ on this function g and receives (i, z) as output. Now, we have two cases:

- $i = 0$. Recall that $i = 0$ implies that $g(U_n)$ is biased towards the value z. Therefore, $\mathcal{A}$ simply invokes $\mathbf{get-state}$ first. This is done not to retrieve the state, but rather to reset the counters of T and T^*. Now, $\mathcal{A}$ uses the biased nature of g on U_n to provide uniform bit $n = \gamma^*\beta$ times. At the end of this process, we have $T^* = \gamma^*\beta$ and the attacker is required to break the scheme within another β steps. After the n inputs, $\mathcal{A}$ invokes $\mathbf{next-ror}$ to receive its challenge response. If this challenge response is equal to z, then we know that $b = 0$, indicating it is the real distribution and not the random distribution.
- $i \neq 0$. Recall that $i \neq 0$ implies that there exists two n-bit strings that differ in one bit, but g produces different evaluations. i is the bit where the strings differ and z is the value for the remaining bits. $\mathcal{A}$ begins by writing down z in its state, and then provides one bit of entropy by randomly sampling x_i. Now, $\mathcal{A}$ uses a "premature" call to **get-next** and receives y as response. With knowledge of z, $\mathcal{A}$ can compute g for two choices of input at the i-th bit and then use y to uniquely determine what was the input at x_i which also helps $\mathcal{A}$ recover the state. This process is repeated γ^* times to provide γ^* bits of entropy. We keep doing this for $\gamma^{*2}\beta^2$ steps, and then, request **next-ror** . However, with knowledge of the state, due to premature next, $\mathcal{A}$ knows the challenge and therefore wins with a non-negligible advantage.

In other words, we have an attacker which can break this scheme, with non-negligible probability, if $q > \gamma^{*2}\beta^2$.[2] ◀

[2] Note, that when $q < \gamma^*\beta$, every PRNG is vacuously secure as there is no need for recovery: at least γ^* steps are needed to inject the required γ^* bits of entropy, and the attacker simply runs out of refresh calls to trigger the security requirement. This, of course, assumes ideal entropy accumulation.

Algorithm $\mathcal{A}$

Set $s = 0$
$\sigma = \perp$
$t = 0$
while $t \leq \gamma^{*2}\beta^2$
  Set $g(\mathbf{x}) = f(s, \mathbf{x})$
  $(i, z) \leftarrow \text{FIND}^g$
  if $i = 0$ **then**
    Invoke **get-state** to get the current state s^*. // This resets $T = 0$.
    for $j = 1$ **to** n:
      Output $X_{t+j} = U_1$ // $\mathrm{H}_\infty(X_{t+j}) = 1$.
    Invoke **next-ror** for challenge δ
    if $\delta = z$ **then return** 0
    else return 1
  else
    Set $X_{t+i} = U_1$ // $\mathrm{H}_\infty(X_{t+i}) = 1$.
    Use z to set X_{t+k} for $k \neq i$. // $\mathrm{H}_\infty(X_{t+k}) = 0$ for $k \neq i$.
    Invoke **get-next** to get output y.
    Let $a_{-i} = z$
    if $g(a_{-i,0}) = y$ **then** $x_{t+i} = 0$
    else $x_{t+i} = 1$
    for $i = 1$ **to** $\alpha\beta$
      $s = \mathsf{refresh}(s, x_{t+i})$
    $(s, y) = \mathsf{next}(s)$
    $t = t + \alpha\beta$
Invoke **next-ror** for challenge δ
if $\mathsf{next}(s) = (\cdot, \delta)$ **then return** 0
else return 1

Figure 3 Pseudocode for $\mathcal{A}$ for Theorem 5.

3.3 Towards Positive Results

The impossibility is, of course, artificial, but it raises questions about how to overcome it, even assuming ideal entropy accumulation and extraction. In Section 4 we abstract the notion of the scheduler which models security against premature next attacks using multiple pools which assume to accumulate entropy optimally (which abstracts away entropy accumulation and extraction).

In this setting, we will first analyze a single-pool scheduler scheme for the special "root pool" in Section 5. This scheme uses a single pool with exponentially decaying time intervals to drain this pool, but the rate of such recovery will depend on the entropy rate *counted from the boot time* (as opposed to the latest compromise in the general notion). The latter point is why we don't want to use this one-pool scheme for the general-purpose PRNG, where we would like to recover from compromise *no matter when it happens.*

For such scenarios, we revisit the round-robin Fortuna scheduler, where [8] observe that this scheme provably overcomes our impossibility result, by assuming all entropy comes at a fixed (but unknown) rate. Instead, in Section 6 we significantly generalize this positive result. The idea is to redefine the notion of entropy we use in a way that makes it more restrictive than traditional (min-) entropy, but not as restrictive as assuming fixed constant rate.[3] Intuitively, our notion of entropy will not allow attacks where the entropy varies too widely within a given round-robin (but can change from one round-robin to another) – in a sense that the attacker will get almost no credit for high-entropy samples when there is at least one low entropy sample within a given round-robin.

[3] We also note that the results about fast entropy accumulation in the register [5] might justify why our new (more restrictive) notion of entropy might be reasonable to expect in practice.

Construction: Premature-Next Robust PRNG

$\underline{\mathsf{refresh}^*(x, \overline{s})}$
Parse $\overline{s}$ as $(s_0, \ldots, s_{k-1}, \rho)$
$\mathsf{in}, \mathsf{out} \leftarrow \mathsf{SC}()$
$s_{\mathsf{in}} \leftarrow \mathsf{refresh}(s_{\mathsf{in}}, x)$
if $\mathsf{out} \neq \perp$ **then**
  $(s_{\mathsf{out}}, R) \leftarrow \mathsf{next}(s_{\mathsf{out}})$
  $\rho \leftarrow \rho \oplus R$
return $\overline{s} = (s_0, \ldots, s_{k-1}, \rho)$

$\underline{\mathsf{next}^*(\overline{s})}$
Parse $\overline{s}$ as $(s_0, \ldots, s_{k-1}, \rho)$
$(Y, \rho) \leftarrow \mathcal{G}(\rho)$
return $\left(\overline{s} = (s_0, \ldots, s_{k-1}, \rho), Y\right)$

Figure 4 Construction of $\mathcal{G} = (\mathsf{refresh}^*, \mathsf{next}^*)$.

4 Seedless Scheduler

For the remainder of this paper, we will assume ideal accumulation and extraction. Further, rather than working with entropy, we will employ the notion of a sequence of weights $\mathbf{w} = (w_1, \ldots, w_q)$ where the weights have been normalized so that $w_i \in [0, 1]$ and a pool is "full" when it has accumulated weight 1. (Specifically, to move between the weight w_i and the entropy γ, one should multiply by the entropy γ^*_{rob} required for a single pool to recover.) See [8].

4.1 Syntax of a Scheduler

We define the syntax of the scheduler below. Note that this scheduler is deterministic and oblivious, i.e., it does not depend on the actual input or its entropy.

▶ **Definition 6** (Syntax of Scheduler). *A (k, q)-scheduler is a deterministic algorithm* SC *that produces q pairs: $\{(\mathsf{in}_i, \mathsf{out}_i)\}_{i=1}^{q}$ where $\mathsf{in}_i \in [k], \mathsf{out}_i \in [k] \cup \{\perp\}$ for $i = 1, \ldots, q$.*

Note that, when the number of "pools" k is not critical to be specified explicitly, a deterministic (k, q)-scheduling scheduler can be thought of as a sequence of values $\{\mathsf{empty}\}_{i=1}^{q}$ corresponding to the time at which each input i with weight w_i is emptied. More formally, we can define: $\mathsf{empty}_i := \min\{j : j > i \wedge \mathsf{out}_j = \mathsf{in}_i\}$

4.2 Seedless PRNG, with Premature Next

Before we venture into the security of such a scheduler, it would be prudent to take a step back and look at an informal composition of a seedless scheduler with PRNGs that are not resilient to premature next in order to achieve security with premature next. Indeed, it is also equally important to frame our composition results, in the face of the impossibility result from Section 3.2 (and also the unrestricted scheduler impossibility later in this section). This is precisely the reason why we do not state a formal composition theorem, as it is vacuous for the most general case. However, the composition is still robust for restricted notions of scheduler security to yield relaxed forms of PRNG security with premature next.

The composition relies on seedless PRNGs which are not secure with premature next. These are typically parametrized by just γ^*, which is the minimum entropy needed for the PRNG to begin producing pseudorandom outputs (see Figure 5). In essence, these have $\alpha = \gamma^*$ and $\beta = 1$ with a reset of all counters when an adversary invokes $\mathbf{get} - \mathbf{next}$ with $\mathsf{corrupt} = \mathsf{true}$. The instantiation of this PRNG can be from the work of Coretti et al. [4]

or from the work of Dodis et al. [6]. Such a PRNG, secure without premature next and parametrized by γ^* is combined with a scheduler. The goal of a scheduler would be to ensure that the input, as it arrives, is allocated a particular pool such that:

- With "enough entropy", a pool is filled, i.e., accumulates γ^* amount of entropy.
- This pool will be emptied within "sufficient time", to recover from compromise.

We will formalize these notions of "enough entropy" and "sufficient time" in the next section.

Formally, we define a seedless PRNG, with construction as follows:

- Let SC be a scheduler with k pools.
- Let $\mathcal{G}_i = (\mathsf{refresh}_i, \mathsf{next}_i)$ be *seedless* PRNGs with input (see Section A), for $i = 0, \ldots, k-1$. For simplicity, we will assume that each $\mathcal{G}_i$ is (m, r)-PRNG. These are PRNGs which are not secure with premature next calls.
- Let $\mathbf{G} : \{0,1\}^m \to \{0,1\}^{2m}$ be a pseudorandom generator (without input).

Then, we construct a PRNG with input $\mathcal{G}(\mathsf{SC}, \{\mathcal{G}_i\}_{i=0}^{k-1}, \mathbf{G}) = (\mathsf{refresh}^*, \mathsf{next}^*)$ as shown in Figure 4, where the scheduler mandates which pool $\mathcal{G}_{\mathsf{in}}$ to use (via $\mathsf{refresh}$) to accumulate entropy from a new sample, and which pool $\mathcal{G}_{\mathsf{out}}$ (if any) to "empty" (via next) into the main register ρ for $\mathbf{G}$.

4.3 Security of a Scheduler

We will define different notions of security for a scheduler. As with PRNGs, (k, q)-scheduler security model is parameterized by two parameters α, β. Informally, it states that if the adversary chooses to provide α units of fresh entropy (i.e., a sequence of w_i values that sum up to α) within a time $t \le q/\beta$, then we guarantee recovery within time $\beta \cdot t \le q$. Formally,

▶ **Definition 7** (General Security of Scheduler). *A (k, q)-scheduler is (α, β)-general-secure if if $\forall t_0, t$ such that $t_1 = t_0 + \beta \cdot t \le q$, and $\forall$ weights $w_1, \ldots, w_q \in [0, 1]$ such that $\sum_{i=1}^{t} w_{t_0+i} \ge \alpha$, the scheme recovers from the compromise in time $t_0 + \beta t$ where recovery occurs if $\exists\ j \in [k]$ and $\exists\ \widehat{T} \in [t_0 + 1, t_0 + \beta \cdot t]$ such that:*

1. $\mathsf{out}_{t_0+1}, \ldots, \mathsf{out}_{\widehat{T}-1} \ne j$ *(pool j has not been emptied before time $\widehat{T}$);*
2. $\mathsf{out}_{\widehat{T}} = j$ *(pool j is emptied at time $\widehat{T}$); and*
3. *(pool j has filled)* $\sum_{\substack{t_0 < i \le \widehat{T} \\ \mathsf{in}_i = j}} w_i \ge 1$.

4.4 Impossibility Result

We can show that, for general security, there exists an impossibility result. Specifically, we will show that for any $k \in \mathbb{N}$, there exists a choice of q such that any (k, q)-scheduler is not (α, β)-secure. In other words, for a suitable choice of q, one can break the scheduler to never recover from compromise for any α, β. Note that this is incomparable to the earlier impossibility result discussed in Section 3.2 as this assumes the existence of pools.

▶ **Theorem 8.** *For any $k \in \mathbb{N}$, there exists $q^* = \alpha^2\beta^2$ such that a given (k, q)-scheduler is not (α, β)-secure for any $q \ge q^*$.*

We defer the proof of this theorem to Section B. The immediate consequence of this impossibility result is the following: there exists input weight sequence $\mathbf{w}$ such that, irrespective of the number of pools, we can inject entropy at a slow rate, such that no scheduler is general-secure. It also implies that we need to make some relaxations to achieve usable security results.

5 Reboot Secure Schedulers

The first relaxation corresponds to the situation when the system is just rebooted, i.e., we are at $t_0 = 0$. We will call this as the "reboot security" of a scheduler. This corresponds to the situation when you just turn on the computer. For this case, we can have a much simpler and better RNG, having only one pool. Like Fortuna, this pool is emptied every β^i steps for gradually increasing values of $i = 0, 1, 2, \ldots$, where β is a small integer (Windows 10 uses $\beta = 3$).

▶ **Definition 9** (Reboot Security of Scheduler). *A (k, q)-scheduler is (α, β)-reboot-secure if for $t_0 = 0$, $\forall t$ such that $t_1 = t_0 + \beta \cdot t \leq q$, and $\forall$ weights $w_1, \ldots, w_q \in [0, 1]$ such that $\sum_{i=1}^{t} w_{t_0+i} \geq \alpha$ the scheme recovers from the compromise in time $t_0 + \beta t$, where the definition of recovery is as defined in Definition 7.*

The composition of such a reboot-secure scheduler with our "not-premature-next" PRNGs will trivially yield a "premature-next" boot PRNG, i.e., the PRNG that is used at the time when the system is booting up.

We start with a lower bound on reboot-security, irrespective of the number of pools k.

▶ **Theorem 10.** *For a (k, q)-scheduler to be (α, β)-reboot secure, $\alpha \geq \lfloor \log_\beta(q) - \log\log q \rfloor - 1$ (i.e., $q \leq \alpha\beta^\alpha$)*

For simplicity let us assume that $q = \alpha\beta^{\ell+1}$, for some $\ell > 0$. Then, divide the time from $\alpha + 1$ to q into intervals of the following form: $(\alpha\beta^{i-1}, \alpha\beta^i]$ for $i = 1$ to $\ell + 1$. We have the following claim:

▷ Claim 11. For any (α, β)-reboot secure scheduler with corresponding emptying sequence $\mathsf{empty}_1, \ldots, \mathsf{empty}_q$ and any $i \in [\ell]$, there must exist a t such that $\mathsf{empty}_t \in (\alpha\beta^i, \alpha\beta^{i+1}]$. (In other words, there must be a pool that is first emptied after roughly β^i steps for every i.)

We defer the proof of this claim to Section B.

Proof of Theorem 10. From Claim 11, we get that there are at least $\lfloor \log_\beta(q/\alpha) \rfloor$ distinct empties, and there needs to be entropy of 1 emptied in each of these empties. By Pigeonhole Principle, we will need $\alpha \geq \lfloor \log_\beta(q/\alpha) \rfloor$ to have any hope of recovery, which implies $\alpha \geq \lfloor \log_\beta(q) - \log\log q \rfloor - 1$. ◀

We now give a scheme that nearly matches the lower bound. This scheme uses the same strategy as Windows 10's "Root RNG" which is used at system startup [9].

▶ **Construction 12** (Reboot Scheme). *The scheme has $k = 1$. $\mathsf{in}_i = 0$ for $i = 1, \ldots, q$.*

$$\mathsf{out}_i = \begin{cases} 0 & \textbf{if } i = \beta^j \\ \perp & \textbf{else} \end{cases}$$

In other words, $\forall i \in [\beta^{j-1}, \beta^j)$, empty at time β^j.

▶ **Theorem 13.** *Construction 12 is (α, β)-reboot secure for $q = \alpha\beta^\alpha$ (i.e., $\alpha \approx \log_\beta q - \log\log q$).*

Proof. Define t to be the time within which the adversary provides α entropy, i.e., $\sum_{i=1}^{t} w_i \geq \alpha$ where these w_i are adversarially chosen. It is clear that $t \geq \alpha$, as we need at least α steps to provide α entropy when $w_i \in [0, 1]$.

Let i be such that $\alpha \in (\beta^{i-1}, \beta^i]$. Now, it is clear that if $t = \alpha$, then the empty at β^i will ensure recovery from compromise. We can induct similar to the proof of Claim 11 to get that if $t \in [\beta^{\ell-1}, \beta^\ell)$ for some $\ell \geq i$, then there $\exists j \in [\beta^{\ell-1}, \beta^\ell)$ such that $w_j = 1$ (or possibly a set of such j's which sum up to 1), which is emptied at β^i, thus recovering from compromise. Specifically, if we have $t \in [\beta^{\ell-1}, \beta^\ell)$, then at each of the preceding $\ell - 1$ intervals (each with an empty), $\mathcal{A}$ provides $1 - \epsilon$ entropy, for some arbitrarily small ϵ. This gives a total of almost $\ell - 1$ entropy across these intervals. Therefore, it follows that the remainder of $\alpha - \ell + 1 > 1$ needs to be provided between $w_{\beta^{\ell-1}}$ and w_t to hit α and all of these are emptied at β^ℓ, recovering from compromise. ◀

6 Repeat Secure Schedulers

A general secure scheme is a stronger model of security than the reboot model. This follows because the value of t_0 is also the choice of the adversary, in addition to the choice of t. However, the impossibility result from Theorem 8 imply a need for relaxation.

Round-Robin Schedulers. Simple round-robin schedulers achieve very good $\alpha \approx \log_\beta(q)$ for the special cases when all of the w_t are equal to some (unknown, adversarially chosen) value w, i.e., $w_1 = w_2 = \ldots = w_q = w$ and setting the number of pools $k \approx \log_\beta(q)$ (so 1 or 2 pools are too little). β is a smaller integer usually 2 or 3 in practice, as in [10, 8]. More formally, such schedulers simply set $\mathsf{in}_t = t \bmod k$ As for out_t, this is set to $\perp$ inside one round (i.e. $t \bmod k \neq 0$). At the the of each round, when $t = k\ell$, one looks at the largest index $i \geq 0$ such that β^i divides ℓ. Then out empties the i-th pool: $\mathsf{out}_t = i$

▶ Remark 14. There is a marginal gain in efficiency when we empty all pools $\leq i$, instead of just the i-th pool. In other words, out is a set, rather than a single index. However, for our analysis below, we will continue to work with the assumption that a single pool is emptied. (More generally, we do not make much of an attempt to optimize the parameters that we achieve. See [8] for an optimized version of similar construction.)

***k*-smooth Sequences.** Our main observation is that we can significantly extend the constant-rate analysis as follows. The idea is to allow support any constant rate within a round-robin (rather than go for a constant (but unknown) rate scheduler). This constant can change arbitrarily once the next round-robin is started. Namely, we don't have to fix the same constant for all q entropies but can change it every $k \ll q$ steps. In practice, this means that while the quality of entropy can change over time, we heuristically assume that it changes rather smoothly, and we rarely have huge jumps within a given round-robin.

▶ **Definition 15** (Repeating Sequences). *$\mathbf{w} = (w_1, \ldots, w_q)$ with $0 \leq w_i \leq 1$ is called k-repeating if $w_{jk+1} = w_{jk+2} = \ldots = w_{jk+k}$ for $j = 0, \ldots, t-1$ where $q = k \cdot t$*

▶ **Definition 16** (Repeat Security of Scheduler). *A (k, q)-scheduler is (α, β, k)-repeat-secure if $\forall\ t_0, t$ such that $t_1 = t_0 + \beta \cdot t \leq q$, and $\forall$ k-repeating weights $w_1, \ldots, w_q \in [0, 1]$ such that $\sum_{i=1}^{t} w_{t_0+i} \geq \alpha$ the scheme recovers from the compromise in time $t_0 + \beta t$, where the definition of recovery is as defined in Definition 7.*

To achieve such repeating sequences, we take any standard $\mathbf{w} = (w_1, \ldots, w_q)$ and apply a k-flattening, as defined below.

▶ **Definition 17** (k-Flattening). *Given a sequence* $\mathbf{w} = (w_1, \ldots, w_q)$ *and a number* $k \geq 1$*, where for simplicity of notation let us assume* $q = kt$*, we define* k-smooth flattening *of* $\mathbf{w}$ *to be* $\mathbf{w}' = (w'_1, \ldots, w'_q)$*, where for any round-robin* $j \in \{0, \ldots t-1\}$ *and* $i \in \{1 \ldots k\}$*, we let*

$$w'_{jk+i} = \min(\ w_{jk+1},\ w_{jk+2}, \ldots,\ w_{(j+1)k}\ )$$

Intuitively, we change the entropy w_j to the smallest of k surrounding entropies inside a given round-robin. Of course, $k = 1$ corresponds to $w'_t = w_t$, but we already know that 1 pool is not enough (as this would give a general scheduler for the unrestricted entropy setting). For larger k, however, the flattened values could be noticeably lower than the original. For example, if $k = 3$ and $\mathbf{w} = \{1, 1/2, 1/3, 1/4, 1/5, 1/6\}$, the 3-flattening of $\mathbf{w}$ is $\mathbf{w}' = \{1/3, 1/3, 1/3, 1/6, 1/6, 1/6\}$. Of course, for a constant rate $w_1 = \ldots w_q = w$, k-flattening does not change anything, which explains why our results below naturally generalize the constant-rate analysis from the work of Dodis et al. [8].

Jumping ahead, we will see that the Fortuna scheduler is "secure" for any (normalized) entropy sequence $\mathbf{w}$, with the understanding that the attacker gets "entropy credit" within a single round-robin equals to k times the *lowest* entropy value in contributes within this round.

New Result. Now, we show that while the original (α, β)-definition above cannot be achieved when applied to $\mathbf{w}$ itself, the analysis for constant-rate schedulers works for general entropy sequences, provided we simply apply it to k-flattening of $\mathbf{w}$ (where $k \approx \log_\beta q$ is the number of pools) instead of $\mathbf{w}$ itself! Namely, a given round only gets "credit" for the smallest entropy (times k) it contributed to any of the k pools. So we do not give the adversary credit if it wildly changes the entropy values within a given round.

We now present our construction, which is parameterized by the number of pools k and a base b. One typically takes $b = 2$ or $b = 3$, and, e.g., $k = 32$ or $k = 64$ in practice, and works for $q \leq b^k$.

▶ **Construction 18** (Smooth scheduler). *Consider the following* $(k, q := b^k)$*-scheduler for integers* $b \geq 2$ *and* $k \geq 1$*:*

- $\mathsf{in}_i = i \mod k$
- $\mathsf{out}_i = \begin{cases} \perp & \textbf{if } i \mod k \neq 0 \\ j & \textbf{if } i = k\ell \end{cases}$ *where* $j \geq 0$ *is the largest* j *such that* $\ell \mod b^j = 0$ *for* $i = k\ell$

We now prove that this scheduler is secure (against k-repeating sequences). For simplicity, we make little attempt to optimize the parameters. See [8] for a carefully optimized version of this result for the special case where the entropy rate is constant (i.e., the case of q-repeating weights).

▶ **Theorem 19.** *For any integers* $b \geq 2$ *and* $k \geq 1$*, Construction 18 is* (α, β, k)*-repeat-secure for*

$$\alpha := 3k - 2 \approx 3 \log_b q; \qquad \text{and} \qquad \beta := 2b\left(1 + \frac{k}{\alpha}\right) \approx \frac{8b}{3} = \frac{8}{3} \cdot q^{1/k}$$

In particular, for $k = \log_b q$ *and* $q \geq b^2$*, we have* $\alpha \leq 3 \log_b q$ *and* $\beta \leq 3b$*.*

Notice, this result explains how the recovery factor β shrinks very quickly as we increase the number of pools k, starting with (roughly) q all the way down to being a constant. In particular, β becomes constant once the number of pools becomes logarithmic in q.

Moreover, up to constant factors in α and β (which, again, we do not attempt to optimize), Theorem 19 is tight. In particular, [8, Proposition 1] proved that even in the "constant-rate" case of q-repeating weights, no scheduler can be (α, β)-secure with $\alpha\beta \leq \log_e q - \log_e \log_e q - 1$. And our scheduler matches this bound (up to a constant factor) when $b = O(1)$ and $k = O(\log q)$. We defer the proof of the theorem to Section B.

References

1 Boaz Barak and Shai Halevi. A model and architecture for pseudo-random generation with applications to /dev/random. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 2005*, pages 203–212, Alexandria, Virginia, USA, November 7–11 2005. ACM Press. doi:10.1145/1102120.1102148.

2 Elaine Barker and John Kelsey. Recommendation for random number generation using deterministic random bit generators. NIST Special Publication 800-90A, 2012.

3 Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988. doi:10.1137/0217015.

4 Sandro Coretti, Yevgeniy Dodis, Harish Karthikeyan, and Stefano Tessaro. Seedless Fruit is the sweetest: Random number generation, revisited. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 205–234, Santa Barbara, CA, USA, August 18–22 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-26948-7_8.

5 Yevgeniy Dodis, Siyao Guo, Noah Stephens-Davidowitz, and Zhiye Xie. No time to hash: On super-efficient entropy accumulation. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 548–576, Virtual Event, August 16–20 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-84259-8_19.

6 Yevgeniy Dodis, Siyao Guo, Noah Stephens-Davidowitz, and Zhiye Xie. Online linear extractors for independent sources. In Stefano Tessaro, editor, *2nd Conference on Information-Theoretic Cryptography, ITC 2021, July 23-26, 2021, Virtual Conference*, volume 199 of *LIPIcs*, pages 14:1–14:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ITC.2021.14.

7 Yevgeniy Dodis, David Pointcheval, Sylvain Ruhault, Damien Vergnaud, and Daniel Wichs. Security analysis of pseudo-random number generators with input: /dev/random is not robust. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 647–658, Berlin, Germany, November 4–8 2013. ACM Press. doi:10.1145/2508859.2516653.

8 Yevgeniy Dodis, Adi Shamir, Noah Stephens-Davidowitz, and Daniel Wichs. How to eat your entropy and have it too: Optimal recovery strategies for compromised rngs. *Algorithmica*, 79(4):1196–1232, 2017. doi:10.1007/s00453-016-0239-3.

9 Niels Ferguson. The windows 10 random number generation infrastructure, October 2019. URL: https://aka.ms/win10rng.

10 Niels Ferguson and Bruce Schneier. *Practical cryptography*. Wiley, 2003.

11 Peter Gazi and Stefano Tessaro. Provably robust sponge-based PRNGs and KDFs. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 87–116, Vienna, Austria, May 8–12 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49890-3_4.

12 Viet Tung Hoang and Yaobin Shen. Security analysis of NIST CTR-DRBG. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 218–247, Santa Barbara, CA, USA, August 17–21 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-56784-2_8.

13 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *BULL. AMER. MATH. SOC.*, 43(4):439–561, 2006.

14 Daniel Hutchinson. A robust and sponge-like PRNG with improved efficiency. In Roberto Avanzi and Howard M. Heys, editors, *SAC 2016*, volume 10532 of *LNCS*, pages 381–398, St. John's, NL, Canada, August 10–12 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-69453-5_21.

15 John Kelsey, Bruce Schneier, and Niels Ferguson. Yarrow-160: Notes on the design and analysis of the yarrow cryptographic pseudorandom number generator. In *In Sixth Annual Workshop on Selected Areas in Cryptography*, pages 13–33. Springer, 1999.

16 John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Cryptanalytic attacks on pseudorandom number generators. In Serge Vaudenay, editor, *FSE'98*, volume 1372 of *LNCS*, pages 168–188, Paris, France, March 23–25 1998. Springer, Heidelberg, Germany. doi:10.1007/3-540-69710-1_12.

17 Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996. doi:10.1006/jcss.1996.0004.

18 Amit Sahai and Salil Vadhan. A complete problem for statistical zero knowledge. *Journal of the ACM*, 50(2):196–249, 2003. Extended abstract in FOCS '97.

19 Thomas Shrimpton and R. Seth Terashima. A provable-security analysis of Intel's secure key RNG. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 77–100, Sofia, Bulgaria, April 26–30 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46800-5_4.

20 Wikipedia contributors. /dev/random – Wikipedia, the free encyclopedia, 2021. [Online; accessed 9-January-2022]. URL: https://en.wikipedia.org/w/index.php?title=/dev/random&oldid=1056079736.

21 Joanne Woodage and Dan Shumow. An analysis of NIST SP 800-90A. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 151–180, Darmstadt, Germany, May 19–23 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17656-3_6.

A PRNG Robustness, without Premature Next

This is an abridged discussion about the robustness security game ROB, for the seedless setting (and with independent samples), but *without* allowing Premature Next calls. The security game is presented as Figure 5. The main difference from the NROB game presented in Figure 1 is that the entropy counter c is reset to 0 with each "premature next" call to $\mathbf{get} - \mathbf{next}$, and also there is no recovery delay parameter β. As the result, there is no need to keep track of the number of steps T^* to accumulate γ^* bits of entropy.

B Deferred Proofs

Proof of Theorem 8. The attack works as follows: we will provide α entropy, in $\alpha^2\beta$ steps. The security requirement is that recovery needs to happen within time $\alpha^2\beta^2$. Recall that recovery occurs if there is a pool that is emptied within $\alpha^2\beta^2$ which has total entropy of 1.

More formally, let I_j denote the j-th interval, of length $\alpha\beta$ starting from 0. This leads us to two cases:

- $\exists j^*$ such that no pool is emptied within I_{j^*}. Formally, there exists no time step i with $\mathsf{empty}_i \in I_{j^*}$.
  Then, set $t_0 = \alpha\beta(j^* - 1) - 1$. After this state compromise, we provide a sequence of 1s of length α, which will set $T^* = \alpha$ and expect recovery in time $t_0 + \beta T^* = \alpha\beta j^* - 1$, which is still inside the interval I_{j^*}. However, we assumed no pool is emptied within I_{j^*}, so no recovery can be possible.

Game The PRNG Robustness Game

ROB

$\sigma = \bot$; $s = 0$; $c = 0$
$b \leftarrow \{0,1\}$; corrupt = true
for $i = 1, \ldots, q$ **do**
  $(\sigma, X_i) \leftarrow \mathcal{A}^{\mathbf{get-next}, \mathbf{get-state}, \mathbf{next-ror}}(\sigma)$
  $x_i \leftarrow X_i$
  $s = \mathsf{refresh}(s, x_i)$
  $c = c + \mathrm{H}_\infty(X_i)$
  if $c \geq \gamma^*$ **then**
    corrupt = false
$b' \leftarrow \mathcal{A}(\sigma)$

get − next

if corrupt = true **then**
  $c = 0$
$(s, y) = \mathsf{next}(s)$
return y

next − ror

$(s, y_0) = \mathsf{next}(s)$
$y_1 \leftarrow \{0,1\}^r$
if corrupt = true **then**
  return y_0
return y_b

get − state

$c = 0$; corrupt = true
return s

Figure 5 The Robustness Game (without Premature Next Calls) $\mathsf{ROB}(\gamma^*, q)$.

- $\forall j$, $\exists i$ such that $\mathsf{empty}_i \in I_j$, meaning at least one pool is emptied within all α intervals I_j.

  Set $t_0 = 0$. Then, for $j = 1, \ldots, \alpha$, pick *one* ℓ such that $\mathsf{empty}_\ell \in I_j$. Set, w_ℓ to be $1 - \epsilon$ for some arbitrarily small ϵ (remaining weights are 0). At the end of this process, the adversary has provided almost α entropy, but there is no recovery, as all of these entropies are completely wasted. By making ϵ arbitrarily small, the result follows. ◀

Proof of Claim 11. We prove this by induction. Define t to be the time within which the adversary provides α entropy, i.e., $\sum_{i=1}^{t} w_i \geq \alpha$ where these w_i are adversarially chosen. Since $w_i \leq 1$, we get that $t \geq \alpha$.

Let us assume to the contrary that there is no emptying in the interval $(\alpha, \alpha\beta]$. Now, if adversary chooses $t = \alpha$. Then, this scheme would never recover as there is no empty in the interval $(\alpha, \alpha\beta]$

Now, let us assume that there is an empty in intervals, $(\alpha, \alpha\beta], (\alpha\beta, \alpha\beta^2], \ldots, (\alpha\beta^{i-1}, \alpha\beta^i]$. We will now show that there needs to be an empty in the interval $(\alpha\beta^i, \alpha\beta^{i+1}]$. To this end, assume to the contrary. Now, note that the adversary can provide the entropy in such a way that every empty in the preceding intervals empties out $1 - \epsilon$, without recovering. This is similar to the attack detailed in the proof of Theorem 8. Further, if $t = \alpha\beta^i$, the scheme has not recovered in time 1 to t and because it has no empty in $(\alpha\beta^i, \alpha\beta^{i+1}]$ it can never hope to recover in time either. Therefore, there is an empty in the interval $(\alpha\beta^i, \alpha\beta^{i+1}]$. ◁

Proof of Theorem 19. Let $w_1, \ldots, w_q$ be k-repeating. Let t_0 and t be such that (1) $t_0 + \beta t \leq q$; and (2) $\sum_{i=1}^{t} w_{t_0+i} \geq \alpha$. We wish to show that in this case the scheduler recovers before time $t_0 + \beta t$, i.e., that there exists a $j \in [k]$ and $\widehat{T} \in [t_0 + 1, t_0 + \beta t]$ such that (1) $\mathsf{out}_{\widehat{T}} = j$; (2) $\mathsf{out}_{t_0+1}, \ldots, \mathsf{out}_{\widehat{T}-1} \neq j$; and (3) $\sum_{\substack{t_0 < i \leq \widehat{T} \\ \mathsf{in}_i = j}} w_i \geq 1$.

Indeed, we take j to be minimal such that $\mathsf{out}_{t_0+1}, \ldots, \mathsf{out}_{t_0+t} \neq j$. In particular, notice that after pool j' is emptied, pool $j' + 1$ is not emptied for the next $k(b^{j'} - 1)$ steps. And, similarly, after pool $j' + 1$ is emptied, pool j' is not emptied for the next $k(b^{j'} - 1)$ steps. It follows that $b^{j-1} \leq t/k + 1$. Since the pool j' is emptied at least once in every $2kb^{j'}$ steps, it follows that we must have $\mathsf{out}_{\widehat{T}} = j$ for some $\widehat{T} - t_0 \leq 2kb^j \leq (2t + 2k)b \leq 2b(1 + k/\alpha)t$, where in the second inequality we have used the fact that $w_i \leq 1$, which implies that $t \geq \alpha$. In particular, $\widehat{T} \leq t_0 + \beta t$, as needed.

And, since the w_i are k-repeating, we must have

$$\sum_{\substack{t_0<i\le\widehat{T}\\ \mathsf{in}_i=j}} w_i \ge \sum_{t_0<i\le\widehat{T}} \sum_{\substack{t_0<i\le t_0+t\\ \mathsf{in}_i=j}} w_i \ge \sum_{\substack{t'_0<i\le t'_0+t'\\ \mathsf{in}_i=j}} w_i = \frac{1}{k}\cdot \sum_{t'_0<i\le t'_0+t'} w_i \ ,$$

where $t'_0 := \lceil t_0/k\rceil k \ge t_0$ and $t' := \lfloor t/k\rfloor k \le t$. And, since $w_i \le 1$, we trivially have that

$$\sum_{t'_0<i\le t'_0+t'} w_i \ge \sum_{t_0<i\le t_0+t} w_i - 2k + 2 \ge \alpha - 2k + 2 \ .$$

Therefore,

$$\sum_{\substack{t_0<i\le t_0+t\\ \mathsf{in}_i=j}} w_i \ge \frac{\alpha}{k} - 2 + 2/k \ge 1 \ ,$$

as needed. ◀

Refuting the Dream XOR Lemma via Ideal Obfuscation and Resettable MPC

Saikrishna Badrinarayanan
Snap, Mountain View, USA

Yuval Ishai
Technion, Haifa, Israel

Dakshita Khurana
University of Illinois, Urbana-Champaign, IL, USA

Amit Sahai
University of California Los Angeles, CA, USA
Center for Encrypted Functionalities, Los Angeles, CA, USA

Daniel Wichs
Northeastern University, Boston, MA, USA
NTT Research, Sunnyvale, CA, USA

Abstract

We provide counterexamples to the "dream" version of Yao's XOR Lemma. In particular, we put forward explicit candidates for hard predicates, such that the advantage of predicting the XOR of many independent copies does not decrease beyond some fixed negligible function, even as the number of copies gets arbitrarily large.

We provide two such constructions:

- Our first construction is in the ideal obfuscation model (alternatively, assuming virtual black-box obfuscation for a concrete class of circuits). It develops a general framework that may be of broader interest, and allows us to embed an instance of a *resettably*-secure multiparty computation protocol into a one-way function. Along the way, we design the first resettably-secure multiparty computation protocol for general functionalities in the plain model with super-polynomial simulation, under standard assumptions.
- The second construction relies on public-coin differing-inputs obfuscation (PCdiO) along with a certain form of hash-function security called extended second-preimage resistance (ESPR). It starts with a previously known counterexample to the dream direct-product hardness amplification based on ESPR, and uses PCdiO to upgrade it into a counterexample for the XOR lemma.

Prior to our work, even completely heuristic counterexamples of this type were not known.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Cryptographic protocols

Keywords and phrases XOR Lemma, Resettable MPC, Obfuscation

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.10

Funding *Saikrishna Badrinarayanan*: Work done while at UCLA.
Yuval Ishai: Supported in part by ERC Project NTSC (742754), BSF grant 2018393, and ISF grant 2774/20.
Dakshita Khurana: Supported in part by DARPA SIEVE project contract No. #HR00112020024, a gift from Visa Research, and a C3AI DTI award.
Amit Sahai: Supported in part from a Simons Investigator Award, DARPA SIEVE award, NTT Research, NSF Frontier Award 1413955, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024.
Daniel Wichs: Partially supported by NSF grants CNS-1750795, CNS- 2055510 and the Alfred P. Sloan Research Fellowship.

1 Introduction

Consider the following standard information-theoretic technique for hardness amplification. Suppose we have a joint distribution (X, B) such that the bit B is "weakly unpredictable" given X, in the sense that B has positive entropy conditioned on X. Then, for t independent samples (x_i, b_i) from (X, B), the XOR of all b_i can only be predicted from $(x_1, \ldots, x_t)$ with $2^{-\Omega(t)}$ advantage over a random guess. The question we address in this work is whether the same holds also in the computational setting.

The computational variant of the above XOR lemma is a central tool in complexity theory that first appeared in presentations of Yao's work [76]. It postulates that if a predicate P of an input $x \in \{0,1\}^\lambda$ is weakly unpredictable by algorithms of a certain complexity, with respect to some input distribution $X = X(\lambda)$, then $P(x_1, \ldots, x_t) = \bigoplus_{i=1}^{t} P(x_i)$ for large enough t is strongly unpredictable by algorithms within a related complexity bound. Yao stated this in the context of one-way functions, where the predicate P can be any "hard-core" bit of a one-way function f: in other words, P is an easy-to-compute Boolean function of the input to f that is hard to predict given the output of f. Here the distribution X is the output distribution of f on a uniformly random input.

Since it was first introduced, different versions of this lemma were proved in the literature, starting with Levin's proof [63], an alternate proof by Impagliazzo [55], and a third one by Goldreich et al. [44]. Unfortunately, all existing proofs of the XOR lemma are stuck at the following barrier: for *any fixed* negligible function $\mu(\cdot)$, no matter how large the polynomial $t(\cdot)$ is, we cannot prove that for large enough λ, the adversary's advantage drops to $\mu(\lambda)$. More concretely, we have no evidence that *any* polynomial $t(\lambda)$ number of repetitions bring the adversary's advantage down even to $\lambda^{-\log \lambda}$, if the original hardness of f was assumed to hold only against all polynomial-sized adversaries.

It is unclear why this barrier exists, and whether it is just an artifact of known proof techniques. Intuitively, it appears that the adversary's advantage should reduce arbitrarily if we perform sufficiently many repetitions. This was previously conjectured and termed the "dream version" of Yao's XOR lemma. It was formalized in [44], and used by [18] to obtain (a stronger flavor of) weak public-key cryptography from strong one-way functions. The dream XOR lemma, if true, would fundamentally change our understanding of how intractability works. It would help arbitrarily bring down errors in security arguments with sufficiently many repetitions; leading to exciting new constructions of primitives like non-interactive non-malleable commitments following [19], or easily obtain multi-instance security [16] with good parameters from standard hardness assumptions.

Previous Explanations. Over the years, there have been some explanations for why the dream XOR lemma eludes a proof. Black-box reduction based proofs of the dream XOR lemma are likely to fail for the following folklore reason. In order to prove that XOR parallel repetition brings the adversary's advantage down to some small probability $\mu(\lambda)$, we would need an efficient reduction that uses an adversary breaking security of the parallel repetition with advantage $\mu(\lambda)$, to break the security of a single instance with significantly larger advantage. But such a reduction cannot obtain any useful information from an attacker unless it succeeds at least once, and this could require $\mathsf{poly}(1/\mu(\lambda))$ attempts. Therefore, this reduction could be efficient for every inverse polynomial $\mu(\cdot)$, but would become inefficient as soon as $\mu(\cdot)$ is a fixed negligible function. This has been attributed to Rudich [44], who also proved that the dream XOR lemma does not hold in a relativized world, by introducing an oracle that inverts every tuple of instances with probability $\mu(\lambda)$.

Shaltiel and Viola [71] initiated a line of work on the impossibility of better black-box hardness amplification results, including the XOR lemma. A recent work by Shaltiel [70], improving on [49], rules out dream XOR lemmas with proofs by so-called "class reductions," which can exploit the efficiency of their oracle. Despite this progress, it is not clear that ruling out (even relaxed forms of) black-box reductions gives a strong evidence against the dream XOR lemma. There are quite a few examples for the surprising power of non-black-box techniques, both in cryptography and in complexity theory.

Another partial explanation was offered by Dodis et al. [34], who gave a counterexample to the "dream version" of the "direct-product lemma". In particular, they showed how to construct a *weak one-way function* that no polynomial-time adversary can invert with probability better than (e.g.,) $\frac{1}{2}$, but any arbitrary polynomial number of independent copies can be simultaneously inverted in polynomial time with advantage greater than $\lambda^{-\log \lambda}$. Their construction relies on an ad-hoc assumption on an un-keyed hash function, which was justified in the random-oracle model with auxiliary input. Since the direct-product lemma implies the XOR lemma without much loss in parameters, a dream version of the former would have implied the latter. Therefore, their work closes off one potential avenue toward proving the dream XOR lemma. Had their weak one-way function also been injective, then their counter-example to the dream direct-product lemma would have also immediately yielded a counter-example to the XOR lemma by taking the hard-core bits of the function. However, their weak one-way function was not injective, in which case the ability to find *some* pre-images and break one-wayness, does not imply the ability to find *the* correct hard-core bits and break the XOR lemma.

In this work, we ask the following question, explicitly left as an open problem in [34]:

Can we build an explicit counterexample to the dream version of the XOR lemma: a one-way function with a (weakly) hardcore bit, for which predicting the XOR of many hardcore bits does not reduce an adversary's advantage beyond a specific negligible function?

We note that, using a trusted setup that generates a trapdoor permutation (and can therefore invert it), one could heuristically obfuscate Rudich's oracle to obtain an explicit counterexample in the structured reference string model; however, beyond the need for trusted setup, this would induce a strong correlation between different instances. Such a correlation is inherently at odds with the idea of the dream XOR lemma, which conjectures hardness of *completely independent and uncorrelated instances*.

1.0.0.1 Are There Simple Heuristic Counterexamples?

To our knowledge, prior to our work, there were no candidate counterexamples to the dream XOR lemma *under any assumption*. In other words, there was not even a *heuristic* construction of an explicit predicate that can be plausibly conjectured to violate the dream XOR lemma. The difficulty of coming up with such heuristic counterexamples is arguably why the conjecture was put forward in the first place.

1.1 Overview of Results

We give two kinds of explicit counterexamples to the dream XOR lemma.

Our first counterexample develops a general framework that allows us to embed an instance of multiparty computation with *resettable* security into a non-interactive cryptosystem, such as a one-way function. It relies on ideal obfuscation, or alternatively, virtual-black-box (VBB)

obfuscation for some specific class of circuits, along with other standard hardness assumptions. Using this framework, the counterexample to the dream XOR lemma is extremely simple. We believe this framework may be of broader interest and may potentially be useful in designing counterexamples to other conjectures in cryptography. Along the way, we also develop the first resettably-secure multiparty computation protocol for general functionalities in the plain model with super-polynomial simulation, under standard assumptions. This protocol requires three rounds and assumes the existence of two-round sub-exponentially secure statistically sender-private OT. This result is of independent interest and achieves general feasibility for resettably-secure multiparty computation in the plain model; we evade prior impossibility results [47] by aiming for super-polynomial simulation.

Our second counterexample is a tailor-made construction whose sole purpose is to defy the dream XOR lemma. However, it avoids the need for ideal or VBB obfuscation. Instead, we can rely on a concrete obfuscation security property called *public-coins differing inputs obfuscation (PCdiO)* [12, 5, 22, 56], along with a hash function security property called *extended second-preimage resistance* (ESPR) [34] and injective one-way functions. The construction uses PCdiO to "upgrade" the counterexample of [34] for the dream direct-product lemma based on ESPR, into a counterexample for the dream XOR lemma.

PcdiO is a clean assumption which, as of today, all existing iO constructions can be conjectured to satisfy. While diO (without the public coin restriction) is known to be implausible [39], all known implausibility results crucially rely on "contrived" auxiliary information. All such negative results therefore do not generalize to PcdiO (we refer the reader to [56] for further discussion of PcdiO). Indeed, given recent progress on constructing iO (e.g. [58]), it is plausible that the PcdiO, that we rely on, may be reduced to well-studied assumptions.

We stress that even with ideal obfuscation, Rudich's oracle does not directly give rise to such counterexamples because it is not efficient. We believe that our techniques for obfuscation-based counterexamples will find other applications.

1.1.0.1 Are the Counterexamples Explicit?

Both of our counterexamples can be made fully explicit by making the same kind of leap of faith one makes when instantiating standard idealized models in cryptography (such as the random oracle model [17] or the generic group model [72]). For the first counterexample, one can use any existing iO construction (such as the one from [58]) as a heuristic substitute for special-purpose obfuscation. A similar heuristic has been suggested in prior works (see, e.g., [39, 42]). For the second, use iO construction as the PC-diO and use SHA-3 as the ESPR. Either way, one gets fully explicit constructions of one-way functions and hard-core bits for which the XOR lemma *provably* does not amplify hardness. Assuming that the function is actually one-way to begin with relies on a strong but explicit assumption. Nevertheless, if one wanted to prove that the dream XOR lemma holds, one would now have to show an attack against the one-wayness of these explicit one-way function candidates. This is very different from the previous oracle-based separations or (generalized) black-box impossibility results, which could be potentially circumvented by finding a novel non-black-box proof technique. We refer the reader to the end of Section 1.2 for additional discussions about the special-purpose obfuscation assumption.

1.2 First Counterexample

A New Paradigm. We suggest a new paradigm for obtaining counterexamples to parallel repetition. We use this paradigm to obtain an explicit counterexample to the dream version of the XOR lemma. Our paradigm can be thought of as implementing Rudich's oracle in a distributed manner between completely independent instances of a one-way function. Such distributed protocols are often cryptographically achievable via secure multi-party computation (MPC). Specifically, one could treat each instance of a one-way function as a participant in an MPC protocol, and implement an ideal functionality that with probability $\mu(\lambda)$, outputs the inverse of all the instances of the one-way function. Clearly, this would allow the XOR of the hardcore bits of individual instances of the one-way function to be predicted with an advantage of at least $\mu(\lambda)$. Indeed, a similar idea was employed by [34] in the context of a counterexample for the direct-product hardness amplification of signature schemes, by having the attacker leverage interaction with the signing oracle to run the protocol.

But in our case there is a crucial type mismatch: we would like to build one-way functions, an inherently non-interactive primitive, by relying on secure MPC, which is an inherently interactive protocol. We resolve this mismatch by relying on obfuscation to *non-interactively implement* the next message function of each party in a secure MPC protocol[1]. Specifically, the output of our one-way function consists of an obfuscation of the next-message function for the appropriate MPC.

When sufficiently many one-way functions are combined, an adversary can execute an MPC protocol between them by appropriately querying their next message functions, as a result, inverting all one-way functions simultaneously with probability $\mu(\lambda)$.

But obfuscating the next message function in this manner exposes individual one-way functions (i.e., "participants" in the MPC protocol) to a new threat model. An adversary can query an obfuscated program repeatedly and in an arbitrary order, amounting to what are called "resetting" attacks [27]. This requires us to confront the need for MPC protocols that are secure against such strong resetting attacks.[2]

Resettable MPC. The question of whether secure MPC can be achieved in a setting where participants can be simultaneously reset has previously been studied by [48, 47] and for the specific setting of zero knowledge in [27, 11, 20, 29, 28, 21, 30]. In particular, the work of [48] considered resetting attacks on only one party and obtained positive results. In the general setting where more than one party can be reset, [47] provided negative results for certain functionalities thereby ruling out a general purpose protocol for all functionalities. In addition, they obtained positive results for a limited class of entropic functionalities.

We observe that this negative result can be side-stepped by allowing our MPC simulator to run in super-polynomial time. Technically, we build on the recent concurrent secure MPC protocols with super-polynomial simulation in [10], which are themselves based on the notion of super-polynomial strong simulation [60]. MPC security with super-polynomial simulation [66, 68] turns out to be sufficient for many scenarios, including for building our counterexamples.

[1] This approach has previously been studied in multiple other contexts [38, 33, 7]. However, in all those cases, the inputs to the MPC are fixed apriori and (implicitly) hardwired into the obfuscated programs. Looking ahead, in our protocol, the inputs cannot be fixed apriori and we resort to using resettable MPC to overcome this crucial issue. Note that this issue is also the reason why we do not know how to use iO to implement our obfuscated next-message function, whereas the earlier works mentioned above were able to use only iO.

[2] A similar issue also came up in [34] when the MPC was embedded in a signing oracle, which is interactive but stateless. It was resolved similarly by relying on some form of resettable MPC.

As a contribution of independent interest, we obtain the first resettable MPC protocol for general functionalities, admitting super-polynomial simulation. Our protocol requires only three rounds of interaction, and assumes the existence of two-round sub-exponentially receiver-private and statistically sender-private OT, which can in turn be based on a variety of standard assumptions including the (sub-exponential) hardness of DDH, LWE, QRA, or DCRA [65, 3, 51, 9, 23, 35]. We state this result in the form of the following informal theorem.

▶ **Theorem 1** (Informal). *Assuming the existence of sub-exponentially receiver-private and statistically sender-private two-round OT, there exists a three-round resettably-secure MPC protocol for general functionalities, admitting a super-polynomial simulator.*

We stress that our resettable MPC protocol does not assume any form of obfuscation.

Our Counterexample to the Dream XOR Lemma. Armed with resettable MPC, we show that one-way functions with hard-core predicates to which the Dream XOR lemma provably does not apply can be obtained in the ideal obfuscation model, as follows: on input $x = (x_1||x_2) \in \{0,1\}^\lambda$, the one-way function f simply outputs an injective one-way function g applied to x_1, the first $\lambda/2$ bits of x, and uses the remaining $\lambda/2$ bits to obfuscate the next-message function of a participant in an MPC protocol. The ideal functionality for this MPC protocol obtains inputs (i.e., the x_1 values) from several participants, and with probability exactly $\mu(\lambda)$, outputs all these x_1 values in the clear. The hardcore bit of f on input $x = (x_1||x_2)$ is defined as the hardcore bit of g on input x_1.

We rely on security of the obfuscation scheme, the resettable MPC protocol and the one-way function g to argue that it is hard to recover x_1 (or predict the hard-core bit) of a single instance of this one-way function with probability significantly larger than $\mu(\lambda) \cdot \mathsf{poly}(\lambda) + \mathsf{negl}(\lambda)$. On the other hand, no matter how many times we repeat in parallel, the ability to execute a co-ordinated MPC program between all instances of the one-way function gives rise to an adversarial strategy that efficiently recovers all x_1 values, and therefore hardcore bits from all parallel instances, with probability at least $\mu(\lambda)$. For $\mu(\lambda) = 2^{-\sqrt{\lambda}}$, we prove the following informal theorem.

▶ **Theorem 2** (Informal). *Assuming resettably-secure MPC with super-polynomial simulation for general functionalities, for target negligible function $\mu(\lambda) = 2^{-\sqrt{\lambda}}$, there exists an explicit counterexample to the dream XOR lemma in the ideal obfuscation model. Furthermore, such a counterexample exists in the plain model under a plausible special-purpose obfuscation assumption.*

We choose to set $\mu(\lambda) = 2^{-\sqrt{\lambda}}$ primarily for the sake of simplicity in exposition, but our technique also generalizes to rule out arbitrary negligible $\mu(\lambda)$. While ideal obfuscation does not exist in the plain model [50, 12], the theorem applies relative to any world in which ideal obfuscation exists. This can refer to any *oracle* (in the complexity-theoretic sense) that enables ideal obfuscation, or given the ability to obfuscate functions using ideal trusted hardware. This counterexample is meaningful even when given ideal obfuscation, because all algorithms are given free access to the obfuscated function, and moreover the model does not introduce any shared randomness; as a result, instances of our one-way function remain *truly independent*.

Replacing Ideal Obfuscation with a Concrete Obfuscation Conjecture. In fact we go one step further and we postulate that a very specific functionality can be obfuscated, in a *virtual black-box* [12] (VBB) manner, *without auxiliary input*. As a result, assuming VBB or special-purpose obfuscation without auxiliary input for a specific class of circuits, we obtain counterexamples to the XOR lemma in the plain model.

How meaningful or plausible is this concrete assumption? While there are known impossibility results for VBB obfuscation of several functionalities, including PRFs *in the presence of auxiliary input*, the only known meaningful negative results on VBB without auxiliary input are the highly contrived "self-eating" programs developed by Barak et al. [12].

Despite it being plausible to embed the circuit family of [12] into *some* specific instantiations of resettable MPC and signatures, it appears extremely unlikely that *every* instantation of resettable MPC and signatures will have a [12]-style counterexample embedded into it. All we need for our counterexample is the existence of *one* VBB-obfuscatable family, which is compatible with all known evidence regarding VBB obfuscation. We stress again that our VBB assumption does not require security with respect to any auxiliary information. Indeed, such special-purpose obfuscation assumptions were used by Garg et al. [39] to prove negative results for differing-inputs obfuscation, and to this date, there are no known refutations of these types of conjectured assumptions for non-contrived circuits without auxiliary input.

Finally, we note that the recent work of [58] has shown how to construct indistinguishability obfuscation (iO) from standard assumptions. In addition, several other constructions of iO from new assumptions that look plausible and are quite simple to state have appeared [1, 57, 6, 2, 40, 41, 24, 25, 74, 31]. While we do not know how to use indistinguishability obfuscation to achieve our result, this recent progress suggests that perhaps achieving VBB obfuscation for circuit families such as ours may also be possible from standard assumptions. Our work offers further motivation for this important line of study.

1.3 Second Counterexample

Our second counterexample begins with the work of [34], which constructs a counterexample to a dream version of *direct-product* hardness amplification. In particular, they construct a hard relation R such that, given a uniformly random instance $\widetilde{x}$, no polynomial time adversary can find a witness w such that $(\widetilde{x}, w) \in R$ except with negligible probability. However, given t independent copies $\widetilde{x}_1, \ldots, \widetilde{x}_t$, the adversarial advantage of finding all t witnesses w_i such that $(\widetilde{x}_i, w_i) \in R$ does not decrease much as t gets large. Concretely, there is a polynomial time adversary that can find all t witnesses with probability (say) $2^{-\sqrt{\lambda}}$, no matter how large t is. This counterexample is based on a non-standard hash function security property called extended second-preimage resistance (ESPR), which is weaker than collision resistance, but is assumed to hold for a fixed (un-keyed) hash function against non-uniform attackers. The work of [34] justifies this assumption by showing that ESPR security holds in the random-oracle model with auxiliary input [73, 32], which models security properties for un-keyed hash functions with respect to non-uniform attackers.

As an initial idea to get a counterexample for the dream XOR lemma, one may hope to simply take a hard-core predicate for the relation R. The main issue is that the witness w for $\widetilde{x}$ is not unique, and so even if we are able to find *some* witness for $\widetilde{x}$, it does not mean we can compute *the* correct hard-core predicate. We resolve this by relying on an injective one way function $\hat{f}$ and a public-coins differing-inputs obfuscation (PCdiO) [5, 22, 56]. PCdiO is a strengthening of indistinguishability obfuscation (iO). All current constructions of iO can be conjectured to also satisfy PCdiO security, although no proofs of security for achieving PCdiO are as-yet known.

For the counterexample, we define a one-way function f that gets as input $x = (\hat{x}, \tilde{x}, r)$ and outputs $y = (\hat{y} = \hat{f}(\hat{x}), \tilde{x}, \tilde{C})$ where $\tilde{C}$ is an obfuscated circuit that takes as input a witness w, and if $(\tilde{x}, w) \in R$ it outputs $\hat{x}$, else $\perp$; we use r as the randomness for the obfuscation. We show that the function f is one-way and moreover, given the output y, it is hard to find $\hat{x}$. Intuitively, this follows because it is hard to find a valid witness w for $\tilde{x}$ that will make the obfuscated circuit output anything useful, and therefore the obfuscated circuit is indistinguishable from one that does not contain $\hat{x}$ and always outputs $\perp$; on the other hand the one-wayness of $\hat{f}$ says that it is hard to compute $\hat{x}$ from $\hat{y}$. We define a predicate $P(x)$ to be Goldreich-Levin hardcore bit of $\hat{x}$, and the above shows that $P(x)$ is a hard-core predicate of $f(x)$. On the other hand, it is easy to see that the dream XOR lemma does not hold for f, P. Given many values $y_i = f(x_i)$ for $i = 1, \ldots, t$, we can find all the witnesses $w_1, \ldots, w_t$ for $\tilde{x}_1, \ldots, \tilde{x}_t$ with probability $2^{-\sqrt{\lambda}}$. We then input these witnesses w_i to the respective obfuscated programs contained in y_i to recover $\hat{x}_i$, which allows us to recover all the hardcore-predicates $P(x_i)$ and therefore also $\bigoplus P(x_i)$.

We obtain the following informal theorem.

▶ **Theorem 3** (Informal). *Assuming the existence of public-coins differing-inputs obfuscation (PCdiO), extended second-preimage resistant (ESPR) hash functions, and injective one-way functions there exists an explicit counterexample to the dream XOR lemma.*

We observe that we do not actually even need PCdiO for the above counterexample, and (public-coins) extractable witness encryption suffices; instead of obfuscating the circuit that takes as input a witness w, and if $(\tilde{x}, w) \in R$ it outputs $\hat{x}$, we use witness encryption to encrypt the message $\hat{x}$ with respect to the statement $\tilde{x}$ for the relation R.

1.4 Counterexamples for the Goldreich-Levin Predicate

We note that, in both our counterexamples, only the choice of the one-way function is "artificial", but the hardcore predicate is just the Goldreich-Levin (GL) predicate, albeit only applied to one of the components of the input, rather than the entire input as a whole. One may ask whether it is possible to get a counterexample where the hardcore predicate is GL applied to the entire input, or whether there is hope that the dream XOR lemma would hold in this case. Although we do not know how to get a counterexample for the GL predicate applied to the pre-image of a one-way function, we can get a counterexample if we generalize to *one-way puzzles* and consider the GL predicate applied to the solution of such a puzzle. In a one-way puzzle, there is a randomized algorithm that generates random hard puzzles together with a solution that can be verified in polynomial time. Security says that no polynomial time attacker can solve a random hard puzzle with better than negligible probability. In this case, we can take the one-way functions from our counterexamples and define a hard puzzle consisting of the one-way function output, while the solution is just the small component of the one-way function's input that we apply GL to. In both examples, we can efficiently verify the solution. To summarize, the above shows that the dream XOR lemma fails, even when restricted to the specific GL predicate, at least when applied to general one-way puzzles.

1.5 Related Work

We note that there is much work in cryptography on designing counter-examples to statements that "should intuitively hold" but don't. Even when it becomes clear that a proof for such statements is lacking, a counter-example provides some tangible understanding of how things

can go wrong. Some such works that provide interesting counterexamples include: parallel repetition of multi-player games [37, 36, 54] and cryptographic protocols [15, 67], hardness amplification [34], circular security of encryption schemes [69, 61, 4, 62, 46, 75, 45], selective opening attack security of encryption and commitments [14, 53, 52], leakage amplification via parallel repetition [64, 59, 34], hardness of prediction via obfuscation [26]. Such counter-examples can point us to refined versions of the statement that may potentially still hold. Furthermore, exactly because such counter-examples "defy intuition", they often capture interesting ideas and techniques that turn out to be of greater value down the line. For example, the techniques developed in the context of circular security counter-examples [46] lead to positive results on obfuscation from LWE [75, 45].

2 Detailed Technical Overview for First Counterexample

As discussed in the introduction, we obtain our counterexample by implementing a variant of Rudich's oracle in a completely decentralized manner, without introducing any correlations between instances of our counterexample. In this section, we outline our construction and give an overview of our proof technique.

A central cryptographic primitive that enables decentralized computation is secure multi-party computation (MPC). MPC enables several mutually distrusting participants to jointly compute a function f of their private inputs while only revealing the output y of f applied to their joint inputs, and revealing no information to each player beyond their own input and the output y.

2.1 XOR lemma counterexample

We design a one-way function $\mathcal{G}$ as our counterexample to the XOR lemma. $\mathcal{G}$ on input uniform randomness $x = (\alpha||\beta)$, outputs $f(\alpha)$ for an injective one-way function f, and uses randomness β to build a "proxy" that participates in an MPC protocol. This proxy is simply the next-message function of a participant in the appropriate MPC protocol. We define a hardcore predicate for $\mathcal{G}$ on input $x = (\alpha||\beta)$ as the output of a hardcore predicate for $f(\alpha)$.

The MPC protocol will allow multiple such proxies to jointly emulate a randomized ideal functionality that obtains the value α as input (from each participating proxy), and with probability $\mu(\lambda)$ for some fixed negligible function $\mu(\cdot)$, outputs all the α values it obtained as input from all proxies, and otherwise outputs $\perp$. If we are able to implement such an MPC protocol, it is clear that no matter how many times we repeat, an adversary would be able to run the MPC protocol between all instances of the one-way function and obtain the α values, and therefore the hardcore predicates of all instances, simultaneously, with probability at least $\mu(\lambda)$.

But in order for this to be a valid counterexample, we also need to ensure that the hard-core predicate for a *single* instance of this one-way function is secure: that is, it cannot be predicted with probability close to 1. In fact, we prove that the hard-core bit cannot be predicted with advantage better than $\mathsf{negl}(\lambda)$. For this, we must ensure that even given a next-message function that has the value α hardwired in it, it is not possible to extract α except with probability $\mathsf{negl}(\lambda)$. As a first step, we *obfuscate* the next-message function instead of releasing it in the clear. Assuming that the next-message circuit can be obfuscated with virtual black-box security, this restricts all information that can be learned from the obfuscated program to information that can be obtained via input-output access alone. Note, however, that we are not done yet, since this still allows an adversary, who has access to the obfuscated circuit, to query the next-message function multiple times on the same partial

transcript, and potentially out of order. We use signatures to fix the latter issue and ensure that the adversary cannot make any meaningful queries on round $(i+1)$ of an MPC execution without querying on round i, for any $i \geq 1$. Unfortunately, this still leaves the obfuscated next-message circuit vulnerable to a "resetting" attack: where an adversary can query such a circuit to obtain multiple executions with the same partial transcript. Therefore, we need to harden our MPC protocol to obtain security against resetting attacks.

Finally, we point out one remaining (subtle) issue. Note that our counterexample is based on VBB-obfuscating the next-message function of a participant in an MPC protocol. In our setting, the adversary – given a single instance of our one-way function – can query this obfuscated program by generating his own next-message functions for imaginary MPC participants. Importantly, the adversary gets to *choose* the identity of these participants. But giving the adversary the freedom to choose the identity of a participant enables it to invoke the same participant (with the same input and randomness), many times using multiple identities in the same protocol. This could, for instance, allow the adversary to make multiple copies of the honest one-way function, and instantiate n parties, all having the same input and random tape, and potentially use this to manipulate the randomness of ideal functionality. More generally, when running an MPC protocol it is assumed that all players have different identities, and the adversary *does not* have the freedom to manipulate the identities of honest parties (or make multiple copies of any given honest identity).

In our setting, since every party is an obfuscated program, we must ensure that messages generated by each program are properly authenticated *under distinct verification keys*. We therefore add an explicit check to our VBB obfuscated program: the program will check that all verification keys are different and all messages in the input transcript are correctly signed. Ensuring that all verification keys are different guarantees that even in the ideal world, every participant of the MPC protocol has a different identity. Once distinct identities are carefully enforced in this way, the underlying MPC protocol (via underlying tools such as non-malleable commitments) ensures that the secret inputs of players to the MPC protocol are all independent of each other. We describe our formal construction assuming the existence of resettable MPC.

Next, we turn to building MPC secure against resetting attacks. As discussed in the introduction, such resettably-secure MPC protocols are only known for ideal functionalities that satisfy a specific property [47][3]. Since our ideal functionality does not satisfy this property, we develop an appropriate resettable MPC protocol for use in our setting. We relax security to allow superpolynomial simulation, because that suffices for our applications. In what follows, we provide an overview of this protocol.

2.2 Resettable MPC

Our construction of MPC with superpolynomial simulation (SPS), secure against resetting attacks, proceeds in two steps.

- **Resettably-Secure SPS MPC against Semi-malicious adversaries.** As a first step, we build resettably secure MPC against semi-malicious adversaries. A semi-malicious adversary always produces messages in the support of honestly generated messages, and

[3] We point out that [47] give resettably secure protocols with polynomial simulation, but only for (very limited) functionalities that satisfy a special property. Informally, they require that there exist compression and decompression algorithms such that with overwhelming probability, for all possible inputs, the compression algorithm generates a small suggestion string s which helps the decompression algorithm correctly predict the output of the adversary in any given session (given the adversary's input-output pairs in prior sessions and its current input, without knowledge of the honest party's input). We refer the reader to [47] for details and a formal description of this property.

writes the input and randomness used to generate these messages on a special witness tape. We compile a semi-malicious MPC protocol that is *not necessarily secure against resetting attacks* into one that is resettably secure against semi-malicious adversaries. Our compiler simply appends a round to the beginning of the protocol; in this round, each party P_i must commit to its input and to a uniformly sampled PRF key K_i. Next, all participants execute the semi-malicious resettably-insecure MPC protocol, except in every round, each party P_i uses randomness that is generated by evaluating the PRF with key K_i on the transcript so far.

The effect of this is that all protocol messages sent by P_i become a deterministic function of the values committed in the first round and the transcript so far. In fact, *any* protocol transcript generated by any combination of honest and semi-malicious adversarial participants is a deterministic function of the first round.

Therefore, any adversary that resets to a previous point in the protocol after round 1 and behaves semi-maliciously, must send exactly the same messages each time. On the other hand, any adversary that resets to the middle of round 1 sees an entirely new execution of the semi-malicious protocol. As a result, we are able to prove that the resulting resettably-secure semi-malicious MPC protocol admits a polynomial time simulator. Next, we compile to obtain an MPC protocol resettably-secure against fully malicious adversaries.

- **Resettably-Secure SPS MPC against Malicious Adversaries.** Our compiler is identical to the one in [10], which builds three round concurrent MPC with superpolynomial simulation. Here, in addition, we prove resettable security. The only difference is that while the compiler in [10] uses as subroutine an underlying stand-alone secure MPC against semi-malicious adversaries; we instead use a *resettably* secure MPC against semi-malicious adversaries (as described above). As ingredients, beyond the resettable semi-malicious protocol, this compiler relies on two round non-malleable commitments and two round zero knowledge arguments with super-polynomial strong simulation [60]. We note that all of these ingredients are secure (or can be easily modified to be secure) under resetting attacks. Furthermore, following [10], we show that the resulting protocol admits a straight-line superpolynomial time simulator. Resettable security of the resulting protocol against malicious adversaries follows by a careful analysis of this simulator and makes use of the resettable security of all ingredients. We formalize these ideas in the full version of the paper.

3 Paper Organization

We describe the first counterexample, assuming resettable MPC, in Appendix B. We first describe some relevant preliminaries in Appendix A. Due to lack of space, we defer the construction of resettable MPC and the second counterexample to the full version of the paper.

References

1 Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In *EUROCRYPT*, 2019.
2 Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In *EUROCRYPT*, 2020.
3 William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, 2001.

4 Navid Alamati and Chris Peikert. Three's compromised too: Circular insecurity for any cycle length from (ring-)lwe. In *CRYPTO*, 2016.

5 Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *IACR Cryptology ePrint Archive*, 2013. URL: `http://eprint.iacr.org/2013/689`.

6 Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In *CRYPTO*, 2019.

7 Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In *CRYPTO*, 2016.

8 Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT*, 2012.

9 Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In *ASIACRYPT*, 2017.

10 Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent MPC via strong simulation. In *TCC*, 2017.

11 Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-sound zero-knowledge and its applications. *FOCS*, 2001.

12 Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

13 Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *FOCS*, 2005.

14 Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, 2009.

15 Mihir Bellare, Russell Impagliazzo, and Moni Naor. Does parallel repetition lower the error in computationally sound protocols? In *FOCS*, 1997.

16 Mihir Bellare, Thomas Ristenpart, and Stefano Tessaro. Multi-instance security and its application to password-based cryptography. In *CRYPTO*, 2012.

17 Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*, 1993.

18 Eli Biham, Yaron J. Goren, and Yuval Ishai. Basing weak public-key cryptography on strong one-way functions. In *TCC*, 2008.

19 Nir Bitansky and Huijia Lin. One-message zero knowledge and non-malleable commitments. In *TCC*, 2018.

20 Nir Bitansky and Omer Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. In *STOC*, 2013.

21 Nir Bitansky and Omer Paneth. On non-black-box simulation and the impossibility of approximate obfuscation. *SIAM J. Comput.*, 2015.

22 Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73. Springer, Heidelberg, February 2014. `doi:10.1007/978-3-642-54242-8_3`.

23 Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In *TCC 2018*, 2018.

24 Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate io from homomorphic encryption schemes. In *EUROCRYPT*, 2020.

25 Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure LWE suffices. *IACR Cryptol. ePrint Arch.*, 2020. URL: `https://eprint.iacr.org/2020/1024`.

26 Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and uces: The case of computationally unpredictable sources. In *CRYPTO*, 2014.

27 Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC*, 2000.

28 Kai-Min Chung, Rafail Ostrovsky, Rafael Pass, Muthuramakrishnan Venkitasubramaniam, and Ivan Visconti. 4-round resettably-sound zero knowledge. In *TCC*, 2014.

29 Kai-Min Chung, Rafail Ostrovsky, Rafael Pass, and Ivan Visconti. Simultaneous resettability from one-way functions. In *FOCS*, 2013.

30 Kai-Min Chung, Rafael Pass, and Karn Seth. Non-black-box simulation from one-way functions and applications to resettable security. *SIAM J. Comput.*, 2016.

31 Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct LWE sampling, random polynomials, and obfuscation. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 256–287. Springer, 2021. doi:10.1007/978-3-030-90453-1_9.

32 Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. 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 473–495. Springer, Heidelberg, April / May 2017. doi:10.1007/978-3-319-56614-6_16.

33 Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *CRYPTO*, 2016.

34 Yevgeniy Dodis, Abhishek Jain, Tal Moran, and Daniel Wichs. Counterexamples to hardness amplification beyond negligible. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 476–493. Springer, Heidelberg, March 2012. doi:10.1007/978-3-642-28914-9_27.

35 Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26954-8_1.

36 Uriel Feige. On the success probability of the two provers in one-round proof systems. In *Structure in Complexity Theory Conference*. IEEE Computer Society, 1991.

37 Lance Fortnow. The complexity of perfect zero-knowledge. *Advances in Computing Research*, 1989.

38 Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *TCC*, 2014.

39 Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *CRYPTO*, 2014.

40 Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In *EUROCRYPT*, 2021.

41 Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *STOC*, 2021.

42 Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing private RAM computation. In *FOCS*, 2014.

43 Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *STOC*, 1989.

44 Oded Goldreich, Noam Nisan, and Avi Wigderson. *On Yao's XOR-Lemma*, pages 273–301. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-22670-0_23.

45 Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *FOCS*, 2017.

46 Rishab Goyal, Venkata Koppula, and Brent Waters. Separating semantic and circular security for symmetric-key bit encryption from the learning with errors assumption. In *EUROCRYPT*, 2017.

47 Vipul Goyal and Hemanta K. Maji. Stateless cryptographic protocols. In *FOCS*, 2011.

48 Vipul Goyal and Amit Sahai. Resettably secure computation. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 54–71. Springer, Heidelberg, April 2009. doi:10.1007/978-3-642-01001-9_3.

49 Aryeh Grinberg, Ronen Shaltiel, and Emanuele Viola. Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs. In *FOCS*, 2018.

50 Satoshi Hada. Zero-knowledge and code obfuscation. In *ASIACRYPT*, 2000.

51 Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *J. Cryptol.*, 2012.

52 Dennis Hofheinz, Vanishree Rao, and Daniel Wichs. Standard security does not imply indistinguishability under selective opening. In *TCC*, 2016.

53 Dennis Hofheinz and Andy Rupp. Standard versus selective opening security: Separation and equivalence results. In *TCC*, 2014.

54 Justin Holmgren and Lisa Yang. The parallel repetition of non-signaling games: counterexamples and dichotomy. In Moses Charikar and Edith Cohen, editors, *51st Annual ACM Symposium on Theory of Computing*, pages 185–192. ACM Press, June 2019. doi:10.1145/3313276.3316367.

55 Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *FOCS*, 1995.

56 Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 668–697. Springer, Heidelberg, March 2015. doi:10.1007/978-3-662-46497-7_26.

57 Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials over r to build io. In *EUROCRYPT*, 2019.

58 Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, 2021.

59 Abhishek Jain and Krzysztof Pietrzak. Parallel repetition for leakage resilience amplification revisited. In *TCC*, 2011.

60 Dakshita Khurana and Amit Sahai. How to achieve non-malleability in one or two rounds. In *FOCS*, 2017.

61 Venkata Koppula, Kim Ramchen, and Brent Waters. Separations in circular security for arbitrary length key cycles. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 378–400. Springer, Heidelberg, March 2015. doi:10.1007/978-3-662-46497-7_15.

62 Venkata Koppula and Brent Waters. Circular security separations for arbitrary length cycles from LWE. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 681–700. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53008-5_24.

63 Leonid A. Levin. One-way functions and pseudorandom generators. In *STOC*, 1985.

64 Allison B. Lewko and Brent Waters. On the insecurity of parallel repetition for leakage resilience. In *51st Annual Symposium on Foundations of Computer Science*, pages 521–530. IEEE Computer Society Press, October 2010. doi:10.1109/FOCS.2010.57.

65 Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, 2001.

66 Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, 2003.

67 Krzysztof Pietrzak and Douglas Wikström. Parallel repetition of computationally sound protocols revisited. In *TCC*, 2007.

68 Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In László Babai, editor, *STOC*, 2004.

69 Ron Rothblum. On the circular security of bit-encryption. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 579–598. Springer, Heidelberg, March 2013. doi:10.1007/978-3-642-36594-2_32.

70 Ronen Shaltiel. Is it possible to improve yao's XOR lemma using reductions that exploit the efficiency of their oracle? In *APPROX/RANDOM*, 2020.

71 Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 2010.

72 Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, 1997.

73 Dominique Unruh. Random oracles and auxiliary input. In *CRYPTO*, 2007.

74 Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. In *EUROCRYPT*, 2021.

75 Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 600–611. IEEE Computer Society Press, October 2017. doi:10.1109/FOCS.2017.61.

76 Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, 1982.

A Preliminaries

Let λ denote the security parameter.

A.1 Virtual Black Box Obfuscation

We recall the definition of Virtual Black-Box (VBB) obfuscation from Barak et al.[12]. In this definition, we don't allow any auxiliary inputs (therefore making our assumptions weaker).

▶ **Definition 4** (Virtual Black-Box Obfuscation). *For any polynomial $t(\cdot)$, circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, a uniform PPT oracle machine* Obf *is a "Virtual Black-Box" Obfuscator for $\mathcal{C}$ if the following conditions are satisfied:*

- *Functionality: For every $\lambda \in \mathbb{N}$ and every input x:*

$$\Pr[(\mathsf{Obf}(C))(x) \neq C(x)] \leq \mathsf{negl}(|C|),$$

  where the probability is over the coins of $C \leftarrow \mathcal{C}_\lambda$.
- *Polynomial Slowdown: There exists a polynomial $p(\cdot)$ such that for every $\lambda \in \mathbb{N}$ and every $C \in \mathcal{C}_\lambda$, $|\mathsf{Obf}(C)| \leq p(|C|)$.*
- *Virtual Black-Box: For every PPT adversary $\mathcal{A}$ there exists a PPT simulator* $\mathsf{Obf.Sim}$*, and a negligible function μ such that for every $\ell \in \mathbb{N}$ and every $C \in \mathcal{C}_\ell$:*

$$\left|\Pr\left[\mathcal{A}\left(\mathsf{Obf}(C)\right) = 1\right] - \Pr\left[\mathsf{Obf.Sim}^C\left(1^{|C|}\right) = 1\right]\right| \leq \mu(|C|) \ ,$$

  where the probabilities are over the coins of $\mathsf{Obf.Sim}$ *and* Obf*.*

A.2 Resettable MPC

We define the notion of Multiparty Computation (MPC) that is resettably secure similar to the definition in Goyal and Maji [47]. The difference in that, in our setting, we consider a simulator that can run in super-polynomial time [66, 13].

Remark. We note that security holds only when the identities of all the parties in the MPC protocol are distinct. Consider n parties $P_1, \ldots, P_n$. An n-party functionality f is a (possibly randomized) mapping of n inputs to n outputs. A secure multi-party computation protocol π with security parameter λ for computing a functionality f is a protocol running in time $\mathsf{poly}(\lambda)$ with the goal of computing the output $f(x_1, \ldots, x_n)$.

The security of a protocol π (with respect to a functionality f) is defined by comparing the real-world execution of the protocol with an ideal-world evaluation of f by a trusted party. More concretely, it is required that for every adversary $\mathcal{A}$, which attacks the real execution of the protocol, there exist an adversary Sim, also referred to as a simulator, which can *achieve the same effect* in the ideal-world. We denote the set of all inputs by $\vec{x} = (x_1, \ldots, x_n)$. The adversary controls a set of parties and at any point during the execution of the protocol it can reset any of the honest parties. We shall consider computational security against parties which have been statically corrupted by the adversary. All honest parties have their random tape independently chosen but when an adversary resets a party, it reuses the same random tape (and the same input).

A.2.0.1 The real execution

In the real execution of the n-party protocol π for computing f in the presence of an adversary $\mathcal{A}$, the honest parties follow the instructions of π. The adversary $\mathcal{A}$, on input λ, outputs the identities of the honest parties, the indices I and identities in 2^λ of corrupted parties, the inputs of the corrupted parties, and an auxiliary input z. $\mathcal{A}$ sends all messages in place of corrupted parties and may follow an arbitrary polynomial-time strategy. The adversary can reset any honest party at any point of time during the execution of the protocol (and potentially even bring them out of sync). Recall that when an adversary resets a party, the reset party reuses the same random tape (and the same input).
The interaction of $\mathcal{A}$ with protocol π defines a random variable $\mathsf{REAL}_{\pi,\mathcal{A}(z),I}(\lambda, \vec{x})$ whose value is determined by the coin tosses of the adversary and the honest players. This random variable contains the output of the adversary (which may be an arbitrary function of its view) as well as the outputs of the uncorrupted parties at the end of the protocol. We let $\mathsf{REAL}_{\pi,\mathcal{A}(z),I}$ denote the distribution ensemble $\{\mathsf{REAL}_{\pi,\mathcal{A}(z),I}(\lambda, \vec{x})\}_{\lambda\in\mathsf{N},\langle\vec{x},z\rangle\in\{0,1\}^*}$.

A.2.0.2 The ideal execution – security with abort

In the ideal world, there is a mutually trusted party which can aggregate the inputs provided to it by the various parties, perform the computation $f(\cdot)$ on their behalf and provide them their respective outputs. An ideal execution for a function f in the presence of an ideal-world adversary (simulator) Sim proceeds as follows:

- **Send inputs to the trusted party:** Honest parties send their inputs to the trusted party; but corrupted parties may decide to send modified inputs to the trusted party, as instructed by Sim. Let x'_i denote the value sent by P_i.
- **Trusted party sends output to the adversary:** The trusted party computes $f(x'_1, \ldots, x'_n) = (y_1, \ldots, y_n)$ and sends $\{y_i\}_{i\in I}$ to the adversary.
- **Adversary instructs trusted party to abort or continue:** This is formalized by having the adversary send either a continue or abort message to the trusted party. In the latter case, the trusted party sends to each uncorrupted party P_i its output value y_i. In the former case, the trusted party sends the special symbol $\perp$ to each uncorrupted party.

- **Resets:** The adversary can reset the ideal world at any point of time. When the adversary decides to reset the ideal world, it requests the trusted party to reset all honest parties and the trusted party sends a reset signal to all honest parties. At this point, the ideal world returns to the first stage where honest parties feed their inputs to the ideal functionality. The honest party inputs do not change between resets.
- **Outputs:** Sim outputs an arbitrary function of its view, and the honest parties output the values obtained from the trusted party.

Sim's interaction with the trusted party defines a random variable $\mathsf{IDEAL}_{f_\perp,\mathcal{A}(z),I}(\lambda,\vec{x})$ that denotes the distribution ensemble $\{\mathsf{IDEAL}_{f_\perp,\mathsf{Sim}(z),I}(\lambda,\vec{x})\}_{\lambda\in\mathsf{N},\langle\vec{x},z\rangle\in\{0,1\}^*}$ where the subscript "$\perp$" indicates that the adversary can abort computation of f. Having defined the real and the ideal worlds, we now proceed to define our notion of security.

▶ **Definition 5** (Resettable MPC with Straight-Line, Black-Box Superpolynomial Simulation)**.** *Let λ be the security parameter. Let f be an n-party randomized functionality, and π be an n-party protocol for $n \in \mathsf{N}$.*

We say that π computes f with resettable straight-line, black-box superpolynomial simulation in the presence of malicious adversaries if for every PPT adversary $\mathcal{A}$ there exists a super-polynomial time simulator Sim that interacts with the adversary via straight-line black-box queries, such that for any $I \subset [n]$:

$$|\Pr[\mathsf{REAL}_{\pi,\mathcal{A}(z),I}(\lambda,\vec{x})=1] - \Pr[\mathsf{IDEAL}_{f_\perp,\mathsf{Sim}(z),I}(\lambda,\vec{x})=1]| = \mathsf{negl}(\lambda)$$

where $\vec{x} = \{x_i\}_{i\in[n]} \in \{0,1\}^*$ and $z \in \{0,1\}^*$.

The simulator Sim is allowed to reset the functionality in the ideal world several times and the number of times the ideal functionality is reset by the simulator could possibly be more than the number of resets performed by the adversary in the real world, though this number must be a polynomial in the security parameter.

A.3 Security Against Semi-Malicious Adversaries

We take this definition verbatim from [8]. A semi-malicious adversary is modeled as an interactive Turing machine (ITM) which, in addition to the standard tapes, has a special witness tape. In each round of the protocol, whenever the adversary produces a new protocol message msg on behalf of some party P_k, it must also write to its special witness tape some pair (x, r) of input x and randomness r that explains its behavior. More specifically, all of the protocol messages sent by the adversary on behalf of P_k up to that point, including the new message m, must exactly match the honest protocol specification for P_k when executed with input x and randomness r.

Also, we assume that the attacker is rushing and hence may choose the message m and the witness (x, r) in each round adaptively, after seeing the protocol messages of the honest parties in that round (and all prior rounds). The adversary may also choose to abort the execution on behalf of P_k in any step of the interaction.

B Counterexample via VBB Obfuscation and Resettable MPC

We start this section by defining hard-core predicates. Next, we state the dream version of Yao's XOR lemma, and then we describe our counterexample.

▶ **Definition 6** ($\epsilon(\lambda)$-Hard Core Predicate.). *Let λ denote a security parameter and $m = m(\lambda), n = n(\lambda)$ be polynomials in λ. An $\epsilon(\lambda)$-hard core predicate of a one-way function $f : \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$ is a predicate $P : \{0,1\}^{m(\lambda)} \to \{0,1\}$ such that for every polynomial-time non-uniform $\mathcal{A}$ and every λ,*

$$\Pr_{x \leftarrow \{0,1\}^{\lambda}}[\mathcal{A}(1^{\lambda}, f(x)) = P(x)] \leq \frac{1}{2} + \epsilon(\lambda)$$

▶ **Lemma 7** (Hard Core Predicate for a One-Way Function). *[43] If f is a one-way function, then $h(x,r) = \langle x, r \rangle (\mathsf{modulo}\ 2)$ is an $\epsilon(\lambda)$-hard core predicate, according to Definition 6, for the one-way function f' defined by $f'(x,r) = (f(x), r)$, for some $\epsilon(\lambda) = \mathsf{negl}(\lambda)$.*

▶ **Conjecture 8** (Dream version of Yao's XOR Lemma). *[18] Fix $\mu(\lambda) = 2^{-\sqrt{\lambda}}$. Let f denote a one-way function and P denote an $\epsilon(\lambda)$-hard core predicate according to Definition 6. Then for any $t = \mathsf{poly}(\lambda)$, $P^{(t)}(x_1, \ldots, x_t) = \bigoplus_{i \in [t]} P(x_i)$ is an $\epsilon'(\lambda)$-hard core predicate for $f'(x_1||x_2|| \ldots ||x_t) \triangleq f(x_1)||f(x_2)|| \ldots ||f(x_t)$, such that $\epsilon'(\lambda) \leq \epsilon(\lambda)^{t(\lambda)} + \mu(\lambda)$.*

We note that this conjecture is a special case of (and is therefore implied by) the dream conjecture first formulated in [44]. Also, we fixed $\mu(\lambda)$ to be an arbitrary negligible function in λ, specifically, we set it to $2^{-\sqrt{\lambda}}$ for ease of exposition. However, we note that our refutation of this conjecture can be generalized to refute arbitrary negligible functions $\mu(\cdot)$ by setting the parameters of our one-way function accordingly. That is, our proof can be generalized to show that for every negligible function $\nu(\cdot)$, there exists a one-way function that refutes Conjecture 8 with $\mu(\cdot) = \nu(\cdot)$. We will now construct a one-way function for which Conjecture 8 does not hold.

B.1 Construction

We define a one-way function $\mathcal{G} : \{0,1\}^{2\lambda} \to \{0,1\}^{p'(\lambda)}$, where $p'(\cdot)$ is a polynomial in λ, the exact value of which will be determined later.

B.1.0.1 Notation and Primitives Used

- Let $g : \{0,1\}^{\lambda} \to \{0,1\}^{p(\lambda)}$ be any injective one way function and h denote the Golreich-Levin [43] hardcore bit for this one way function.
- Let π^{Res} denote any resettably secure MPC protocol with superpolynomial simulation. Let $(\pi^{\mathsf{Res}}.\mathsf{NextMsg}_1, \pi^{\mathsf{Res}}.\mathsf{NextMsg}_2, \ldots, \pi^{\mathsf{Res}}.\mathsf{NextMsg}_{\mathsf{n}})$ denote the algorithms used by each party to compute the messages in each of the rounds and $\pi^{\mathsf{Res}}.\mathsf{OUT}$ denote the algorithm used by each party to compute its final output. Also, let Trans_i denote all messages sent in an execution of π^{Res} up to round i. Let $\mathsf{Res.Sim}$ denote the straight-line (super-polynomial) simulator for this protocol. Let $(\mathsf{Res.Sim.NextMsg}_1, \ldots, \mathsf{Res.Sim.NextMsg}_{\mathsf{n}})$ denote the algorithms used by the simulator to compute the messages in each of the rounds and $\mathsf{Res.Sim.Out}$ denote the algorithm used to compute the final output on behalf of the honest parties. Let $\ell(\lambda)$ denote the length of the randomness required by each party on inputs of length λ. Let $s(\lambda)$ denote the maximum size of the circuit representation of $(\pi^{\mathsf{Res}}.\mathsf{NextMsg}_1, \pi^{\mathsf{Res}}.\mathsf{NextMsg}_2, \ldots, \pi^{\mathsf{Res}}.\mathsf{NextMsg}_{\mathsf{n}}, \pi^{\mathsf{Res}}.\mathsf{OUT})$ in this protocol.
- Let $(\mathsf{Obf}, \mathsf{Eval})$ be a VBB obfuscation scheme and $\mathsf{Obf.Sim}$ denote its simulator. Let $r(\lambda)$ denote the length of the randomness used to obfuscate programs of size $s(\lambda)$, and $s'(\lambda)$ denote the size of the obfuscated program.
- Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a sub-exponentially unforgeable signature scheme.
- Let $\mathsf{PRG} : \{0,1\}^{\lambda} \to \{0,1\}^{\sqrt{\lambda}+\lambda+r(\lambda)+\ell(\lambda)}$ be a pseudorandom generator.

B.1.0.2 Construction

The one-way function $\mathcal{G}$, on input $x = (a, b)$, where $|a| = |b| = \lambda$, is:

- Compute $y = g(a)$.
- Compute $(\alpha, \beta, \gamma, \delta) \leftarrow \mathsf{PRG}(b)$ where α is of length $\sqrt{\lambda}$, β is of length λ, γ is of length $\ell(\lambda)$ and δ is of length $r(\lambda)$. Set $\mathsf{r}^{\mathsf{Res}} = \gamma$.
- Generate $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda; \beta)$.
- Compute $\widehat{\mathsf{C}} = \mathsf{Obf}(1^\lambda, \mathsf{C})$ using randomness δ where program C is described in Figure 1.
- Output $(\widehat{\mathsf{C}})$. We set $p'(\lambda) = |\widehat{\mathsf{C}}|$.

Hardwired values: $(a, y, \alpha, \mathsf{sk}, \mathsf{vk}, \mathsf{r}^{\mathsf{Res}})$

1. Output y. Additionally, do the following.
2. If input = ("Reveal Identity for MPC"): output vk.
3. If input = ("MPC", "Begin", "Party" $\mathsf{j}, \overrightarrow{\mathsf{vk}}$):
   - Output $\perp$ if any of the following happen:
     - $\mathsf{vk}^j \neq \mathsf{vk}$,
     - $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct,
   - Output $\sigma = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||j))$
4. If input = ("MPC", "Round" 1, "Party" $\mathsf{j}, \overrightarrow{\mathsf{vk}}, \sigma$):
   - Output $\perp$ if any of the following happen:
     - $\mathsf{vk}^j \neq \mathsf{vk}$,
     - $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct,
     - $\mathsf{Verify}(\mathsf{vk}, \sigma, (\overrightarrow{\mathsf{vk}}||j)) \neq 1$.
   - Begin resettable MPC protocol π^{Res} amongst λ^2 parties $\mathsf{P}_1, \ldots, \mathsf{P}_{\lambda^2}$ playing the role of P_j with identity vk. Use input α, randomness $\mathsf{r}^{\mathsf{Res}}$ to evaluate functionality f (Figure 2). Let ℓ be the number of rounds of π^{Res}.
   - Compute $\mathsf{msg}_1 = \pi^{\mathsf{Res}}.\mathsf{NextMsg}_1(1^\lambda, \alpha; \mathsf{r}^{\mathsf{Res}})$, $\sigma_1 = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\mathsf{msg}_1||1||j))$.
   - Output $(\mathsf{msg}_1, \sigma_1)$.
5. If input = ("MPC", "Round" i, "Party" $\mathsf{j}, \overrightarrow{\mathsf{vk}}, \overrightarrow{\mathsf{msg}}_{i-1}, \sigma_{i-1}, \mathsf{Trans}_{i-2}$):
   - Let Trans_{i-2} denote the transcript till round $(i-2)$ and $\overrightarrow{\mathsf{msg}}_{i-1}$ denote the set of messages sent by all parties in round $(i-1)$ of protocol π.
   - Output $\perp$ if any of the following happen:
     - $\mathsf{vk}^j \neq \mathsf{vk}$,
     - $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct,
     - $\mathsf{Verify}(\mathsf{vk}, \sigma_{i-1}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-2}||\mathsf{msg}_{i-1}||i-1||j)) \neq 1$.
   - If round number $i \leq \ell$, do:
     - Continue π^{Res}. Compute $\mathsf{msg}_i = \pi^{\mathsf{Res}}.\mathsf{NextMsg}_i(\mathsf{Trans}_{i-2}, \overrightarrow{\mathsf{msg}}_{i-1}, \alpha; \mathsf{r}^{\mathsf{Res}})$.
     - Set $\mathsf{Trans}_{i-1} = (\mathsf{Trans}_{i-2}||\overrightarrow{\mathsf{msg}}_{i-1})$, $\sigma_i = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-1}||\mathsf{msg}_i||\ i||j))$.
     - Output $(\mathsf{msg}_i, \sigma_i)$.
   - If round number $i = (\ell + 1)$, do:
     - Compute $z = \pi^{\mathsf{Res}}.\mathsf{OUT}(\mathsf{Trans}_{\ell-1}, \overrightarrow{\mathsf{msg}}_\ell, \alpha; \mathsf{r}^{\mathsf{Res}})$.
     - If $z = 0$, output a. Else, output $\perp$.

Figure 1 Description of program C.

Input: For each $i \in [\lambda^2]$, party P_i has input α_i of length $\sqrt{\lambda}$. **Output:** If $(\alpha_1 \oplus \ldots \oplus \alpha_{\lambda^2}) = 0^{\sqrt{\lambda}}$, output 0. Otherwise, output $\perp$.

Figure 2 Description of Functionality f.

B.1.0.3 Hardcore Predicate

Recall that the Goldreich-Levin hardcore predicate [43] for the one-way function g is h. We now define the hardcore predicate H for $\mathcal{G}$ as: $\mathsf{H}(x) = h(a)$, where $x = (a, b)$ such that $|a| = |b|$.

B.2 Security

We conjecture the following about the existence of special-purpose obfuscation.

▶ **Conjecture 9** (Special-purpose obfuscation)**.** *There exists a secure signature scheme and a resettable MPC protocol according to Definition 5 for which, for the class of circuits $\mathsf{C} = \{\mathsf{C}_\lambda\}_{\lambda \in \mathbb{N}}$ where for $\lambda \in \mathbb{N}$, C_λ is depicted in Figure 1, where $a \in \{0,1\}^\lambda, y \in \{0,1\}^{p(\lambda)}, \alpha \in \{0,1\}^{\sqrt{\lambda}}, (\mathsf{SK}, \mathsf{VK}) \in \mathsf{Supp}(\mathsf{Gen}(1^\lambda)), \mathsf{r}^{\mathsf{Res}} \in \{0,1\}^{\ell(\lambda)}$, there exists a virtual black-box obfuscator according to Definition 4.*

▶ **Theorem 10.** *Assuming Conjecture 9 and resettably secure MPC with super-polynomial simulation against malicious adversaries according to Definition 5, Conjecture 8 is false. In particular, assuming Conjecture 9, and either sub-exponential DDH or LWE, Conjecture 8 is false.*

This also implies that the above result holds in the ideal obfuscation model. Here we model obfuscation as an ideal functionality with two interfaces. An "Obfuscate" interface takes as input a program P and outputs a handle $\tilde{P}$. The handle can simply be a counter which is incremented on each invocation. The ideal functionality keeps a list of such tuples $(\tilde{P}, P)$. An "Evaluate" interface takes as input a handle $\tilde{P}$ and an input x, and finds the corresponding tuple $(\tilde{P}, P)$ in the list; if such a tuple exists it outputs $P(x)$ else $\perp$. We rely on the fact that Conjecture 9 holds relative to this oracle and that our counter-example uses the obfuscator in a black-box manner. This gives us the following corollary.

▶ **Corollary 11.** *Assuming sub-exponential DDH or LWE, Conjecture 8 is false in the ideal obfuscation model.*

We now describe the proof of this theorem. First, we prove that H defined above is indeed a hardcore predicate for function $\mathcal{G}$. We observe that for our construction of $\mathcal{G}$ and H, this also automatically proves that $\mathcal{G}$ is a one way function. This is for the following reason: suppose for contradiction that $\mathcal{G}$ is not a one way function. Then, there exists a PPT adversary that, given a value $\mathcal{G}(x = (a, b)) = (\widehat{\mathsf{C}})$, can compute an inverse $x' = (a', b')$ with non-negligible probability. However, observe that since g is an injective one way function, $a' = a$. Further, since $\mathsf{H}(x = (a, b)) = h(a)$, $\mathcal{A}$ can compute $\mathsf{H}(x') = \mathsf{H}(x)$. Thus, given just $\mathcal{G}(x)$, $\mathcal{A}$ can compute $\mathsf{H}(x)$ with non-negligible probability which contradicts the fact that H is a hardcore predicate for function $\mathcal{G}$. Therefore, it suffices to formally show that for every PPT adversary $\mathcal{A}$,

$$\mathsf{Pr}[\mathcal{A}(\mathcal{G}(x)) = \mathsf{H}(x)] = \frac{1}{2} + \mathsf{negl}(\lambda)$$

where the probability is over the randomness of x and $\mathcal{A}$. We do this via the following series of computationally indistinguishable hybrids $\mathsf{Hyb}_0, \ldots, \mathsf{Hyb}_6$. We defer the rest of the proof to the full version of the paper.

B.3 Parallel Repetition Attack

We now describe the counterexample to the dream XOR lemma by setting $t = \lambda^2$. That is, we will construct a PPT adversary $\mathcal{A}$ that, given λ^2 samples of the outputs of the one way function $\mathcal{G}$, can compute the XOR of their respective hardcore bits with respect to predicate H with probability greater than or equal to $2^{-\sqrt{\lambda}}$ thus disproving Conjecture 8. Formally, we construct an adversary $\mathcal{A}$ such that : $\mathsf{Pr}[\mathcal{A}(\mathcal{G}(x_1), \ldots, \mathcal{G}(x_{\lambda^2})) = \mathsf{H}(x_1) \oplus \ldots \oplus \mathsf{H}(x_{\lambda^2})] \geq 2^{-\sqrt{\lambda}}$ where the probability is over the random choices of the values $(x_1, \ldots, x_{\lambda^2})$ and the randomness of the adversary.

B.3.0.1 Adversary's Strategy

Adversary $\mathcal{A}$, given $\{\mathcal{G}(x_i) = (y_i, \widehat{\mathsf{C}}_i)\}_{i \in [\lambda^2]}$, does:

1. Run an execution of the resettable MPC protocol π^{Res} amongst (λ^2) parties for functionality f (Figure 2) using obfuscated programs $\widehat{\mathsf{C}}_1, \ldots, \widehat{\mathsf{C}}_{\lambda^2}$. The protocol messages are forwarded appropriately to all the obfuscations.
2. Abort if any obfuscated program outputs $\perp$. Else, let a_i be the value output by $\widehat{\mathsf{C}}_i$ at the end of the protocol. For each $i \in [\lambda^2]$, compute $h(a_i)$.
3. Output $h(a_1) \oplus \ldots \oplus h(a_{\lambda^2})$.

B.3.0.2 Analysis

For each input $x_i = (a_i, b_i)$, recall that α_i is the first $\sqrt{\lambda}$ bits of $\mathsf{PRG}(b_i)$. For randomly chosen inputs $x_1, \ldots, x_{\lambda^2}$, $Pr[(\alpha_1 \oplus \ldots \oplus \alpha_{\lambda^2}) = 0^{\sqrt{\lambda}}] \geq 2^{-\sqrt{\lambda}}$. Therefore, by the correctness of the obfuscation scheme, the resettable MPC protocol π^{Res} and the signature scheme, it is easy to see that for randomly chosen inputs $x_1, \ldots, x_{\lambda^2}$, for every $j \in [\lambda^2]$, the adversary learns the preimage a_i with probability $\geq 2^{-\sqrt{\lambda}}$. Thus, $\mathsf{Pr}[\mathcal{A}(\mathcal{G}(x_1), \ldots, \mathcal{G}(x_{\lambda^2})) = \mathsf{H}(x_1) \oplus \ldots \oplus \mathsf{H}(x_{\lambda^2})] \geq 2^{-\sqrt{\lambda}}$ and this completes the proof.

Revisiting Collision and Local Opening Analysis of ABR Hash

Chandranan Dhar
Indian Statistical Institute, Kolkata, India

Yevgeniy Dodis
New York University, NY, USA

Mridul Nandi
Indian Statistical Institute, Kolkata, India

Abstract

The question of building the most efficient tn-to-n-bit collision-resistant hash function H from a smaller (say, $2n$-to-n-bit) compression function f is one of the fundamental questions in symmetric key cryptography. This question has a rich history, and was open for general t, until a recent breakthrough paper by Andreeva, Bhattacharyya and Roy at Eurocrypt'21, who designed an elegant mode (which we call ABR) achieving roughly $2t/3$ calls to f, which matches the famous Stam's bound from CRYPTO'08. Unfortunately, we have found serious issues in the claims made by the authors. These issues appear quite significant, and range from verifiably false statements to noticeable gaps in the proofs (e.g., omissions of important cases and unjustified bounds).

We were unable to patch up the current proof provided by the authors. Instead, we prove from scratch the security of the ABR construction for the first non-trivial case $t = 11$ (ABR mode of height 3), which was incorrectly handled by the authors. In particular, our result matches Stam's bound for $t = 11$. While the general case is still open, we hope our techniques will prove useful to finally settle the question of the optimal efficiency of hash functions.

2012 ACM Subject Classification Security and privacy → Cryptography

Keywords and phrases ABR hash, collision resistance, local opening

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.11

Funding *Yevgeniy Dodis*: Partially supported by gifts from VMware Labs and Google, and NSF grants 1815546 and 2055578.
Mridul Nandi: Partially supported by the project "Study and Analysis of IoT Security" under Government of India at R.C.Bose Centre for Cryptology and Security, Indian Statistical Institute, Kolkata.

1 Introduction

The *Merkle-Damgård construction* [3, 7] is a sequential construction which is used in MD5, SHA-1 and SHA-2 and many other hash functions. On the other hand, the *Merkle tree* [6] is a parallel construction that is used in hash-based signatures (of interest due to their post-quantum security), version control systems such as git, and cryptocurrencies such as Ethereum. It is well known that the Merkle-Damgård construction and the Merkle tree are collision-resistant provided so are the compression functions. The number of compression function calls is (essentially) the same for both constructions. When we use $2n$-to-n-bit compression functions, we can process t blocks of messages by making t or $(t-1)$ calls to the compression function.

Although both of these widely used constructions are rather efficient, and only rely on the collision-resistance of the compression function, practical compression functions are believed to have more properties than mere collision resistance. As such, it is interesting to study the

question of designing the most efficient way to build a t-to-1 collision-resistant hash function, even if modeling the compression function as ideal (i.e. a random oracle). In particular, to see whether the classical Merkle-Damgård and Merkle tree constructions can be improved under such idealized modeling. This question has received a lot of attention from the cryptography community, which we survey below.

Lower Bound on the Number of Calls. We start with lower bounds (i.e., attacks). In [2], Black et al. formally analyze the security-efficiency trade-off of compression functions, showing that a $2n$-to-n-bit compression function making a single call to a fixed-key n-bit block cipher can not achieve collision resistance. Later Rogaway and Steinberger [9] generalized the result for permutation-based hash. For a general hash function based on a compression function, Stam [11] conjectures a lower bound on the number of compression function calls. In particular, a collision with at most $2^{n(\lambda-(t-0.5)/r)}$ queries on a t-to-1 block hash function can be found after making r calls to λ-to-1 block compression functions. Equivalently, for optimal birthday security, the number of hash calls must be at least $r \geq (2t-1)/(2\lambda-1)$. This bound is popularly known as the Stam's bound. Stam has shown the bound for some cases under a uniformity assumption. Later by Steinberger [12] and by Steinberger, Sun and Yang [13], a formal proof of the Stam's bound is shown.

Hence, for the most widely studied case of $\lambda = 2$, we have a lower bound $r \geq (2t-1)/3$, leaving a factor 1.5 efficiency gap when compared to the Merkle-Damgård and Merkle trees.

Upper Bound on the Number of Calls. For the upper bounds, much of earlier work concentrated on the setting of the "non-compressing" case of $\lambda = 1$, and often focused on the case of small t (e.g., $t = 2$), implicitly suggesting that – once the 2-to-1 function is built, – one should do further extensions with either Merkle-Damgård and Merkle trees. For example, Shrimpton and Stam [10] proposed a 2-to-1 compression function based on three calls of non-compressing function, which matches Stam's bound for $\lambda = 1$ and $t = 2$. Rogaway and Steinberger [8] designed similar results when the non-compressing primitive is an invertible permutation, which they also showed is optimal for this setting [9].

For general (large) t, Mennink and Preneel [5] also considered the non-compressing case $\lambda = 1$ and proposed an elegant tree-based mode of operation making $(2t-1)$ calls to the non-compressing round function, which matches Stam's bound. Unfortunately, they could only prove below-birthday security of $2^{n/3}$ queries for this construction. They also conjectured that the construction achieves optimal birthday security $2^{n/2}$, but could only prove it for a very restricted special-case attacker. These attacks make all their random oracle calls "layer-by-layer" (as opposed to in any order). As acknowledged by the authors, the simplifying assumption significantly helps with the proof of this special case and appears to be with a great loss of generality. In fact, they presented evidence that their existing analysis is unlikely to work for proving optimal security against unrestricted attackers.

Recently, two papers have appeared to tackle the compressing case $\lambda = 2$. In [4], Dodis et al. optimally settled the case $t = 5$, by introducing the $T5$ construction that processes five n-bit message blocks using three $2n$-to-n-bit compression function calls, which matches Stam's bound for $t = 5$ and $\lambda = 2$. Further, they suggested extending the $T5$ construction to a larger value of t using either Merkle-Damgård or Merkle trees. In both cases, they already achieve non-trivial saving compared to the earlier efficiency of these modes (equal to t compression calls): both variants now make roughly $3t/4$ calls to the compression functions. Still, once $t > 5$, this does not match the current lower bound of $2t/3$ calls. [4] also mentioned a natural, but more aggressive, variant of this extended construction for

the case of Merkle trees. However, they remark that this construction – even if proven collision-resistant (which is open), – would lose the efficient "local opening" properties of their simpler tree construction with $3t/4$ compression calls. Namely, one can no longer open one message block by only opening $O(\log t)$ internal values in the tree (as any such opening cannot have birthday security, despite satisfying correctness).

Finally, a breakthrough result of Andreeva, Bhattacharyya and Roy at Eurocrypt'21 [1] have claimed to settle the general case in the affirmative. They proposed a hash function ABR_l based on a perfect binary tree of height l. The hash ABR_l can process $t = (2^l + 2^{l-1} - 1)$ blocks with $r = (2^l - 1)$ calls of compression functions. This matches Stam's bound $r \geq (2t - 1)/3$. Somewhat interestingly, the ABR construction looks very similar to the tree construction of Mennink and Preneel [5] from non-compressing primitives, except all the compression calls at the leaf level now have an extra input (due to $\lambda = 2$ instead of $\lambda = 1$), while the internal calls to the compression function can also process an extra input, but using a slightly trickier rule involving two simple XOR operations. So, at least in the intuitive sense, the authors must have resolved the difficulty of [5] of dealing with general adversaries, for a construction very similar to the one of [5].

As an additional bonus feature, the work of [1] even claimed that the ABR_l mode also has attractive local opening properties, at the expense of slightly longer proof length ($2l$ instead of l of Merkle trees), but still having only l compression calls to verify such local opening.

Are We Done? Unfortunately, we have found serious issues in many claims made by the authors of [1], whom we call ABR hereafter. These issues appear quite significant, and range from verifiably false statements to noticeable gaps in the proof (e.g., omissions of important cases and unjustified bounds). Unfortunately, at this stage, we are unable to fix these issues in any simple way.

1.1 Our Results

Our results can be roughly divided into 3 categories:

(1) explicit refutation of some claims made by [1];
(2) serious technical issues in the proof provided by [1];
(3) a correct (but very different from [1]) proof for the for the ABR_3 construction (i.e. $t = 11$ and $r = 7$), which is incorrectly handled by ABR.

We detail these below.

Local Opening Insecurity of ABR. As we mentioned, ABR proposed a very efficient local opening for ABR_l. It opens about $2l$ blocks and makes l calls to verify. However, we have shown that a collision pair of the verification function can be found in $O(2^{n/2l})$ queries, which is significantly below birthday security already for $l = 2$. Hence, the suggested local opening can be broken in the above complexity. Moreover, we have shown that *any* non-trivial local opening of ABR_l satisfying a "by-pass verification" property (which is a natural class of openings that seems to include any natural opening one can think of) is broken below the birthday bound. For example, even opening $(t-1)$ out of t inputs cannot be birthday-secure, where $t = 2^l + 2^{l-1} - 1 = 2^{\Omega(l)}$. In contrast, previous tree-like constructions (e.g., [4]) achieve birthday security with logarithmic opening length $O(l)$. This is discussed in Section 4.

There are two surprising aspects to this mistake. First, our attack is completely standard (using standard generalized birthday attack [14]). Second, the local opening subsection in the ABR-paper does not even mention anything about security, only focusing on the correctness of the opening. We found this quite surprising.

Mistakes with the Main Proof. While the local opening mistake above is indisputable, the technical mistakes in the main collision resistance proof of ABR are harder to explain in detail (at least in the Introduction, before the technical notation is developed). They are also harder to state with conviction, since they often do a combination of the following pitfalls:

(a) involve imprecise statements,
(b) state a bound which might be true, but which appears completely non-obvious to us (to the extent of being the most difficult part of the proof);
(c) point to an "analogous" earlier case, but we fail to see why the previous argument generalizes;
(d) state some bound which appears to be correct only if one makes some restricting assumption on the attacker (but no such assumptions are made by the authors, who claim a fully general result!);
(e) silently omitting an important special case of the proof (i.e., the proof is non-exhaustive).

The totality of these issues make the proof presented by [1] at best unverifiable, and at worst incorrect. In particular, we still believe that the end result is correct, but fixing it would require a substantially harder proof.

At a very high level, the correct collision analysis for a tree-based function like ABR_l is complex mostly due to the *adaptive nature* of queries, and the queries made to different layers in the tree might not come in monotone order (i.e., may not be in order of the level of the nodes). Indeed, this is precisely the reason why the earlier birthday security result of Mennink and Preneel [5] only held for "in order" adversaries. Fortunately, the outputs of the leaf nodes can be given beforehand, as the input of those has no role in finding a collision. More formally, we can make a simple argument to force the attacker "evaluate" the first layer compression calls before any of the subsequent calls as follows. We give the attacker q random outputs (where q is the total number of queries made by the attacker) at the very beginning, but allow the adversary to *arbitrarily label* the corresponding input values at any point in the game. This is fine, since those input values do not participate in any other computation, but now all the outputs in the first layer are known before a single compression call is made to the lower layers. This allows for relatively simple analysis for the special case $l = 2$, and the authors of [1] indeed start with the correct analysis of this special case.[1]

Unfortunately, this argument completely fails after the first layer. (Indeed, handling this case will be one of the most difficult parts of our analysis, when we provide a correct proof for $l = 3$ in this paper.) In particular, we see the following high-level issues with the proof presented by [1] for $l \geq 3$. (More lower-level issues are discussed in Section 5.3 in the paper.)

1. ABR claimed a relation between collision and the number of computable hash outputs (termed as load). We will show in Section 5.4 that the relation is not true in general by giving a counterexample. This seems to hold for ABR if queries to the root node are performed at the end (which is the case for ABR_2). However, it seems non-obvious to us why a similar relation holds when the adversary makes out-of-order queries.
2. We have also found issues while bounding load. ABR consider "input multi-collision" for every node up to $O(n)$. However, due to the multiplicative nature of the number of multi-collisions as one goes down in the tree, we find that $O(n^i)$ multi-collision must be considered for the nodes at the i-th level. This would degrade the bound for load

[1] Another correct proof for $t = 5$ (corresponding to tree depth $l = 2$) was made for the $T5$ compression function by [4]. Interestingly, the authors did not notice the simplifying non-adaptivity argument above, and had to work relatively hard to handle out-of-order queries (e.g., it involved a careful expectation analysis and applying Markov's inequality; see proof of Proposition 5 in [4], which is over a page). This shows that handling out-of-order attackers is indeed highly non-trivial.

claimed by ABR, and invalidate the claimed birthday security at the end (unless the number of levels i is constant, in which case one can hide the extra n^i bound in the "O-tilde"-notation). This will be discussed in Step 1 of Section 5.3.

3. In fact, even if the load analysis is somehow fixed, ABR seem to consider the last query happens in the final node (or at the node where the load is considered). This is effectively equivalent to in order adversaries, but does not seem to be the case for general attackers. See Step 2 of Section 5.3.
4. Moreover, both messages of a collision pair can be generated due to a single query response (termed as *twin collision pair*). ABR completely ignore this case. This is discussed in detail in the last paragraph of Section 5.3.

We leave a more detailed explanation of these (and other issues (a)-(e)) later in the paper.

Collision Analysis of ABR_3. On a positive, our main technical result shows that the ABR_3 construction for $t = 11$ indeed achieves birthday security (roughly $n^5q^2/2^n$, where q is the number of compression function queries) with an optimally small number of $r = 7$ compression calls (see Section 6). While forming only the first step in recouping the incorrect results of [1], we are optimistic that our approach could be extended to finally settle the general case correctly. For example, compared to best known correct proofs for $t = 5$ (e.g., ABR_2 from [1], or the $T5$ compression function from [4]), we can no longer assume that the second layer calls to the compression function are made before all the third-layer calls, which is the main (unresolved) difficulty in the work of [5], and one of the key mistakes in the analysis of [1] (as we explained above). Thus, our proof is the first which handles non-trivial "out-of-order" adversaries correctly.

We also hope our proof of ABR_3 provides a sharp contrast to the flawed proof of [1], even for this special case. For example, we already mentioned handling general "out-of-order" adversaries. In a different vein, we also consider the twin-collision analysis for ABR_3 which is completely missing from [1]. This analysis requires a non-trivial multi-collision analysis on a sum of our compression functions, and we also need to bound some other failure events to analyze the non-twin collision security of ABR_3. None of these arguments appeared in [1].

2 Security Definitions

2.1 Notations

We call elements of $\{0,1\}^n$ blocks. A k-to-r (block) function or random oracle has domain $\{0,1\}^{kn}$ and range $\{0,1\}^{rn}$. We write the set $[k] = \{1, 2, \ldots, k\}$. A partial function τ from D to R is a subset $\tau \subseteq D \times R$ such that for every $x \in D$, there are at most one y with $(x, y) \in \tau$. We define domain $\mathrm{dom}(\tau) := \{x : \exists y, (x, y) \in \tau\}$ and range $\mathrm{ran}(\tau) = \{y : \exists x, (x, y) \in \tau\}$ of a partial function τ. We use the shorthand notation $A \cup x$ and $A \setminus x$ to denote $A \cup \{x\}$ and $A \setminus \{x\}$ respectively. For any q-tuple x^q, we define $\mathrm{mc}(x^q) = \mathsf{max}_a|\{i : x_i = a\}|$. For two lists $\mathcal{L}_1$ and $\mathcal{L}_2$, we define $\mathrm{mc}(\mathcal{L}_1 \oplus \mathcal{L}_2) = \mathsf{max}_a|\{(i, i') : L_i \oplus L_{i'} = a, L_i \in \mathcal{L}_1, L_{i'} \in \mathcal{L}_2\}|$. It can be similarly extended for xor of more than two lists.

2.2 Generic Hash Mode

Let $\mathtt{H}^f$ be a t-to-1 hash function which uses an n-bit compression function (i.e. λ-to-1 compression function f for some $\lambda > 1$) as an oracle. Note that a mode can use more than one compression functions $f_1, \ldots, f_r$. However, as we analyze in the random oracle model, independent random oracles can be obtained from a single random oracle with a little bit

larger domain by using the standard domain separation method. In this paper, we only consider fixed-length input and also assume r is the same for all messages. Moreover, the hash function calls f_i on i-th call and so the domains of every call are separated by domain separation. We also denote the family $f := (f_i : i \in [r])$ by f and we call λ-to-1 r r.o. (random oracle). We denote $\tau_{\mathtt{H}}(M \mid f) := \{((1, x_1), y_1), \ldots, ((r, x_r), y_r)\}$ where x_i denotes the input of i-th call of its oracle tuple while computing $\mathtt{H}^f(M)$ and $y_i = f_i(x_i) := f(i, x_i)$. A λ-to-1 **transcript** τ is a partial function from $[r] \times \{0,1\}^{\lambda n}$ to $\{0,1\}^n$. For a λ-to-1 r r.o. f, we have

$$\forall (i, x) \notin \mathrm{dom}(\tau), y \in \{0,1\}^n, \quad \mathrm{Prob}(f(i, x) = y \mid \tau \subseteq f) = 2^{-n}.$$

▶ **Definition 1** (transcript-based hash computation). *Given a partial function $\tau \subseteq f$, let $\mathtt{H}^\tau = \{(M, \mathtt{H}^f(M)) : \tau_{\mathtt{H}}(M \mid f) \subseteq \tau\}$ be a partial hash function. In other words, $\mathtt{H}^\tau$ consists of all pairs (M, z) such that $\mathtt{H}^f(M)$ can be computed by simply using the transcript τ and z is the final value. The elements of the set $\mathrm{dom}(\mathtt{H}^\tau)$ are called τ-computable messages. As $\tau \subseteq f$, we must have $\mathtt{H}^\tau \subseteq \mathtt{H}^f$.*

2.3 Collision Game

Let $\mathcal{A}$ be an adversary having oracle access of f which makes q queries to each f_i adaptively. As we assume an unbounded time adversary, there is no loss in assuming that $\mathcal{A}$ is deterministic. Thus, the i-th query (x_i, v_i) of $\mathcal{A}$ depends on τ^{i-1} (the transcript of query-responses after $(i-1)$ queries). After the query-response phase, $\mathcal{A}$ returns a pair of distinct messages (M, M') such that both M, M' are transcript-computable. We say $\mathsf{coll}_{\mathtt{H}}$ holds if $\mathtt{H}^\tau(M) = \mathtt{H}^\tau(M')$, called a *computable collision pair*. We define $\mathbf{Adv}^{\mathsf{coll}}_{\mathtt{H}^f}(\mathcal{A}) := \Pr(\mathsf{coll}_{\mathtt{H}})$.

▶ **Definition 2** (cross collision). *Let $\mathtt{H}$ and $\mathtt{H}'$ be two hash functions. A* cross-collision τ-computable pair *is a pair (M, M') (not necessarily distinct) such that $\mathtt{H}^\tau(M) = \mathtt{H}'^\tau(M')$. We denote $\mathsf{coll}^\tau_{\mathtt{H},\mathtt{H}'} := \{M \in \mathrm{dom}(\mathtt{H}^\tau) : \exists M',\ \mathtt{H}^\tau(M) = \mathtt{H}'^\tau(M')\}$.*

2.4 Local Opening

We now define the local opening security of a hash function output (viewed as a commitment of a message). Given a hash function mode $\mathtt{H}^f$, a local opening Open^f for $\mathtt{H}$ maps a pair (M, i) to $\pi = (m_i, i, \pi')$ (called proof) where $M = (m_1, m_2, \ldots, m_c)$ is a message (a tuple of blocks) and $i \in [c]$ is an index.

Correctness of Local Opening. There is an efficient function Ver^f such that for all message M, all index i, $\mathsf{Ver}^f(\mathsf{Open}^f(M, i), \mathtt{H}^f(M)) = 1$.

Security of Local Opening. In the local opening security, the adversary wins if it produces an output h corresponding to two contradicting local openings for some position i.

▶ **Definition 3** (local opening advantage). *Let $\mathtt{H}$ be a hash function and Open be a correct local opening for $\mathtt{H}$ with verification function Ver. For any adversary $\mathcal{A}$, we define the local opening advantage as*

$$\mathbf{Adv}^{local}_{H}(\mathcal{A}) = \Pr\Big[\mathsf{Ver}(i, m, \pi, h) = \mathsf{Ver}(i, m', \pi', h) = 1, m \neq m' \mid (i, m, m', \pi, \pi', h) \leftarrow \mathcal{A}^f\Big]$$

By-Pass Hash Computation. We say that $\mathtt{H}$ has a *by-pass computation* $(\mathtt{H}_i : i \in [c])$ corresponding to a local opening Open if for all M, $i \in [c]$,

$$\mathtt{H}_i^f(\mathsf{Open}^f(M,i)) = \mathtt{H}^f(M).$$

In other words, given a proof (output of the Open) and the message block for the index (for which the proof is produced), we can compute the hash output of the message (without knowing the other blocks of the message). The verification algorithm simply checks whether the hash value computed through the by-pass hash is the same as what was committed before. As f is treated as an oracle, it is natural to assume that for all M and for all i,

$$\tau_{\mathsf{Open}}(M,i \mid f) \cup \tau_{\mathtt{H}_i}(\mathsf{Open}^f(M,i) \mid f) = \tau_{\mathtt{H}}(M \mid f).$$

We now define the *inter-collision* advantage for by-pass computation $\mathtt{H}_i$ as

$$\mathbf{Adv}^{\mathsf{coll}*}_{\mathtt{H}_i}(\mathcal{A}) = \Pr\left[\mathtt{H}_i(m,\pi) = \mathtt{H}_i(m',\pi') \text{ and } m \neq m' \mid (m,\pi,m',\pi') \leftarrow \mathcal{A}^f\right].$$

Thus, it is the same as the collision game, except that the adversary needs to find a collision pair for which $m \neq m'$. Suppose $\mathcal{A}$ finds a collision pair $((m,\pi),(m',\pi'))$ for $\mathtt{H}_i$, and let $h = \mathtt{H}_i(m,\pi)$. Then $\mathcal{A}$ can commit h and later on, it can successfully open for either of two messages m and m' as required. Now we make the following simple observation

$$\mathbf{Adv}^{local}_{\mathtt{H}}(q') = \max_{\mathcal{A}} \max_{i} \mathbf{Adv}^{\mathsf{coll}*}_{\mathtt{H}_i}(\mathcal{A}). \tag{1}$$

The above observation (see [4] for details) helps us to reduce the local opening security to inter-collision security problem for the by-pass hash family.

2.5 Stam's Tradeoff between Security and Performance

Stam's bound states that there always exists a collision attack with at most $2^{n(\lambda-(t-0.5)/r)}$ queries on a t-to-1 block hash function making r calls to λ-to-1 block compression functions.

3 Re-introduction of the ABR Hash due to [1]

We first start by defining a generalized tree hash structure, and then re-introduce the ABR Hash as a special tree hash, as opposed to introducing as it is in [1]. This is because we feel some things have not been properly defined by the authors there, and these issues need to be addressed properly.

A *full binary tree* (FBT) is a binary tree in which every node v other than the leaves has two children, denoted as v_{L} (left child) and v_{R} (right child). A *perfect binary tree* (PBT) is a full binary tree in which all the leaf nodes are at the same level (called height of the tree).

▶ **Example 4** (perfect binary tree of height l)**.** Let l be a fixed positive integer and $\mathcal{T}$ be a perfect binary tree of height l over all vertices (j,b), $j \in [l], b \in [2^{l-j}]$ with $(l,1)$ being the root. For every two vertices (j,b) and $(j+1,\lceil b/2 \rceil)$, we associate an edge. We call $(j-1,2b-1)$ and $(j-1,2b)$ the left and right child of (j,b) respectively. Note that $\mathcal{T} = \mathcal{T}_{(l,1)}$.

3.1 Some Notations and Definitions on Binary Trees

For a binary tree $\mathcal{F}$, let $\mathcal{L}_{\mathcal{F}}$ and $V(\mathcal{F})$ denote the set of leaf nodes and all nodes of $\mathcal{F}$ respectively. Any non-leaf node is called an intermediate node. For a non-root intermediate node v of $\mathcal{F}$, we consider the following two full binary trees:

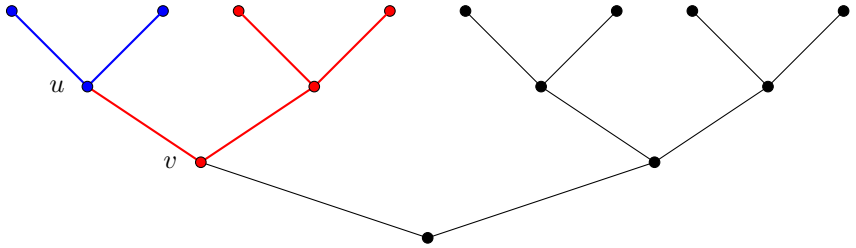

Figure 1 In this figure, $\mathcal{F}_v$ is the sub-tree rooted at v, i.e. the union of the red and blue sub-trees, $\mathcal{F}_{-v}$ is the black sub-tree, and $\mathcal{F}_{v-u}$ is the red sub-tree.

1. $\mathcal{F}_v$: the full binary sub-tree rooted at v.
2. $\mathcal{F}_{-v}$: the sub-tree $(\mathcal{F} \setminus \mathcal{F}_v) \cup v$.

For a tree $\mathcal{F}$, and a vertex v of $\mathcal{F}$, we write V_v, $\mathcal{L}_v$ and V_v^* to denote the set of all nodes, leaf nodes and intermediate (non-leaf) nodes respectively for the tree $\mathcal{F}_v$. For any $u \in V_v^* \setminus v$, we write $\mathcal{F}_{v-u} = (\mathcal{F}_v \setminus \mathcal{F}_u) \cup u$. We write V_{v-u} to denote the set of vertices of $\mathcal{F}_{v-u}$. For the sake of notational simplicity we ignore the suffix v when v is the root. In this section we only consider trees of the form $\mathcal{F}_v$ and $\mathcal{F}_{v-u}$. Refer to Figure 1 for a pictorial representation.

To each node $v \in V$ of a perfect binary tree $\mathcal{T}$, an independent 2-to-1 block compression function (modeled as a random oracle) f_v is assigned. We use the notation f to denote the collection of random oracles $\{f_v : v \in \mathcal{T}\}$.

▶ **Definition 5** (message for tree hash). *A message m for any full binary sub-tree $\mathcal{F}$ of a perfect binary tree $\mathcal{T}$ having the same root is a function $m : V(\mathcal{F}) \to \{0,1\}^n \cup \{0,1\}^{2n}$ such that for all $u \in \mathcal{L}_\mathcal{F} \cap \mathcal{L}_\mathcal{T}$, $m(u) \in \{0,1\}^{2n}$, otherwise, $m(u) \in \{0,1\}^n$. A complete message m is a message at the root of $\mathcal{T}$.*

Thus, for every leaf node of $\mathcal{F}$ (which is also a leaf node of the perfect binary tree), we associate $2n$ bit messages. For all other vertices, we associate an n bit message. We write $\mathbb{M}_\mathcal{F}$ to denote the set of all messages for $\mathcal{F}$. We simply write $\mathbb{M}_v$ and $\mathbb{M}_{v-u}$ instead of $\mathbb{M}_{\mathcal{T}_v}$ and $\mathbb{M}_{\mathcal{T}_{v-u}}$ respectively.

For a message m for $\mathcal{T}_v$ (also called m at the node v), and $u \in V_v$, we write $m|_u = m|_{\mathcal{T}_u}$, the message restricted to $\mathcal{T}_u$. Similarly, we write $m_\mathsf{L} := m|_{v_\mathsf{L}}$ and $m_\mathsf{R} := m|_{v_\mathsf{R}}$. We also write $m|_{v-u\to h}$ to denote a message for $\mathcal{T}_{v-u}$ which is same as the restricted function $m|_{\mathcal{T}_{v-u}}$, except at u, where it assigns h (instead of $m(u)$). In the context of our work, this basically means we replace the message $m(u)$ at node u by the intermediate hash output of $\mathcal{T}_u$, the tree rooted at u, and consider the message for the remaining tree, $\mathcal{T}_{v-u}$.

▶ **Definition 6** (Generalized Tree Hash). *Let $\mathcal{F}$ be a full binary sub-tree of a perfect binary tree $\mathcal{T}$ and let $m \in \mathbb{M}_\mathcal{F}$. For every $v \in \mathcal{F}$, we associate an intermediate hash output O_v and an intermediate input I_v recursively as follows:*

1. *$v \in \mathcal{L}_\mathcal{F} \setminus \mathcal{L}_\mathcal{T}$, $|m(v)| = n$: $O_v = m(v)$ and there is no input,*
2. *$v \in \mathcal{L}_\mathcal{F} \cap \mathcal{L}_\mathcal{T}$, $|m(v)| = 2n$: $O_v = f_v(m(v))$, $I_v = m(v)$,*
3. *otherwise: $|m(v)| = n$ and we define*

$$I_v = \big(O_{v_\mathsf{L}} \oplus m(v),\ O_{v_\mathsf{R}} \oplus m(v)\big),\ \text{and}\ O_v = f_v(O_{v_\mathsf{L}} \oplus m(v),\ O_{v_\mathsf{R}} \oplus m(v)) \ \oplus\ O_{v_\mathsf{R}}.$$

O_ω is the final hash output corresponding to $\mathcal{F}$ where ω is the root of $\mathcal{F}$. We also call I_ω final input.

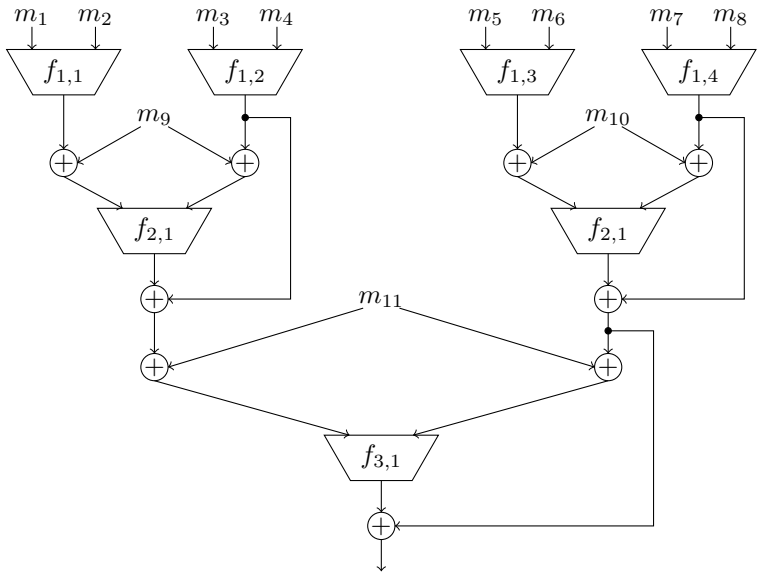

Figure 2 ABR of height 3.

Let us see what this means. If $\mathcal{F} = \mathcal{T}$, the above definition implies that for a leaf node v, the message at v, which itself is the input, is $2n$ bits long, and the output is just $f_v(m(v))$, where f_v is the 2-to-1 block compression function attached to it, and for an intermediate node, the message is n bits long, and the input and output are as defined above. If $\mathcal{F}$ is a proper sub-tree of $\mathcal{T}$, then there might exist vertices, which are leaves of $\mathcal{F}$, but not of $\mathcal{T}$. For such a vertex v, the message is n bits long, and the message itself is considered the output of the vertex. This vertex doesn't have any input.

The ABR Hash Function. The ABR hash is the hash output based on a perfect binary tree $\mathcal{T}$ of height l. In terms of Definition 6, the case $\mathcal{F} = \mathcal{T}$ corresponds to a ABR tree, and the final hash output is the ABR hash. Thus, ABR_l hash is a $(2^l + 2^{l-1} - 1)$-to-1 block hash function, $l > 1$. We refer to Figure 2 for a pictorial view of ABR with $l = 3$. For a trivial tree $\mathcal{F} = \{w\}$, with a message $m(\omega) \in \{0,1\}^{2n}$, $\mathcal{F}(m) = f_\omega(m)$.

We write $\mathtt{H}^\tau(m)$ and $\mathsf{in}^\tau(m)$ to denote the transcript based hash and the final input respectively, whenever defined for the message m for a tree $\mathcal{F}$. If $\mathtt{H}^\tau(m)$ is defined we call m τ-computable or simply computable message. We write $\perp$ to mean that it is undefined. Note that a tree is uniquely determined from the message. We write dom_v^τ and dom_{v-u}^τ to denote the set of all computable messages at v and for $\mathcal{T}_{v-u}$ respectively. Similarly, we write ran_v^τ and ran_{v-u}^τ to denote the set of all computable hashes at v and for $\mathcal{T}_{v-u}$ respectively. The size of the set ran_v^τ, called *load at* v, is denoted as $L_{\tau,v}$.

4 Local Opening Analysis of ABR Hash Function

In section 3, we have defined hash function based on a tree $\mathcal{F}$ for a message over the tree $\mathcal{F}$. In this section, we consider a variant of the message function and a hash function for the variant message. This is required to properly define the local opening of the ABR tree.

Message for a Full Binary Tree. Let $\mathcal{F}$ be a full binary tree and $L \subseteq \mathcal{L}_{\mathcal{F}}$. Let $\mathcal{M}_{\mathcal{F},L}$ be the set of all functions $m : V(\mathcal{F}) \to \{0,1\}^n \cup \{0,1\}^{2n}$ such that for all $v \in \mathcal{L}_{\mathcal{F}} \setminus L$, $m(v) \in \{0,1\}^{2n}$ and for all other vertices v, $m(v) \in \{0,1\}^n$. We call m a message (or a message function) for $\mathcal{F}$.

▶ **Definition 7** (Generalized Tree Hash, a variant)**.** *Let $m \in \mathcal{M}_{L,\mathcal{F}}$ be a message function for $\mathcal{F}$. For every $v \in \mathcal{F}$, the intermediate hash output O_v is defined recursively as follows:*

1. *$v \in L$, $|m(v)| = n$: $O_v = m(v)$,*
2. *$v \in \mathcal{L}_{\mathcal{F}} \setminus L$, $|m(v)| = 2n$: $O_v = f_v(m(v))$, $I_v = m(v)$,*
3. *$v \notin \mathcal{L}_{\mathcal{F}}$: we define*

$$I_v = \big(h_1 \oplus m(v),\ h_2 \oplus m(v)\big) \text{ and } O_v = f_v(h_1 \oplus m(v),\ h_2 \oplus m(v)) \oplus h_2,$$

where $h_1 = O_{v_{\mathsf{L}}}$ and $h_2 = O_{v_{\mathsf{R}}}$.

The hash output corresponding to $\mathcal{F}$ is defined as $\mathcal{F}^f(m) := O_\omega$ where ω is the root of $\mathcal{F}$. We also call $I_\omega := \mathcal{F}^f_{\mathsf{in}}(m)$ final input. It is clear from the definition that for any node $v \notin L$, $\mathcal{F}^f_v(m|_v) = O_v$ and $\mathcal{F}^f_{v,\mathsf{in}}(m|_v) = I_v$.

Visualizing the tree is not difficult. As an example, when $\mathcal{F} = \mathsf{ABR}_3$, we have Figure 2, where L is a subset of the leaf nodes, say $(1,1)$ and $(1,2)$. We now define local opening of the Generalized Tree Hash.

▶ **Definition 8.** *Let m be a message for a perfect binary tree $\mathcal{T}$. For any full binary sub-tree $\mathcal{F}$ and a set $\mathcal{L}_{\mathcal{F}} \setminus \mathcal{L}_{\mathcal{T}} \subseteq L \subseteq \mathcal{L}_{\mathcal{F}}$, we define a message $m' := \mathsf{Open}^f_{\mathcal{F},L}(m) \in \mathcal{M}_{\mathcal{F},L}$ for $\mathcal{F}$ as follows.*

1. *$v \in L$: $m'(v) = \mathcal{T}^f_v(m_v)$.*
2. *Otherwise: $m'(v) = m(v)$.*

Now, we first analyze the local opening security of ABR_l proposed by [1] and then show that no non-trivial opening of ABR can achieve birthday bound security.

4.1 Local Opening Analysis of ABR Hash due to [1]

We describe the by-pass hash corresponding to the message block m_1 for ABR_l. It is based on the full sub-tree $\mathcal{F}$ consisting of nodes $\{(i,1) : i \in [l]\} \cup \{(i,2) : i \in [l-1]\}$ and $L = \{(1,2),(2,2),\ldots,(l-1,2)\}$. Refer to Figure 3. Note that the number of blocks in $\mathsf{Open}_{\mathcal{F},L}(m)$ is $2l$, and in the sub-tree $\mathcal{F}$ corresponding to $\mathsf{Open}_{\mathcal{F},L}(m)$, the number of calls to underlying compression function f is l. According to Stam's bound, there exists a collision attack with at most $2^{n/2l}$ queries. We give an attack that matches this bound.

Let $I(i,1) = (u_{i,1}, v_{i,1})$ be the input of $f_{i,1}$ and let $y_{i,1}$ be the output. Let $h_{i,2}$ be the message for node $(i,2)$. Let $h_{i,1} = y_{i,1} \oplus h_{i-1,2}$ for $i > 1$ and $h_{1,1} = y_{1,1}$. Then, $h_{i,1}$ is the output at node $(i,1)$. Also, let $m_{i,1}$ be the message associated with a non-leaf node $(i,1)$. We wish to find a collision at the output of node $(l,1)$, i.e. we need to find two messages m' and m'' for $\mathcal{F}$ such that $\mathcal{F}_{(l,1)}(m') = \mathcal{F}_{(l,1)}(m'')$. Given any message for $\mathcal{F}$, the output at node $(l,1)$ is given by $h_{l,1}$.

Note that $h_{1,1} = f_{1,1}(u_{1,1}, v_{1,1})$. After computing $h_{i-1,1}$, we proceed to compute $h_{i,1}$. We note that $h_{i-1,2}$ is a message block for $\mathcal{F}$. The input at node $(i,1)$, $I(i,1) = (h_{i-1,1} \oplus m_{i,1}, h_{i-1,2} \oplus m_{i,1}) = (u_{i,1}, v_{i,1})$ and the output at node $(i,1)$ is:

$$\begin{aligned} h_{i,1} = f_{i,1}(I(i,1)) \oplus h_{i-1,2} &= f_{i,1}(u_{i,1}, v_{i,1}) \oplus h_{i-1,2} \\ &= f_{i,1}(u_{i,1}, v_{i,1}) \oplus u_{i,1} \oplus v_{i,1} \oplus h_{i-1,1} \\ &= g_{i,1}(u_{i,1}, v_{i,1}) \oplus h_{i-1,1} \end{aligned}$$

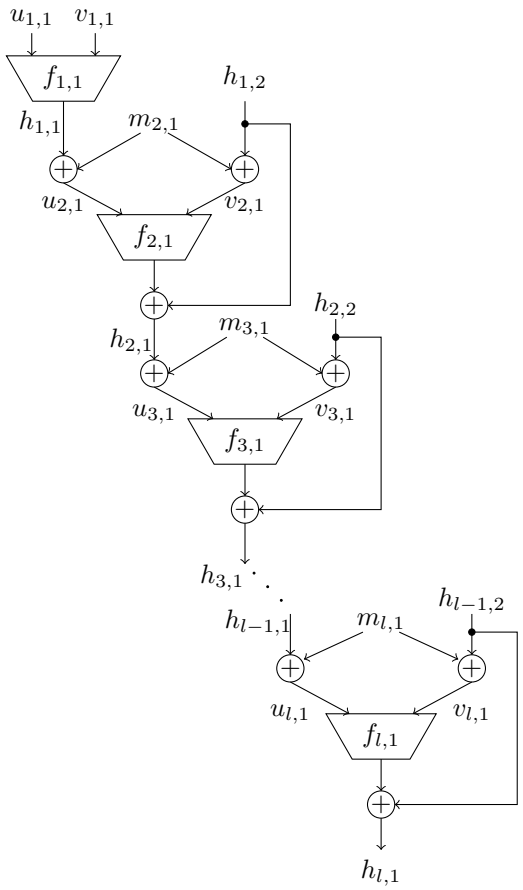

Figure 3 A specific local opening of ABR_3.

where $g_{i,1}(u_{i,1}, v_{i,1}) = f_{i,1}(u_{i,1}, v_{i,1}) \oplus u_{i,1} \oplus v_{i,1}$. By induction, the final hash computation is

$$h_{l,1} = g_{l,1}(u_{l,1}, v_{l,1}) \oplus g_{l-1,1}(u_{l-1,1}, v_{l-1,1}) \oplus \ldots \oplus g_{1,1}(u_{1,1}, v_{1,1}).$$

Since the functions $f_{i,1}$ are random and independent so are $g_{i,1}$'s. Thus $h_{l,1}$ is the XOR of l random functions. Thus, a collision is expected at node $(l, 1)$ with $2^{n/2l}$ queries. One can also apply a generalized birthday attack with complexity $2^{n/(1+\lceil \log 2l \rceil)}$.

Now, let us look at the target collision resistance of the above local opening of ABR_l. Target Collision Resistance describes the ability of an adversary to find a second pre-image for a fixed message. Target collision resistance has many practical applications. For example, if a client sends a file F to the server and then wants the server to send part of the file F_i along with a proof of correctness then, as long as the server does not control the choice of the file F, the server would need to find a targeted collision to break security and reveal an incorrect value F_i'.

Here, for a fixed message m, the final hash computation $h_{l,1}$ is fixed. Hence, for target collision resistance we wish the XOR of l random functions to collide with this value of $h_{l,1}$. This collision is expected with $2^{n/l}$ queries.

4.2 Decomposition of ABR Hash

Now we decompose ABR hash computation on $\mathcal{T}$ through a full binary proper sub-tree $\mathcal{F}$ sharing the same root and a set L.

▶ **Lemma 9** (decomposition lemma for any full binary tree)**.** *For all full binary sub-tree $\mathcal{F}$ of a perfect binary tree $\mathcal{T}$ and a set of nodes $\mathcal{L}_{\mathcal{F}} \setminus \mathcal{L}_{\mathcal{T}} \subseteq L \subseteq \mathcal{L}_{\mathcal{F}}$, we have*

$$\mathcal{T}^f = \mathcal{F}^f \circ \mathsf{Open}^f_{\mathcal{F},L}.$$

Proof. Let m be a message for $\mathcal{T}$. $\mathcal{T}^f(m)$ represents the hash output based on the perfect binary tree $\mathcal{T}$. For any node v of $\mathcal{T}$, the restricted message over $\mathcal{T}_v$ is m_v. Hence, $\mathcal{T}^f(m)$ computes $\mathcal{T}^f_v(m_v)$ for all nodes $v \in \mathcal{T}$.

For any full binary sub-tree $\mathcal{F}$ of $\mathcal{T}$, $m' = \mathsf{Open}^f_{\mathcal{F},L}$ is defined as above. For any $v \in L$: $m'(v) = \mathcal{T}^f_v(m_v)$. We calculate the hash outputs for the restricted messages on these nodes first. Since for all other $v \in \mathcal{F}$, $m'(v) = m(v)$, and $\mathcal{F}$ is a sub-tree of $\mathcal{T}$, $\mathcal{F}^f(m')$ actually computes $\mathcal{T}^f_v(m_v)$ for all $v \in \mathcal{F} \setminus \mathcal{L}_{\mathcal{F}}$. Thus, $\mathcal{F}^f \circ \mathsf{Open}^f_{\mathcal{F}}(m)$ also computes $\mathcal{T}^f_v(m_v)$ for all nodes $v \in \mathcal{T}$ and produces the same output $\mathcal{T}^f(m)$. ◀

If $\mathcal{F} = \mathcal{T}$ and $L = \emptyset$ then $\mathsf{Open}^f_{\mathcal{F},L}(m) = m$. For any other proper local opening we cannot ensure birthday bound security. We prove the following theorem:

▶ **Theorem 10.** *No non-trivial opening of* ABR *can achieve birthday bound security.*

Proof. Stam's bound states that there exists a collision attack with at most $2^{n(\lambda-(t-0.5)/r)}$ queries on a t-to-1 block hash function making r calls to λ-to-1 block compression functions. We have $\lambda = 2$. If we want to achieve $2^{n/2}$ collision security, $t \leq 1.5r + 0.5$. In other words, if $t > 1.5r + 0.5$, then we have a collision attack with query complexity $2^{\frac{n}{2}(1-\delta/r)}$, $\delta := t - 1.5r - 0.5$.

For ABR of height l, we have $t = 2^l + 2^{l-1} - 1$ and $r = 2^l - 1$. This satisfies $t = 1.5r + 0.5$, and it is optimal. We show that for any non-trivial opening $\mathsf{Open}_{\mathcal{F},L}$ of ABR, $\mathcal{F}$ satisfies $t > 1.5r + 0.5$. Let us consider the simplest non-trivial opening, corresponding to $L = \{(1,1)\}$. Then, for $m = (m_1, m_2, m')$, where m_1, m_2 are the first two message blocks and m' is the remaining part, $\mathsf{Open}_{\mathcal{F},L}(m) = (f_{1,1}(m_1, m_2), m')$. Then, $t = 2^l + 2^{l-1} - 2$, and $r = 2^l - 2$ ($f_{1,1}$ is not called). This satisfies $t > 1.5r + 0.5$. If $\mathsf{Open}_{\mathcal{F},L}$ consists of only one sub-tree computation of height h, then for $\mathcal{F}$, we have $t = (2^l + 2^{l-1} - 1) - (2^h + 2^{h-1} - 1) + 1$ and $r = 2^l - 2^h$, which satisfies $t > 1.5r + 0.5$.

A general opening $\mathsf{Open}_{\mathcal{F},L}$ of ABR may consist of more than one complete sub-tree computation. Let the number of complete sub-tree computations in $\mathsf{Open}_{\mathcal{F},L}$ be k, and for each $1 \leq i \leq k$, let h_i be the height of the i-th sub-tree. Then, for $\mathcal{F}$, we have

$$t = (2^l + 2^{l-1} - 1) - \sum_{i=1}^{k} (2^{h_i} + 2^{h_i - 1} - 1) + k, \qquad r = (2^l - 1) - \sum_{i=1}^{k} (2^{h_i} - 1).$$

It can be easily seen that $t > 1.5r + 0.5$. Thus, no non-trivial opening of ABR can achieve birthday bound security. ◀

5 Collision Analysis of ABR hash

In this section, we first define certain items which will be required to analyze the collision.

▶ **Definition 11** (input multi-collision). *For any $x \in \{0,1\}^n$, let $\mathrm{MC}_v^\tau(x)$, called input multi-collision set at v (with x as input multi-collision value), denote the set of all messages m at v with $\mathsf{in}^\tau(m) = x$. also, let*

$$\mathrm{mc}_v^\tau(x) = |\mathrm{MC}_v^\tau(x)|, \quad \mathrm{mc}_v^\tau = \max_{x\in\{0,1\}^n} \mathrm{mc}_v^\tau(x).$$

When v is the root node, we skip the notation v.

We define the newly generated messages and the hashes at a node v due to addition of the query-response (x, y) to the transcript τ as

$$\mathsf{New}_v^\tau(x,y) := \mathrm{dom}_v^{\tau\cup(x,y)} \setminus \mathrm{dom}_v^\tau, \quad \mathsf{NewH}_v^\tau(x,y) := \mathrm{ran}_v^{\tau\cup(x,y)} \setminus \mathrm{ran}_v^\tau.$$

Clearly, $\mathsf{NewH}_v^\tau(x,y) = \mathtt{H}^{\tau\cup(x,y)}(\mathsf{New}_v^\tau(x,y))$ (image set of $\mathtt{H}^{\tau\cup(x,y)}$ for the domain $\mathsf{New}_v^\tau(x,y)$). Note that x need not be queried at v. However, to have a new computable message, x should be queried at some node, say u, in $\mathcal{T}_v$. Analyzing the behavior of the set $\mathsf{New}_v^\tau(x,y)$ (or its size) is easy when $u = v$ or when u is one of the children of v. However, it becomes more complex when u is far away from v.

- Case $u = v$: $\mathsf{New}_v^\tau(x,y) = \mathrm{MC}_v^\tau(x)$ (and does not depend on y) and we call these messages freshly generated **immediate** messages.
- Case $u \in \mathcal{T}_v \setminus v$: The newly generated messages at v is

$$\mathsf{New}_v^\tau(x,y) = \{m|_v : \mathsf{in}^\tau(m|_u) = x, m|_{v-u\to h} \in \mathrm{dom}_{v-u}^\tau, h = y \oplus \mathtt{H}^\tau(m|_{u\mathsf{R}})\}.$$

So, we have $\mathbb{E}_y(|\mathsf{New}_v^\tau(x,y)|) = \dfrac{\mathrm{mc}_u^\tau(x) \times |\mathrm{dom}_{v-u}^\tau|}{2^n}$.

Now we discuss how the size of the computable message space $|\mathrm{dom}_{v-u}^\tau|$ can be written when u is one of the children or grandchildren of v.

▶ **Example 12.** Suppose $u = v_{\mathsf{R}}$. In this case,

$$\begin{aligned}\mathsf{New}_v^\tau(x,y) =& \{m|_v : \mathsf{in}^\tau(m_{\mathsf{R}}) = x, y = \mathtt{H}^\tau(m_{\mathsf{RR}}) \oplus \mathtt{H}^\tau(m_{\mathsf{L}}) \oplus x_1 \oplus x_2,\\ & \mid (v,(x_1,x_2)) \in \mathrm{dom}(\tau), m(v) = x_1 \oplus \mathtt{H}^\tau(m_{\mathsf{L}})\}.\end{aligned}$$

So, $\mathbb{E}_y(|\mathsf{New}_v^\tau(x,y)|) \le \dfrac{\mathrm{mc}_{v_{\mathsf{R}}}^\tau(x) \times |\mathrm{ran}_{v_{\mathsf{L}}}^\tau| \times |\tau_v|}{2^n}$, where τ_v denotes the set of elements in the transcript of the form $((v,x),y)$.

▶ **Example 13.** In the previous case, we could write the expectation of number of newly generated messages in terms of input multi-collision and range size of tree hash. Now, we consider $u = v_{\mathsf{RR}}$, i.e. u is a grandchild of v. Refer to Figure 4. Let $h = y \oplus \mathtt{H}^\tau(m_{\mathsf{RRR}})$. First, let us look at $|\mathrm{dom}_{v-v_{\mathsf{RR}}}^\tau|$.

$$\begin{aligned}|\mathrm{dom}_{v-v_{\mathsf{RR}}}^\tau| =& \{m|_v : H_1 \oplus h = x_1' \oplus x_2', H_2 \oplus y' \oplus h = x_1'' \oplus x_2'',\\ & \mid H_1 = \mathtt{H}^\tau(m_{\mathsf{RL}}), H_2 = \mathtt{H}^\tau(m_{\mathsf{L}}), (v_{\mathsf{R}},(x_1',x_2'),y'), (v,(x_1'',x_2''),*) \in \tau\}.\end{aligned}$$

Note that this implies $H_1 \oplus H_2 \oplus \bar{y'} = x_1'' \oplus x_2''$, where $\bar{y'} = x_1' \oplus x_2' \oplus y'$. Thus,

$$|\mathrm{dom}_{v-v_{\mathsf{RR}}}^\tau| = \mathrm{mc}(\mathrm{ran}_{v_{\mathsf{L}}}^\tau \oplus \mathrm{ran}_{v_{\mathsf{RL}}}^\tau \oplus \bar{f}_{v_{\mathsf{R}}}) \times |\tau_v|,$$

where $\bar{f}_{v_{\mathsf{R}}}(u_1,u_2) = u_1 \oplus u_2 \oplus f_{v_{\mathsf{R}}}(u_1,u_2)$. Hence,

$$\mathbb{E}_y(|\mathsf{New}_v^\tau(x,y)|) \le \frac{\mathrm{mc}_{v_{\mathsf{RR}}}^\tau(x) \times \mathrm{mc}(\mathrm{ran}_{v_{\mathsf{L}}}^\tau \oplus \mathrm{ran}_{v_{\mathsf{RL}}}^\tau \oplus \bar{f}_{v_{\mathsf{R}}}) \times |\tau_v|}{2^n}.$$

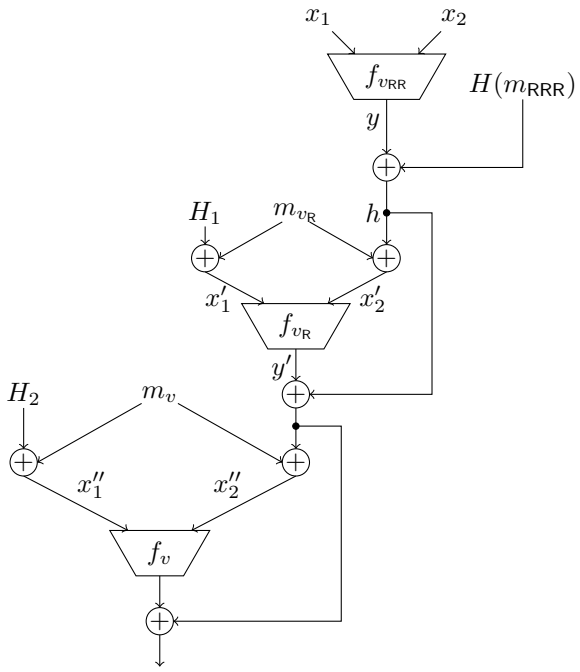

Figure 4 The graph of $\mathcal{T}_{v-u}$ when u is the rightmost grandchild of v.

Adversary and Its Queries. Let $\mathcal{L}_v$ denote the lists of all responses of f_v, for all leaf node v. We can assume that these lists are given to the adversary at the beginning of the game. This is without loss of generality as the inputs to f_v's have no role in the collision event. However, this is not true for all intermediate nodes (the non-leaf nodes) and so adaptivity of intermediate nodes must be considered. We assume that an adversary makes exactly q queries to each node. Let $q' := qr$ denote the total number of queries where $r = |V^*|$ and V^* is the set of non-leaf or intermediate nodes. Let Q_v denote the set of query numbers for the node v, $v \in V^*$. So for all non-leaf node v, $|Q_v| = q$. Let (x_i, y_i) denote the i-th query-response pair made to the node v_i. So given transcript τ^{i-1} (transcript after $(i-1)$ queries), the distribution of y is uniform over $\{0,1\}^n$. For notational simplicity, we use simply i in a superscript instead of τ^i (the transcript after i-th query) in all above notations defined so far. For example, $\mathtt{H}^i(m)$ denotes the transcript based hash of m where the transcript is τ^i. We write New_v^i instead of $\mathsf{New}_v^{\tau^{i-1}}(x_i, y_i)$, which represents the set of all newly generated computable messages at node v immediately after obtaining i-th query-response. We also ignore the superscript τ^i completely when we all the queries have been made, i.e. $i = q'$. For example, we write $\mathrm{mc}_v(x)$ instead of $\mathrm{mc}_v^\tau(x)$, when τ is the final transcript, obtained at the end of all the queries.

- For any computable message m at v, we write $\mathrm{Fin}(m) := i$ to encode the final query index after which m is computable.

- For all m for which $m_{\mathsf{L}}, m_{\mathsf{R}}$ are τ-computable, we define $\mathrm{Fin}^*(m) = i$ such that $\max\{\mathrm{Fin}(m_{\mathsf{L}}), \mathrm{Fin}(m_{\mathsf{R}})\} = i$, (i.e. immediately after i-th query the final-input for the message m is computable).

5.1 Steps of Collision Analysis

Proper Internal Collision. We say that a **proper internal collision** happens at $v = (j, b)$ for a transcript τ if for some distinct messages m, m' at v, (i) $\mathtt{H}^\tau(m) = \mathtt{H}^\tau(m')$, (ii) $\mathsf{in}^\tau(m) \neq \mathsf{in}^\tau(m')$, and (iii) no collision happens for $\mathtt{H}^\tau_u$ for all $u \in V(\mathcal{T}_v)$, $u \neq v$. By using standard reduction, a collision of ABR must have proper internal collision at some node. So it is sufficient to bound the probability of a proper internal collision at the root node of ABR as $\mathtt{H}_v$ is identical to ABR_s where s denotes the level of the node v. We write $\mathsf{coll} := \mathsf{coll}_l$ to denote the proper internal collision at the root node of $\mathcal{T}$ of height l. The probability of collision of ABR_l can be then bounded as $\sum_{i \le l} 2^{l-i} \Pr(\mathsf{coll}_i)$.

Now, there are two types of collision which can happen for any proper collision at the root. Let us consider the i-th query. This query itself can generate two new computable messages for which the collision occurs. This is the first type of collision. Also, the hash output of one among the new computable messages generated by the i-th query can match with one of the hash outputs generated by the previous queries. We formalize them here:

▶ **Definition 14** (types of collision)**.**

- *We call a collision pair (M, M') **twin** at the i-th query, $i \in [q']$ if $M, M' \in \mathsf{New}^i$. In this case $\mathsf{in}^i_{v_i}(M) = \mathsf{in}^i_{v_i}(M') = x_i$, where v_i is the node where the i-th query is made.*
- *The collision pair is called **non-twin** at the i-th query if exactly one of M and M' is a member of New^i, and the other message is τ^{i-1}-computable.*

We write coll^i to denote that the proper internal collision happens at the i-th query. Moreover, if it is a twin-collision (or non-twin collision) we denote the event as $\mathsf{coll}^{i,\mathrm{tw}}$ (or $\mathsf{coll}^{i,\mathrm{ntw}}$ respectively). Thus,

$$\mathsf{coll} = \bigcup_{i \in [q']} \left(\mathsf{coll}^{i,\mathrm{ntw}} \cup \mathsf{coll}^{i,\mathrm{tw}}\right).$$

It is easy to see that twin-collision at the root node is not possible as a collision at the right child of the root node is necessary. In notation, $\mathsf{coll}^{i,\mathrm{tw}} = \emptyset$, whenever $v_i = \omega$.

5.1.1 Non-Twin Collision Analysis

For any non-root, non-leaf node v, we consider cross-collision between $\mathtt{H}_{-v}$ and $\mathtt{H}_\omega$. Let CC^i_v denote the set of all pairs (m, m') such that (i) m is a complete message, m' is a message for $\mathcal{T}_{-v}$ and (ii) $\mathtt{H}^i(m) = \mathtt{H}^i_{-v}(m')$. Now, a *non-twin collision* can happen at the i-th query (to the node v_i) if freshly generated hash of a message at v_i matches with the v_i-th message block of m' for a cross-collision pair (m, m') of CC^{i-1}_v. Thus,

$$\Pr(\mathsf{coll}^{i,\mathrm{ntw}}) \le \frac{\mathrm{mc}^{i-1}_v(x_i) \times |\mathsf{CC}^{i-1}_v|}{2^n}. \tag{2}$$

Now, if $v = \omega$ then the freshly generated hash at the root node is a hash. So, we have,

$$\Pr(\mathsf{coll}^{i,\mathrm{ntw}}) \le \frac{\mathrm{mc}^{i-1}_\omega(x_i) \times L}{2^n}. \tag{3}$$

5.1.2 Twin Collision Analysis

For any non-root, non-leaf node v and $\delta \in \{0,1\}^n \setminus \{0^n\}$, let $\mathsf{C}_{\delta,v}$, called δ-collision, denote the set of all pairs (m, m') such that $\mathtt{H}^\tau_{-v}(m) = \mathtt{H}^\tau_{-v}(m')$ and $m(v) \oplus m'(v) = \delta$. We have seen that no twin collision possible at the root node. We define a set

$$\Delta^i = \{\mathtt{H}^{i-1}(m_{\mathsf{R}}) \oplus \mathtt{H}^{i-1}(m'_{\mathsf{R}}) : m, m' \in \mathrm{MC}^{i-1}_{v_i}(x_i)\}.$$

Now,

$$\Pr(\mathsf{coll}^{i,\mathrm{tw}}) \leq \frac{\sum_{\delta\in\Delta} \mathrm{mc}_v^{i-1}(x_i) \times |\mathsf{C}_{\delta,v}^{i-1}|}{2^n}. \tag{4}$$

Note that the size of Δ can be at most $(\mathrm{mc}_v^{i-1}(x_i))^2$.

Thus, we have seen a collision analysis requires to bound the following random variables.

1. $\mathrm{mc}_v^{i-1}(x_i)$ for all i (and so for all nodes v),
2. L: load of the hash,
3. $|\mathrm{dom}_{-v}^{i-1}|$: load for $\mathcal{T}_{-v}$ which is required to bound the load L,
4. $|\mathsf{C}_{\delta,v}^{i-1}|$: size of δ-collision, and
5. $|\mathsf{CC}_v^{i-1}|$: size of cross-collision.

In the following subsection, we present the collision analysis of ABR_2 in which we only need the input multi-collision and load (which is also bounded in terms of input multi-collision). We also present a collision analysis of ABR_3 for which the above terms are present.

5.2 Collision Analysis of ABR_2 by ABR

As discussed above, we can assume that all queries to the compression functions at the leaf node have been made beforehand and let q denote the number of queries to each oracle. Let $\mathcal{L}_1, \mathcal{L}_2$ be two lists of outputs of the leaf node functions and let $\omega := (2, 1)$ denote the root (the only non-leaf node for $\mathcal{T}$ of height 2). Note that the proper collision at height 1 is the same as the collision of the lists $\mathcal{L}_1, \mathcal{L}_2$. The proper collision at a leaf node can happen with probability at most $q^2/2^n$.

So, we now consider collision at the root $(2, 1)$. For this, we now define a bad event mc_ω that $\mathrm{mc}_\omega^q > n$. Equivalently, the event can be expressed as $\mathsf{mc}(\mathcal{L}_1 \oplus \mathcal{L}_2) > n$. Note that we do not have any non-leaf node other than root node. So, the load for hash values L can be upper bounded as nq, given that mc_ω^q does not hold. Moreover, cross-collision and δ-collision is also not possible as we do not have any non-leaf, non-root node. Now, it is well known that

$$\Pr(\mathsf{mc}(\mathcal{L}_1 \oplus \mathcal{L}_2) > n) \leq \frac{q^2}{2^n}$$

(see [1] for details). Thus, the collision probability is bounded by $\frac{(n^2+2)q^2}{2^n}$.

5.3 Collision Analysis of $\mathsf{ABR}_h, h \geq 3$ by [1]

The proof of [1] is divided into two main parts: (i) bounding the load and (ii) bounding proper collision probability in terms of the load. ABR fix a parameter ρ (which is chosen to be $n+1$, however, the exact value is not relevant to our discussion). Let $L_{i,v} = \sum_{j\leq i, j\in Q_v} |\mathsf{New}\mathtt{H}_v^i|$ represent the total number of generated hash values at v after all i queries. If there is no collision (which is true while we consider proper internal collision), $L_{i,v}$ is same as the size of the set $|\mathrm{dom}_v^i|$. To bound load, ABR considered the following bad events (in our notations):

1. $\mathsf{bad}_{1,v}$: $\mathrm{mc}_v^{q'} > \rho$ at v. Let $\mathsf{bad}_1 := \cup_v \mathsf{bad}_{1,v}$.
2. $\mathsf{bad}_{2,v}$: $L_{q',v} \geq \rho q$. Let $\mathsf{bad}_2 := \cup_v \mathsf{bad}_{2,v}$.

Given $\mathsf{bad}_1, \mathsf{bad}_2$ do not hold, clearly $L \leq 2\rho q$.

5.3.1 Step-1: Bounding $\Pr(\mathsf{bad}_1)$

Let $\mathsf{bad}_{1,\le i} = \cup_{(j,b):j\le i}\mathsf{bad}_{1,v}$. So it is sufficient to bound $\Pr(\mathsf{bad}_{1,(j,b)} \wedge \neg\mathsf{bad}_{1,<j})$. Let us fix a query x at $v = (j,b)$. Now, ABR implicitly claimed the following:

▷ Claim ([1]). If $\mathrm{MC}^{q'}_{(j,b)}(x) \supseteq \{m_1,\ldots,m_\rho\}$ then $\mathsf{in}_{(j-1,2b)}(m_{i,\mathsf{R}})$'s are distinct.

We note that this claim is not correct. As there can be ρ multi-collision at node $(j-1,2b)$, each query can potentially give at most ρ multi-collision at node (j,b). Hence we can have ρ^2 multi-collision at node (j,b). Thus, a corrected version of the above claim requires to revise the parameter ρ depending on the level. So, we may redefine $\mathsf{bad}_{1,(j,b)}$: $\mathrm{mc}_v > \rho^j$ which could solve the issue. This is a fixable minor issue (but will have an impact on the claimed bound).

Now to continue with the bound, let us assume that $\mathrm{MC}^{q'}_v(x) \supseteq \{m_1,\ldots,m_\rho\}$ such that $\mathsf{in}(m_{i,\mathsf{R}})$'s are distinct and $x = (a,b)$. So we can choose ρ query indices out of q queries to $v_2 := v_\mathsf{R}$ in $\binom{q}{\rho}$ ways. For any such choices of ρ tuple $(i_1,i_2,\ldots,i_\rho)$ (all queried to v_2), we have

$$\Pr(f(x_{i_1}) \oplus \mathtt{H}(m_{1,\mathsf{RR}}) = b,\ldots,f(x_{i_\rho}) \oplus \mathtt{H}(m_{\rho,\mathsf{RR}}) = b) = \frac{2\rho q}{2^{n\rho}}$$

as there ρq many choices of $\mathtt{H}(m_{i,\mathsf{RR}})$ values (as we assume the load at v_{RR} is less than $2\rho q$). However, the above is true when we consider the cases where $\mathrm{Fin}^*(m_i) = j_i$ where $v_{j_i} = v_2$ for all i. The most important case in which the input multi-collision is contributed due to the final queries which are not on right child is not considered in the proof by [1].

5.3.2 Step-2: Bounding $\Pr(\mathsf{bad}_2)$

Let $\mathsf{bad}_{2,\le i} = \cup_{(j,b):j\le i}\mathsf{bad}_{2,v}$. So it is sufficient to bound

$$\Pr(\mathsf{bad}_{2,(j,b)} \wedge \neg\mathsf{bad}_{1,<j} \wedge \neg\mathsf{bad}_1 \wedge \neg\mathsf{coll}).$$

The main idea to bound the above probability is to bound the expected number of newly generated hash at $v = (j,b)$ over all queries. Then the bad event probability can be bounded by applying Markov's inequality. We have already seen that

$$\mathbb{E}_y(|\mathsf{New}^i_v| \mid \tau^{i-1}) = \frac{\mathrm{mc}^\tau_{v_i}(x_i) \times |\mathrm{dom}^i_{v-v_i}|}{2^n}.$$

Moreover, we have shown that bounding $|\mathrm{dom}^i_{v-v_i}|$ becomes more complex when v_i is neither v nor a child of v (see Example 13). [1] tried to argue in a different way. ABR showed a bound expectation of load due to all queries of its children (see Example 12). Then, they continued this argument for two levels up (i.e. for the queries on grandchildren as we consider in Example 13). However, they did not analyze this case properly. In particular, they did not consider to bound the $\mathrm{mc}(\mathrm{ran}^\tau_{v_\mathsf{L}} \oplus \mathrm{ran}^\tau_{v_{\mathsf{RL}}} \oplus \bar{f}_{v_\mathsf{R}})$. Finally, they claimed the general case by using induction which is clearly unverifiable.

5.3.3 Step-3: Proving Collision in terms of Load

ABR stated that as analyzed for ABR_2, given (i) no collision for all primitive, (ii) $\neg\mathsf{bad}_{1,\le l}$ and (iii) $\neg\mathsf{bad}_{2,\le l}$, the proper internal collision probability at the root node is $\mathbb{E}(L^2)/2^n$ where L is the total number computable hash values.

There is a fundamental gap in the high level of the proof. As ABR did not explain anything supporting his claim, we show that this statement is not true in general. In particular, we show (in the next subsection) a *hash mode based on* 2*-to-*1 *compression function whose load is at most* q^2 *(for any* q*-query adversary), however, a collision can be found in* $O(n)$ *queries.* So the above claim cannot be made in general.

5.3.4 Missing Step: Twin-Collision Analysis

We find that the twin-collision analysis of the ABR hash is missed completely. The bound for δ-collision is not obvious and it requires bounding the probability of some more bad events. In the following section, we have analyzed ABR_3 in which the twin-collision analysis requires a bad event dealing with the multi-collision of xor of random oracle compression function outputs for two distinct inputs. We do not know any method to bound the number of cross-collision pairs for a general height tree.

5.4 Relationship between Load and Collision Probability

A hash function with a high load is unlikely to be collision-resistant. For example, $\mathsf{xor}(x_1, \ldots, x_r) = f_1(x_1) \oplus \cdots \oplus f_r(x_r)$ has load 2^r after 2 queries to each oracle f_i. It is easy to see that the hash function xor is not collision-resistant. Let $r = n$. Then, after making two queries to each function, we have sufficiently many computable messages. It is then very easy to find computable collision pairs by solving a linear system of equations. In general, if the load becomes the order of $2^{n/2}$ then one may expect a collision. However, the converse need not be true. In other words, we have a hash function where load can not be high, but still, a collision pair can be generated efficiently.

Example of Collision Insecure Hash Functions with Low Load

Let MD^f be the MD hash which takes n blocks and initial value is also replaced by one message block (so exactly $n-1$ calls of f is required). We define $MD^f_n(M) = MD^{f_1}(M) \| \cdots \| MD^{f_n}(M)$ which is n^2-to-n^2 hash function. Now we define a hash function $\mathtt{H}(M_1, M_2)$ for $M_1, M_2 \in \{0,1\}^{n^2}$:

1. Let $(C_1, C_2) = (MD^f_n(M_1) \oplus M_2, MD^g_n(C_1) \oplus M_1)$ (two round LR construction which is invertible).
2. Let $h_1, \ldots, h_n$ be $2n$-to-n functions. The final hash output is defined as $h_1(x_1) \oplus \cdots \oplus h_n(x_n)$ where $C_1 \| C_2 = x_1 \| \cdots \| x_n$, $x_i \in \{0,1\}^{2n}$.

Note that we cannot compute (C_1, C_2) for more than q^2 messages assuming there is no collisions in f and g functions. So, $L(q) \le q^2$ for any q-query adversary.

A Collision Attack. Now, we construct a collision finding algorithm for the above hash. It first finds collision pair for xor function $h_1 \oplus \cdots \oplus h_n$ (can be achieved easily by making $2n$ queries altogether). Let (C, C') be a collision pair. We can easily invert C and C' to obtain M and M' respectively. Clearly, (M, M') is a computable collision pair.

6 Analysis of ABR of height 3

In this section, we show that the ABR_3 construction achieves birthday security. In particular, we prove the following theorem:

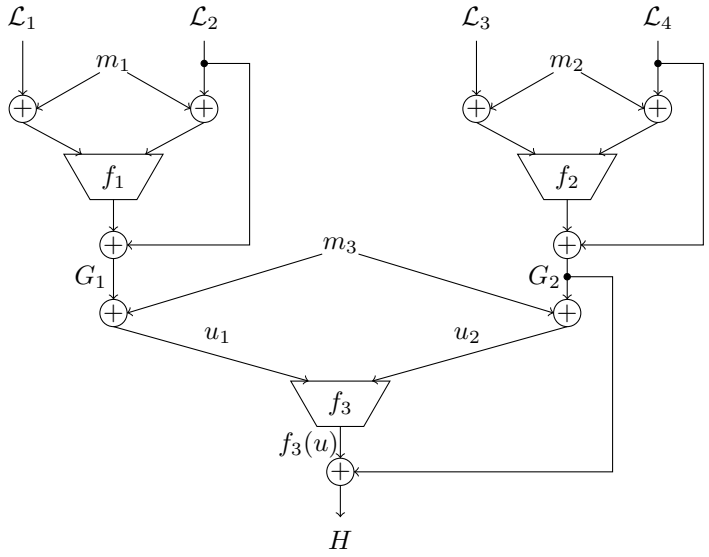

Figure 5 ABR_3 according to our new notation when the query $u = (u_1||u_2)$ is made to f_3.

▶ **Theorem 15** (collision theorem for ABR_3). *For any adversary $\mathcal{A}$ making at most q queries to each compression function modeled to be random oracle, we have*

$$\mathbf{Adv}^{\mathsf{coll}}_{\mathrm{H}^f}(\mathcal{A}) \leq \frac{6n^5q^2 + 3n^4q^2 + 2n^4q + 2n^2q^2 + 13q^2}{2^n}. \tag{5}$$

Let $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4$ be the four lists of size q each corresponding to the outputs of $f_{1,1}, f_{1,2}, f_{1,3}, f_{1,4}$ respectively. We can assume that these lists are given to the adversary at the beginning of the game. This is without loss of generality as the inputs to $f_{1,i}$'s are independent from the rest of the transcripts. Also, for ease of notation, from now on we denote $f_{2,1}$ by f_1, $f_{2,2}$ by f_2 and $f_{3,1}$ by f_3. If the input to any of the functions is $u = (u_1, u_2)$, we define $u^{\oplus} = u_1 \oplus u_2$. Also, if $f_3(u) = v$, then we define $\bar{f}_3(u) = u^{\oplus} \oplus v$. As f_3 is a random oracle, the output distributions of $\bar{f}_3$ are uniform and independent. Let Q_j be the set of queries to f_j. We assume $|Q_j| = q$ for $j = 1, 2, 3$. Also, let Q_j^i denote the set of queries to Q_j up to the i-th query (including the i-th one). Let $\mathcal{G}_1 = O_{(2,1)}$ denote the set of intermediate hash outputs at node $(2, 1)$ and $\mathcal{G}_2 = O_{(2,2)}$. Let $\mathcal{H}$ denote the set of final hash outputs of ABR_3. Refer to Figure 5 for a pictorial representation. We follow the general approach as described before. We have already shown the collision bound for ABR_2 and so it is sufficient to bound proper collision at the root for ABR_3.

As we have seen above, the collision analysis requires us to bound some random variables. We first define some bad events to bound these random variables.

▶ **Definition 16** (list collision). *The first bad event we consider is:*

- *$B0$: There exists a collision in at least one of the lists $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, $\mathcal{L}_4$, $\{f_1(u) : u \in Q_1\}$, $\{f_2(u) : u \in Q_2\}$, $\{f_3(u) : u \in Q_3\}$.*

Since f is modeled as a random function, the collision probability in any of the lists is at most $q^2/2^n$. Hence, $\Pr(B0) \leq 7q^2/2^n$.

▶ **Definition 17** (bad event on input multi-collision). *We define the following bad events:*

- *$B1$: $\mathrm{mc}(\mathcal{L}_1 \oplus \mathcal{L}_2) > n$, or $\mathrm{mc}(\mathcal{L}_3 \oplus \mathcal{L}_4) > n$,*
- *$B2 : \mathrm{mc}(\mathcal{G}_1 \oplus \mathcal{G}_2) > n^2$.*

We now state some simple observations related to input multi-collision:

1. Given that $B1$ does not hold, $\mathrm{mc}_{(2,1)}, \mathrm{mc}_{(2,2)} \leq n$ and so $|\mathcal{G}_1|, |\mathcal{G}_2| \leq nq$.
2. Given that $B2$ does not hold, $\mathrm{mc}_{(3,1)} \leq n^2$ and so $L_{(3,1)} \leq n^2 q$.
3. Note, $|\mathrm{dom}(\mathcal{T}_{-(2,1)})|, |\mathrm{dom}(\mathcal{T}_{-(2,2)})| \leq nq^2$. So, $\mathbb{E}(L_{(2,1)}), \mathbb{E}(L_{(2,2)}) \leq n^2 q^3/2^n$. By Markov's inequality, $\Pr(L > 3n^2 q) \leq 2q^2/2^n$ ($3n^2 q$ because we include $L_{(3,1)}$ as well).
4. By using a similar argument as we applied for multi-collision, we have $\Pr(B1) \leq 2q^2/2^n$.
5. Now, given that $B1$ does not hold and $B2$ holds, there must exist at least n distinct inputs to f_2 leading to n^2 input multi-collision. So, we can similarly prove $\Pr(B2) \leq q^2/2^n$.

We say that bad_{mc} holds if either $B1$ or $B2$ happens, or $L > 3n^2 q$. Then, from above, $\Pr(\mathsf{bad}_{mc}) \leq 3q^2/2^n$. We now define bad events which would be used to bound cross-collision.

▶ **Definition 18** (bad event on cross-collision)**.** *We define the following bad events:*

- $B3$*:* $|\{(G_2, f_3(u), H) : G_2 \oplus f_3(u) \oplus H = 0; G_2 \in \mathcal{G}_2, u \in Q_3, H \in \mathcal{H}\}| > 3n^4 q$.
- $B4$*:* $|\{(G_1, \bar{f}_3(u), H) : G_1 \oplus \bar{f}_3(u) \oplus H = 0; G_1 \in \mathcal{G}_1, u \in Q_3, H \in \mathcal{H}\}| > 3n^4 q$.

We say that bad_{cc} *holds if any one of the above happens.*

If the i-th query is made at f_2, an intermediate hash output G_2 generated at this level due to this query can match with a query u already done to f_3 to generate a final hash output H which was already previously generated by the first $i-1$ queries. The event $B3$ implies that the number of such triplets $(G_2, f_3(u), H)$ is more than $3n^4 q$. $B4$ has a similar implication when we consider $\mathcal{G}_1$ instead of $\mathcal{G}_2$.

▶ **Lemma 19.** $\Pr(\mathsf{bad}_{cc} \wedge \neg \mathsf{bad}_{mc}) \leq 2q^2/2^n$.

Proof. $\Pr(\mathrm{mc}(\mathcal{G}_2 \oplus \mathrm{ran}(f_3)) > n^2) \leq q^2/2^n$. The proof is similar to that of event $B2$. Hence, for a fixed $H \in \mathcal{H}$, we have

$$\Pr[|\{(G_2, f_3(u), H) : G_2 \oplus f_3(u) \oplus H = 0; G_2 \in \mathcal{G}_2, u \in Q_3\}| > n^2] \leq q^2/2^n.$$

Now, there are $3n^2 q$ choices for H. Therefore, $\Pr(B3 \wedge \neg \mathsf{bad}_{mc}) \leq q^2/2^n$. A similar argument works for $B4$. Hence,

$$\Pr(\mathsf{bad}_{cc} \wedge \neg \mathsf{bad}_{mc}) \leq 2q^2/2^n. \qquad \blacktriangleleft$$

Given that bad_{cc} does not hold, $|\mathsf{CC}_{(2,1)}| \leq 3n^4 q$ (or $|\mathsf{CC}_{(2,2)}| \leq 3n^4 q$ respectively). We finally define bad events which would be used to bound δ-collision pairs.

▶ **Definition 20** (bad event on δ-collision)**.** *We define the following bad event:*

- $B5$*:* $\mathsf{mc}(\bar{f}_3(u) \oplus \bar{f}_3(u')) > n$.

We say that bad_δ *holds if the above happens.*

▶ **Lemma 21.** $\Pr(\mathsf{bad}_\delta) \leq \frac{q^2}{2^n}$.

Proof. Since $f_3(u)$ is random, $\bar{f}_3(u) = f_3(u) \oplus u^{\oplus}$ is also random. Therefore, bounding $B5$ is similar to bounding $B1$. ◀

Given that bad_δ does not hold, $|\mathsf{C}_\delta| \leq n$. Let $\mathsf{bad} = B0 \cup \mathsf{bad}_{mc} \cup \mathsf{bad}_{cc} \cup \mathsf{bad}_\delta$. Then, $\Pr(\mathsf{bad}) \leq \frac{13q^2}{2^n}$.

Collision Analysis

We assume that bad does not hold. Since $\mathsf{coll} = \bigcup_{i\in[q],v\in V\setminus\mathcal{L}} \left(\mathsf{coll}_v^{i,\mathrm{ntw}} \cup \mathsf{coll}_v^{i,\mathrm{tw}}\right)$, we need to bound $\mathsf{coll}_v^{i,\mathrm{ntw}}$ and $\mathsf{coll}_v^{i,\mathrm{tw}}$ for $v = (2,1),(2,2),(3,1)$. In the following lemmas, we bound them, assuming bad does not occur. We already know that $\mathsf{coll}_{(3,1)}^{i,\mathrm{tw}}$ does not occur.

▶ **Lemma 22.** $\Pr(\mathsf{coll}_{(3,1)}^{i,\mathrm{ntw}}|\neg\mathsf{bad}) \le \frac{3n^4q}{2^n}$.

Proof. As seen above in equation 3, $\Pr(\mathsf{coll}_{(3,1)}^{i,\mathrm{ntw}}) \le \dfrac{\mathrm{mc}_{(3,1)}^{i-1}(x_i) \times L}{2^n}$.

Given ¬bad, $\mathrm{mc}_{(3,1)} \le n^2$ and $L \le 3n^2q$. Hence, $\Pr(\mathsf{coll}_{(3,1)}^{i,\mathrm{ntw}}|\neg\mathsf{bad}) \le \dfrac{3n^4q}{2^n}$. ◀

▶ **Lemma 23.** $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{ntw}}|\neg\mathsf{bad}) \le \frac{3n^5q}{2^n}$.

Proof. As seen above in equation 3, $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{ntw}}) \le \dfrac{\mathrm{mc}_{(2,1)}^{i-1}(x_i) \times |\mathsf{CC}_{(2,1)}^{i-1}|}{2^n}$.

Given ¬bad, $\mathrm{mc}_{(2,1)} \le n$ and $|\mathsf{CC}_{(2,1)}| \le 3n^4q$. Hence, $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{ntw}}|\neg\mathsf{bad}) \le \dfrac{3n^5q}{2^n}$. ◀

▶ **Lemma 24.** $\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{ntw}}|\neg\mathsf{bad}) \le \frac{3n^5q}{2^n}$.

Proof. This proof is similar to that of the previous lemma.

$\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{ntw}}) \le \dfrac{\mathrm{mc}_{(2,2)}^{i-1}(x_i) \times |\mathsf{CC}_{(2,2)}^{i-1}|}{2^n}$.

Given ¬bad, $\mathrm{mc}_{(2,2)} \le n$ and $|\mathsf{CC}_{(2,2)}| \le 3n^4q$. Hence, $\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{ntw}}|\neg\mathsf{bad}) \le \dfrac{3n^5q}{2^n}$. ◀

▶ **Lemma 25.** $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{tw}}|\neg\mathsf{bad}) \le \frac{n^4}{2^n}$.

Proof. As seen above in equation 4, $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{tw}}) \le \dfrac{\sum_{\delta\in\Delta} \mathrm{mc}_{(2,1)}^{i-1}(x_i) \times |\mathsf{C}_{\delta,(2,1)}^{i-1}|}{2^n}$.

Given ¬bad, $\mathrm{mc}_{(2,1)} \le n$, $|\Delta| \le (\mathrm{mc}_{(2,1)}^{i-1}(x_i))^2 \le n^2$ and $|\mathsf{C}_{\delta,(2,1)}| \le n$. Hence,

$\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{tw}}|\neg\mathsf{bad}) \le \dfrac{n^4}{2^n}$. ◀

▶ **Lemma 26.** $\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{tw}}|\neg\mathsf{bad}) \le \frac{n^4}{2^n}$.

Proof. This proof is similar to that of the previous lemma.

$\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{tw}}) \le \dfrac{\sum_{\delta\in\Delta} \mathrm{mc}_{(2,2)}^{i-1}(x_i) \times |\mathsf{C}_{\delta,(2,2)}^{i-1}|}{2^n}$.

Given ¬bad, $\mathrm{mc}_{(2,2)} \le n$, $|\Delta| \le (\mathrm{mc}_{(2,2)}^{i-1}(x_i))^2 \le n^2$ and $|\mathsf{C}_{\delta,(2,2)}| \le n$. Hence,

$\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{tw}}|\neg\mathsf{bad}) \le \dfrac{n^4}{2^n}$. ◀

From the above lemmas, we have

$$\Pr(\mathsf{coll}|\neg\mathsf{bad}) \le \sum_{i\in[q],v\in V\setminus\mathcal{L}} \Pr(\mathsf{coll}_v^{i,\mathrm{ntw}}|\neg\mathsf{bad}) + \Pr(\mathsf{coll}_v^{i,\mathrm{tw}}|\neg\mathsf{bad}) \le \frac{6n^5q^2 + 3n^4q^2 + 2n^4q}{2^n}.$$

Therefore, $\Pr(\mathsf{coll}) \le \Pr(\mathsf{coll}|\neg\mathsf{bad}) + \Pr(\mathsf{bad}) \le \dfrac{6n^5q^2 + 3n^4q^2 + 2n^4q + 13q^2}{2^n}$.

Note that we have bound the proper collision probability at the root for ABR_3. Since $B0$ does not occur, collision does not occur at the leaf node. As seen in section 5.2, the probability that proper collision occurs at node $(2,1)$ (resp. $(2,2)$) is bounded above by $\frac{n^2q^2}{2^n}$. Hence, the theorem is proved.

7 Conclusion

In this paper, we revisit the collision security of the ABR hash. We found that there is a serious gap in the analysis of collision security. Some missing and important cases have also been identified. In this paper, we have shown collision security for level 3. Several new bad events have been identified in ABR_3 which were not considered for the general hash. We leave the collision security analysis open for general hash. Thus, the optimality of Stam's bound remains open for an arbitrary domain hash.

We have also found that the ABR hash cannot have any non-trivial local opening which can give birthday bound security. This shows a limitation in terms of applications in local opening. In particular, the efficient local opening proposed by [1] can be broken in $O(2^{n/2l})$ query complexity.

References

1 Elena Andreeva, Rishiraj Bhattacharyya, and Arnab Roy. Compactness of hashing modes and efficiency beyond merkle tree. In *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*, pages 92–123. Springer, 2021.

2 John Black, Martin Cochran, and Thomas Shrimpton. On the impossibility of highly-efficient blockcipher-based hash functions. In *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 526–541. Springer, 2005.

3 Ivan Damgård. A design principle for hash functions. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.

4 Yevgeniy Dodis, Dmitry Khovratovich, Nicky Mouha, and Mridul Nandi. T5: Hashing five inputs with three compression calls. In *2nd Conference on Information-Theoretic Cryptography, ITC 2021, July 23-26, 2021, Virtual Conference*, pages 24:1–24:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

5 Bart Mennink and Bart Preneel. Efficient parallelizable hashing using small non-compressing primitives. *Int. J. Inf. Sec.*, 15(3):285–300, 2016.

6 Ralph C. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and Privacy*, pages 122–134. IEEE Computer Society, 1980.

7 Ralph C. Merkle. One way hash functions and DES. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.

8 Phillip Rogaway and John P. Steinberger. Constructing cryptographic hash functions from fixed-key blockciphers. In *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 433–450. Springer, 2008.

9 Phillip Rogaway and John P. Steinberger. Security/efficiency tradeoffs for permutation-based hashing. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 220–236. Springer, 2008.

10 Thomas Shrimpton and Martijn Stam. Building a collision-resistant compression function from non-compressing primitives. In *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 643–654. Springer, 2008.

11 Martijn Stam. Beyond uniformity: Better security/efficiency tradeoffs for compression functions. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 397–412. Springer, 2008.

12 John P. Steinberger. Stam's collision resistance conjecture. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 597–615. Springer, 2010.

13 John P. Steinberger, Xiaoming Sun, and Zhe Yang. Stam's conjecture and threshold phenomena in collision resistance. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 384–405. Springer, 2012.

14 David A. Wagner. A generalized birthday problem. In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.

A Fully-Constructive Discrete-Logarithm Preprocessing Algorithm with an Optimal Time-Space Tradeoff

Lior Rotem
School of Computer Science and Engineering, Hebrew University of Jerusalem, Israel

Gil Segev
School of Computer Science and Engineering, Hebrew University of Jerusalem, Israel

Abstract

Identifying the concrete hardness of the discrete logarithm problem is crucial for instantiating a vast range of cryptographic schemes. Towards this goal, Corrigan-Gibbs and Kogan (EUROCRYPT '18) extended the generic-group model for capturing "preprocessing" algorithms, offering a tradeoff between the space S required for storing their preprocessing information, the time T required for their online phase, and their success probability. Corrigan-Gibbs and Kogan proved an upper bound of $\widetilde{O}(ST^2/N)$ on the success probability of any such algorithm, where N is the prime order of the group, matching the known preprocessing algorithms.

However, the known algorithms assume the availability of truly random hash functions, without taking into account the space required for storing them as part of the preprocessing information, and the time required for evaluating them in essentially each and every step of the online phase. This led Corrigan-Gibbs and Kogan to pose the open problem of designing a discrete-logarithm preprocessing algorithm that is fully constructive in the sense that it relies on explicit hash functions whose description lengths and evaluation times are taken into account in the algorithm's space-time tradeoff.

We present a fully constructive discrete-logarithm preprocessing algorithm with an asymptotically optimal space-time tradeoff (i.e., with success probability $\widetilde{\Omega}(ST^2/N)$). In addition, we obtain an algorithm that settles the corresponding tradeoff for the computational Diffie-Hellman problem. Our approach is based on derandomization techniques that provide rather weak independence guarantees. On the one hand, we show that such guarantees can be realized in our setting with only a minor efficiency overhead. On the other hand, exploiting such weak guarantees requires a more subtle and in-depth analysis of the underlying combinatorial structure compared to that of the known preprocessing algorithms and their analyses.

2012 ACM Subject Classification Security and privacy → Information-theoretic techniques; Security and privacy → Mathematical foundations of cryptography; Theory of computation → Computational complexity and cryptography

Keywords and phrases Discrete logarithm, Preprocessing

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.12

Funding Supported by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253).
Lior Rotem: Supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

1 Introduction

Identifying the concrete hardness of the discrete logarithm problem in prime-order groups is crucial for instantiating a vast range of cryptographic schemes. Shoup's seminal work [23] introduced the generic-group model, capturing all computations that do not exploit any

specific property of the representation of the underlying group, and provided a tight bound on the hardness of the discrete logarithm problem in this model. Specifically, Shoup proved that any generic-group algorithm that runs in time at most T (thus, in particular, performs at most T group operations), computes the discrete logarithm of a uniformly-distributed group element with probability $O(T^2/N)$, where N is the prime order of the group.[1]

Although generic-group algorithms seem somewhat restricted, the generic hardness of the discrete logarithm problem may nevertheless be used for setting concrete security parameters in any group in which non-generic discrete logarithm algorithms do not seem to outperform the known generic ones (most notably, in popular elliptic-curve groups [16, 13]). However, as recently observed by Corrigan-Gibbs and Kogan [6], the bound established by Shoup and Maurer does not apply to "preprocessing" algorithms, as introduced by Hellman in the context of the function inversion problem [15, 12, 7]. For the discrete logarithm problem, a preprocessing algorithm may first preprocess the group in an offline phase. Then, in an online phase, the algorithm receives a group element $h \in \mathbb{G}$, and may use the preprocessing information to compute its discrete logarithm. The efficiency of such algorithms is measured via the tradeoff between the space S required for storing their preprocessing information, the time T required for their online phase, and their success probability.

Motivated by elegant preprocessing algorithms for the discrete logarithm problem due to Lee, Cheon, and Hong [17] and by Bernstein and Lange [4], Corrigan-Gibbs and Kogan extended the generic-group model to capture preprocessing algorithms, and proved an upper bound on the success probability of any such algorithm in computing discrete logarithms. Specifically, for essentially any S and T, they proved that any preprocessing algorithm computes the discrete logarithm of a uniformly-distributed group element with probability $\widetilde{O}(ST^2/N)$. Alternatively, denoting by ϵ the success probability of such algorithms, they proved the lower bound $ST^2 = \widetilde{\Omega}(\epsilon N)$ on the required space and time resources.

The tradeoff established by Corrigan-Gibbs and Kogan matches the performance provided by the algorithms of Lee et al. and of Bernstein and Lange. However, these algorithms assume the availability of truly random hash functions, without taking into account the space required for storing them as part of the preprocessing information, and the time required for evaluating them in essentially each and every step of the online phase.[2] A standard approach in the design and analysis of algorithms for eliminating this assumption is to rely on derandomization techniques based on k-wise independent hash functions, guaranteeing that their outputs are independently and uniformly distributed when restricted to any set of most k inputs. Unfortunately, as we discuss in Section 1.3, for the level k of independence that seems required for the probabilistic analysis of the above two algorithms, explicit constructions of k-wise independent hash functions inherently result in a significant increase either in the space required for storing them as part of the preprocessing information or in the time required for evaluating them in the online phase [24].

This state of affairs has led Corrigan-Gibbs and Kogan to pose the open problem of designing a preprocessing algorithm for computing discrete logarithms that is fully constructive in the sense that it relies on explicit hash functions whose description lengths and evaluation times are taken into account in the algorithm's space-time tradeoff. That is:

---

[1] An alternative generic-group model was later introduced by Maurer [18], admitting the same tight bound on the hardness of the discrete logarithm problem.

[2] The lower bound of Corrigan-Gibbs and Kogan allows the offline and online phases to share an arbitrary-long common random string which is not accounted for in the space required for storing the preprocessing information. Thus, on the one hand, their lower bound applies even to algorithms that assume the availability of truly random hash functions. On the other hand, however, when taking the required storage into account, a comparable yet more direct solution is to just store the discrete logarithms of all group elements.

Is there an explicit (i.e., fully constructive) *preprocessing algorithm for computing discrete logarithms that matches the* $ST^2 = \widetilde{\Theta}(\epsilon N)$ *tradeoff?*

This question is in fact relevant not only to the discrete logarithm problem, but also to various other problems in prime-order groups for which the known preprocessing algorithms assume the availability of truly random functions without taking into account the space required for storing them and the time required for evaluating them. These include, in particular, the computational Diffie-Hellman problem, for which Corrigan-Gibbs and Kogan proved a similar $ST^2 = \widetilde{\Omega}(\epsilon N)$ lower bound.

1.1 Existing Approaches

The seminal work of Fiat and Naor [12] considered the function-inversion variant of this problem, given that Hellman's preprocessing function inversion algorithm [15] assumed the availability of truly random hash functions in a similar manner. Fiat and Naor presented an explicit algorithm that relies on concrete hash functions, and were able to match the tradeoff established by Hellman. The algorithm of Fiat and Naor can be seen as an explicit preprocessing algorithm for computing discrete logarithms. However, the bound that it achieves when applied to the discrete logarithm problem is $S^2T = \Theta(\epsilon N^2)$, far from matching the $ST^2 = \widetilde{\Theta}(\epsilon N)$ tradeoff. It is worth stressing that this gap stems from the fact that the setting considered by Fiat and Naor is much more general: Their algorithm can invert *any* function, and thus cannot exploit the algebraic structure of the underlying group in the discrete logarithm problem. Nevertheless, one could hope that relying on the same ideas of Fiat and Naor, one could make the algorithms of Lee, Cheon, and Hong [17] and of Bernstein and Lange [4] explicit. Unfortunately, as we discuss in Section 1.3, the standard amortization technique that enabled Fiat and Naor to rely on concrete hash functions while still matching Hellman's tradeoff does not seem applicable for the known preprocessing algorithms for the discrete logarithm problem.

The recent work of Maurer, Portman, and Zhu [19] considered the task of replacing the random function assumed by Lee, Cheon, and Hong [17] and of Bernstein and Lange [4] by an explicit k-wise independent hash function for a suitable choice of k. However, the focus of their work is different, as they consider idealized models which only account for the number of oracle queries that the online algorithm issues. As a result, their analysis does not take into account the *time* required for evaluating the k-wise independent hash function, and they only consider the space required for representing it. As we discuss in Section 1.3, instantiating their approach in the standard model results in a space-time tradeoff that is far from optimal.

Finally, it should be noted that preprocessing algorithms that assume the availability of truly random hash functions can be viewed as algorithms within the random-oracle model [2].[3] When instantiating the random oracle with cryptographic hash functions, most applications rely on the standard assumption that such functions are "sufficiently random" with respect to a polynomial (or, say, moderately super-polynomial) number of queries. However, the known preprocessing algorithms issue a nearly exponential number of queries (e.g., $N^{1/3}$ queries), and this holds even when considering only the queries issued in their online phase. From the theoretical perspective, this requires a substantially stronger assumption regarding the heuristic security of cryptographic hash functions. Moreover, from the more

[3] See also [25, 11, 5], and the references therein, for the related line of work on bounding the usefulness of auxiliary inputs in the random-oracle model.

practical perspective of setting concrete security parameters against preprocessing attacks, this unnecessarily ties the assumed concrete security of discrete logarithm problem (and of additional related problems) to that of cryptographic hash functions.

1.2 Our Contributions

In this work we resolve the above-stated question by presenting an explicit (i.e., fully constructive) discrete logarithm preprocessing algorithm that is asymptotically optimal in terms of its space-time tradeoff. In fact, our algorithm does not explicitly settle the space-time tradeoff only for the discrete logarithm problem, but also yields an explicit algorithm that settles the corresponding tradeoff for the computational Diffie-Hellman problem (CDH).

Within the unit-cost RAM model, which is the standard model for analyzing the efficiency of explicit data structures and algorithms (see Section 2), we prove the following theorem:

▶ **Theorem 1** (informal). *For any integers S and T such that $S - T = \Omega(S)$, there exists an explicit algorithm $A = (A_0, A_1)$ such that for any cyclic group $(\mathbb{G}, N, g)$ it holds that*

$$\Pr_{x \leftarrow \mathbb{Z}_N} \left[A_1 \left(A_0(\mathbb{G}, N, g), g^x \right) = x \right] = \widetilde{\Omega}\left(\frac{S \cdot T^2}{N} \right),$$

where the offline algorithm A_0 outputs S bits of preprocessing information, and the online algorithm A_1 runs in time T.

Note that our algorithm A consists of a pair (A_0, A_1) of algorithms: The offline algorithm A_0 takes as input the description $(\mathbb{G}, N, g)$ of a cyclic group of order N that is generated by $g \in \mathbb{G}$ and produces the preprocessing information, and the online algorithm A_1 takes as input a uniformly-distributed group element $g^x \in \mathbb{G}$ (together with the preprocessing information) and tries to compute its discrete logarithm $x \in \mathbb{Z}_N$ with respect to the given generator g. Similarly to the previously-known algorithms [17, 4], our algorithm does not rely on any specific property of the representation of the underlying group, and can be formally presented within Shoup's generic-group model [23].

In addition, note that we require the parameters S and T to satisfy $S - T = \Omega(S)$ (i.e., we require that $(1-\alpha)S \geq T$ for some constant $\alpha > 0$), and this results from the additional space overhead we incur for explicitly storing descriptions of hash functions.[4] This is a somewhat natural restriction given the nature of preprocessing algorithms (relying on preprocessing information in order to reduce the online running time), which captures in particular the choice of $S = \Theta(N^{1/3})$ and $T = \Theta(N^{1/3})$ that balances the space and time resources of the algorithm, as well as any other choice of $S = \Theta(N^{1-2\beta})$ and $T = \Theta(N^{\beta})$ for $\beta \geq 1/3$.

Finally, note that any preprocessing algorithm for the discrete logarithm problem directly yields a preprocessing algorithm for the computational Diffie-Hellman (CDH) problem with the same space-time tradeoff. Thus, given that Corrigan-Gibbs and Kogan [6] additionally proved the lower bound $ST^2 = \widetilde{\Omega}(\epsilon N)$ for the CDH problem with preprocessing, the following corollary of Theorem 1 provides an explicit preprocessing algorithm that asymptotically matches the optimal tradeoff for the CDH problem as well:

▶ **Corollary 2** (informal). *For any integers S and T such that $S - T = \Omega(S)$, there exists an explicit algorithm $A = (A_0, A_1)$ such that for any cyclic group $(\mathbb{G}, N, g)$ it holds that*

$$\Pr_{x,y \leftarrow \mathbb{Z}_N} \left[A_1 \left(A_0(\mathbb{G}, N, g), g^x, g^y \right) = g^{xy} \right] = \widetilde{\Omega}\left(\frac{S \cdot T^2}{N} \right),$$

where the offline algorithm A_0 outputs S bits of preprocessing information, and the online algorithm A_1 runs in time T.

[4] An interesting technical question is whether the requirement $S - T = \Omega(S)$ can be avoided while still matching the $ST^2 = \widetilde{\Theta}(\epsilon N)$ tradeoff with an explicit algorithm.

1.3 Overview of Our Approach

Our starting point: Preprocessing with a truly random function

The starting point for our explicit preprocessing algorithms is the approach which underlies the preprocessing algorithms of Lee, Cheon, and Hong [17] and Bernstein and Lange [4]. This approach relies on the existence of a truly random hash function $f : \mathbb{G} \to \mathbb{Z}_N$, shared between the preprocessing algorithm A_0 and the online algorithm A_1. This function defines a random walk on the elements of $\mathbb{G}$ via the step function $h \to h \cdot g^{f(h)}$.

The preprocessing algorithm A_0 preforms S such random walks, starting from S uniformly-random group elements $g^{\alpha_1}, \ldots, g^{\alpha_S}$, and taking T steps in each walk. It then stored the end point h_i of each walk, together with its discrete logarithm β_i with respect to g. Note that A_0 can indeed compute β_i, since $\beta_i = \alpha_i + \sum_{j=1}^{T} f^{(j)}(g^{\alpha_i})$, where $f^{(1)}(\cdot) = f(\cdot)$ and $f^{(j)}(\cdot) = f(f^{(j-1)}(\cdot))$ for $j \geq 2$. The endpoints $h_1, \ldots, h_S$ and their respective discrete logarithms $\beta_1, \ldots, \beta_S$ are passed as the state to the online algorithm A_1.

The online algorithm A_1 receives as input the above state and a challenge group element $h = g^x$, and its goal is to compute x. To this end, it performs a random walk of length at most $2T$, starting from the challenge h. The hope is that eventually, this walk will "hit" one of the stored endpoints $h_1, \ldots, h_S$. Say that the online walk hits h_i after ℓ steps. In this case, we know that $g^{x+\sum_{j=1}^{\ell} f^{(j)}(h)} = h_i = g^{\beta_i}$. Since g is a generator of the group, this implies that $x = \beta_i - \sum_{j=1}^{\ell} f^{(j)}(h)$. Since β_i and h are both known to A_1, it can compute and output x.

The description length of the state passed from A_0 to A_1 is roughly S (assuming that the function f does not need to be stored as part of the state). The computation executed by A_1 involves roughly $2T$ invocations of H, and T exponentiations in the group, resulting in a running time of roughly T (when ignoring the time required for evaluating f).[5]

We now sketch the analysis for bounding the success probability of these algorithms (see also [6] and the references therein). First, observe that if the online random walk collides with at least one of the precomputed paths within the first T steps, then it will inevitably hit a precomputed endpoint and A_1 will successfully output the discrete logarithm of the challenge element h. Hence, it is sufficient to bound the probability that such a collision occurs. This bound follows from two simple probabilistic arguments. The first argument shows that with constant probability, the expected number of distinct group elements "touched" by the precomputed paths is at least $\Omega(ST)$. The second argument shows that when the precomputed paths touch at least $\Omega(ST)$ group elements, the collision probability that we wish to bound is at least $\Omega(ST^2/N)$, since each new group element in the online walk hits any of the elements touched by the precomputed elements with probability $\Omega(ST/N)$. Both of these probabilities are taken also over the choice of the truly random function f, and both arguments inherently rely on the assumption that f is sampled uniformly at random from the set of all functions from the group $\mathbb{G}$ to $\mathbb{Z}_N$.

Accounting for the description length of the hash function

As previously mentioned, the above attack attains the optimal tradeoff between the preprocessed state size, the online running time and the success probability only when we do not take into count the description length of the truly random hash function f as part of the state size. When we do account for the description length of the function f, the state size

[5] Both the state size and the running time ignore $\log N$ factors.

blows up to more than $N \cdot \log N$ bits. This clearly renders the entire approach useless, since with this many bits, the preprocessing algorithm can simply pass to the online algorithm the discrete logarithm of every group element.

A standard way of reducing the representation size of such functions while still enjoying certain independence guarantees, is by sampling them from a family of k-wise independent hash functions for a suitable choice of k. Observe that f is applied by A_0 and A_1 to $O(ST)$ group elements. Hence, instead of using a truly uniform function, one can use a hash function sampled from a family of $O(ST)$-wise independent functions while keeping the analysis intact and without damaging the success probability. Alas, this is still far from satisfactory, since the space for storing a function from such families and its evaluation time will yield a tradeoff which is far from optimal. For example, instantiating such functions via randomly-sampled polynomials of degree $O(ST)$ will result in a state of size $S' = O(ST)$ and online running time of $T' = O(ST^2)$. In other words, the success probability is only $\Omega(T'/N)$, smaller by a factor of roughly $S'T'$ from the lower bound of Corrigan-Gibbs and Kogan. This non-optimality seems to be inherent when instantiating f using full-fledged k-wise independence: A lower bound by Siegel [24] shows that any construction reducing the evaluation time of k-wise independent functions below $\Omega(k)$ entails a polynomial increase in the space required for representing each function, thus once again leading to a non-optimal tradeoff.

Relying on $O(T)$-wise independence via a local analysis

As a first step, we present a more nuanced analysis for the success probability of the above algorithms, which enables us to reduce the level of independence required for the family of hash functions from $O(ST)$ to just $O(T)$. Note that indeed, due to its global nature, the analysis presented above for a truly random hash function breaks down when replacing it with a function sampled from an $O(T)$-wise independent family. In particular, we can no longer argue that the random walks in the preprocessing stage cover $\Omega(ST)$ distinct group element with high enough probability. The key observation is that such a global argument is unnecessary. What we are really interested in is the probability that the online walk collides with *one* of the walks from the preprocessing stage. Indeed, this probability can be sufficiently large even if the fraction of group elements covered by the preprocessed walks is very small. Intuitively, this is even likely: The more skewed the distribution over endpoints of T-step random walks is, the greater the probability is for a collision.[6]

Our refined analysis relies on a local argument that only considers the application of the hash function f on $O(T)$ group elements at a time. Therefore, it holds even when f is sampled from a family of $O(T)$-wise independent functions. First, we prove that the probability that the online walk collides with any specific precomputed walk within T steps is at least $\Omega(T^2/N)$. Then, we prove that the probability that it collides with any *two* specific precomputed walks within T steps is at most $O(T^4/N^2)$. Since these arguments only consider at most $2T$ and $3T$ distinct group elements, respectively, we are able to prove them relying on H being sampled from a $3T$-wise independent family. Finally, we use the inclusion-exclusion principle to bound the probability that the online walk hits a precomputed walk with T steps by $\Omega(ST^2/N)$.

[6] As an extreme example, consider a function f^* that maps every group element $h \in \mathbb{G}$ to an integer $\alpha \in \mathbb{Z}_N$ such that $h \cdot g^\alpha = g$. If this function is used, the preprocessed walks cover at most $S+1$ group elements, but the online walk collides with all of them with probability 1. Of course, the function f^* essentially computes the discrete logarithm of every group element without the random-walks-based algorithms. We use it here merely to exemplify the above point.

The remaining gap: The evaluation time of k-wise independent functions

Sampling the hash function f from a family of $O(T)$-wise independent functions still does not suffice to match the lower bound of Corrigan-Gibbs and Kogan. Consider again using a randomly-sampled polynomial of degree $O(T)$. In this case, the size of the state passed from A_0 to A_1 is indeed reduced to $S' = O(S + T)$, simplifying to $S' = O(S)$ in the natural case in which $S - T = \Omega(S)$. However, the running time of A_1 is much greater than T. Each evaluation of a degree $O(T)$ polynomial takes time at least $\Omega(T)$, and at worst, A_1 makes $2T$ such evaluations. This results in a running time of at least $T' = \Omega(T^2)$. In other words, the success probability is only $\Omega(S'T'/N)$, a factor of roughly T' away from the lower bound of Corrigan-Gibbs and Kogan. The lower bound of Siegel [24] again suggests that other instantiations of $O(T)$-wise independent families will also result in far-from-optimal tradeoffs.

The unsuitability of Fiat-Naor's derandomization

The work of Fiat and Naor [12] undertakes a similar endeavor to ours, presenting explicit preprocessing algorithms for the *function inversion problem*. They too start from (a modification of) previously-known algorithms that assume the existence of truly random functions shared between the preprocessing and the online algorithms [15] and encounter a similar problem to the one described above: Trying to instantiate the random functions naively by choosing them independently from a k-wise independent family (for a suitable value of k) results in a sub-optimal running time. Their solution to this problem is to choose these functions in a pairwise independent manner, instead of choosing them completely independently. Concretely, each function in their construction is a random polynomial of degree $k - 1$, but the coefficients of the different polynomials are sampled using pairwise independent functions. Fiat and Naor show that this change does not hurt the success probability of the attack too much, while at the same time, it enables a valuable speed-up in the online running time by evaluating these polynomials concurrently using the Fast Fourier Transform. Such an approach does not seem to fit our setting, where we are need to derandomize only a *single* truly random function. Moreover, attempts to modify the algorithms of Lee, Cheon, and Hong [17] and Bernstein and Lange [4] to use several different random functions (to instantiate them in a correlated manner like Fiat and Naor did) seem to yield sub-optimal tradeoffs.

Our Solution: Reducing running time via weaker independence

To reduce the overhead in the online running time caused by the T hash evaluations, we prove that the $O(T)$-wise independent family can be replaced with a function family that offers weaker independence guarantees. Concretely, we sample our hash function f from function families put forth by Pagh and Pagh [21] (following Siegel [24]). Functions within these families can be evaluated in constant time in the standard unit-cost RAM model (as described in Section 2), effectively eliminating the overhead in running time relative to using full-fledged $O(T)$-wise independent families. The process of sampling a function from these families can be thought of as occurring in two steps: First, a function family $\mathcal{F}$, parameterized by a parameter $k \in \mathbb{N}$, is drawn from a collection of families. Then, a function f is sampled from $\mathcal{F}$. Simplifying, the independence guarantee is that for any *specific* set $\mathcal{S}$ of k elements in the domain, $\mathcal{F}$ is "fully" independent with respect to $\mathcal{S}$ with high probability. This differs from the standard notion of k-wise independence, which requires a randomly chosen function to satisfy this property for *any* set $\mathcal{S}$ of size k.

We would like to argue that the above analysis, for a function f sampled from an $O(T)$-wise independent family, carries over to the case where f is sampled from a family with this weaker independence guarantee (where the parameter k is set to be $O(T)$). The problem, though, is that according to the original analysis of Pagh and Pagh, the family $\mathcal{F}$ is guaranteed to be k-wise independent with high probability only with respect to subsets which are fixed before $\mathcal{F}$ is chosen. It immediately follows that $\mathcal{F}$ is k-wise independent with high probability also with respect to subsets which are sampled from a distribution which is independent of the choice of $\mathcal{F}$. In our case, however, the analysis of the discrete log algorithms requires $\mathcal{F}$ to be fully independent with respect to random subsets that do depend on the choice $\mathcal{F}$. Concretely, we want the function f, sampled from $\mathcal{F}$, to behave like a random function on the union of two or three T-step random walks induced by f. Fortunately, a lemma proved by Berman, Haitner, Komargodski, and Naor [3] in a different context implies that $\mathcal{F}$ remains essentially fully independent with sufficiently high probability on such adaptively-chosen subsets as well. Overcoming various additional technical difficulties, this enables us to rely on explicit hash functions whose description lengths and evaluation times are taken into account in the algorithm's asymptotically optimal space-time tradeoff.

2 Preliminaries

In this section we present the basic notions and standard cryptographic primitives that are used in this work. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. 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 $\mathcal{X}$.

The computational model

We consider the unit-cost RAM model which has been the subject of much research, and is the standard model for analyzing the efficiency of explicit data structures and algorithms in terms of the running time of their operations (see, for example, [20, 14, 8, 21] and the references therein). In this model, any operation in a rather minimal instruction set can be executed in constant time on w-bit operands, where $w = O(\log u)$, and all elements are taken from a universe of size u. In our case, u may be any polynomial in the order N of the underlying group $\mathbb{G}$ in which we wish to compute discrete logarithms, and thus $w = O(\log N)$. We consider the standard instruction set for the unit-cost RAM model, which includes integer addition, subtraction, bit-wise Boolean operations, left and right bit shifts, and integer multiplication (we emphasize that the integers considered in the analysis of our algorithms will all be in the range $0, \ldots, N-1$).

Additionally, for an underlying cyclic group $\mathbb{G}$ of order p we denote by t_{mult} and t_{exp} the running times of computing the group operation and the group exponentiation, respectively. Within the unit-cost RAM model we then state the running time of our algorithms as functions of t_{mult} and t_{exp}. Assuming that both operations can be implemented in time polynomial in $\log N$, this translates into (at most) a multiplicative lower-order factor of $\mathsf{poly}(\log N)$.

Uniform hashing

A function family $\mathcal{H}$ is said to be uniform over a set $\mathcal{S}$ of elements in its domain, if a uniformly-sampled function $h \leftarrow \mathcal{H}$ is indistinguishable from a truly random function when evaluated on $\mathcal{S}$. This is formally captured via the following definition:

▶ **Definition 3.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be sets and let $\mathcal{S} = \{x_1, \ldots, x_k\} \subseteq \mathcal{X}$ be a subset of size k. We say that a function family $\mathcal{H}$ mapping $\mathcal{X}$ to $\mathcal{Y}$ is* uniform over $\mathcal{S}$ *if for every tuple $(y_1, \ldots, y_k) \in \mathcal{Y}^k$ it holds that*

$$\Pr_{h \leftarrow \mathcal{H}}\left[\forall i \in [k] \ : \ h(x_i) = y_i\right] = \frac{1}{|\mathcal{Y}|^k}.$$

*We say that $\mathcal{H}$ is k-*wise independent *if it is uniform over all subsets of $\mathcal{X}$ of size at most k.*

Functions families which are k-wise independent have repeatedly proven to be useful for the design and analysis of data structures in general, and for cryptographic preprocessing attacks in particular [12]. Alas, all known constructions of such families provide functions which take time at least k to evaluate; this will, unfortunately, prove to be too costly for our attack. However, Pagh and Pagh [21], following Siegel [24], constructed families of functions which can be evaluated in constant time, offering weaker guarantees than full-fledged k-wise independence, but such that still suffice for our needs. Concretely, they consider a randomized algorithm which generates a random family $\mathcal{H}$ of functions, such that for any predetermined set $\mathcal{S}$ of at most k elements in the domain, the family $\mathcal{H}$ is uniform over $\mathcal{S}$ with high probability.

▶ **Theorem 4** ([21] – simplified)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be sets. Then, there exists an algorithm* HashGen *that on input any integer $k \in \mathbb{N}$ and any constant $c > 0$, outputs a description of a function family $\mathcal{H}$ mapping $\mathcal{X}$ to $\mathcal{Y}$ such the following hold:*

1. *For every set $\mathcal{S} \subseteq \mathcal{X}$ of size at most k it holds that*

$$\Pr_{\mathcal{H} \leftarrow \mathsf{HashGen}(k,c)}\left[\mathcal{H} \text{ is uniform over } \mathcal{S}\right] \geq 1 - \frac{1}{k^c}.$$

2. *Every function in $\mathcal{H}$ can be represented using at most $2k \cdot \log|\mathcal{Y}| + O(k + \log\log|\mathcal{X}|)$ bits, and evaluated on any input in constant time within the unit-cost RAM model.*

We note that the construction of Pagh and Pagh was later improved in various ways (see, for example, [10, 9, 1]), but the parameters it offers already suffice for our needs. In addition, note that Theorem 4 guarantees only that $\mathcal{H}$ sampled by HashGen is uniform with high probability over sets of elements which are a-priori fixed, and do not depend on $\mathcal{H}$. Looking ahead, we will want to reason about the output distribution of $\mathcal{H}$ on sets of elements that *do depend* on $\mathcal{H}$. To this end, we will rely on a lemma by Berman et al. [3] who proved that $\mathcal{H}$ sampled by HashGen is uniform with high probability also on sets of elements which are chosen by an unbounded adversary which queries a random function in $\mathcal{H}$ at most k times.

▶ **Lemma 5** ([3] – simplified)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be sets, let k be an integer, let $\mathcal{F}_k$ be a k-wise independent family of functions $f : \mathcal{X} \to \mathcal{Y}$, and let* HashGen *be the algorithm guaranteed by Theorem 4 producing families of functions $f : \mathcal{X} \to \mathcal{Y}$. Then, for any k-query algorithm D and constant $c > 0$ it holds that*

$$\left|\Pr_{\substack{\mathcal{H} \leftarrow \mathsf{HashGen}(k,c) \\ f \leftarrow \mathcal{H}}}\left[D^f() = 1\right] - \Pr_{f \leftarrow \mathcal{F}_k}\left[D^f() = 1\right]\right| \leq O\left(\frac{1}{k^c}\right).$$

3 Our Discrete-Logarithm Preprocessing Algorithm

In this section we present our preprocessing algorithm for computing discrete logarithms in a cyclic group $\mathbb{G}$ of order N relative to a generator $g \in \mathbb{G}$. For simplicity, throughout the section we fix the group $\mathbb{G}$, the order N and the generator g, and note that these can in

fact be provided as inputs to our algorithm. Our algorithm A consists of a pair (A_0, A_1) of algorithms, where A_0 and A_1 are the preprocessing algorithm and the on-line algorithm, respectively. Additionally, our algorithm is parameterized by integers $\ell, s \in \mathbb{N}$ and by a constant $c > 0$, and uses as a building block the algorithm $\mathsf{HashGen}$ described in Section 2 for producing families of hash functions $f : \mathbb{G} \to \mathbb{Z}_N$.

The preprocessing algorithm A_0

Input: A description $(\mathbb{G}, N, g)$ of a cyclic group $\mathbb{G}$ of order N that is generated by $g \in \mathbb{G}$, integers ℓ and s, and a constant $c > 0$.

1. Sample $\mathcal{H} \leftarrow \mathsf{HashGen}(3\ell, c)$ and $f \leftarrow \mathcal{H}$.
2. For each $i = 1, \ldots, s$:
   a. Sample $x_{i,1} \leftarrow \mathbb{Z}_N$ and compute $g_{i,1} = g^{x_{i,1}}$.
   b. For each $j = 2, \ldots, \ell$ compute $x_{i,j} = x_{i,j-1} + f(g_{i,j-1})$ and $g_{i,j} = g^{x_{i,j}}$.
   c. Set $y_i = x_{i,\ell}$ and $g_i = g_{i,\ell}$.
3. Output $\mathsf{st} = \left(f, \{(g_i, y_i)\}_{i \in [s]}\right)$.

The online algorithm A_1

Input: A description $(\mathbb{G}, N, g)$ of a cyclic group, a group element $h \in \mathbb{G}$, and a state $\mathsf{st} = \left(f, \{(g_i, y_i)\}_{i \in [s]}\right)$ produced by A_0.

1. If $h = g_i$ for some $i \in [s]$, then output y_i and terminate.
2. Set $h_1 = h$ and $\Delta_1 = 0$, and for each $i = 2, \ldots, 2\ell$:
   a. Compute $\delta_i = f(h_{i-1})$ and $\Delta_i = \Delta_{i-1} + \delta_i$.
   b. Compute $h_i = h_{i-1} \cdot g^{\delta_i}$.
   c. If $h_i = g_j$ for some $j \in [s]$, then output $x = y_j - \Delta_i$ and terminate.
3. Output $\perp$.

Note that the description of the online algorithm A_1 includes two non-trivial lookup operations in Steps 1 and 2c for the elements h and h_i, respectively. For avoiding a noticeable overhead in the running time of A_1, these lookup operations can be implemented by having the preprocessing algorithm A_0 store the pairs $\{(g_i, y_i)\}_{i \in [s]}$ within an explicit data structure that supports efficient lookup operations and uses linear space (i.e., $O(s)$ space). In the unit-cost RAM model, existing such data structures range, for example, from the most basic solution of a sorted list that supports lookup operations in time $O(\log s)$, to more advanced solutions such as cuckoo hashing that supports lookup operations in constant time [22].

The following theorem states our bounds on the amount of space required for storing the state produced by the preprocessing algorithm A_0, on the running time of the online algorithm A_1, and on the success probability of A_1 in computing the discrete logarithm $\mathsf{dlog}_g(h)$ of a uniformly-distributed group element h.

▶ **Theorem 6.** *Let $\mathbb{G}$ be a cyclic group of order N that is generated by $g \in \mathbb{G}$. Let s and ℓ be any integers such that $s \cdot \ell^2 \leq N/64$, and let $c > 0$ be any constant. Then,*

$$\Pr\left[A_1\left(\mathbb{G}, N, g, h, \mathsf{st}\right) = \mathsf{dlog}_g(h)\right] \geq \frac{1}{8} \cdot \frac{s \cdot \ell^2}{N} - O\left(\frac{1}{(3\ell)^c}\right),$$

where $\mathsf{st} \leftarrow A_0(\mathbb{G}, p, g)$ and $h \leftarrow \mathbb{G}$. In addition, A_0 outputs $O((\ell + s) \cdot \log N)$ bits, and A_1 runs in time $O(\ell \cdot t_{\mathsf{exp}})$ in the unit-cost RAM model, where t_{exp} denotes the running time of exponentiation in the group $\mathbb{G}$.

Assuming that exponentiation in the group $\mathbb{G}$ can be implemented in time $t_{\mathsf{exp}} = \mathsf{poly}(\log N)$, the following corollary captures the specific setting of $s = \ell = O(N^{1/3})$:

▶ **Corollary 7.** *When setting $s = \ell = O(N^{1/3})$, the preprocessing algorithm A_0 outputs $S = \widetilde{O}(N^{1/3})$ bits, and the online algorithm A_1 runs in time $T = \widetilde{O}(N^{1/3})$ in the unit-cost RAM model and succeeds with a constant probability.*

We now turn to the proof of Theorem 6.

Proof. First, note that the state $\mathsf{st} = \left(f, \{(g_i, y_i)\}_{i\in[s]}\right)$ produced by A_0 consists of the description of a hash function f sampled from the family $\mathcal{H}$ produced by $\mathsf{HashGen}(3\ell, c)$, and of s pairs (g_i, y_i) where $g_i \in \mathbb{G}$ and $y_i \in \mathbb{Z}_N$ for each $i \in [s]$. By Theorem 4 the description of f is of length at most $2 \cdot 3\ell \cdot O(\log N) + O(\ell + \log\log N)$ bits, and additionally each pair (g_i, y_i) can be represented using $O(\log N)$ bits. Therefore, A_0 outputs $O((\ell + s) \cdot \log N)$ bits.

Second, note that A_1's running time is dominated by that of Step 2, which is repeated for at most 2ℓ iterations. Each such iteration consists of an evaluation of the hash function f (which by Theorem 4 takes constant time in the unit-cost RAM model), a group multiplication, a group exponentiation, and an additional constant number of constant-time operations. Therefore, overall A_1 runs in time $O(\ell \cdot t_{\mathsf{exp}})$.

In the remainder of this proof, we analyze the success probability of our algorithm. For any function $f : \mathbb{G} \to \mathbb{Z}_N$, define the function $\widehat{f} : \mathbb{G} \to \mathbb{G}$ by $\widehat{f}(h) = h \cdot g^{f(h)}$.

▷ Claim 8. Let $\mathcal{H}$ be a family of functions $f : \mathbb{G} \to \mathbb{Z}_N$ and let $\mathcal{G}_{\mathcal{H}} = \left\{\widehat{f}\right\}_{f\in\mathcal{H}}$. Then, for any integer $k \in \mathbb{N}$, if $\mathcal{H}$ is k-wise independent then $\mathcal{G}_{\mathcal{H}}$ is k-wise independent.

Proof. Assume that $\mathcal{H}$ is k-wise independent, and let $h_1, \ldots, h_k \in \mathbb{G}$ be distinct group elements. Then, for every k group elements $u_1, \ldots, u_k \in \mathbb{G}$ it holds that

$$\begin{aligned}
\Pr_{\widehat{f}\leftarrow\mathcal{G}_{\mathcal{H}}}\left[\forall i \in [k] \ : \ \widehat{f}(h_i) = u_i\right] &= \Pr_{f\leftarrow\mathcal{H}}\left[\forall i \in [k] \ : \ \widehat{f}(h_i) = u_i\right] \\
&= \Pr_{f\leftarrow\mathcal{H}}\left[\forall i \in [k] \ : \ h_i \cdot g^{f(h_i)} = u_i\right] \\
&= \Pr_{f\leftarrow\mathcal{H}}\left[\forall i \in [k] \ : f(h_i) = \mathsf{dlog}_g(u_i) - \mathsf{dlog}_g(h_i)\right] \\
&= \left(\frac{1}{N}\right)^k \qquad (1)
\end{aligned}$$

where for a group element $u \in \mathbb{G}$, $\mathsf{dlog}_g(u)$ is the unique $\mathbb{Z}_N$ element x such that $g^x = u$, and Eq. (1) follows from the k-wise independence of $\mathcal{H}$. ◁

For a group element $u \in \mathbb{G}$, a function f and an integer $k \in \mathbb{N}$, denote

$$C_{u,f,k} = \left(\widehat{f}^{(j)}(u)\right)_{j\in\{0,\ldots,k-1\}},$$

where $\widehat{f}^{(j)}(u) = \widehat{f}(\widehat{f}^{(j-1)}(u))$ and $\widehat{f}^{(0)}(u) = u$. That is, $C_{u,f,k}$ is the ordered multi-set of all group elements visited by a $(k-1)$-step walk in $\mathbb{G}$, which starts from u and progresses according to the function $\widehat{f}$. For $k = 0$ we use the convention that $C_{u,f,0}$ is the empty set. We define the following random variables:

- Let F denote the random variable corresponding to the hash function f chosen by A_0 in Step 1 by sampling $\mathcal{H} \leftarrow \mathsf{HashGen}(3\ell, c)$ and $f \leftarrow \mathcal{H}$.
- For each $i \in [s]$ let $G_{i,1}$ denote the random variable corresponding to the group element $g_{i,1} \in \mathbb{G}$ sampled uniformly by A_1 in Step 2a.
- Let H be the random variable corresponding to the uniformly-distributed group element $h \in \mathbb{G}$ that is given as input to A_1.

Note that using this notation, for each $i \in [s]$ it holds that $C_{G_{i,1},F,\ell}$ is a random variable corresponding to the multi-set of group elements computed by A_0 in each iteration of Step 2. Similarly, $C_{H,F,2\ell}$ is a random variable corresponding to the multi-set of group elements computed by A_1 in Step 2.

Observe that if $C_{H,F,\ell}$ (which contains the first ℓ elements computed by A_1) intersects $C_{G_{j,1},F,\ell}$ for some $j \in [s]$, then A_1 successfully outputs X for which $g^X = H$. This is the case since $C_{H,F,\ell} \cap C_{G_{j,1},F,\ell} \neq \emptyset$ implies that $G_j \in C_{H,F,2\ell}$, where G_j is included in the state st along with the corresponding exponent Y_j such that $G_j = g^{Y_j}$. Moreover, if G_j is the ith element computed by A_1, then the integer Δ_i computed by A_1 satisfies $G_j = H \cdot g^{\Delta_i}$. Therefore, we obtain

$$g^{Y_j} = G_j = H \cdot g^{\Delta_i}$$

which implies that

$$H = g^{Y_j - \Delta_i},$$

and that the output $X = Y_j - \Delta_i$ of A_1 is indeed the discrete logarithm of H with respect to g. Hence, in the remainder of the proof we will focus on bounding the probability that $C_{H,F,\ell} \cap C_{G_{j,1},F,\ell} \neq \emptyset$ for some $j \in [s]$. By the inclusion-exclusion principle, it holds that

$$\begin{aligned}\Pr\left[\bigvee_{j\in[s]} C_{H,F,\ell} \cap C_{G_{j,1},F,\ell} \neq \emptyset\right] \\ \geq \sum_{1\leq j\leq s} \Pr\left[C_{H,F,\ell} \cap C_{G_{j,1},F,\ell} \neq \emptyset\right] - \sum_{1\leq i<j\leq s} \Pr\left[\begin{array}{c} C_{H,F,\ell} \cap C_{G_{i,1},F,\ell} \neq \emptyset \\ \wedge\ C_{H,F,\ell} \cap C_{G_{j,1},F,\ell} \neq \emptyset \end{array}\right]. \quad (2)\end{aligned}$$

We now bound each of the sums in Eq. (2) separately, in Claims 9 and 10 below.

▷ Claim 9. For every $j \in [s]$ it holds that

$$\Pr\left[C_{H,F,\ell} \cap C_{G_{j,1},F,\ell} \neq \emptyset\right] > \frac{1}{2\sqrt{e}} \cdot \frac{\ell^2}{N} - O\left(\frac{1}{(3\ell)^c}\right).$$

Proof. Let $j \in [s]$, and let F^* denote a random variable describing a function from $\mathbb{G}$ to $\mathbb{Z}_N$ distributed uniformly in a 3ℓ-wise independent family $\mathcal{F}$. We will prove that

$$\Pr\left[C_{H,F^*,\ell} \cap C_{G_{j,1},F^*,\ell} \neq \emptyset\right] > \frac{1}{2\sqrt{e}} \cdot \frac{\ell^2}{N},$$

and the claim then follows immediately from Lemma 5. By total probability,

$$\begin{aligned}\Pr\left[C_{H,F^*,\ell} \cap C_{G_{j,1},F^*,\ell} \neq \emptyset\right] \quad \geq \quad & \Pr\left[C_{H,F^*,\ell} \cap C_{G_{j,1},F^*,\ell} \neq \emptyset \;\middle|\; |C_{H,F^*,\ell}| = \ell \wedge |C_{G_{j,1},F^*,\ell}| = \ell\right] \\ & \times \Pr\left[|C_{H,F^*,\ell}| = \ell \wedge |C_{G_{j,1},F^*,\ell}| = \ell\right].\end{aligned}$$

Moreover, it holds that

$$\begin{aligned}\Pr\left[|C_{H,F^*,\ell}| = \ell \wedge |C_{G_{j,1},F^*,\ell}| = \ell\right] \quad &= \quad 1 - \Pr\left[|C_{H,F^*,\ell}| < \ell \vee |C_{G_{j,1},F^*,\ell}| < \ell\right] \\ &\geq \quad 1 - 2 \cdot \Pr\left[|C_{H,F^*,\ell}| < \ell\right] \quad (3)\end{aligned}$$

where Eq. (3) follows from the union bound and the fact that the random variables $|C_{H,F^*,\ell}|$ and $|C_{G_{j,1},F^*,\ell}|$ are identically distributed. We now turn to bound $\Pr\left[|C_{H,F^*,\ell}| < \ell\right]$. By total probability

$$\Pr\left[|C_{H,F^*,\ell}| < \ell\right] = \sum_{h\in\mathbb{G}} \Pr\left[H = h\right] \cdot \Pr\left[|C_{h,F^*,\ell}| < \ell\right]. \quad (4)$$

Since for every $h \in \mathbb{G}$, the events $\{|C_{h,F^*,i-1}| = i-1 \wedge |C_{h,F^*,i}| < i\}_{i\in[\ell]}$ form a partition of the event $|C_{h,F^*,\ell}| < \ell$, for every $h \in \mathbb{G}$ it holds that

$$\begin{aligned}
\Pr\left[|C_{h,F^*,\ell}| < \ell\right] &= \sum_{i=1}^{\ell} \Pr\left[|C_{h,F^*,i-1}| = i-1 \wedge |C_{h,F^*,i}| < i\right] \\
&\leq \sum_{i=1}^{\ell} \Pr\left[|C_{h,F^*,i-1}| = i-1 \mid |C_{h,F^*,i}| < i\right] \\
&= \sum_{i=1}^{\ell} \sum_{\substack{h_2,\ldots,h_{i-1}\in\mathbb{G}: \\ \forall 1\leq k<t\leq i-1,\ h_k\neq h_t}} \Pr\left[\begin{array}{l}\forall k \in [i-2] : \widehat{F^*}(h_k) = h_{k+1} \\ \wedge\ \widehat{F^*}(h_{i-1}) \in \{h_1,\ldots,h_{i-1}\}\end{array}\right] \\
&= \sum_{i=1}^{\ell} \sum_{\substack{h_2,\ldots,h_{i-1}\in\mathbb{G}: \\ \forall 1\leq k<t\leq i-1,\ h_k\neq h_t}} \sum_{m\in[i-1]} \Pr\left[\begin{array}{c}\forall k \in [i-2] : \widehat{F^*}(h_k) = h_{k+1} \\ \wedge\ \widehat{F^*}(h_{i-1}) = h_m\end{array}\right],
\end{aligned}$$

where $h_1 = h$ and $\widehat{F^*}$ is the function define by $\widehat{F^*}(u) = u \cdot g^{F^*(u)}$. By the fact that $\mathcal{G}_{\mathcal{F}}$ is 3ℓ-wise independent, then it is in particular ℓ-wise independent, and thus the following holds: For every $i \in [\ell]$, for every $h_1,\ldots,h_{i-1} \in \mathbb{G}$ and for every $m \in [i-1]$, the fraction of functions $\widehat{f}$ in $\mathcal{G}_{\mathcal{F}}$ which satisfy $\widehat{f}(h_k) = h_{k+1}$ for all $k \in [i-2]$ and $\widehat{f}(h_{i-1}) = h_m$ is $N^{-(i-1)}$. Hence, we obtain that for every $h \in \mathbb{G}$,

$$\begin{aligned}
\Pr\left[|C_{h,F^*,\ell}| < \ell\right] &\leq \sum_{i=1}^{\ell} \sum_{\substack{h_2,\ldots,h_{i-1}\in\mathbb{G}: \\ \forall 1\leq k<t\leq i-1,\ h_k\neq h_t}} (i-1)\cdot N^{-(i-1)} \\
&< \sum_{i=1}^{\ell} \frac{i-1}{N} \\
&\leq \frac{\ell^2}{N}.
\end{aligned}$$

Together with Eq. (3) and (4), this implies that

$$\begin{aligned}
\Pr\left[|C_{H,F^*,\ell}| = \ell \wedge |C_{G_{j,1},F^*,\ell}| = \ell\right] &\geq 1 - 2\sum_{h\in\mathbb{G}} \Pr\left[H = h\right]\cdot\frac{\ell^2}{N} \\
&= 1 - \frac{2\ell^2}{N}. \qquad (5)
\end{aligned}$$

For each $i \in [\ell]$, let E_i denote the event in which

$$\left(C_{H,F^*,i-1} \cap C_{G_{j,1},F^*,\ell} = \emptyset\right) \wedge \left(C_{H,F^*,i} \cap C_{G_{j,1},F^*,\ell} \neq \emptyset\right).$$

That is, E_i is the event in which the ith element in $C_{H,F^*,\ell}$ is the first element in $C_{H,F^*,\ell}$ that also appears in $C_{G_{j,1},F^*,\ell}$. Then, the 3ℓ-wise independence of $\mathcal{F}$ implies in particular its 2ℓ-wise independence, and thus for each $i \in [\ell]$ it holds that

$$\begin{aligned}
\Pr\left[\mathsf{E}_i \mid |C_{H,F^*,\ell}| = \ell \wedge |C_{G_{j,1},F^*,\ell}| = \ell\right] &= \left(\prod_{k=1}^{i}\left(1 - \frac{\ell+k-1}{N}\right)\right)\cdot\frac{\ell}{N} \\
&\geq \left(1-\frac{2\ell}{N}\right)^i \cdot \frac{\ell}{N},
\end{aligned}$$

and since $1 - x \leq e^{-2x}$ for all $x \in [0, 1/2]$, the fact that $\ell \leq N/4$ implies that

$$\Pr\left[\mathsf{E}_i \;\middle|\; |C_{H,F^*,\ell}| = \ell \wedge |C_{G_{j,1},F^*,\ell}| = \ell\right] \quad \geq \quad e^{-2\ell i/N} \cdot \frac{\ell}{N}.$$

It thus follows that

$$\begin{aligned}
&\Pr\left[C_{H,F^*,\ell} \cap C_{G_{j,1},F^*,\ell} \neq \emptyset \;\middle|\; |C_{H,F^*,\ell}| = \ell \wedge |C_{G_{j,1},F^*,\ell}| = \ell\right] \\
&\qquad = \sum_{i=1}^{\ell} \Pr\left[\mathsf{E}_i \;\middle|\; |C_{H,F^*,\ell}| = \ell \wedge |C_{G_{j,1},H,\ell}| = \ell\right] \\
&\qquad \geq \frac{\ell}{N} \cdot \sum_{i=1}^{\ell} e^{-2\ell i/N} \\
&\qquad \geq \frac{\ell}{N} \cdot \ell \cdot e^{-2\ell^2/N}. \qquad (6)
\end{aligned}$$

Taken together, Eq. (5) and Eq. (6) imply that

$$\Pr\left[C_{H,F^*,\ell} \cap C_{G_{j,1},F^*,\ell} \neq \emptyset\right] > \left(1 - \frac{2\ell^2}{N}\right) \cdot \frac{\ell^2}{N} \cdot e^{-2\ell^2/N},$$

and since $\ell \leq \sqrt{N}/2$, we obtain

$$\Pr\left[C_{H,F^*,\ell} \cap C_{G_{j,1},F^*,\ell} \neq \emptyset\right] > \frac{1}{2\sqrt{e}} \cdot \frac{\ell^2}{N}.$$

By Lemma 5, this implies that

$$\Pr\left[C_{H,F,\ell} \cap C_{G_{j,1},F,\ell} \neq \emptyset\right] > \frac{1}{2\sqrt{e}} \cdot \frac{\ell^2}{N} - O\left(\frac{1}{(3\ell)^c}\right). \qquad \lhd$$

▷ Claim 10. For every $1 \leq i < j \leq s$ it holds that

$$\Pr\left[C_{H,F,\ell} \cap C_{G_{i,1},F,\ell} \neq \emptyset \wedge C_{H,F,\ell} \cap C_{G_{j,1},F,\ell} \neq \emptyset\right] \leq \frac{8\ell^4}{N^2} + O\left(\frac{1}{(3\ell)^c}\right).$$

Proof. Let i, j such that $1 \leq i < j \leq s$, and as before let F^* denote a random variable describing a function from $\mathbb{G}$ to $\mathbb{Z}_N$ distributed uniformly in a 3ℓ-wise independent family $\mathcal{F}$. We will prove that

$$\Pr\left[C_{H,F^*,\ell} \cap C_{G_{i,1},F^*,\ell} \neq \emptyset \wedge C_{H,F^*,\ell} \cap C_{G_{j,1},F^*,\ell} \neq \emptyset\right] \leq \frac{8\ell^4}{N^2},$$

and the claim then follows immediately from Lemma 5. Since the event $C_{H,F^*,\ell} \cap C_{G_{j,1},F^*,\ell} \neq \emptyset$ is contained in the event $C_{G_{j,1},F^*,\ell} \cap \left(C_{H,F^*,\ell} \cup C_{G_{i,1},F^*,\ell}\right) \neq \emptyset$, it holds that

$$\begin{aligned}
&\Pr\left[C_{H,F^*,\ell} \cap C_{G_{i,1},F^*,\ell} \neq \emptyset \wedge C_{H,F^*,\ell} \cap C_{G_{j,1},F^*,\ell} \neq \emptyset\right] \\
&\qquad \leq \Pr\left[C_{H,F^*,\ell} \cap C_{G_{i,1},F^*,\ell} \neq \emptyset \wedge C_{G_{j,1},F^*,\ell} \cap \left(C_{H,F^*,\ell} \cup C_{G_{i,1},F^*,\ell}\right) \neq \emptyset\right] \\
&\qquad \leq \Pr\left[C_{H,F^*,\ell} \cap C_{G_{i,1},F^*,\ell} \neq \emptyset\right] \qquad (7) \\
&\qquad\quad \cdot \Pr\left[C_{G_{j,1},F^*,\ell} \cap \begin{pmatrix} C_{H,F^*,\ell} \\ \cup C_{G_{i,1},F^*,\ell} \end{pmatrix} \neq \emptyset \,\middle|\, C_{H,F^*,\ell} \cap C_{G_{i,1},F^*,\ell} \neq \emptyset\right] \qquad (8)
\end{aligned}$$

For upper bounding Eq. 7, note that $|C_{H,F^*,\ell}| \leq \ell$ and $|C_{G_{i,1},F^*,\ell}| \leq \ell$, and therefore the 3ℓ-wise independence of $\mathcal{F}$ (which implies, in particular, 2ℓ-wise independence) guarantees that

$$\Pr\left[C_{H,F^*,\ell} \cap C_{G_{i,1},F^*,\ell} \neq \emptyset\right] \leq \left(1 - \left(1 - \frac{\ell}{N}\right)^{\ell}\right).$$

Similarly, for upper bounding Eq. 8, note that $|C_{G_{j,1},F^*,\ell}| \leq \ell$ and $|C_{H,F^*,\ell} \cup C_{G_{i,1},F^*,\ell}| \leq 2\ell$, and therefore the 3ℓ-wise independence of $\mathcal{F}$ guarantees that

$$\Pr\left[C_{G_{j,1},F^*,\ell} \cap \left(C_{H,F^*,\ell} \cup C_{G_{i,1},F^*,\ell}\right) \neq \emptyset \,\middle|\, C_{H,F^*,\ell} \cap C_{G_{i,1},F^*,\ell} \neq \emptyset\right] \leq \left(1 - \left(1 - \frac{2\ell}{N}\right)^{\ell}\right).$$

Since $1 - (1-x)^y \leq 2xy$ for all $x, y \in \mathbb{N}$ such that $x \leq 1/2$, and since $\ell \leq p/4$, these imply that

$$\Pr\left[C_{H,F^*,\ell} \cap C_{G_{i,1},F^*,\ell} \neq \emptyset \wedge C_{H,F^*,\ell} \cap C_{G_{j,1},F^*,\ell} \neq \emptyset\right] \leq \frac{2\ell^2}{N} \cdot \frac{4\ell^2}{N} = \frac{8\ell^4}{N^2},$$

and from Lemma 5 we obtain that

$$\Pr\left[C_{H,F,\ell} \cap C_{G_{i,1},F,\ell} \neq \emptyset \wedge C_{H,F,\ell} \cap C_{G_{j,1},F,\ell} \neq \emptyset\right] \leq \frac{8\ell^4}{N^2} + O\left(\frac{1}{(3\ell)^c}\right). \qquad \triangleleft$$

From Claims 9 and 10 and from Eq. (2), we obtain that

$$\begin{aligned}
\Pr\left[A_1\left(\mathbb{G}, N, g, h, \mathsf{st}\right) = \mathsf{dlog}_g(h)\right] &> \sum_{1 \leq j \leq s} \frac{1}{2\sqrt{e}} \cdot \frac{\ell^2}{N} - \sum_{1 \leq i < j \leq s} \frac{8\ell^4}{N^2} - O\left(\frac{1}{(3\ell)^c}\right) \\
&> \frac{1}{2\sqrt{e}} \cdot \frac{s \cdot \ell^2}{N} - 8 \cdot \frac{s^2 \cdot \ell^4}{N^2} - O\left(\frac{1}{(3\ell)^c}\right) \\
&> \frac{1}{4} \cdot \frac{s \cdot \ell^2}{N} - 8 \cdot \frac{s^2 \cdot \ell^4}{N^2} - O\left(\frac{1}{(3\ell)^c}\right).
\end{aligned}$$

Since $s \cdot \ell^2 \leq N/64$, this implies that

$$8 \cdot \frac{s^2 \cdot \ell^4}{N^2} \leq \frac{1}{8} \cdot \frac{s \cdot \ell^2}{N}$$

and hence

$$\Pr\left[A_1\left(\mathbb{G}, N, g, h, \mathsf{st}\right) = \mathsf{dlog}_g(h)\right] > \frac{1}{8} \cdot \frac{s \cdot \ell^2}{N} - O\left(\frac{1}{(3\ell)^c}\right).$$

This concludes the proof of Theorem 6. ◀

References

1. Martin Aumüller, Martin Dietzfelbinger, and Philipp Woelfel. Explicit and efficient hash families suffice for cuckoo hashing with a stash. *Algorithmica*, 70(3):428–456, 2014.
2. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
3. Itay Berman, Iftach Haitner, Ilan Komargodski, and Moni Naor. Hardness-preserving reductions via cuckoo hashing. *Journal of Cryptology*, 32(2):361–392, 2019.
4. Daniel J. Bernstein and Tanja Lange. Non-uniform cracks in the concrete: The power of free precomputation. In *Advances in Cryptology – ASIACRYPT '13*, pages 321–340, 2013.
5. Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In *Advances in Cryptology – CRYPTO '18*, Lecture Notes in Computer Science, pages 693–721, 2018.
6. Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. In *Advances in Cryptology – EUROCRYPT '18*, pages 415–447, 2018.

7 Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In *Advances in Cryptology – CRYPTO '10*, pages 649–665, 2010.

8 Martin Dietzfelbinger and Rasmus Pagh. Succinct data structures for retrieval and approximate membership. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, pages 385–396, 2008.

9 Martin Dietzfelbinger and Michael Rink. Applications of a splitting trick. *ICALP 2009: Automata, Languages and Programming*, pages 354–365, 2009.

10 Martin Dietzfelbinger and Philipp Woelfel. Almost random graphs with simple hash functions. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 629–638, 2003.

11 Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In *Advances in Cryptology – EUROCRYPT '17*, volume 10211, pages 473–495, 2017.

12 Amos Fiat and Moni Naor. Rigorous time/space trade-offs for inverting functions. *SIAM Journal on Computing*, 29(3):709–803, 1999.

13 David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(2):224–280, 2010.

14 Torben Hagerup, Peter Bro Miltersen, and Rasmus Pagh. Deterministic dictionaries. *Journal of Algorithms*, 41(1):69–85, 2001.

15 Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transaction on Information Theory*, 26(4):401–406, 1980.

16 Neal Koblitz, Alfred Menezes, and Scott A. Vanstone. The state of elliptic curve cryptography. *Designs, Codes and Cryptography*, 19(2/3):173–193, 2000.

17 Hyung Tae Lee, Jung Hee Cheon, and Jin Hong. Accelerating ID-based encryption based on trapdoor DL using pre-computation. Cryptology ePrint Archive, Report 2011/187, 2011.

18 Ueli Maurer. Abstract models of computation in cryptography. In *Proceedings of the 10th IMA International Conference on Cryptography and Coding*, pages 1–12, 2005.

19 Ueli Maurer, Christopher Portmann, and Jiamin Zhu. Unifying generic group models. Cryptology ePrint Archive, Report 2020/996, 2020.

20 Peter Bro Miltersen. Cell probe complexity – a survey. In *Proceedings of the 19th Conference on the Foundations of Software Technology and Theoretical Computer Science, Advances in Data Structures Workshop*, 1999.

21 Anna Pagh and Rasmus Pagh. Uniform hashing in constant time and optimal space. *SIAM Journal on Computing*, 38(1):85–96, 2008.

22 Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *Journal of Algorithms*, 51(2):122–144, 2004.

23 Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology – EUROCRYPT '97*, pages 256–266, 1997.

24 Alan Siegel. On universal classes of extremely random constant-time hash functions. *SIAM Journal on Computing*, 33(3):505–543, 2004.

25 Dominique Unruh. Random oracles and auxiliary input. In *Advances in Cryptology – CRYPTO '07*, pages 205–223, 2007.

Revisiting the Uber Assumption in the Algebraic Group Model: Fine-Grained Bounds in Hidden-Order Groups and Improved Reductions in Bilinear Groups

Lior Rotem
School of Computer Science and Engineering, Hebrew University of Jerusalem, Israel

Abstract

We prove strong security guarantees for a wide array of computational and decisional problems, both in hidden-order groups and in bilinear groups, within the algebraic group model (AGM) of Fuchsbauer, Kiltz and Loss (CRYPTO '18). As our first contribution, we put forth a new fine-grained variant of the Uber family of assumptions in hidden-order groups. This family includes in particular the repeated squaring function of Rivest, Shamir and Wagner, which underlies their time-lock puzzle as well as the main known candidates for verifiable delay functions; and a computational variant of the generalized BBS problem, which underlies the timed commitments of Boneh and Naor (CRYPTO '00). We then provide two results within a variant of the AGM, which show that the hardness of solving problems in this family in a less-than-trivial number of steps is implied by well-studied assumptions. The first reduction may be applied in any group (and in particular, class groups), and is to the RSA assumption; and our second reduction is in RSA groups with a modulus which is the product of two safe primes, and is to the factoring assumption.

Additionally, we prove that the hardness of any computational problem in the Uber family of problems in bilinear groups is implied by the hardness of the q-discrete logarithm problem. The parameter q in our reduction is the maximal degree in which a variable appears in the polynomials which define the specific problem within the Uber family. This improves upon a recent result of Bauer, Fuchsbauer and Loss (CRYPTO '20), who obtained a similar implication but for a parameter q which is lower bounded by the maximal *total* degree of one of the above polynomials. We discuss the implications of this improvement to prominent group key-exchange protocols.

2012 ACM Subject Classification Security and privacy → Cryptography; Theory of computation → Cryptographic primitives

Keywords and phrases Algebraic group model, Uber assumption, Bilinear groups, Hidden-order groups

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.13

Related Version *Full Version*: `https://eprint.iacr.org/2022/584`

Funding 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).

1 Introduction

The algebraic group model (the AGM) was introduced by Fuchsbauer, Kiltz and Loss[1] [28] with the aim of striking a middle ground between the generic group model (the GGM) and the standard model. In contrast to the GGM, algorithms within the AGM (also known as algebraic algorithms) do receive the representation of group elements and may use it in any

[1] Following Abdalla, Benhamouda and Mackenzie [3] and Bernhard, Fischlin and Warinschi [10]. Additionally, the earlier works of Boneh and Venkatesan [18] and Paillier and Vergnaud [40] considered algebraic *reductions*, rather than algebraic adversaries.

way they see fit. The restriction, however, is that whenever an algebraic algorithm outputs a group element, it must provide alongside it a representation of it in the basis of its input group elements. This representation serves as an explanation as to how the output element was computed from the input elements in an algebraic manner. Given the less restrictive nature of the AGM when compared to the GGM,[2] a central line of research over the last couple of years has focused on establishing the security of cryptographic schemes and assumptions within the AGM; see for example [28, 39, 38, 1, 6, 16, 7, 24, 25, 29, 34, 37, 44, 2, 33, 35].

The sequentiality of repeated squaring

The "repeated squaring" function in hidden-order groups, first suggested by Rivest, Shamir and Wagner [43], serves as the basis for the main candidate constructions of both time-lock puzzles and of verifiable delay functions [13, 41, 47]. For years, however, the sequentiality of this function remained purely as an assumption, and there was no known reduction (in idealized models or otherwise) relating it to the hardness of a better-established assumptions. Recently, Katz, Loss and Xu [37] presented a strengthened version of the AGM (the strong AGM) and provided evidence for the sequentiality of repeated squaring within this model.[3] Concretely, they showed that any strongly-algebraic algorithm which manages to speed-up the repeated squaring function in the group QR_N of quadratic residues modulo N (where N is the product of two safe primes), can be used in order to factor the modulus N. Their result provides a novel and important corroboration for the sequentiality of repeated squaring in the group QR_N. However, it is limited in two respects: Firstly, it inherently relies on the algebraic structure of QR_N and does not apply to other hidden-order groups of interest, such as RSA groups or the class group of imaginary quadratic number fields [23, 11, 15, 47]. Secondly, it addresses only the repeated squaring function, and leaves out other possible fine-grained problems in hidden-order groups.

The Uber problem in bilinear groups

The Uber family of problems in bilinear groups was introduced by Boneh, Boyen and Goh [14, 19] as a unified framework for reasoning about computational problems in such groups. A problem in the family is parameterized by three tuples of multivariate polynomials $\vec{F}, \vec{H}, \vec{K}$ and three polynomials Q_1, Q_2, Q_T and is defined by the following task: Given the group elements $\{g_1^{F_i(\vec{x})}\}_i, \{g_2^{H_i(\vec{x})}\}_i, \{g_T^{K_i(\vec{x})}\}_i$ for a random vector $\vec{x}$, compute $g_1^{Q_1(\vec{x})}$, $g_2^{Q_2(\vec{x})}$ and $g_T^{Q_T(\vec{x})}$, where g_1, g_2 and g_T are the generators of the source groups and the target group, respectively. Bauer, Fuchsbauer and Loss [7] recently showed that in the AGM, the hardness of any problem in this family, as long as it does not admit a trivial solution, is implied by the hardness of the q-DLOG problem in one of the source groups.[4] Their result provides a clean and succinct characterization of the Uber family, reducing its hardness to that of a seemingly simpler and better-understood family of problems. However, the parameter q in their result is lower bounded by the *maximal total degree* of the polynomials in $\vec{F}, \vec{H}, \vec{K}$, which may not be optimal. To see why that is, consider the following toy problem for any integer n: Given a generator g and a tuple $(g^{x_1}, g^{x_2}, g^{x_3}, g^{x_1 x_3}, g^{x_2 x_3})$ of group elements for randomly-chosen

[2] The AGM may also be instantiated in the standard model from falsifiable assumptions, as demonstrated by the elegant work of Agrikola, Hofheinz and Kastner [5]. This is in contrast to the GGM [27].

[3] Another recent result [45], proving the equivalence of speeding-up repeated squaring and factoring within the generic ring model is discussed in Section 3.

[4] The q-DLOG problem in a cyclic group $\mathbb{G}$ is defined as: Given a generator g and the q group elements $g^x, \ldots, g^{x^q}$ for a randomly chosen x, compute x.

x_1, x_2, x_3, compute $g^{x_1 x_2 x_3}$. On the one hand, the result of Bauer et al. can be used to conclude that the hardness of this problem in the AGM is implied by the hardness of the 2-DLOG problem. On the other hand, it is not hard to see that this problem is actually equivalent to the Computational Diffie-Hellman (CDH) problem. The CDH problem was proven equivalent to the DLOG problem (i.e., 1-DLOG) in the AGM [28], suggesting that the bound of Bauer et al. might not be optimal with respect to the parameter q.[5]

2 Our Results

In this work, we provide stronger hardness results within the AGM, both for a new fine-grained Uber family of problems in hidden-order groups that we put forth, and for the Uber family of problems in bilinear groups.

2.1 Our Results for Fine-Grained Computations in Hidden-Order Groups

A fine-grained Uber family

As our first contribution, we present a univariate and fine-grained variant of the Uber family of problems in hidden-order groups. Our family of problems generalizes the repeated squaring function [43], as well as well as a computational variant[6] of the generalized BBS problem underlying Boneh and Naor's timed commitments [12, 17] and refinements thereof [31] (see also [30, 32]). A problem in this family is parameterized by integers $u_1, \ldots, u_\ell$ and an integer w, and requires that the adversary computes x^w for a uniformly-chosen group element x, given $(x^{u_1}, \ldots, x^{u_\ell})$ as input. Of course, if one can efficiently express w as a linear combination of $u_1, \ldots, u_\ell$ with integer coefficients, then one can trivially compute x^w from $(x^{u_1}, \ldots, x^{u_\ell})$ using a polynomial number of group operations. Therefore, we carefully define what it means for a strongly-algebraic algorithm to non-trivially solve a problem in our new Uber family, also accounting for the possibility of parallel computations. Looking ahead, the repeated squaring function is obtained by setting $u_1 = 1$ and $w = 2^T$, whereas the generalized BBS problem is obtained by setting $u_1 = 0$, $u_i = 2^{2^{i-2}}$ for $i = 2, \ldots, k+2$ and $w = 2^{2^{k+1}}$. Moreover, the repeated squaring function cannot be trivially solved (according to our triviality notion) in less than T steps, whereas the generalized BBS problem cannot be trivially solved is less than 2^k steps.

The algebraic hardness of the fine-grained Uber problem

Within the strong AGM of Katz, Loss and Xu [37], we provide evidence for the hardness of our new family of problems. Firstly, we present a general hardness result which may be applied in any cryptographic group, and prove that the hardness of any problem in the fine-grained Uber family in a group $\mathbb{G}$ (i.e., the hardness of computing the target group element x^w in a less-than-trivial number of steps), is implied by the RSA assumption in the same group.

[5] Though in the specific case of the toy problem considered above, the result of Fuchsbauer, Kiltz and Loss [28] already shows it to be equivalent to the DLOG problem, this is not the case for general instances of the Uber family in bilinear groups, motivating Theorem 3 below.

[6] In practice, one can always achieve pseudorandomness from this computational variant heuristically by applying a cryptographic hash function (e.g., SHA) onto the output of the problem [8].

▶ **Theorem 1** (informal). *Let $\mathbb{G}$ be a group, let $\ell \in \mathbb{N}$ and let $u_1, \ldots, u_\ell, w \in \mathbb{Z}$. Let* A *be a strongly-algebraic algorithm for the $(u_1, \ldots, u_\ell, w)$-univariate fine-grained Uber problem in the group $\mathbb{G}$, which makes a less-than-trivial number of steps. Then, there exists an algorithm* B *for the RSA problem in $\mathbb{G}$ whose running time and success probability are polynomially-related to those of* A*.*

Theorem 1 immediately implies that any strongly-algebraic algorithm that computes the repeated squaring function in less than T steps in some group $\mathbb{G}$, or solves the generalized BBS problem in less than 2^k steps, can be used in order to solve the RSA problem in the group. Note that Theorem 1 assumes nothing about the group $\mathbb{G}$, and in particular can be applied in any group in which the RSA assumption is believed to hold, such as RSA groups, class groups of imaginary quadratic number fields, and the group QR_N with respect to arbitrary bi-prime moduli. Importantly, Theorem 1 provides evidence for the sequentiality of repeated squaring in class groups, as the RSA problem has been considered and studied in these groups for a while now (see for example [23, 11, 26] and the references therein), whereas the sequentiality of repeated squaring in these groups is a much newer assumption [15, 47]. As far as we are aware, this is the first result supporting the sequentiality of repeated squaring in class groups.

Our second hardness result for the fine-grained Uber problem considers RSA groups with a modulus N which is the product of two safe primes. Informally, within the strong AGM, we prove that in such groups, the hardness of any problem in the fine-grained Uber family is implied by the hardness of factoring N.

▶ **Theorem 2** (informal). *Let N be the product of two safe primes and let $\ell \in \mathbb{N}$ and let $u_1, \ldots, u_\ell, w \in \mathbb{Z}$. Let* A *be a strongly-algebraic algorithm for the $(u_1, \ldots, u_\ell, w)$-univariate fine-grained Uber problem in $\mathbb{Z}_N^*$ which makes a less-than-trivial number of steps. Then, there exists an algorithm* B *for factoring N whose running time and success probability are polynomially-related to those of* A*.*

Observe that Theorem 2 strictly strengthens the result of Katz, Loss and Xu [37] in two respects. Firstly, by considering our new fine-grained Uber family, which captures in particular the sequentiality of the repeated squaring function (considered by Katz, Loss and Xu), but also other problems, such as the generalized BBS problem [17]. Secondly, Theorem 2 considers RSA groups with respect to moduli which are the product of two safe primes, whereas Katz, Loss and Xu consider the group QR_N with respect to the same family of moduli. Since a uniformly-random element in $\mathbb{Z}_N^*$ is in QR_N with probability $1/4$, the hardness of any problem within the fine-grained Uber family with respect to $\mathbb{Z}_N^*$ implies in particular its hardness with respect to QR_N.

Interestingly, to the best of our knowledge, Theorems 1 and 2 are the first applications of the algebraic group model (or a variant thereof) in non-cyclic groups. As far as we are aware, all previous reductions in the AGM were either in cyclic groups of prime order or in the cyclic group QR_N where N is the product of two safe primes. Hence, our work exemplifies for the first time the applicability of the AGM beyond cyclic groups.

2.2 Our Results for Bilinear Groups

The Algebraic Hardness of the Uber problem in bilinear groups

We strengthen the characterization of Bauer, Fuchsbauer and Loss [7] of the Uber family framework in bilinear groups. Concretely, let $\vec{F}, \vec{H}, \vec{K}$ be vectors of polynomials and let Q_1, Q_2, Q_T be polynomials. We prove that within the AGM, as long as these polynomials do

not admit a trivial solution, the hardness of their respective problem in the Uber family is implied by the q-DLOG assumption, where q is lower bounded by the maximal degree in which *a variable appears* in $\vec{F}, \vec{H}, \vec{K}$.

▶ **Theorem 3** (informal). *Let $\mathcal{G}$ be a bilinear group, let $\vec{F}, \vec{H}, \vec{K}$ be vectors of m-variate polynomials and let Q_1, Q_2, Q_T be m-variate polynomials that do not admit a trivial solution to the $(\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2, Q_T)$-Uber problem. Let q be the maximal degree in which a variable appears in $\vec{F}, \vec{H}, \vec{K}$. Then, for any algebraic algorithm A for the $(\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2, Q_T)$-Uber problem in $\mathcal{G}$, there exists an algorithm B for the q-DLOG problem in one of the source groups, whose running time and success probability are polynomially-related to those of A.*

Theorem 3 strengthens the result by Bauer, Fuchsbauer and Loss, since the total degree of a polynomial is always at least the degree of each variable in it, and in many typical cases it is indeed strictly greater. For example, consider again our toy example from before: Given $g, g^{x_1}, g^{x_2}, g^{x_3}, g^{x_1 x_3}$ and $g^{x_2 x_3}$, compute $g^{x_1 x_2 x_3}$. Since the polynomials defining the input elements of this problem are all multilinear,[7] Theorem 3 implies that within the AGM, its hardness is implied by the hardness of the discrete logarithm problem. Recall that the result of Bauer et al. bases the hardness of the aforesaid problem only on the hardness of the seemingly easier 2-DLOG problem.

Application: Group Key Exchange

The toy-problem example might seem contrived at first sight, but it is actually a special case of the Group Computational Diffie-Hellman (G-CDH) problem, which underlies the highly-influential group key-exchange protocols of Bresson, Chevassut, Pointcheval and Quisquater [22, 20, 21]. This problem is parameterized by an integer n (which in group key-exchange applications represents the number of users in the group) and a collection Γ of subsets of $\{1, \ldots, n\}$. The adversary is given a generator g of the group, alongside the group elements $\left\{g^{\prod_{i \in S} x_i}\right\}_{S \in \Gamma}$ for uniformly-chosen $x_1, \ldots, x_n$, and is asked to compute $g^{x_1 \cdots x_n}$. Typically, in group key-exchange protocols Γ includes at least one subset of size $n-1$. In such cases, Theorem 3 reduces the hardness of the (n, Γ)-G-CDH problem to the hardness of the discrete logarithm problem, whereas the previous bound of Bauer et al. reduces it to the hardness of the $(n-1)$-DLOG problem, where n may be a very large integer and perhaps not even a-priori bounded.

The decisional Uber problem in bilinear groups

As our second contribution in bilinear groups, we extend Theorem 3 to the decisional setting. Within the decisional algebraic group model (DAGM) of Rotem and Segev [44] which we extend to accommodate analysis in asymmetric bilinear groups, we prove that the hardness of the *decisional* Uber problem in bilinear groups is implied by the hardness of the q-DLOG problem. Again, the parameter q in our results is the maximal degree in which some variable appears in polynomials defining the problem.

▶ **Theorem 4** (informal). *Let $\mathcal{G}$ be a bilinear group, let $\vec{F}, \vec{H}, \vec{K}$ be vectors of m-variate polynomials and let Q_T be an m-variate polynomial which does not admit a trivial solution to the $(\vec{F}, \vec{H}, \vec{K}, Q_T)$-Uber problem. Let q be the maximal degree in which a variable appears in $\vec{F}, \vec{H}, \vec{K}, Q_T$. Then, for any algorithm A for the decisional $(\vec{F}, \vec{H}, \vec{K}, Q_T)$-Uber problem in $\mathcal{G}$, there exists an algorithm B for the q-DLOG problem in one of the source groups, whose running time and success probability that are polynomially-related to those of A.*

[7] These are the polynomials $X_1, X_2, X_3, X_1 X_3$ and $X_2 X_3$.

The parameters $\vec{F}, \vec{H}, \vec{K}, Q_T$ play a similar role in the decisional setting to their role in the computational setting; for a formal definition of the decisional Uber problem, see the full version. Theorem 4 improves upon a result of Rotem and Segev [44], who showed a similar result, but in their work the parameter q is strictly greater than the maximal total degree of the polynomials defining the problem.

3 Additional Related Work

Bauer, Fuchsbauer and Loss [7] also considered additional variants of the (computational) Uber problem in bilinear groups. Concretely, they proved that their result extends to: The flexible Uber problem, in which the adversary can choose the target polynomials Q_1, Q_2, Q_T; the Uber problem for rational functions, in which the polynomials $\vec{F}, \vec{H}, \vec{K}, Q_1, Q_2$ and Q_T are replaced by rational functions; the Uber problem with decisional oracles, in which the adversary is given access to an oracle for checking whether tuples of group elements satisfy a polynomial relation in the exponent; and the flexible "GeGenUber" problem in which the adversary may choose the generators with respect to which the problem is defined. It seems that the techniques used by Bauer et al. to extend their results to all of the aforesaid variants can be used essentially as is in order to extend our results (Theorems 3 and 4) to accommodate these variants as well, but we leave this task to future work. We refer the reader to [7] for a formal definition of these variants of the Uber problem and for the techniques used by Bauer et al. in order to extend their result to these variants.

The concurrent work of van Baarsen and Stevens [46] also considered reductions supporting the sequentiality of repeated squaring in hidden-order groups. They propose a strengthening of the strong algebraic group model and within it, they reduce the task of speeding up repeated squaring to that of finding a multiple of the group's order. It seems that the results of van Baarsen and Stevens are incomparable to our results in hidden-order groups. On the one hand, we consider the more general fine-grained Uber problem, which we put forth, while they only consider the repeated squaring problem. Additionally, the model of van Baarsen and Stevens seems more restrictive for general hidden-order groups. It assumes that either every algorithm receives as input a set of generators or the ability to uniformly sample group elements (as discussed in their paper, the two assumptions are essentially equivalent). This is essentially the case for $\mathbb{Z}_N^*$, but in general groups, this poses an additional assumption. In contrast, our reduction from the RSA problem is algebraic and does not require the ability to publicly sample uniformly-random group elements. It also works if the group element x specifying the fine-grained Uber problem instance is sampled uniformly by the challenger in a private-coin manner, or if it comes from a non-uniform distribution. In the latter case, we solve that RSA problem for the same distribution. Finally, the reduction of van Baarsen and Stevens loses a factor of n in running time, where n is an upper bound on the number of generators of the group. Potentially, n can be as large as logarithmic in the order of the group. In comparison, our reductions are essentially tight. On the other hand, the reduction of van Baarsen and Steve is from the problem of finding the group's order, which is harder than the RSA problem we consider in general groups. In particular, their result establishes the equivalence, within the strong AGM (which is equivalent to their strengthened model in $\mathbb{Z}_N^*$, where sampling is easy), of speeding up repeated squaring in RSA groups and factoring the RSA modulus, for general bi-prime moduli (whereas we only establish this equivalence when the modulus is the product of two safe primes).

Rotem and Segev [45] put forth the notion of generic-ring delay functions within the generic ring model of Aggarwal and Maurer [4], and the notion of a *sequentiality depth* of such functions. Informally, they proved that in the ring $\mathbb{Z}_N$ where N is an RSA modulus,

generically computing a generic-ring delay function in a number of steps which is less than its sequentiality depth is equivalent to factoring the modulus N. In particular, they showed that this implies that within the generic ring model, computing the repeated squaring function [43] in $\mathbb{Z}_N$ with respect to delay parameter T in less than T sequential steps is equivalent to factoring. Their results are incomparable to ours for several reasons. First, the generic ring model is incomparable to the strong AGM: On the one hand, the generic ring model withholds the elements' representation from the adversary (which the strong AGM does not do); but on the other hand, it allows the adversary to apply all of the ring operation onto pairs of ring elements, whereas the strong AGM requires that the adversary explains how its output was computed solely using the group operation. Secondly, Rotem and Segev only consider RSA groups (within the ring $\mathbb{Z}_N$), whereas Theorem 1 may be applied in any group. Finally, when considering the specific case of RSA groups, our result (Theorem 2) is restricted to RSA groups with respect to a modulus N which is the product of two safe primes, whereas the result of Rotem and Segev is not.

4 Overview of Our Contributions

In this section we provide an informal overview of the main technical ideas underlying our contributions. We start by presenting the techniques we use to derive Theorems 1 and 2 in hidden-order groups, and then move on to describe our techniques in bilinear groups for deriving Theorem 3. For brevity, we do not discuss here how we extend Theorem 3 to the decisional setting to obtain Theorem 4. Due to space limitations, the reader is referred to the full version for further details and formal theorem statements and proofs.

4.1 Our Reductions in Hidden-Order Groups

For simplicity of presentation in this informal overview, when presenting our reduction from the factoring problem to the fine-grained Uber problem in RSA group (Theorem 2), and our reduction from the RSA problem to the fine-grained Uber problem in general groups (Theorem 1), we restrict our attention to the problem of speeding-up the repeated squaring function of Rivest, Shamir and Wagner [43]. The reader is referred to the full version for a formal definition of the fine-grained Uber problem, and for our theorem statements and reductions in their full generality (applying to all problems within the fine-grained Uber family, and not just to speeding-up repeated squaring).

The strong AGM

We prove our results for the fine-grained Uber problem in hidden-order groups in the strong algebraic group model (the SAGM) put forth by Katz, Loss and Xu [37]. Informally speaking, the SAGM strengthens the AGM, by requiring that whenever a strongly-algebraic algorithm A outputs a group element y, it outputs alongside it not only a representation of it in the basis of the input group elements, but also the entire sequence of group operations used to derive this representation. An important feature of this model, is that the length of this sequence is dominated by the running time of the algorithm. Hence, if we view the input elements to a strongly-algebraic algorithm A as polynomials of degree at most d in some underlying indeterminates, then the output y can be viewed as a polynomial of degree at most d^{2^t} in these indeterminates, where t is the running time of A. This is the case since each group operation at most doubles the degree of the highest degree polynomial in the computation up to it. Note that this observation remains true even if A runs in time t on many processors that may perform group operations in parallel. See the full version for a formal definition of the model.

The reduction of Katz, Loss and Xu in QR_N

Recall the reduction of Katz et al. from the factoring problem to the problem of speeding-up the repeated squaring. They focused on the group QR_N of quadratic residues modulo N, for a modulus N which is the product of two safe primes; that is $N = (2p+1)\cdot(2q+1)$, where $p, q, 2p+1$ and $2q+1$ are all primes. Let $T \in \mathbb{N}$ and consider a strongly-algebraic algorithm A, that given a uniformly random group element $x \leftarrow QR_N$ computes x^{2^T} in time $t < T$. The reduction samples such an element x and invokes A on it. Since A is strongly-algebraic, it produces alongside its output y an integer $\alpha \leq 2^t$ such that $y = x^\alpha$. Whenever A succeeds in computing x^{2^T}, it means that $x^\alpha = x^{2^T}$, or equivalently, $x^{2^T-\alpha} = 1$ (all equalities are in the group QR_N). Since $\alpha \leq 2^t < 2^T$, this implies that $\omega = 2^T - \alpha$ is a non-zero multiple of the order of x in QR_N. The analysis of Katz et al. proceeds by observing that when N is the product of two safe primes, then almost all elements in the group are generators, and that the order of the group is $\varphi(N)/4 = p \cdot q$, where $\varphi(\cdot)$ is Euler's totient function. Thus, if A succeeds, then 4ω is almost surely a multiple of $\varphi(N)$, and the reduction is completed by invoking a well-known algorithm for factoring N given a multiple of $\varphi(N)$ (e.g., [36, Theorem 8.50]).

Our reduction in RSA groups (Theorem 2)

Unlike in the group QR_N, when moving to consider the RSA group $\mathbb{Z}_N^*$, the group is no longer cyclic and hence a random element in the group is never a generator. However, our generalization of the result of Katz et al. to the RSA group $\mathbb{Z}_N^*$ is based on the observation that their reduction actually does not rely on the fact that the sampled x is a generator of QR_N. Instead, it only uses the fact that with overwhelming probability over the choice of x, its order satisfies a certain relation with $\varphi(N)$. We use this observation to prove that the reduction of Katz et al. can be applied not only in QR_N when N is the product of two safe primes, but also in $\mathbb{Z}_N^*$ when N is of this form. Concretely, denoting $N = (2p+1)\cdot(2q+1)$, we use the isomorphism of $\mathbb{Z}_N^*$ to the product group $\mathbb{Z}_2^2 \times \mathbb{Z}_p \times \mathbb{Z}_q$ in order to argue that almost all elements in $\mathbb{Z}_N^*$ have order either $p \cdot q = \varphi(N)/4$ or order $2p \cdot q = \varphi(N)/2$. Therefore, it is still the case that that whenever A succeeds in computing x^{2^T}, it must be that $4\omega = 4(2^T - \alpha)$ is a multiple of $\varphi(N)$ and the correctness of the reduction follows.

Our reduction in general groups (Theorem 1)

Let $\mathbb{G}$ be an abelian group. The goal of our reduction is to solve the RSA problem in $\mathbb{G}$, where the problem is parameterized by an integer e which is coprime to the order of $\mathbb{G}$. That is, given a uniformly random $u \in \mathbb{G}$, we need to find a group element $w \in \mathbb{G}$ such that $w^e = u$ (meaning, w is the eth root of u). Let $T \in \mathbb{N}$ and consider a strongly-algebraic algorithm A, that given a uniformly random group element $x \leftarrow \mathbb{G}$ computes x^{2^T} in time $t < T$. Our reduction starts by invoking A on input u (the input to the RSA problem). As before, since A is strongly-algebraic, it produces alongside its output y an integer $\alpha \leq 2^t$ such that $y = u^\alpha$. Following a similar argument as in the previous reduction, if y is indeed equal to u^{2^T}, then $\omega = 2^T - \alpha$ is a non-zero multiple of the order of u in $\mathbb{G}$. The new idea underlying our reduction from the RSA problem, is that we can use this information about the order of u to find its eth root. Suppose that we could find an inverse d of e modulo ω; i.e., an integer d satisfying $ed = n \cdot \omega + 1$ for some integer n. Then, we would be done, since u^d would be the eth root of u:

$$\left(u^d\right)^e = u^{ed} = u^{n\cdot\omega+1} = (u^\omega)^n \cdot u = 1 \cdot u = u, \tag{1}$$

where $u^{\omega} = 1$ because ω is a multiple of u's order. The problem is that e might not have an inverse modulo ω, as the two integers might not be relatively prime. To remedy this situation, we factor out from ω any common divisors that it shares with e, by computing $\omega' = \omega/\mathsf{gcd}(\omega, e)$. On the one hand, we are now guaranteed that ω' and e are coprime, and we can find an inverse d' of e modulo ω'. On the other hand, the key observation is that ω' must still be a multiple of u's order in $\mathbb{G}$. This is because e is coprime to the order of $\mathbb{G}$ and thus, by Lagrange's theorem, it is also coprime to the order of u. This means that if we write $\omega = \mathsf{order}(u) \cdot c$ for some integer c, then

$$\omega' = \frac{\omega}{\mathsf{gcd}(\omega, e)} = \frac{\mathsf{order}(u) \cdot c}{\mathsf{gcd}(\mathsf{order}(u) \cdot c, e)} = \mathsf{order}(u) \cdot \frac{c}{\mathsf{gcd}(c, e)}$$

and ω' is indeed a multiple of $\mathsf{order}(u)$. Hence, our reduction may simply output $u^{d'}$, and the same analysis from Eq. (1) still applies, replacing ω and d with ω' and d' respectively.

4.2 Our Reduction in Bilinear Groups

For simplicity of presentation, we focus here on the case of symmetric bilinear groups. In this setting, we consider a single source group $\mathbb{G}$ of prime order $p \in \mathbb{N}$ that is equipped with a bilinear map e, mapping pairs of elements in $\mathbb{G}^2$ to elements in a target group $\mathbb{G}_T$. We also restrict our attention to a simple problem within the Uber family, referred to below as the (f, h)-Uber problem, as the reduction in this case already captures the gist of the techniques used in our proof of Theorem 3. In the (f, h)-Uber problem the adversary is given g and $g^{f(\vec{x})}$ and is required to compute $g_T^{h(\vec{x})}$, where g is a generator of $\mathbb{G}$, $g_T = e(g, g)$ is a generator of $\mathbb{G}_T$, f and h are polynomials parameterizing the problem, and $\vec{x} = (x_1, \ldots, x_m)$ is a m-tuple of elements in $\mathbb{Z}_p$ chosen independently and uniformly at random. We assume that there are no integers $\alpha, \beta, \gamma \in \mathbb{Z}_p$ such that $h = \alpha + \beta \cdot f + \gamma \cdot f^2$ over $\mathbb{Z}_p$, as otherwise the problem is trivial to solve and we cannot hope to reduce the q-DLOG problem (or any other problem which we believe to be hard) to it.[8] We start by recalling the reduction of Bauer et al. [7] and then move to describe our reduction and how it improves upon it.

The reduction of Bauer, Fucshbauer and Loss

Let A be an algebraic algorithm for the (f, h)-Uber problem. The reduction receives as input $g, g^x, \ldots, g^{x^q}$ for a uniformly sampled $x \leftarrow \mathbb{Z}_p$, and its goal is to find x. The idea of Bauer et al. was to have the reduction "plant" m independent randomized versions of the secret exponent x as the m secret exponents in a random instance of the (f, h)-Uber problem and then invoke A on this instance. This is done by sampling $\alpha_i, \beta_i \leftarrow \mathbb{Z}_p$ for each $i \in [m]$ and invoking A on input $(g, g^{f(\vec{x'})})$ where $\vec{x'} = (\alpha_1 \cdot x + \beta_1, \ldots, \alpha_m \cdot x + \beta_m)$. Since A is an algebraic algorithm, it provides alongside its output $y \in \mathbb{G}_T$ three integers $\alpha, \beta, \gamma \in \mathbb{Z}_p$ such that $y = e(g, g)^{\alpha} \cdot e(g, g^{f(\vec{x'})})^{\beta} \cdot e(g^{f(\vec{x'})}, g^{f(\vec{x'})})^{\gamma}$. Whenever A succeeds, it means that $y = g_T^{h(\vec{x'})}$, and hence, since g_T is a generator of $\mathbb{G}_T$, we obtain that

$$h(\vec{x'}) = \alpha + \beta \cdot f(\vec{x'}) + \gamma \cdot \left(f(\vec{x'})\right)^2. \tag{2}$$

[8] If such integers existed, the adversary could simply compute and output $e(g, g)^{\alpha} \cdot e(g, g^{f(\vec{x})})^{\beta} \cdot e(g^{f(\vec{x})}, g^{f(\vec{x})})^{\gamma}$.

Observe that Eq. (2) is a univariate equation over the finite field $\mathbb{F}_p$. Roughly speaking, the non-triviality of the (f,h)-Uber problem guarantees that with overwhelming probability, Eq. (2) is not the trivial equation. Therefore, the reduction simply uses any efficient polynomial factorization algorithm (e.g., Berlekamp's algorithm [9, 42]) to find all solutions to Eq. (2) and then checks each of them to see if it is the secret exponent x. Note that in order to compute $g^{f(\vec{x'})}$ given as input to A, the reduction needs access to $g, g^x, \ldots, g^{x^q}$ for q that is the *total* degree of f.

Our improved reduction (Theorem 3)

In order to reduce the parameter q needed by the reduction, our idea is to plant the secret exponent x in place of just one of the exponents $x_1, \ldots, x_m$ in a random instance of the (f,h)-Uber problem, and sample the rest of the exponents uniformly at random. Concretely, our reduction samples a random index $i^* \leftarrow [m]$ and plants x instead of x_{i^*}: It samples $m-1$ additional values $x_1, \ldots, x_{i^*-1}, x_{i^*+1}, \ldots, x_m \leftarrow \mathbb{Z}_p$ and invokes A on input $(g, g^{f(\vec{x''})})$ where $\vec{x''} = (x_1, \ldots, x_{i^*-1}, x, x_{i^*+1}, \ldots, x_m)$. Observe that indeed, in order to compute the group element $g^{f(\vec{x''})}$ given as input to A, all the reduction needs is access to $g, g^x, \ldots, g^{x^q}$ for a parameter q which is at least the degree of x_{i^*} in f. In particular, the reduction can be efficiently implemented as long as q is at least the maximal degree in which some variable appears in f. As before, since A is algebraic, it outputs alongside its output $y \in \mathbb{G}_T$ three integers $\alpha, \beta, \gamma \in \mathbb{Z}_p$ such that $y = e(g,g)^\alpha \cdot e(g, g^{f(\vec{x''})})^\beta \cdot e(g^{f(\vec{x''})}, g^{f(\vec{x''})})^\gamma$, and whenever $y = g_T^{h(\vec{x''})}$, it holds that

$$h(\vec{x''}) = \alpha + \beta \cdot f(\vec{x''}) + \gamma \cdot \left(f(\vec{x''})\right)^2 . \tag{3}$$

Alas, we can no longer simply solve Eq. (3) for x. The problem is that for our choice of $x_1, \ldots, x_{i^*-1}, x_{i^*+1}, \ldots, x_m$, Eq. (3) might be a trivial equation, yielding no information about x. Worse still, this situation may arise with high probability over the choice of i^* and of $x_1, \ldots, x_{i^*-1}, x_{i^*+1}, \ldots, x_m$. So instead, our reduction uses the following observation by Rotem and Segev [44], which in turn generalizes the previous work of Fuchsbauer, Kiltz and Loss [28]. For a non-zero polynomial $\ell(X_1, \ldots, X_m)$ in the indeterminates $X_1, \ldots, X_m$, we define a cascade of multivariate polynomials recursively: We set $\ell_1 = \ell$; and for every $i \in \{2, \ldots, m\}$, we let $\ell_i(X_i, \ldots, X_m)$ be the first non-zero coefficient of $\ell_{i-1}(X_{i-1}, \ldots, X_m)$, when ℓ_{i-1} is written as a univariate polynomial in the indeterminate X_{i-1} with coefficients which are polynomials in $X_i, \ldots, X_m$. Rotem and Segev proved that for every vector $\vec{z} = (z_1, \ldots, z_m) \in \mathbb{Z}_p^m$ such that $\ell(\vec{z}) = 0$, there exists $t^* \in [m]$ such that: The univariate polynomial $v(X) = \ell_{t^*}(X, z_{t^*+1}, \ldots, z_m)$ is not the zero polynomial and additionally $v(z_{t^*}) = 0$.[9] Using this observation, our reduction considers the m-variate polynomial $\ell(\vec{X}) = \alpha + \beta \cdot f(\vec{X}) + \gamma \cdot \left(f(\vec{X})\right)^2 - h(X)$, and computes the polynomial $\ell_{i^*}(X_{i^*+1}, \ldots, X_m)$ from it as described by the above recursive procedure, where i^* is the index sampled by the reduction when preparing the (f,h)-Uber instance to A. It then factors the univariate polynomial $v(X) = \ell_{i^*}(X, x_{i^*+1}, \ldots, x_m)$ to find all of its roots, where $x_{i^*+1}, \ldots, x_m$ are the random $\mathbb{Z}_p$ values chosen by the reduction for generating the (f,h)-Uber instance. As for the success probability of the reduction, whenever A succeeds in computing the $g_T^{h(\vec{x''})}$, it holds that $\vec{x''}$

[9] Fuchsbauer, Kiltz and Loss [28] essentially observed that this holds for bi-linear polynomials to reduce the hardness of the Computational Diffie-Hellman (CDH) problem to that of the discrete log problem within the AGM.

is a root of the m-variate polynomial ℓ. By the observation of Rotem and Segev, this means that there is an index $t^* \in [m]$ such that if the index i^* guessed by the reduction matches it, then $v(X) = \ell_{i^*}(X, x_{i^*+1}, \ldots, x_m)$ is not the zero polynomial, and x is a root of it. Hence, if A succeeds and $i^* = t^*$, then the reduction will successfully retrieve the secret exponent x of the q-DLOG problem.

References

1 Michel Abdalla, Manuel Barbosa, Tatiana Bradley, Stanislaw Jarecki, Jonathan Katz, and Jiayu Xu. Universally composable relaxed password authenticated key exchange. In *Advances in Cryptology – CRYPTO '20*, pages 278–307, 2020.

2 Michel Abdalla, Manuel Barbosa, Jonathan Katz, Julian Loss, and Jiayu Xu. Algebraic adversaries in the universal composability framework. In *Advances in Cryptology – ASIACRYPT '21*, pages 311–341, 2021.

3 Michel Abdalla, Fabrice Benhamouda, and Philip MacKenzie. Security of the J-PAKE password-authenticated key exchange protocol. In *IEEE Symposium on Security and Privacy*, pages 571–587, 2015.

4 Divesh Aggarwal and Ueli Maurer. Breaking RSA generically is equivalent to factoring. In *Advances in Cryptology – EUROCRYPT '09*, pages 36–53, 2009.

5 Thomas Agrikola, Dennis Hofheinz, and Julia Kastner. On instantiating the algebraic group model from falsifiable assumptions. In *Advances in Cryptology – EUROCRYPT '20*, pages 96–126, 2020.

6 Benedikt Auerbach, Federico Giacon, and Eike Kiltz. Everybody's a target: Scalability in public-key encryption. In *Advances in Cryptology – EUROCRYPT '20*, pages 475–506, 2020.

7 Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In *Advances in Cryptology – CRYPTO '20*, pages 121–151, 2020.

8 Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

9 Elwyn Ralph Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24(111):713–735, 1970.

10 David Bernhard, Marc Fischlin, and Bogdan Warinschi. On the hardness of proving CCA-security of signed ElGamal. In *Public-Key Cryptography – PKC ,16*, pages 47–69, 2016.

11 LenIngrid Biehl, Johannes Buchmann, Safuat Hamdy, and Andreas Meyer. A signature scheme based on the intractability of computing roots. *Designs, Codes and Cryptography*, 25(3):223–236, 2002.

12 Lenore Blum, Manuel Blum, and Michael Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

13 Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Advances in Cryptology – CRYPTO '18*, pages 757–788, 2018.

14 Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology – EUROCRYPT '05*, pages 440–456, 2005.

15 Dan Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712, 2018.

16 Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. Cryptology ePrint Archive, Report 2020/081, 2020.

17 Dan Boneh and Moni Naor. Timed commitments. In *Advances in Cryptology – CRYPTO '00*, pages 236–254, 2000.

18 Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In *Advances in Cryptology – EUROCRYPT '98*, pages 59–71, 1998.

19 Xavier Boyen. The Uber-assumption family – A unified complexity framework for bilinear groups. In *Proceedings of the 2nd International Conference on Pairing-based Cryptography*, pages 39–56, 2008.

20 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.

21 Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Provably secure authenticated group diffie-hellman key exchange. *ACM Transactions on Information and System Security*, 10(3):10:1–10:45, 2007.

22 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.

23 Johannes Buchmann and Safuat Hamdy. A survey on IQ cryptography. In *In Proceedings of Public Key Cryptography and Computational Number Theory*, pages 1–15, 2001.

24 Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *Advances in Cryptology – EUROCRYPT '20*, pages 738–768, 2020.

25 Geoffroy Couteau and Dominik Hartmann. Shorter non-interactive zero-knowledge arguments and ZAPs for algebraic languages. In *Advances in Cryptology – CRYPTO '20*, pages 768–798, 2020.

26 Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In *Advances in Cryptology – EUROCRYPT '02*, pages 256–271, 2002.

27 Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In *Advances in Cryptology – ASIACRYPT '02*, pages 100–109, 2002.

28 Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In *Advances in Cryptology – CRYPTO '18*, pages 33–62, 2018.

29 Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In *Advances in Cryptology – EUROCRYPT '20*, pages 63–95, 2020.

30 Juan A. Garay and Markus Jakobsson. Timed release of standard digital signatures. In *Financial Cryptography*, pages 190–207, 2002.

31 Juan A. Garay, Philip D. MacKenzie, Manoj Prabhakaran, and Ke Yang. Resource fairness and composability of cryptographic protocols. In *Proceedings of the 4th Theory of Cryptography Conference*, pages 404–428, 2006.

32 Juan A. Garay and Carl Pomerance. Timed fair exchange of standard signatures. In *Financial Cryptography*, pages 190–207, 2003.

33 Ashrujit Ghoshal and Stefano Tessaro. Tight state-restoration soundness in the algebraic group model. In *Advances in Cryptology – CRYPTO '21*, pages 64–93, 2021.

34 Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 2007–2023, 2020.

35 Julia Kastner, Julian Loss, and Jiayu Xu. On pairing-free blind signature schemes in the algebraic group model. To appear in *Public-Key Cryptography — PKC '22* (available at `https://eprint.iacr.org/2020/1071.pdf`), 2022.

36 Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography (2nd edition)*. Chapman & Hall/CRC press, 2014.

37 Jonathan Katz, Julian Loss, and Jiayu Xu. On the security of time-lock puzzles and timed commitments. In *Proceedings of the 18th Theory of Cryptography Conference*, pages 390–413, 2020.

38 Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2111–2128, 2019.

39 Taiga Mizuide, Atsushi Takayasu, and Tsuyoshi Takagi. Tight reductions for Diffie-Hellman variants in the algebraic group model. In *Topics in Cryptology – CT-RSA '19*, pages 169–188, 2019.

40 Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *Advances in Cryptology – ASIACRYPT '05*, pages 1–20, 2005.

41 Krzysztof Pietrzak. Simple verifiable delay functions. In *Proceedings of the 10th Conference on Innovations in Theoretical Computer Science*, pages 60:1–60:15, 2019.

42 Michael Oser Rabin. Probabilistic algorithms in finite fields. *SIAM Journal on Computing*, 9(2):273–280, 1980.

43 R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto, 1996.

44 Lior Rotem and Gil Segev. Algebraic distinguishers: From discrete logarithms to decisional uber assumptions. In *Proceedings of the 18th Theory of Cryptography Conference*, pages 366–389, 2020.

45 Lior Rotem and Gil Segev. Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. In *Advances in Cryptology – CRYPTO '20*, pages 481–509, 2020.

46 Aron van Baarsen and Marc Stevens. On time-lock cryptographic assumptions in abelian hidden-order groups. In *Advances in Cryptology – ASIACRYPT '21*, pages 367–397, 2021.

47 Benjamin Wesolowski. Efficient verifiable delay functions. In *Advances in Cryptology – EUROCRYPT '19*, pages 379–407, 2019.

Universally Composable Almost-Everywhere Secure Computation

Nishanth Chandran
Microsoft Research, Bangalore, India

Pouyan Forghani
Texas A&M University, College Station, TX, USA

Juan Garay
Texas A&M University, College Station, TX, USA

Rafail Ostrovsky
University of California, Los Angeles, CA, USA

Rutvik Patel
Texas A&M University, College Station, TX, USA

Vassilis Zikas
Purdue University, West Lafayette, IN, USA

Abstract

Most existing work on secure multi-party computation (MPC) ignores a key idiosyncrasy of modern communication networks, that there are a limited number of communication paths between any two nodes, many of which might even be corrupted. The problem becomes particularly acute in the information-theoretic setting, where the lack of trusted setups (and the cryptographic primitives they enable) makes communication over sparse networks more challenging. The work by Garay and Ostrovsky [EUROCRYPT'08] on *almost-everywhere MPC* (AE-MPC), introduced "best-possible security" properties for MPC over such incomplete networks, where necessarily some of the honest parties may be excluded from the computation.

In this work, we provide a universally composable definition of *almost-everywhere security*, which allows us to automatically and accurately capture the guarantees of AE-MPC (as well as AE-communication, the analogous "best-possible security" version of secure communication) in the Universal Composability (UC) framework of Canetti. Our results offer the first simulation-based treatment of this important but under-investigated problem, along with the first simulation-based proof of AE-MPC. To achieve that goal, we state and prove a general composition theorem, which makes precise the level or "quality" of AE-security that is obtained when a protocol's hybrids are replaced with almost-everywhere components.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols; Security and privacy → Information-theoretic techniques; Security and privacy → Formal security models

Keywords and phrases Secure multi-party computation, universal composability, almost-everywhere secure computation, sparse graphs, secure message transmission

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.14

Related Version *Full Version*: `https://eprint.iacr.org/2021/1398` [16]

Funding *Juan Garay*: Work partially supported by NSF grants CNS-2001082 and CNS-2055694. *Rafail Ostrovsky*: Supported in part by DARPA under Cooperative Agreement HR0011-20-2-0025, NSF grant CNS-2001096, US-Israel BSF grant 2015782, Cisco Research Award, Google Faculty Award, JP Morgan Faculty Award, IBM Faculty Research Award, Xerox Faculty Research Award, OKAWA Foundation Research Award, B. John Garrick Foundation Award, Teradata Research Award, Lockheed-Martin Research Award and Sunday Group. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official

policies, either expressed or implied, of DARPA, the Department of Defense, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copyright annotation therein.
Vassilis Zikas: Research supported in part by NSF grants CNS-2055599 and CNS-2001096, BSF grant 2020277, and by Sunday Group.

Acknowledgements The authors are grateful to Ran Canetti for useful discussions during preliminary stages of this work.

1 Introduction

Secure multi-party computation (MPC) allows n parties communicating over a network to compute a function on their private inputs so that an adversary corrupting some of the parties can neither disrupt the computation (correctness) nor learn more than (what can be inferred from) the output of the function being computed (privacy).

Despite great progress on the problem since it was first introduced and proven feasible [44, 27, 8, 18] involving hundreds, if not thousands, of publications in cryptography and security, and, more recently, even implemented systems, the overwhelming majority of the solutions assume a *complete* communication network of either authenticated (aka reliable) or secure (both authenticated and private) point-to-point channels. In fact, with only a few exceptions, this is the case for both practical and theoretical works on MPC, and in particular for works on composable security of MPC – indeed, the latter almost exclusively assume a network that cannot be disconnected by the adversary. This creates a disconnect between the vast MPC literature and modern *ad-hoc* networks, such as the Internet, where the communication might be occurring over an incomplete graph, with some nodes even being routing nodes.

At first approximation, there are two situations that might present themselves in such an incomplete network: Either the adversary is able to disconnect the communication graph – by corrupting nodes whose edges are in cuts of the graph – or not. In the former case, it is known that if the parties do not share an authentication-enabling setup, such as a PKI, then the best that can be achieved is the so-called *secure computation without authentication* [6]: The adversary is able to break down the player set into connected components, so that parties in different connected components compute different instances of the function with inputs from the component – and all other inputs chosen by the adversary, and potentially different for each component. Even this weak form of security is only achievable for computationally bounded adversaries; if one is after information-theoretic (aka unconditional) security, where the adversary is unbounded, then the above guarantee is too much to ask for.

Notwithstanding, even in the latter case, where the adversary cannot disconnect the network, the situation is trickier than one might expect. Indeed, if a PKI-like setup is not assumed[1] then it is known that secure communication between any two parties requires the existence of $O(n)$ paths among them (known to or discoverable by the receiver), the majority of which must remain uncorrupted. This is the well-known *secure message transmission* (SMT) problem [20]. The result holds even for the *reliable message transmission* (RMT) problem, in which only correctness is required.

That leads to the following natural question: What is the "best-possible" MPC security we can obtain in such a situation where SMT cannot be in general guaranteed? Towards answering this question, Garay and Ostrovsky [25] introduced the properties of *almost-everywhere MPC* (AE-MPC), which extended the concept of AE reliable communication previously studied by Dwork et al. [21]. In a nutshell, the paradigm of *almost-everywhere*

[1] A PKI trivializes this case as a complete graph can be built by gossip (i.e., flooding) of signed messages.

security (AE-security) recognizes that when even all-to-all SMT is not possible (and only AE-SMT is available), then inevitably there will be uncorrupted parties for which we are unable to offer the security guarantees that honest parties enjoy in MPC (privacy, correctness, etc). The core mission is then to minimize the number of such left-out (aka *doomed*) parties in an AE-secure construction, while tolerating the maximum number of corruptions.

However, despite a number of elegant combinatorial arguments to achieve the above goal, the security definition used by these constructions has not caught up with the state of the art in MPC security. In particular, to the best of our knowledge, there exists no simulation-based treatment of AE-security. This means that one cannot directly compose the elegant constructions of AE-secure primitives into a higher level protocol. For example, one would hope to be able to prove that running a standard MPC protocol over an AE-SMT network yields an AE-MPC protocol which does not leave more doomed parties than the underlying AE-SMT construction. Given the state of the art, such a modular statement would be impossible, and one would need to prove AE-MPC security from scratch. Instead, a simulation-based treatment in one of the composable security frameworks would inherit a modular composition theorem making such statements tractable and simpler.

This work's main goal is to derive such a treatment in the Universal Composability (UC) framework of Canetti [13]. A major challenge, which we tackle, is to obtain a generic definition of AE-security which can be applied to any type of functionality and captures both AE-communication and AE-computation, two primitives whose treatment has been very different. In fact, we achieve this goal by introducing a generic, composition-preserving transformation from a secure variant of a functionality to its AE-secure counterpart. We show that the derived AE-secure functionalities for secure communication (AE-RMT and AE-SMT) and for secure MPC (AE-MPC): (1) preserve all the desired properties of the previous definitions, and (2) are realized by (straightforward UC adaptations of) classical AE-secure protocols. Since our treatment preserves composability of the (AE-)security statements, we obtain, as a simple corollary, the first simulation-based proof of AE-MPC.

In passing, we note that although we adopt the language of UC, our definitional framework is generic and can be applied to any of the main-stream composable security frameworks for cryptographic protocols [3, 15, 37, 31, 11, 4]. Before providing more details on our results, we first provide some necessary literature background.

1.1 Related Work

The origins of the "almost-everywhere" (AE) notion can be traced back to the work of Dwork et al. [21], who considered the task of Byzantine agreement [39, 36] over sparse communication networks. In such networks, correctness cannot be guaranteed for all honest parties, since for example the adversary can isolate a node by corrupting all its neighbors. Thus, some honest parties must be given up, and correctness is guaranteed only almost-everywhere, i.e., only for the remaining honest parties. The AE notion can be applied to other distributing computing tasks as well: Given a set of parties P and an adversary who corrupts $T \subseteq P$, the parties in some set $D \subseteq P - T$ (D for "doomed") are considered abandoned and the correctness conditions of the task are only guaranteed for the parties in $W = P - T - D$ (called "privileged"). Note that both D and W are functions of T as well as of the underlying protocol and graph. The number of doomed parties thus becomes another parameter to the problem, and the goal is to construct a low-degree network (ideally of constant degree) admitting a protocol that tolerates a large number t of corruptions (ideally, a constant fraction) while dooming as few nodes as possible (ideally $O(t)$ for constant-degree networks).

Returning to the problem of Byzantine agreement, Dolev [19] showed that it requires connectivity at least $2t + 1$ to solve, which implies that every node in the network must have degree $\Omega(t)$. Given this high connectivity requirement, Dwork et al. [21] proposed the notion

of *AE agreement*, in which the agreement and validity properties are guaranteed only for the privileged parties. They showed how to simulate, over an incomplete network, an agreement protocol designed for a complete network by replacing the point-to-point communication with a transmission scheme that works over multiple paths between any two nodes. Thus, they reduced AE agreement to AE reliable message transmission (AE-RMT), which guarantees that any two privileged nodes can communicate perfectly reliably.

Dwork et al. gave a number of constructions achieving AE-RMT with various combinations of parameters; the most important is a constant-degree graph admitting an AE-RMT scheme tolerating $t = O(n/\log n)$ corruptions while dooming $O(t)$ nodes. Several follow-up works have obtained improved parameters for AE-RMT (and thus also for AE agreement). Upfal [43] gave a transmission scheme tolerating $t = O(n)$ corruptions and dooming $O(t)$ nodes in a network of constant degree, which is the optimal set of parameters, but the protocol is inefficient. Chandran et al. [17] proposed a scheme tolerating $t = O(n)$ corruptions and dooming $O(t/\log n)$ nodes in a network of polylogarithmic degree. Most recently, Jayanti et al. [32] used the probabilistic method to show the existence of a logarithmic-degree graph admitting an AE-RMT scheme with the same parameters, strictly improving the [17] result.

Due to the results in [19, 20], standard MPC (guaranteeing correctness and privacy for all honest parties) is possible only in networks with connectivity at least $2t + 1$. To circumvent this high-connectivity requirement and still obtain a meaningful notion of (property-based) MPC over sparse networks, Garay and Ostrovsky [25] introduced the notion of AE-MPC, which guarantees correctness and privacy only for the privileged parties[2].

"Regular" information-theoretic MPC (i.e., MPC over a complete network) requires $t < n/3$ [8, 18]. In the AE setting, the effect of dooming nodes is equivalent to letting the adversary corrupt some additional t' nodes (which are doomed) by requesting the corruption of t nodes (which are actually corrupted). As shown by Garay and Ostrovsky, AE-MPC in the information-theoretic setting can be achieved when $t + t' < n/3$. Their approach resembles that of Dwork et al. [21] for simulating a protocol meant for a complete network, but to replace point-to-point secure channels, they introduced a new model for the existing (perfectly) SMT problem termed *secure message transmission by public discussion* (SMT-PD).

The original SMT problem [20] considers two honest parties, a sender S and a receiver R, connected by n disjoint "wires". The task is for S to send a message to R in the presence of a computationally unbounded adversary $\mathcal{A}$ who can adaptively corrupt up to t of the wires. SMT requires that the message be conveyed perfectly reliably to R, and also that no information about the message leaks to $\mathcal{A}$. While the simpler task of RMT (with no secrecy requirement) can be achieved for $t < n/2$ by simply duplicating the message over all wires, Dolev et al. [20] showed that SMT is also possible if and only if $t < n/2$. Since their initial feasibility result, much more efficient protocols have been introduced [40, 42, 1, 35, 28, 41].

The SMT-PD model overcomes the necessity of $2t + 1$ wires in SMT by allowing access to an authentic and reliable public channel. Given such a channel (which can be constructed using, e.g., a broadcast protocol), Garay and Ostrovsky [25] gave a protocol that is secure as long as one wire remains honest, at the cost of a small error. To use their SMT-PD protocol over sparse networks (in effect achieving AE-SMT), the wires are replaced by multiple paths between a pair of nodes and the public channel is replaced by AE broadcast. Garay and Ostrovsky showed how to construct graphs that admit SMT-PD from any of the networks in

[2] Technically, they considered the related task of *secure function evaluation* (SFE). We do the same, although for consistency we still refer to the functionality that we realize as AE-MPC.

the AE agreement literature, with asymptotically preserved parameters. Finally, they showed how to "compile" a standard information-theoretic MPC protocol into an AE-MPC protocol over any such graph, dooming the same number of parties as the underlying network.

To reiterate, all the above constructions are shown secure in a property-based manner. Other related notions include hybrid failure models (e.g., [26, 23]) and the model of Alon et al. [2]. In the AE setting, adversarial corruptions also have the effect of indirectly influencing the behavior of some of the honest parties (those who are doomed), but in our model this other type of failure is defined structurally, based on the graph and the set of corruptions. Also related is the work by King and Saia [34] and follow-ups (e.g., [9, 10]), who considered full (not AE) Byzantine agreement over complete networks, but without all-to-all communication.

1.2 Overview of Our Results

In this work we put forth the first composable (simulation-based) definition and treatment of AE-security. In particular, we devise a definition in Canetti's UC framework [13] and prove that the (UC adaptation of) existing AE-secure communication/computation protocols achieve this definition. We emphasize that all of our constructions tolerate adaptive corruptions.

There are several challenges associated with such a task. First, as should be evident from the above discussion, the related literature – from RMT/SMT, to Byzantine agreement, to MPC, and even their AE counterparts – treats the underlying network in different ways: e.g., in MPC, the network is typically a complete graph of point-to-point channels, whereas the literature on (AE-)RMT assumes multiple paths (wires or indirect paths) between two parties. Thus, to derive a formulation general enough to capture the security of the above constructions, one first needs to develop a unified approach. Towards this goal, we adopt the graph model as a basis for all these protocols, and express the wires in the RMT/SMT literature as a simple graph which for each wire includes a path going through a unique "wire-party." We can then model corrupted wires as standard (party) corruptions in UC.

The second, and more thorny challenge is regarding the (simulation of) doomed parties. Recall that those are parties that due to their poor connectivity (which might be the result of the sparsity of the graph and the corruption choices of the adversary) cannot enjoy the security guarantees that the protocol is designed to offer to honest parties (e.g., correctness and privacy for an MPC protocol). A strawman approach would be to capture those parties plainly as corrupted. This, however, is problematic in several ways: First, corrupted parties lose their security guarantees as soon as they become corrupted, unlike doomed parties who might, at the adversary's discretion, still be allowed some level of security. In particular, the real-world adversary might allow those parties to receive their outputs, which would mean that in the ideal world, the simulator would also need to allow them to produce an output on their output tape, which is not allowed by the UC corruption mechanism.

An attempt to fix the above issue would be to define weaker corruption types corresponding to the flexible guarantees offered to the doomed parties. This, however, is also problematic, as corruptions in UC are by default known to (and declared by) the adversary/environment, whereas the actual identities of doomed parties are not, and depend on the behavior of the adversary (not just the identities of malicious parties). In particular, an adversary following, e.g., a random strategy might not even be aware who is becoming doomed by this strategy.

A third attempt would be to completely change the corruption mechanism of UC so that certain corruptions are not to be declared by the environment. But this would immediately invalidate the composition theorem, which defeats the purpose of using UC in the first place.

It might seem like we are in a deadlock, but the second attempt above is the one that breaks through. In particular, we observe that although the adversary might not include in

its view the identities of the doomed parties, its behavior still defines these identities and the corresponding guarantees they receive. This is similar to how inputs of corrupted parties are treated in standard UC security: It is the job of the simulator to extract them from the adversary and hand them over to the functionality.

Accordingly, instead of modifying the foundations of UC, we define a class of functionalities which take requests from their adversary (simulator) to mark parties as doomed, and allow the simulator to use these parties as if they were corrupted, but without declaring them as corrupted to the framework and without grounding their input/output tapes (e.g., the simulator might still instruct this new functionality to deliver output for doomed parties). In fact, this is done in a black-box manner, by *wrapping* an underlying (non-AE) functionality.

In more detail, our AE wrapper builds the entire infrastructure of UC around it (including a fake corruption directory), and whenever a doom request comes in, the wrapper pretends towards its wrapped functionality to be an adversary that corrupts this party. This way, the party remains honest as far as the UC experiment is concerned, but the wrapper now has the ability to give full control over this party to the simulator it interacts with.

The final piece of the puzzle is capturing different ratios of corrupted vs doomed parties while making a composable statement. Here we use an idea inspired by [5]: We parameterize the wrapper by the set of all allowable corruption/doom patterns, and make sure that any request outside this set is ignored. For example, to prove security of AE-MPC with $t < \alpha n$ corruptions and $d < \beta n$ doomed parties, we can parameterize the wrapper with the pair (α, β) and ignore requests of simulators that do not respect the above requirements.

In fact, to allow for the tightest possible results that accurately translate non-threshold corruption/doom patterns (the types of results we get by using structural properties of the underlying graph), we draw inspiration from the mixed general adversary literature [29, 7]. That is, we parameterize the wrapper with a corruption/doom structure ("doom structure" for short), consisting of all allowed pairs (C, D) where parties in D can be doomed simultaneously to parties in C being corrupted. As is common in the general adversary literature, such a structure might be exponentially large. Although this is not an issue in our definition, we note that all our concrete instantiations consider structures that have a polynomial representation.

We then apply our definitional framework to capture known AE-secure constructions, as well as (simulation-based) AE-MPC. Next, we describe our results in greater detail.

Almost-Everywhere RMT and SMT. We start in Section 3 by modeling the tasks of RMT and SMT (with a dedicated sender and receiver connected by a number of corruptible wires). As part of this, we show how these primitives, which have classically only been considered for an honest majority of wires, can be captured so that their security is defined independently of the number of corrupted wires. To apply a unified treatment, we cast the problem by modeling each wire with a (corruptible) dummy party called a "wire-party," which simply relays messages between S and R. In Section 3.1, we confirm that classical RMT/SMT protocols [20] are UC-secure (in the ordinary, non-AE sense) in our model against corrupted minorities of wire-parties. To handle corrupted majorities (and more generally to capture AE-security), in Section 3.2 we introduce an *AE wrapper functionality* that is parameterized by a doom structure as defined above. We can then state the AE-security of RMT/SMT protocols, by using a simple doom structure like the one that allows dooming S or R when a majority of the wire-parties are corrupted. We finish up in Section 3.3 with a universally composable treatment of the SMT-PD problem [25]. We model the public channel using access to the same functionality that we use to capture RMT security. Looking ahead, we will need SMT-PD when we want to elevate RMT to SMT over some classes of sparse graphs.

Almost-Everywhere *Remote* RMT and SMT. In Section 4, we consider the more complicated case where an incomplete graph connects several parties and yet all-to-all communication is desired. Interestingly, we show that the same wrapper from Section 3.2, which allowed for the simulation-based treatment of tasks like RMT and SMT even against corrupted majorities of wires, can also be used to model AE-security of the all-to-all versions of those tasks. In particular, in Section 4.1 we use the same ideal functionalities and wrapper (with more complex doom structures) from Section 3 to provide the first universally composable treatment of (AE) reliable communication over the sparse graphs constructed in [21, 43, 17, 32], which we refer to as AE *remote* RMT. In Section 4.2, we extend our treatment to AE *remote* SMT for all of these graphs. First, we show that a perfect SMT protocol from [20] can be adapted to realize perfectly secure AE-SMT over a class of sparse graphs constructed in [21]. In general, the same approach cannot be directly extended to achieve privacy for other graphs. To overcome this, we adapt an SMT-PD protocol from [25] to realize AE-SMT over the graphs in [43, 17, 32], at the cost of obtaining only statistical UC security.

Almost-Everywhere Secure Computation. Lastly, we study the composability of AE-security guarantees, with the ultimate goal of realizing AE-MPC. In Section 5.1, we prove a general composition theorem, which makes precise the level or "quality" of AE-security (as captured in a doom structure) that is obtained when a protocol's hybrids are replaced with AE counterparts. We emphasize that this *AE compiler* need not replace all of the hybrids with AE-wrapped versions using the *same* doom structure; thus, we are able to explain, for example, what happens when a protocol uses subprotocols that provide differing levels of AE-security. Our composition theorem applies even to protocols that already carry some level of AE-security, and therefore the compiled protocol can easily be composed with higher-level protocols. As a simple corollary, we show that a protocol achieving standard (non-AE) security using a single hybrid can be compiled into an AE-secure protocol while preserving the doom structure associated with the wrapped hybrid. In Section 5.2, we apply this corollary to obtain the first simulation-based proof of AE-MPC, over any of the classes of sparse graphs considered in the AE agreement literature [21, 43, 17, 32]. Depending on which class of sparse graphs is used, we obtain either perfect or statistical UC security.

Next, we review some preliminaries. Due to space limitations, some of the functionalities, protocols, and proofs are presented in the appendix or in the full version of the paper [16].

2 Preliminaries

2.1 UC Basics

We briefly summarize the UC framework [13] here. Protocol machines, ideal functionalities, the adversary, and the environment are all modeled as interactive Turing machine (ITM) instances, or ITIs. An execution of protocol π consists of a series of activations of ITIs, starting with the environment $\mathcal{Z}$ who provides inputs to and collects outputs from the parties and the adversary $\mathcal{A}$; parties can also give input to and collect output from sub-parties, and $\mathcal{A}$ can communicate with parties via messages. Corruption of parties is modeled by a special `corrupt` message sent from $\mathcal{A}$ to the party; upon receipt of this message, the party sends its entire local state to $\mathcal{A}$, and in all future activations follows the instructions of $\mathcal{A}$. Note that a party p_i can only be corrupted once $\mathcal{A}$ receives a special (`corrupt` p_i) input from $\mathcal{Z}$. Denote by $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ the probability distribution ensemble corresponding to the output of $\mathcal{Z}$ at the end of an execution of π with adversary $\mathcal{A}$. The ideal-world process for functionality $\mathcal{F}$ is simply defined as an execution of the ideal protocol $\text{IDEAL}_{\mathcal{F}}$, in which the so-called "dummy" parties just forward inputs from $\mathcal{Z}$ to $\mathcal{F}$ and forward outputs from $\mathcal{F}$ to $\mathcal{Z}$. The corresponding ensemble is denoted by $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

We are interested in unconditional security. Thus, we say that a protocol π *UC-realizes* an ideal functionality $\mathcal{F}$ if for any computationally unbounded adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ (polynomial in the complexity of $\mathcal{A}$) such that for any computationally unbounded environment $\mathcal{Z}$, we have $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \equiv \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. *Statistical* UC-realization requires only that the two ensembles be indistinguishable. When π is a $(\mathcal{G}_1,\ldots,\mathcal{G}_n)$-hybrid protocol (i.e., making subroutine calls to $\text{IDEAL}_{\mathcal{G}_1},\ldots,\text{IDEAL}_{\mathcal{G}_n}$), we say that π UC-realizes $\mathcal{F}$ in the $(\mathcal{G}_1,\ldots,\mathcal{G}_n)$-hybrid model. It turns out that (regular) UC-realization is equivalent to UC-realization with respect to the "dummy" adversary $\mathcal{D}$, which simply follows the instructions of $\mathcal{Z}$ on which messages to send, and reports all received messages to $\mathcal{Z}$.

We will assume synchronous computation (i.e., our protocols proceed in rounds), and the adversary is assumed to be rushing. Although our treatment is in the (G)UC setting, to avoid over-complicating the exposition we use the standard round-based language of, e.g., [12, 38]. Notwithstanding, such specifications can be translated to the synchronous UC model of Katz et al. [33] by assuming a clock functionality and bounded (zero) delay channels.

2.2 Building Blocks

Here we present some building blocks that we will use in our constructions, as well as (somewhat informal) property-based definitions to contrast with our UC formulations.

▶ **Definition 1** (SMT). *A protocol* Π *achieves (t-)*SMT *if it allows* S *to send a message* $m \in \mathcal{M}$ *to* R *such that the following hold for any adversary* $\mathcal{A}$ *corrupting up to* t *of the wires:*

- *Reliability: R correctly outputs $m' = m$.*
- *Secrecy: $\mathcal{A}$ learns no information about m.*

We define RMT by omitting the secrecy property, and AE-RMT and AE-SMT are defined by only requiring the properties to hold for privileged S and R (over some sparse graph).

For simplicity, we will use the 3-phase SMT protocol $\Pi_{\text{DDWY}}(\vec{\gamma},\tau,m)$ presented in Figure 3 (Appendix A), which is essentially the FastSMT protocol from [20]. The n wires are denoted by $\vec{\gamma} = (\gamma_1,\ldots,\gamma_n)$, and $\tau = \lceil \frac{n}{2} \rceil - 1$ specifies how many corrupted wires can be tolerated. Although the protocol assumes access to an authenticated channel between S and R, this can be implemented by simply sending the message over all wires and having R take majority.

We will sometimes need the SMT-PD protocol $\Pi_{\text{PUB-SMT}}(\vec{\gamma},Pub,m,l)$ presented in Figure 4 (Appendix A), which was given in [25] and tolerates $n-1$ wire corruptions, assuming access to a public channel Pub and allowing a small probability of error (determined by l).

Finally, we present the security definition for (property-based) AE-MPC that was given in [25]. Recall that W is the set of privileged nodes, as a function of the set of corruptions.

▶ **Definition 2** (AE-MPC, [25]). *An n-player two-phase protocol* Π *achieves* AE-MPC *if for any initial value x_i for party P_i for each $i \in [n]$, any probabilistic polynomial-time computable function f, and any adversary $\mathcal{A}$ corrupting a set T of parties, there exists a subset W of honest parties such that the following properties hold at the end of the respective phases:*

Commitment phase: *During this phase, all players commit to their inputs.*

- *Binding: For all P_i there is a uniquely defined value x_i^*; if $P_i \in W$, then $x_i^* = x_i$.*
- *Privacy: For all $P_i \in W$, x_i^* is information-theoretically hidden.*

Computation phase:

- *Correctness: All $P_i \in W$ output $f(x_1^*,\ldots,x_n^*)$.*
- *Privacy: For all $P_i \in W$, no information about x_i^* beyond what can be inferred from the output of the corrupted parties leaks to $\mathcal{A}$ by this phase.*

3 Almost-Everywhere RMT and SMT

In this section, we use the UC framework to capture classical RMT and SMT protocols, which work in a model where the sender S and receiver R are connected by n disjoint *wires*, as in the abstract formulation of [20]. Although this is a simple model, here we give a novel treatment of these tasks that also serves as a warm-up to our later results, which look at these tasks over sparse graphs. Since the classical protocols may not provide security when enough of the wires are corrupted, we also introduce an AE *wrapper* that allows parties interacting with the underlying functionality to be marked as "doomed" in such cases. In Section 4, where we consider *remote* RMT and SMT, we will realize the same functionalities for RMT and SMT defined in this section, just in a wrapped form with different parameters.

We begin by modeling the disjoint wires from the classical setting as virtual wires that are represented by UC parties, which we call *wire-parties* and denote by $W_1, \ldots, W_n$ ($\vec{W}$ for short). The idea is that a wire-party can securely forward a message from S to R or vice versa as long as it is not corrupted, just as a wire in the classical model can securely transmit a message between S and R as long as it is free of corruptions. Since the basic communication model in UC is completely unprotected, we assume access to the ideal secure channel functionality $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ (see the full version [16] for a formal specification), which provides secure communication between an honest S or R and an honest wire-party over a single round. Looking ahead, this functionality is very similar to the functionality we use to capture secure channels between every pair of nodes connected by an edge in a sparse graph.[3]

For convenience, we use $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ to realize the wire channel functionality $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ presented in Figure 5 (Appendix A), which abstracts the process of sending a message to a wire-party, who then forwards it to S or R. The functionality actually allows sending a potentially different message through each wire-party in parallel, and it provides security for a given message as long as S, R, and the wire-party in question are all honest. Since we are considering virtual wires that consist of just one intermediate node, the functionality requires two rounds to generate output. In $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ (and all of our functionalities), $l(\cdot)$ refers to length and INFL is short for "influence" (see, e.g., [24]).

We can use the protocol $\Pi_{\text{WC}}(S, R, \vec{W})$, which simply routes each message m_i from S to R (or R to S) via W_i using two instances of $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$, to realize $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$. The proof, as well as a formal specification of $\Pi_{\text{WC}}(S, R, \vec{W})$, can be found in the full version [16].

▶ **Proposition 3.** *Protocol $\Pi_{WC}(S, R, \vec{W})$ UC-realizes $\mathcal{F}_{WC}^{S,R,\vec{W}}$ in the $\mathcal{F}_{SC}^{S,R,\vec{W}}$-hybrid model.*

3.1 Universally Composable RMT and SMT

We model the task of RMT in UC with the authenticated channel functionality $\mathcal{F}_{\text{AUTH}}^{\mathcal{P},\mathsf{rnd}}$ (see the full version [16] for a formal specification), which is essentially Canetti's $\mathcal{F}_{\text{AUTH}}$ [14] with synchrony (the rnd parameter). There is also a parameter $\mathcal{P}$ representing the set of possible senders and receivers (the functionality itself is single-use). This parameter allows the functionality to verify that the actual sender and receiver can be identified as specific nodes in the network topology over which it is being realized, which is necessary because the realizing protocol will need to perform the same verification.

[3] Our RMT protocols only require reliable edges. However, we eventually need secure channels to achieve SMT and MPC, so for simplicity we present everything in the secure channels hybrid model.

To realize $\mathcal{F}_{\text{AUTH}}^{\mathcal{P},\mathsf{rnd}}$ in the wire-party model ($\mathcal{P} = \{S, R\}$) assuming only a minority of the wire-parties get corrupted, we can simply duplicate the message through all wire-parties using a single instance of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$, and have the receiver (who may actually be S) take majority. We formally define protocol $\Pi_{\text{AUTH}}(S, R, \vec{W})$ and give a proof in the full version [16].

▶ **Theorem 4.** *Protocol $\Pi_{AUTH}(S, R, \vec{W})$ UC-realizes $\mathcal{F}_{AUTH}^{\{S,R\},\mathsf{rnd}}$ for $\mathsf{rnd} = 2$ in the $\mathcal{F}_{WC}^{S,R,\vec{W}}$-hybrid model, against an adversary corrupting up to a minority of the wire-parties.*

Next, we capture SMT in UC with the secure channel functionality $\mathcal{F}_{\text{SMT}}^{\mathcal{P},\mathsf{rnd}}$ (see the full version [16] for a formal specification), which is essentially Canetti's $\mathcal{F}_{\text{SMT}}$ [14] with synchrony.

To realize $\mathcal{F}_{\text{SMT}}^{\mathcal{P},\mathsf{rnd}}$ in the wire-party model assuming only a minority of the wire-parties get corrupted, we can use protocol $\Pi_{\text{SMT}}(S, R, \vec{W})$ (outlined in the full version [16]), which is essentially the FastSMT protocol from [20] adapted for our UC treatment. That is, the sender (who may actually be R) runs protocol $\Pi_{\text{DDWY}}(\vec{\gamma}, \tau, m)$ (Figure 3) with the receiver, using $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ in phase 1 as a substitute for sending messages through the wires in $\vec{\gamma}$, and separate instances of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$ in phases 2 and 3 as a substitute for the authenticated channel.

▶ **Theorem 5.** *Protocol $\Pi_{SMT}(S, R, \vec{W})$ UC-realizes $\mathcal{F}_{SMT}^{\{S,R\},\mathsf{rnd}}$ for $\mathsf{rnd} = 6$ in the $(\mathcal{F}_{WC}^{S,R,\vec{W}}, \mathcal{F}_{AUTH}^{\{S,R\},2})$-hybrid model, against an adversary corrupting up to a minority of the wire-parties.*

3.2 Corrupted Majorities of Wire-Parties

In the wire-party model, $\mathcal{F}_{\text{AUTH}}$ and $\mathcal{F}_{\text{SMT}}$ can only be realized when the adversary is restricted to corrupting only a minority of wire-parties. When corrupted majorities are allowed, the sender and receiver may essentially become doomed. To allow the simulator to handle such cases, we introduce an *AE wrapper functionality* (Figure 1) that allows parties to be marked as doomed according to the current set of corruptions. The wrapper accepts "doom" requests according to an adversary structure, and it processes them by simply having the underlying functionality treat doomed parties as fully corrupted. Recall that an adversary structure is a set of c-vectors of subsets of a participant set $\mathcal{P}$, where each component of a vector represents corruptions of a certain type. We consider adversary structures that consist of *doubles* of subsets, corresponding to a corrupted set and a doomed set, respectively, although the two may intersect[4]. We call such structures *doom structures*.

In the wire-party model, we can realize *wrapped* $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ and $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ with doom structure $\mathscr{D}_{\text{PSMT}}$, defined as follows using participant set $\mathcal{P} = \{S, R, W_1, \ldots, W_n\}$:

- $(T_i, D_i) \in \mathscr{D}_{\text{PSMT}}$ if and only if either $|T_i - \{S, R\}| < \frac{n}{2}$ and $D_i = \emptyset$ or $|T_i - \{S, R\}| \geq \frac{n}{2}$ and $D_i \subseteq \{S, R\}$

▶ **Theorem 6.** *Protocol $\Pi_{AUTH}(S, R, \vec{W})$ UC-realizes $\mathcal{W}_{AE}^{\mathscr{D}_{PSMT}}(\mathcal{F}_{AUTH}^{\{S,R\},\mathsf{rnd}})$ for $\mathsf{rnd} = 2$ in the $\mathcal{F}_{WC}^{S,R,\vec{W}}$-hybrid model, even against corrupted majorities of wire-parties.*

To realize wrapped $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, define protocol $\Pi'_{\text{SMT}}(S, R, \vec{W})$ by replacing invocations of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$ in protocol $\Pi_{\text{SMT}}(S, R, \vec{W})$ with invocations of $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{AUTH}}^{\{S,R\},2})$.

▶ **Theorem 7.** *Protocol $\Pi'_{SMT}(S, R, \vec{W})$ UC-realizes $\mathcal{W}_{AE}^{\mathscr{D}_{PSMT}}(\mathcal{F}_{SMT}^{\{S,R\},\mathsf{rnd}})$ for $\mathsf{rnd} = 6$ in the $(\mathcal{F}_{WC}^{S,R,\vec{W}}, \mathcal{W}_{AE}^{\mathscr{D}_{PSMT}}(\mathcal{F}_{AUTH}^{\{S,R\},2}))$-hybrid model, even against corrupted majorities of wire-parties.*

Next we turn to SMT-PD, which offers an alternative way to achieve SMT against a corrupted majority of wires, in the presence of a public channel.

[4] This is a technicality, which simplifies some of our definitions and results. For example, the definition of AE-monotonicity (Section 5.1) would not be quite as short and intuitive otherwise.

Wrapper Functionality $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ The wrapper is parameterized by a doom structure $\mathscr{D} = \{(T_1, D_1), \ldots, (T_m, D_m)\}$, where each $(T_i, D_i) \in 2^{\mathcal{P}} \times 2^{\mathcal{P}}$. The underlying functionality is $\mathcal{F}$. Let T be the set of currently corrupted parties and let D be the set of currently doomed parties, both initialized to $\emptyset$.

- Upon receiving $(\text{CORRUPT}, sid, P_i)$ from the adversary for $P_i \in \mathcal{P}$: If $(T \cup \{P_i\}, D) \in \mathscr{D}$, then set $T \leftarrow T \cup \{P_i\}$, relay the message to $\mathcal{F}$, and send back $\mathcal{F}$'s response.
- Upon receiving (DOOM, sid, P_i) from the adversary for $P_i \in \mathcal{P}$: If $(T, D \cup \{P_i\}) \in \mathscr{D}$, then set $D \leftarrow D \cup \{P_i\}$, send $(\text{CORRUPT}, sid, P_i)$ to $\mathcal{F}$, and send back $\mathcal{F}$'s response.
- Any other request from any party or the adversary is simply relayed to $\mathcal{F}$ without any further action and the output is relayed to the destination specified by $\mathcal{F}$.

Figure 1 AE wrapper functionality.

3.3 Universally Composable SMT-PD

To capture SMT-PD in UC, we use our wire-party model from before, with the public channel modeled by assuming access to $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$ for some rnd'. Protocol $\Pi_{\text{SMT-PD}}(S, R, \vec{W}, l)$ (see the full version [16] for a formal specification) is essentially the Pub-SMT protocol from [25] adapted for our UC treatment. In particular, the sender transmits a message v to the receiver by essentially executing protocol $\Pi_{\text{PUB-SMT}}(\vec{\gamma}, Pub, v, l)$ (Figure 4), where $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ is used in the first phase as a substitute for sending messages through the wires in $\vec{\gamma}$, and separate instances of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$ are used in the remaining three phases as a substitute for Pub.

▶ **Theorem 8.** *Protocol* $\Pi_{SMT\text{-}PD}(S, R, \vec{W}, l)$ *statistically UC-realizes* $\mathcal{F}_{SMT}^{\{S,R\},\mathsf{rnd}}$ *for* $\mathsf{rnd} = 2 + 3 \cdot \mathsf{rnd}'$ *in the* $(\mathcal{F}_{WC}^{S,R,\vec{W}}, \mathcal{F}_{AUTH}^{\{S,R\},\mathsf{rnd}'})$*-hybrid model, against an adversary corrupting all but one of the wire-parties.*

4 Almost-Everywhere *Remote* RMT and SMT

In this section, we consider *remote* – i.e. over a sparse graph G_n – RMT and SMT. As in Section 3, we model the network topology using the parameterized secure channel functionality $\mathcal{F}_{\text{SC}}^{G_n}$ presented in Figure 6 (Appendix A), which provides secure channels only between parties that are connected in G_n. Instead of always working directly with $\mathcal{F}_{\text{SC}}^{G_n}$, we also use it to realize the remote secure channel functionality $\mathcal{F}_{\text{R-SC}}^{G_n}$ (see the full version [16] for a formal specification), the counterpart to $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ from Section 3. This functionality provides secure communication over a single path, as long as no node on the path is corrupted. Using protocol $\Pi_{\text{R-SC}}(G_n)$ (see the full version [16] for details), we can realize $\mathcal{F}_{\text{R-SC}}^{G_n}$ by simply forwarding the message along the path, which leads to the following statement (proof omitted).

▶ **Proposition 9.** *Protocol* $\Pi_{R\text{-}SC}(G_n)$ *UC-realizes* $\mathcal{F}_{R\text{-}SC}^{G_n}$ *in the* $\mathcal{F}_{SC}^{G_n}$*-hybrid model.*

4.1 AE Remote RMT

We first show how classical AE-RMT protocols from the AE agreement literature can be adapted to UC-realize our wrapped $\mathcal{F}_{\text{AUTH}}^{\mathcal{P},\mathsf{rnd}}$ functionality, using doom structures that are derived from the protocol and the underlying sparse graph.

Graphs of Constant Degree. We first describe a scheme due to Dwork et al. [21], which guarantees AE *reliable* communication in classes of constant-degree graphs (such as their "butterfly" network) that admit a certain three-phase transmission scheme. At a high-level,

the scheme associates with every node v a *fan-in* set $\Gamma_{\text{in}}(v)$ and a *fan-out* set $\Gamma_{\text{out}}(v)$ of a fixed size s. In addition, (not necessarily vertex-disjoint) paths from a node to its sets are specified, as well as (vertex-disjoint) paths from one node's fan-out set to another node's fan-in set. Node u transmits a message to node v by first sending it to all members of $\Gamma_{\text{out}}(u)$; each member then forwards the message to its connected (via a path) node in $\Gamma_{\text{in}}(v)$; and finally each member of $\Gamma_{\text{in}}(v)$ forwards the message to v, who simply takes majority.

Let $G_{\text{DPPU}} = (V_{\text{DPPU}}, E_{\text{DPPU}})$ be a graph that admits such a scheme. To realize wrapped $\mathcal{F}_{\text{AUTH}}^{V_{\text{DPPU}},\mathsf{rnd}}$, we use protocol $\Pi_{\text{R-AUTH}}^{\text{DPPU}}$ (outlined in the full version [16]), which is a straightforward adaptation of the scheme in the $\mathcal{F}_{\text{R-SC}}^{G_{\text{DPPU}}}$-hybrid model, and the doom structure $\mathscr{D}_{\text{DPPU}}$:

- For any corruption set T_i, let $D_{\text{DPPU}}(T_i)$ be a subset of participants P such that at least $\frac{1}{8}$ of the paths from P to $\Gamma_{\text{out}}(P)$ or at least $\frac{1}{8}$ of the paths from $\Gamma_{\text{in}}(P)$ to P are corrupted.
- Now, let $(T_i, D_i) \in \mathscr{D}_{\text{DPPU}}$ if and only if $|T_i| < s/4$ and $D_i \subseteq D_{\text{DPPU}}(T_i)$.

▶ **Theorem 10.** *Protocol $\Pi_{R\text{-}AUTH}^{DPPU}$ UC-realizes $\mathcal{W}_{AE}^{\mathscr{D}_{DPPU}}(\mathcal{F}_{AUTH}^{V_{DPPU},\mathsf{rnd}})$ for some $\mathsf{rnd} \in O(\log n)$ in the $\mathcal{F}_{R\text{-}SC}^{G_{DPPU}}$-hybrid model, against an adversary corrupting less than $s/4$ nodes.*

Upfal [43] proposed an alternative transmission scheme for constant-degree graphs, which actually works over *any* graph; however, his optimal result is achieved only on constant-degree expander graphs with specific parameters. The main limitation of the scheme is that it is computationally expensive. Node u transmits a message to node v by sending it through all the simple paths connecting them. As the message travels along a path to v, each node on the path appends the ID of the previous node to the message. This way each message received from a corrupted path will contain at least one ID of a corrupted node, and the receiver can enumerate over all the possible corruption sets to recover the message.

Let $G_n^{\text{UPFAL}} = (V_{\text{UPFAL}}, E_{\text{UPFAL}})$ be a d-regular expander graph. To realize wrapped $\mathcal{F}_{\text{AUTH}}^{V_{\text{UPFAL}},\mathsf{rnd}}$, we use protocol $\Pi_{\text{R-AUTH}}^{\text{UPFAL}}$ (outlined in the full version [16]), which is a straightforward adaptation of Upfal's scheme in the $\mathcal{F}_{\text{SC}}^{G_n^{\text{UPFAL}}}$-hybrid model, and the following doom structure $\mathscr{D}_{\text{UPFAL}}$:

- First, define $D_{\text{UPFAL}}(T_i)$ by the following iterative process: Starting with the set $S = T_i$, repeatedly add all participants $Q \notin S$ such that at least $\frac{1}{5}$ of Q's neighbors are in S.
- Now, let $(T_i, D_i) \in \mathscr{D}_{\text{UPFAL}}$ if and only if $|T_i| < t < 1/72n$ and $D_i \subseteq D_{\text{UPFAL}}(T_i)$.

▶ **Theorem 11.** *Protocol $\Pi_{R\text{-}AUTH}^{UPFAL}$ UC-realizes $\mathcal{W}_{AE}^{\mathscr{D}_{UPFAL}}(\mathcal{F}_{AUTH}^{V_{UPFAL},\mathsf{rnd}})$ for some $\mathsf{rnd} \in O(\log n)$ in the $\mathcal{F}_{SC}^{G_n^{UPFAL}}$-hybrid model, against an adversary corrupting less than $1/72n$ nodes.*

Although the simulator we construct is inefficient, that seems reasonable since the protocol itself runs in exponential time.

Graphs of Poly-Logarithmic Degree. Chandran et al. [17] proposed an AE-RMT scheme over a randomly constructed graph of poly-logarithmic degree. A very high-level idea of their construction is to transmit a message via multiple paths, while also performing some sort of error correction along the way. Their graph is comprised of some randomly generated, overlapping, fully connected committees that are themselves connected via the butterfly network. They also assign and connect to each node a number of those committees as *helpers*. Node u transmits a message to node v by sending it to all of u's helper committees, who then transmit it to v's helper committees using the transmission scheme of Dwork et al. [21] at the committee level. Finally, v takes majority over the values received from its helpers. As the message travels from one committee to another, error correction occurs using a *differential agreement* protocol [22] run by the nodes in the destination committee.

Let $G_n^{\text{CGO}} = (V_{\text{CGO}}, E_{\text{CGO}})$ be a graph constructed as above. To realize wrapped $\mathcal{F}_{\text{AUTH}}^{V_{\text{CGO}},\mathsf{rnd}}$, we use protocol $\Pi_{\text{R-AUTH}}^{\text{CGO}}$ (outlined in the full version [16]), which is a straightforward adaptation of the above scheme in the $\mathcal{F}_{\text{SC}}^{G_n^{\text{CGO}}}$-hybrid model, and the following doom structure $\mathscr{D}_{\text{CGO}}$:

- First, for any set of corruptions T_i, let $D_{\text{CGO}}(T_i)$ be the set of all participants P such that $P \in T_i$ or at least $\frac{1}{6}$ of P's helper committees are unprivileged. A committee is considered corrupted if more than $\frac{1}{4}$ of its members are corrupted, and committees are categorized as unprivileged according to the $D_{\text{DPPU}}(\cdot)$ function defined above.
- Now, let $(T_i, D_i) \in \mathscr{D}_{\text{CGO}}$ if and only if corrupting the nodes in T_i causes at most $\frac{n \log^k n}{4 \log(n \log^k n)}$ committees to become corrupted, and $D_i \subseteq D_{\text{CGO}}(T_i)$.

Chandran et al. [17] proved that there exist constants $\alpha_{\text{CGO}}, \beta_{\text{CGO}}$ such that for any T_i with $|T_i| < \alpha_{\text{CGO}} n$, it holds that $|D_{\text{CGO}}(T_i)| < \beta_{\text{CGO}} \frac{|T_i|}{\log n}$. For those constants we have:

▶ **Theorem 12.** *Protocol* $\Pi^{CGO}_{\text{R-AUTH}}$ *UC-realizes* $\mathcal{W}^{\mathscr{D}_{CGO}}_{AE}(\mathcal{F}^{V_{CGO},\mathsf{rnd}}_{AUTH})$ *for* $\mathsf{rnd} \in O(\log n \cdot \log\log n)$ *in the* $\mathcal{F}_{SC}^{G_n^{CGO}}$ *-hybrid model, against an adversary corrupting less than* $\alpha_{CGO} n$ *nodes.*

Graphs of Logarithmic Degree. Jayanti et al. [32] recently proposed a transmission scheme over logarithmic-degree graphs. Their graphs consist of multiple layers that are all constructed using the same method but over randomly permuted sets of nodes. In each layer, nodes are partitioned into committees of size s that are connected via the butterfly network and have Upfal's [43] expander graph instantiated inside them. We call this family of graphs $\mathcal{G}_{\text{JRV}}$. To transmit a message from node u to node v, in each layer u sends the message to all its committee members using Upfal's transmission scheme, and then the committee transmits it to v's committee using the transmission scheme of Dwork et al. [21] at the committee level. Finally, all of v's committee members send the value to v so that it can take majority over what is received from all the layers combined. There is also some type of error correction when messages travel from one committee to another.

Let $G_n^{\text{JRV}} = (V_{\text{JRV}}, E_{\text{JRV}}) \in \mathcal{G}_{\text{JRV}}$ with $|V_{\text{JRV}}| = n$. To realize wrapped $\mathcal{F}_{\text{AUTH}}^{V_{\text{JRV}},\mathsf{rnd}}$, we use protocol $\Pi^{\text{JRV}}_{\text{R-AUTH}}$ (outlined in the full version [16]), which is a straightforward adaptation of the above scheme in the $\mathcal{F}_{\text{SC}}^{G_n^{\text{JRV}}}$-hybrid model, and the following doom structure $\mathscr{D}_{\text{JRV}}$:

- First, in each layer of G_n^{JRV}, if a committee contains more than $\frac{1}{72}s$ corruptions, call it *bad*. If the total number of bad committees in a layer exceeds $\frac{n/s}{4\log(n/s)}$, call the layer *bad*.
- Next, for any set of corruptions T_i, let $D_{\text{JRV}}(T_i)$ be the set of all participants P such that $P \in T_i$ or P is doomed in more than $\frac{1}{10}z$ layers among all the good layers. A node is considered doomed in a layer if it is located in a doomed committee (with respect to $D_{\text{DPPU}}(\cdot)$) or is doomed itself within its committee (with respect to $D_{\text{UPFAL}}(\cdot)$).
- Now, let $(T_i, D_i) \in \mathscr{D}_{\text{JRV}}$ if and only if corrupting the nodes in T_i causes at most $\frac{1}{5}$ of the layers to become bad, and $D_i \subseteq D_{\text{JRV}}(T_i)$.

Jayanti et al. [32] proved there exists a graph $G_n^{\text{JRV}} \in \mathcal{G}_{\text{JRV}}$ and constants $\alpha_{\text{JRV}}, \beta_{\text{JRV}}$ such that for T_i with $|T_i| < \alpha_{\text{JRV}} n$, it holds that $|D_{\text{JRV}}(T_i)| < \beta_{\text{JRV}} \frac{|T_i|}{\log n}$. For such a graph we have:

▶ **Theorem 13.** *Protocol* $\Pi^{JRV}_{\text{R-AUTH}}$ *UC-realizes* $\mathcal{W}^{\mathscr{D}_{\text{JRV}}}_{AE}(\mathcal{F}^{V_{\text{JRV}},\mathsf{rnd}}_{AUTH})$ *for some* $\mathsf{rnd} \in O(\log n \cdot \log\log\log n)$ *in the* $\mathcal{F}_{SC}^{G_n^{JRV}}$ *-hybrid model, against adversaries corrupting less than* $\alpha_{\text{JRV}} n$ *nodes.*

4.2 AE Remote SMT

To achieve AE *secure* communication over the constant-degree graphs studied by Dwork et al. [21], we can apply the approach that we used in Section 3.2 to obtain AE-SMT in the wire-party model. That is, we can adapt protocol $\Pi'_{\text{SMT}}(S, R, \vec{W})$ to work over the three-phase paths in G_{DPPU}, and the resulting protocol $\Pi^{\text{DPPU}}_{\text{R-SMT}}$ (formally outlined in the full version [16]) realizes wrapped $\mathcal{F}_{\text{SMT}}^{V_{\text{DPPU}},\mathsf{rnd}'}$ for some rnd' with the same doom structure $\mathscr{D}_{\text{DPPU}}$ from Section 4.1. Let ℓ denote the maximum length of any of the three-phase paths.

▶ **Theorem 14.** *Protocol* $\Pi^{DPPU}_{R\text{-}SMT}$ *UC-realizes* $\mathcal{W}^{\mathscr{D}_{DPPU}}_{AE}(\mathcal{F}^{V_{DPPU},2\cdot\mathsf{rnd}+\ell}_{SMT})$ *in the* $(\mathcal{W}^{\mathscr{D}_{DPPU}}_{AE}(\mathcal{F}^{V_{DPPU},\mathsf{rnd}}_{AUTH}), \mathcal{F}^{G_{DPPU}}_{R\text{-}SC})$*-hybrid model for* $\mathsf{rnd} \in O(\log n)$*, against an adversary corrupting less than* $s/4$ *nodes.*

The above technique cannot in general be extended to other AE-RMT schemes, because it requires a majority of honest paths between any pair of privileged nodes to realize a secure link between them. Many transmission schemes, such as Upfal's [43], do not guarantee such a property for privileged nodes. To realize AE-SMT using other transmission schemes, one approach is to use SMT-PD, which only requires a single honest path between sender and receiver to establish a secure channel, assuming access to a public channel. This approach can be used to make any AE-RMT scheme secure, since these schemes realize an authenticated channel (between privileged nodes) and guarantee at least one honest path between any pair of privileged nodes. The downside is that only statistical security is obtained.

We first introduce some notation. Given a doom structure $\mathscr{D}$ (with participant set $\mathcal{P}$), denote by $\mathsf{dom}(\mathscr{D})$ the set of values that appear as a first component in $\mathscr{D}$ (in other words, the set of all corruption sets allowed by $\mathscr{D}$). Say that $\mathscr{D}$ is *t-complete* if $\max_{(T_i,D_i)\in\mathscr{D}} |T_i| = t$, and $T \in \mathsf{dom}(\mathscr{D})$ for all $T \subseteq \mathcal{P}$ with $|T| \leq t$ (in other words, if all possible sets of corruptions of size at most t are allowed by $\mathscr{D}$). Moreover, say that a doom structure $\mathscr{D}$ is *D-monotone* if whenever $(T_j, D_j) \in \mathscr{D}$ and $D_i \subseteq D_j$, it holds that $(T_j, D_i) \in \mathscr{D}$. We note that all of our doom structures are t-complete and D-monotone.

Now, let $G_n = (V, E)$ be a graph with polynomially many paths of length at most ℓ specified between every pair of nodes. Suppose we already know how to realize wrapped $\mathcal{F}^{V,\mathsf{rnd}}_{\text{AUTH}}$ for some rnd, with respect to a doom structure $\mathscr{D}_{\text{SMT-PD}}$ (with $\mathcal{P} = V$) that is t-complete and D-monotone and moreover satisfies the following condition: For all $T \subseteq V$ with $|T| \leq t$, at least one of the specified paths between any pair of nodes in $V - T - \cup_{(T,D_i)\in\mathscr{D}_{\text{SMT-PD}}} D_i$ is completely contained in $V - T$. Then, we can realize wrapped $\mathcal{F}^{V,\mathsf{rnd}'}_{\text{SMT}}$ using protocol $\Pi_{\text{R-SMT-PD}}(G_n)$ (formally outlined in the full version [16]), which is essentially protocol $\Pi_{\text{SMT-PD}}(S, R, \vec{W}, l)$ from Section 3.3 adapted to work over the specified paths in G_n, and the same doom structure $\mathscr{D}_{\text{SMT-PD}}$. For such a doom structure we obtain the following statement:

▶ **Theorem 15.** *Protocol* $\Pi_{R\text{-}SMT\text{-}PD}(G_n)$ *statistically UC-realizes* $\mathcal{W}^{\mathscr{D}_{SMT\text{-}PD}}_{AE}(\mathcal{F}^{V,\ell+3\cdot\mathsf{rnd}}_{SMT})$ *in the* $(\mathcal{F}^{G_n}_{R\text{-}SC}, \mathcal{W}^{\mathscr{D}_{SMT\text{-}PD}}_{AE}(\mathcal{F}^{V,\mathsf{rnd}}_{AUTH}))$*-hybrid model against a* t*-adversary.*

Observe that t-completeness allows for statements against threshold adversaries, and D-monotonicity is required in the simulation since the simulator can only doom parties one by one. According to [21], all the realizable doom structures for AE remote RMT satisfy the above condition. Therefore, protocol $\Pi_{\text{R-SMT-PD}}(G_n)$ can be used with any of the classes of sparse graphs discussed in Section 4.1 to achieve AE remote SMT with statistical security.

5 Almost-Everywhere Secure Computation

In this section, we consider general UC-secure computation in the almost-everywhere setting. We start by proving a composition theorem that shows how to compile a protocol Π realizing some functionality $\mathcal{F}$ with the help of several hybrids into an *almost-everywhere* version of Π, by wrapping each hybrid with a potentially different doom structure $\mathscr{D}_i$. These structures can be arbitrary, subject only to a certain monotonicity property, although they must correspond to the same participant set (indeed, composition would not make much sense otherwise); the compiled protocol is then shown to realize a *wrapped* version of $\mathcal{F}$, using a new doom structure $\mathscr{D}'$. Moreover, we allow the original protocol Π to itself realize a wrapped functionality associated with some doom structure $\mathscr{D}$. This, along with the fact that the monotonicity property carries over to the new doom structure $\mathscr{D}'$, make the compiled protocol readily

Compiler $\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi)$ Apply the following modifications to protocol Π (which uses $\mathcal{F}_1,\ldots,\mathcal{F}_m$ as hybrids):

1. For each $i \in [m]$, instead of using $\mathcal{F}_i$, parties use $\mathcal{W}_{\text{AE}}^{\mathscr{D}_i}(\mathcal{F}_i)$ (which has the same input/output format to the parties).

Figure 2 The AE compiler.

amenable to further composition. We conclude by applying a special case of the composition theorem to obtain AE-MPC over the sparse graphs that were considered in Section 4. Rather than constructing protocols from scratch, we simply apply our generic AE compiler to replace the secure channels that are used in standard MPC protocols with AE remote SMT.

5.1 A General Composition Theorem

Let us first introduce some notation. Say that a doom structure $\mathscr{D}$ is *AE-monotone* if whenever $(T_i, D_i) \in \mathscr{D}$ and $T_i \subseteq T_j$ for $T_j \in \mathsf{dom}(\mathscr{D})$, it holds that $(T_j, D_i) \in \mathscr{D}$. Different from the standard notion of monotonicity in the general adversary literature, AE-monotonicity captures the intuitive property that when additional parties are corrupted, parties that were previously doomed are still doomed (or newly corrupted). AE-monotonicity seems to be important for simulatability; for example, the simulator may want to make a doom request for a newly doomed party only after some additional parties are corrupted in the meantime, and in such a case the doom structure needs to admit that request. Fortunately, all of our doom structures are AE-monotone.

The AE compiler is shown in Figure 2. It takes as input a protocol Π realizing some wrapped functionality $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ in the $(\mathcal{F}_1,\ldots,\mathcal{F}_m)$-hybrid model and turns it into a protocol that works in the $(\mathcal{W}_{\text{AE}}^{\mathscr{D}_1}(\mathcal{F}_1),\ldots,\mathcal{W}_{\text{AE}}^{\mathscr{D}_m}(\mathcal{F}_m))$-hybrid model. Of course, the compiled protocol will not in general realize wrapped $\mathcal{F}$ with the same doom structure $\mathscr{D}$. In the following theorem, we construct a new doom structure $\mathscr{D}'$ representing the level of AE-security that is retained. Since we consider general adversaries, the compiled protocol can tolerate a set T' of corruptions only if T' can be tolerated by *all* of the assumed doom structures (i.e., $\mathscr{D}$ as well as $\mathscr{D}_1,\ldots,\mathscr{D}_m$). Furthermore, the set of parties in the compiled protocol that are considered doomed (relative to T') can consist of, roughly speaking, parties that are doomed with respect to *any* of the wrapped hybrids (such parties are collected in $D(T')$ below) or that would have been doomed in the original protocol Π (the parties denoted by A). In fact, since Π may already carry some level of AE-security, as captured by $\mathscr{D}$, we must expand the latter set to include parties that only become doomed when some or all of the parties in the former set are actually corrupted. Thus, we require that $T' \cup D(T')$ is also tolerated by $\mathscr{D}$.

▶ **Theorem 16.** *Let $\mathscr{D}, \mathscr{D}_1, \ldots, \mathscr{D}_m$ be AE-monotone doom structures over the same participant set $\mathcal{P}$. Let $\mathcal{T} = \mathsf{dom}(\mathscr{D})$ and $\mathcal{T}' = (\bigcap_{i=1}^m \mathsf{dom}(\mathscr{D}_i)) \cap \mathcal{T}$. For any $T' \in \mathcal{T}'$, define*

$$D(T') = \bigcup_{i=1}^{m} \left(\bigcup_{(T', D_j) \in \mathscr{D}_i} D_j \right).$$

Suppose that for all $T' \in \mathcal{T}'$, it holds that $T' \cup D(T') \in \mathcal{T}$. If protocol Π UC-realizes $\mathcal{W}_{AE}^{\mathscr{D}}(\mathcal{F})$ in the $(\mathcal{F}_1,\ldots,\mathcal{F}_m)$-hybrid model against a $\mathcal{T}$-adversary, then $\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi)$ UC-realizes $\mathcal{W}_{AE}^{\mathscr{D}'}(\mathcal{F})$ in the $(\mathcal{W}_{AE}^{\mathscr{D}_1}(\mathcal{F}_1),\ldots,\mathcal{W}_{AE}^{\mathscr{D}_m}(\mathcal{F}_m))$-hybrid model against a $\mathcal{T}'$-adversary, where $\mathscr{D}'$ is defined as follows: For all $T' \in \mathcal{T}'$, we have $(T', D \cup A) \in \mathscr{D}'$ if $D \subseteq D(T')$ and $(T' \cup D(T'), A) \in \mathscr{D}$. Moreover, $\mathscr{D}'$ is AE-monotone.

In the specific case that Π realizes an *unwrapped* functionality $\mathcal{F}$ (indeed, one can always apply our AE wrapper to $\mathcal{F}$ with a doom structure of the form $\{(T_i, \emptyset)\}_i$, which is trivially AE-monotone, in order to obtain an equivalent functionality) in the $\mathcal{G}$-hybrid model against a threshold adversary, we obtain the following corollary:

▶ **Corollary 17.** *Let $\mathscr{D}$ be a t'-complete, D-monotone, and AE-monotone doom structure. Let $t = \max_{|T'|=t'} \left| \left(\bigcup_{(T', D_i) \in \mathscr{D}} D_i \right) \cup T' \right|$. If protocol Π UC-realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model against a t-adversary, then $\mathcal{C}^{\mathscr{D}}(\Pi)$ UC-realizes $\mathcal{W}_{AE}^{\mathscr{D}}(\mathcal{F})$ in the $\mathcal{W}_{AE}^{\mathscr{D}}(\mathcal{G})$-hybrid model against a t'-adversary.*

Observe that t'-completeness allows the simulator to handle a threshold adversary that can corrupt *any* t' parties, and D-monotonicity is needed for the doom structure $\mathscr{D}$ used to wrap $\mathcal{G}$ to be preserved when wrapping $\mathcal{F}$. By construction, all of our doom structures satisfy these two properties. We remark that t has a natural interpretation: the maximum number of parties that can become unprivileged (with respect to $\mathscr{D}$) when t' parties are corrupted.

5.2 AE-MPC

We now present our main result: how to achieve almost-everywhere MPC over several classes of sparse graphs in a composable manner. We assume a protocol that achieves "regular" MPC over a complete network of point-to-point secure channels, and show how to transform it into a protocol that achieves AE-MPC (with a lower corruption threshold) over a sparse graph with secure channels only between connected parties, using our AE compiler. To capture the MPC task for n-ary function f, we use the functionality $\mathcal{F}_{\text{MPC}}^{f,\mathcal{P},\mathsf{rnd}}$ (see the full version [16] for a formal specification), which is essentially Canetti's $\mathcal{F}_{\text{SFE}}$ [14] with synchrony.

Although standard information-theoretic MPC protocols tolerating $t < \frac{n}{3}$ corruptions are known [8, 18], they assume access to a broadcast channel, noting that broadcast can be achieved when $t < \frac{n}{3}$. However, [30] showed that classical broadcast protocols are not adaptively secure in a simulation-based setting, and gave a VSS-based protocol that does in fact realize adaptively secure broadcast with perfect security for $t < \frac{n}{3}$, assuming only secure channels. Therefore, there exists a protocol that UC-realizes $\mathcal{F}_{\text{MPC}}^{f,\mathcal{P},\mathsf{rnd}}$ for any n-ary function f and some rnd in the $\mathcal{F}_{\text{SMT}}^{\mathcal{P},1}$-hybrid model, against an adversary corrupting less than $\frac{n}{3}$ parties. It is clear that this holds even in the $\mathcal{F}_{\text{SMT}}^{\mathcal{P},\ell}$-hybrid model, for arbitrary ℓ. Now, by invoking Corollary 17 (which of course also offers statistical security) and then applying the (regular) UC composition theorem in tandem with our results in Theorems 14 and 15 showing how to achieve AE-SMT over several classes of sparse graphs with either perfect or statistical security, we obtain the following corollaries showing how to achieve AE-MPC over those classes of graphs, with different combinations of parameters (recall that the maximum number of doomed nodes is encoded into each doom structure), for any n-ary function f:

▶ **Corollary 18.** *There exists a protocol that UC-realizes $\mathcal{W}_{AE}^{\mathscr{D}_{DPPU}}(\mathcal{F}_{MPC}^{f,V_{DPPU},\mathsf{rnd}})$ in the $\mathcal{F}_{SC}^{G_n^{DPPU}}$-hybrid model against a t-adversary, for some rnd and $t \in O(\frac{n}{\log n})$.*

▶ **Corollary 19.** *Let $\mathtt{x} \in \{UPFAL, CGO, JRV\}$. There exists a protocol statistically UC-realizing $\mathcal{W}_{AE}^{\mathscr{D}_{\mathtt{x}}}(\mathcal{F}_{MPC}^{f,V_{\mathtt{x}},\mathsf{rnd}})$ in the $\mathcal{F}_{SC}^{G_n^{\mathtt{x}}}$-hybrid model against a t-adversary, for some rnd and $t \in O(n)$.*

References

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

2 Bar Alon, Eran Omri, and Anat Paskin-Cherniavsky. MPC with friends and foes. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 677–706. Springer, 2020. doi:10.1007/978-3-030-56880-1_24.

3 Michael Backes, Birgit Pfitzmann, and Michael Waidner. A composable cryptographic library with nested operations. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washington, DC, USA, October 27-30, 2003*, pages 220–230. ACM, 2003. doi:10.1145/948109.948140.

4 Christian Badertscher, Ran Canetti, Julia Hesse, Björn Tackmann, and Vassilis Zikas. Universal composition with global subroutines: Capturing global setup within plain UC. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 1–30. Springer, 2020. doi:10.1007/978-3-030-64381-2_1.

5 Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. 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 324–356. Springer, 2017. doi:10.1007/978-3-319-63688-7_11.

6 Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. *J. Cryptol.*, 24(4):720–760, 2011. doi:10.1007/s00145-010-9075-9.

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, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 231–250. Springer, 2008. doi:10.1007/978-3-540-78524-8_14.

8 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.

9 Elette Boyle, Ran Cohen, Deepesh Data, and Pavel Hubácek. Must the communication graph of MPC protocols be an expander? 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 III*, volume 10993 of *Lecture Notes in Computer Science*, pages 243–272. Springer, 2018. doi:10.1007/978-3-319-96878-0_9.

10 Elette Boyle, Ran Cohen, and Aarushi Goel. Breaking the o($\sqrt{}$ n)-bit barrier: Byzantine agreement with polylog bits per party. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, pages 319–330. ACM, 2021. doi:10.1145/3465084.3467897.

11 Jan Camenisch, Stephan Krenn, Ralf Küsters, and Daniel Rausch. iuc: Flexible universal composability made simple. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 191–221. Springer, 2019. doi:10.1007/978-3-030-34618-8_7.

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

13 Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145. IEEE Computer Society, 2001. doi:10.1109/SFCS.2001.959888.

14 Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, December 2005. Latest version at https://ia.cr/2000/067.

15 Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. 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 61–85. Springer, 2007. doi:10.1007/978-3-540-70936-7_4.

16 Nishanth Chandran, Pouyan Forghani, Juan Garay, Rafail Ostrovsky, Rutvik Patel, and Vassilis Zikas. Universally composable almost-everywhere secure computation. Cryptology ePrint Archive, Report 2021/1398, 2021. URL: https://ia.cr/2021/1398.

17 Nishanth Chandran, Juan A. Garay, and Rafail Ostrovsky. Improved fault tolerance and secure computation on sparse networks. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 249–260. Springer, 2010. doi:10.1007/978-3-642-14162-1_21.

18 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (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 11–19. ACM, 1988. doi:10.1145/62212.62214.

19 Danny Dolev. Unanimity in an unknown and unreliable environment. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 159–168. IEEE Computer Society, 1981. doi:10.1109/SFCS.1981.53.

20 Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 36–45. IEEE Computer Society, 1990. doi:10.1109/FSCS.1990.89522.

21 Cynthia Dwork, David Peleg, Nicholas Pippenger, and Eli Upfal. Fault tolerance in networks of bounded degree (preliminary version). In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 370–379. ACM, 1986. doi:10.1145/12130.12169.

22 Matthias Fitzi and Juan A. Garay. Efficient player-optimal protocols for strong and differential consensus. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003, Boston, Massachusetts, USA, July 13-16, 2003*, pages 211–220. ACM, 2003. doi:10.1145/872035.872066.

23 Matthias Fitzi, Martin Hirt, and Ueli M. Maurer. Trading correctness for privacy in unconditional multi-party computation (extended abstract). In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 121–136. Springer, 1998. doi:10.1007/BFb0055724.

24 Juan A. Garay, Aggelos Kiayias, and Hong-Sheng Zhou. A framework for the sound specification of cryptographic tasks. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010, Edinburgh, United Kingdom, July 17-19, 2010*, pages 277–289. IEEE Computer Society, 2010. doi:10.1109/CSF.2010.26.

25 Juan A. Garay and Rafail Ostrovsky. Almost-everywhere secure computation. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 307–323. Springer, 2008. doi:10.1007/978-3-540-78967-3_18.

26 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, November 2-4, 1992, Proceedings*, volume 647 of *Lecture Notes in Computer Science*, pages 153–165. Springer, 1992. doi:10.1007/3-540-56188-9_11.

27 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred 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.

28 Jacopo Griggio. *Perfectly secure message transmission protocols with low communication overhead and their generalization*. PhD thesis, Universiteit Leiden, 2012.

29 Martin Hirt and Ueli M. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In James E. Burns and Hagit Attiya, editors, *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, Santa Barbara, California, USA, August 21-24, 1997*, pages 25–34. ACM, 1997. doi:10.1145/259380.259412.

30 Martin Hirt and Vassilis Zikas. Adaptively secure broadcast. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 466–485. Springer, 2010. doi:10.1007/978-3-642-13190-5_24.

31 Dennis Hofheinz and Victor Shoup. GNUC: A new universal composability framework. *J. Cryptol.*, 28(3):423–508, 2015. doi:10.1007/s00145-013-9160-y.

32 Siddhartha Jayanti, Srinivasan Raghuraman, and Nikhil Vyas. Efficient constructions for almost-everywhere secure computation. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 159–183. Springer, 2020. doi:10.1007/978-3-030-45724-2_6.

33 Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In Amit Sahai, editor, *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, volume 7785 of *Lecture Notes in Computer Science*, pages 477–498. Springer, 2013. doi:10.1007/978-3-642-36594-2_27.

34 Valerie King and Jared Saia. From almost everywhere to everywhere: Byzantine agreement with $\tilde{o}(n^{3/2})$ bits. In Idit Keidar, editor, *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*, volume 5805 of *Lecture Notes in Computer Science*, pages 464–478. Springer, 2009. doi:10.1007/978-3-642-04355-0_47.

35 Kaoru Kurosawa and Kazuhiro Suzuki. Truly efficient 2-round perfectly secure message transmission scheme. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 324–340. Springer, 2008. doi:10.1007/978-3-540-78967-3_19.

36 Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982. doi:10.1145/357172.357176.

37 Ueli Maurer and Renato Renner. Abstract cryptography. In Bernard Chazelle, editor, *Innovations in Computer Science - ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 1–21. Tsinghua University Press, 2011. URL: http://conference.iiis.tsinghua.edu.cn/ICS2011/content/papers/14.html.

38 Jesper Buus Nielsen. *On Protocol Security in the Cryptographic Model*. PhD thesis, University of Aarhus, 2003.

39 Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980. doi:10.1145/322186.322188.

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

41 Gabriele Spini and Gilles Zémor. Perfectly secure message transmission in two rounds. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 286–304, 2016. doi:10.1007/978-3-662-53641-4_12.

42 K. Srinathan, Arvind Narayanan, and C. Pandu Rangan. Optimal perfectly secure message transmission. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 545–561. Springer, 2004. doi:10.1007/978-3-540-28628-8_33.

43 Eli Upfal. Tolerating linear number of faults in networks of bounded degree. In Norman C. Hutchinson, editor, *Proceedings of the Eleventh Annual ACM Symposium on Principles of Distributed Computing, Vancouver, British Columbia, Canada, August 10-12, 1992*, pages 83–89. ACM, 1992. doi:10.1145/135419.135437.

44 Andrew Chi-Chih Yao. Space-time tradeoff for answering range queries (extended abstract). In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 128–136. ACM, 1982. doi:10.1145/800070.802185.

A Functionalities and Protocols

Protocol $\Pi_{\text{DDWY}}(\vec{\gamma}, \tau, m)$

1. **(Phase 1)** The sender S sends $n\tau + 1$ strong pads SP_1,SP_2, ..., $SP_{n\tau+1}$. To send each strong pad, S chooses a random polynomial $f(x) \in \mathbb{Z}_q(x)$ of degree τ and sets $pad = f(0)$. Then for each $i \in [n]$ S chooses an additional random polynomial $h_i(x) \in \mathbb{Z}_q$ of degree τ such that $h_i(0) = f(i)$. Finally, for each $i \in [n]$, S sends $h_i(\cdot)$ with a vector of checking pieces $C_i = (c_{1i}, c_{2i}, \ldots, c_{ni})$ to R using wire γ_i where for all $i, j \in [n]$, $c_{ji} = h_j(i)$.
2. **(Phase 2)** For each $k \in [n]$, let T_k be received in the attempted transmission of SP_k and g_i, D_i be possibly corrupted information received as h_i, C_i. If for any T_a all the checking pieces c_{ji} and all polynomials $h_i(\cdot)$ are consistent then R interpolates the pad_a from T_a and sends "a, OK" to S over the authenticated channel. Otherwise, R finds an l such that $\{\text{conflicts of } T_l\} \subseteq \cup_{m \neq l}\{\text{conflicts of } T_m\}$, where any unordered pair (i, j) is called a conflict of T_k if $d_{ji} \neq g_j(i)$. Then R sends l and all T_m, $m \neq l$ back to S using authenticated channel.
3. **(Phase 3)**
   - If "a, OK" received over the authenticated channel in phase 2, then S sends $z = m + pad_a$ to R using the authenticated channel. Otherwise, S preforms error detection on all T_j's received from R and sends detected faults and $z = m + pad_i$ to R using authenticated channel.
   - If R previously sent "a, OK" to S in phase 1, then s/he computes $m = z - pad_a$. Otherwise, R corrects the faults in T_i, obtains pad_i and computes $m = z - pad_i$.

Figure 3 The SMT protocol from [20].

Protocol $\Pi_{\text{PUB-SMT}}(\vec{\gamma}, Pub, m, l)$

1. The sender S sends n uniformly random bit strings $R_1, R_2, \ldots, R_n$ of length $15l$ to the receiver R through wires $\gamma_1, \gamma_2, \ldots, \gamma_n$, respectively. Let $R'_1, R'_2, \ldots, R'_n$ be the strings received by R. R rejects all wires where $|R'_i| \neq 15l$.
2. For $i \in [n]$, S generates R^*_i by replacing $12l$ randomly chosen positions of R_i with "$*$." Then S sends $R^*_1, R^*_2, \ldots, R^*_n$ to R over Pub.
3. For any $i \in [n]$, if R^*_i and R'_i differ in any "opened" bits, R marks γ_i as "faulty." Then R sends an n-bit string to S over Pub that identifies faulty wires. Let $\vec{\overline{\gamma}} = \{\overline{\gamma_1}, \overline{\gamma_2}, \ldots, \overline{\gamma_s}\}$, $s \leq n$ denote the set of non-faulty wires, and $\overline{R_i}$, $|\overline{R_i}| = 12l$, $1 \leq i \leq s$, denote the corresponding string of unopened bits; let $\overline{R'_i}$ be the corresponding string in R's possession.
4. For $1 \leq i \leq s$, S chooses m_i such that $m = m_1 \oplus m_2 \oplus \cdots \oplus m_s$, and sends $S_i = E(m_i) \oplus \overline{R_i}$, $1 \leq i \leq s$, over Pub. R computes $m'_i = D(S_i \oplus \overline{R'_i})$ for all $1 \leq i \leq s$. Then R outputs $m' = m'_1 \oplus m'_2 \oplus \cdots \oplus m'_s$.

Figure 4 The SMT-PD protocol from [25].

Functionality $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ The functionality is parameterized by the identities of the sender S, the receiver R, and the n wire-parties $\vec{W} = (W_1, \ldots, W_n)$. At the first activation, verify that $sid = (P_s, P_r, sid')$, where $\{P_s, P_r\} = \{S, R\}$. Initialize variables $m_1, \ldots, m_n$ to $\perp$.

- Upon receiving input $(\text{SEND}, sid, W_i, v_i)$ from P_s in round ρ (which is the same for all W_i), record $m_i \leftarrow v_i$. If any $P \in \{P_s, P_r, W_i\}$ is marked as corrupted, then send $(\text{SENDLEAK}, sid, W_i, m_i)$ to the adversary; otherwise send $(\text{SENDLEAK}, sid, W_i, l(m_i))$.
- Upon receiving $(\text{INFLSEND}, sid, W_i, m_i')$ from the adversary: If any $P \in \{P_s, P_r, W_i\}$ is corrupted, and $(\text{SENT}, sid, W_i, m_i)$ has not yet been sent to P_r, then set $m_i \leftarrow m_i'$.
- Upon receiving (FETCH, sid, W_i) from P_r in round ρ': If P_r is corrupted, then send $(\text{FETCHLEAK}, sid, W_i)$ to the adversary; otherwise, if $\rho' = \rho + 2$, then output $(\text{SENT}, sid, W_i, m_i)$ to P_r if it has not yet been sent.
- Upon receiving $(\text{OUTPUT}, sid, W_i)$ from the adversary: If P_r is corrupted, then output $(\text{SENT}, sid, W_i, m_i)$ to P_r if it has not yet been sent.
- Upon receiving $(\text{CORRUPT}, sid, P)$ from the adversary for $P \in \{P_s, P_r, W_1, \ldots, W_n\}$, mark P as corrupted. If P is some wire-party W_i, then send $(\text{SENDLEAK}, sid, m_i)$ to the adversary; otherwise, send $(\text{SENDLEAK}, sid, m_1, \ldots, m_n)$. If $P = P_r$, then additionally leak any previous fetch requests made by P_r.

Figure 5 The Wire Channel functionality.

B Proofs

Proof of Theorem 5. Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no environment can distinguish whether it is interacting with $\Pi_{\text{SMT}}(S, R, \vec{W})$ and $\mathcal{A}$, or with $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ and $\mathcal{S}$. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$, $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$, and the parties in a simulated execution of the protocol. All inputs from $\mathcal{Z}$ are forwarded to $\mathcal{A}$, and all outputs from $\mathcal{A}$ are forwarded to $\mathcal{Z}$. Moreover, whenever $\mathcal{A}$ corrupts a party in the simulation, $\mathcal{S}$ corrupts the same party in the ideal world by interacting with $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ (except if the party is a wire-party), and if the corruption was direct (i.e., not via one of the aiding functionalities), then $\mathcal{S}$ sends $\mathcal{A}$ the party's state and follows $\mathcal{A}$'s instructions thereafter for that party.

The simulated execution starts upon $\mathcal{S}$ receiving $(\text{SENDLEAK}, sid, \hat{m})$ from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ in round ρ for $sid = (P_s, P_r, sid')$, where $\hat{m} \in \{m, l(m)\}$ and m is the message to be sent. Now, $\mathcal{S}$ executes the first two phases of the protocol honestly, by simulating sending random strong pads (shares $h_i(\cdot)$ and checking pieces $\vec{C}_i = (c_{1i}, \ldots, c_{ni})$) from P_s to P_r through the n wire-parties (i.e., by simulating leakage from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ to $\mathcal{A}$, and responding to corruption and influence requests directed from $\mathcal{A}$ to $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$) and by simulating sending the response from P_r to P_s over the authenticated channel (i.e., by appropriately playing the role of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$ for $\mathcal{A}$). For the third phase of the protocol, $\mathcal{S}$ simulates honestly, except for choosing z when P_s and P_r are both honest in which case $\mathcal{S}$ simulates sending a random value z from P_s to P_r over $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$ instead of $z = m \oplus Pad$. When P_s or P_r is corrupted by $\mathcal{A}$, $\mathcal{S}$ learns m via leakage from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ and thus can send $z = m \oplus Pad$ just like in the real protocol. Note that the simulated P_s may need to abort, and that if the simulated P_r aborts by outputting $\perp$, then $\mathcal{S}$ can influence $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, since this can only happen if $\mathcal{A}$ corrupts P_s or P_r.

Functionality $\mathcal{F}_{\text{SC}}^{G_n}$ The functionality is parameterized by a graph $G_n = (V, E)$ of party identities and communication edges. At the first activation, verify that $sid = (P_i, P_j, sid')$, where $(P_i, P_j) \in E$; else halt. Initialize variable m to $\perp$.

- Upon receiving input (SEND, sid, v) from P_i in round ρ, record $m \leftarrow v$. If P_i or P_j is marked as corrupted, then send $(\text{SENDLEAK}, sid, m)$ to the adversary; otherwise send $(\text{SENDLEAK}, sid, l(m))$.
- Upon receiving $(\text{INFLSEND}, sid, m')$ from the adversary: If P_i or P_j is corrupted, and (SENT, sid, m) has not yet been sent to P_j, then set $m \leftarrow m'$.
- Upon receiving (FETCH, sid) from P_j in round $\rho + 1$, output (SENT, sid, m) to P_j if it has not yet been sent.
- Upon receiving $(\text{CORRUPT}, sid, P)$ from the adversary for $P \in \{P_i, P_j\}$, mark P as corrupted and send $(\text{SENDLEAK}, sid, m)$ to the adversary.

Figure 6 The Secure Channel functionality for (incomplete) graph G_n.

Next, we describe how $\mathcal{S}$ simulates P_r's response to a FETCH input from $\mathcal{Z}$ in the real world. If P_r is corrupted by $\mathcal{A}$, then $\mathcal{S}$ can wait to receive $(\text{FETCHLEAK}, sid)$ from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, upon which it possibly leaks the fetch to $\mathcal{A}$ and then sends INFLSEND and OUTPUT messages to $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ as appropriate. If P_s is corrupted by $\mathcal{A}$, then $\mathcal{S}$ needs to constantly influence $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ during the second phase of the protocol, so that the dummy P_r fetches the correct value. If neither P_s nor P_r is corrupted, then $\mathcal{S}$ can simply let the dummy P_r fetch from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ when instructed by $\mathcal{Z}$. An important case is when both P_s and P_r are honest in the beginning of the third phase (at the time $\mathcal{S}$ decides the value of z) and then at least one of them gets corrupted before the protocol ends (before the output is fetched). In this case, $\mathcal{A}$ receives enough leakage from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ to interpolate the pad and compute the value of the message from z. Since z is chosen randomly by $\mathcal{S}$, the message learned by $\mathcal{A}$ deviates from what is sent by P_s, causing $\mathcal{Z}$ to distinguish the real and ideal worlds. In such a situation, $\mathcal{S}$ learns the actual value of m via leakage from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, and hence it can cheat by calculating a fake pad' satisfying $z = m \oplus pad'$ and then simulate leaking from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ to result in pad'.

The simulation is perfect. Indeed, by corrupting at most τ wires, $\mathcal{A}$ learns nothing about $h_i(0)$ for honest wires, because the $h_i(\cdot)$'s are independent random polynomials of degree τ. Moreover, $f(\cdot)$ is also a random polynomial of degree τ so $\mathcal{A}$ learns nothing about $f(0)$ (i.e., pad used by the protocol looks uniformly random to $\mathcal{Z}$). Thus, choosing a random value z by $\mathcal{S}$ looks perfectly indistinguishable from the real protocol execution to $\mathcal{Z}$. Besides, $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$ acts like an authenticated channel in the protocol, and hence in the real world P_r outputs the sender's input. (See the full version [16] for more details). ◀

Proof of Theorem 7 (Sketch). We construct a simulator $\mathcal{S}$ that is very similar to the simulator in the proof of Theorem 5. However, $\mathcal{S}$ now interacts with a wrapped functionality, and corruption messages for wire-parties are indeed sent because they can now be processed by the wrapper. Another difference concerns the case in which P_s and P_r are not corrupted by $\mathcal{A}$. If $\mathcal{A}$ corrupts only a minority of the wire-parties, then $\mathcal{S}$ can simply use a random value of z in the third phase of the protocol, and let the dummy P_r fetch its output as before. Otherwise, as soon as enough wire-parties are corrupted, $\mathcal{S}$ sends a DOOM message for P_s to the wrapper, which will be accepted by definition of $\mathscr{D}_{\text{PSMT}}$, and obtains m as leakage. Now, $\mathcal{S}$ can use $z = m \oplus pad$ in the third phase, and influences the wrapper every time the value that the real-world P_r would have output changes (these influence messages will be accepted by the wrapper). Another issue that comes up in the case that P_s and P_r remain honest is

that $\mathcal{A}$ might exceed a minority of wire-party corruptions only after $\mathcal{S}$ has already chosen a random z. However, $\mathcal{S}$ can handle this by cheating and computing a fake pad consistent with m, like the simulator in the proof of Theorem 5 does. Finally, $\mathcal{S}$ may need to simulate sender or receiver aborts when $\mathcal{A}$ corrupts a majority of wire-parties but not P_s or P_r. ◀

Proof of Theorem 8. Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no environment $\mathcal{Z}$ can distinguish whether it is interacting with $\Pi_{\text{SMT-PD}}(S, R, \vec{W}, l)$ and $\mathcal{A}$, or with $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ and $\mathcal{S}$. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$, $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$, and the parties in a simulated execution of the protocol. All inputs from $\mathcal{Z}$ are forwarded to $\mathcal{A}$, and all outputs from $\mathcal{A}$ are forwarded to $\mathcal{Z}$. Moreover, whenever $\mathcal{A}$ corrupts a party in the simulation, $\mathcal{S}$ corrupts the same party in the ideal world by interacting with $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ (except if the party is a wire-party), and if the corruption was direct (i.e., not via either of the aiding functionalities), then $\mathcal{S}$ sends $\mathcal{A}$ the party's state and thereafter follows $\mathcal{A}$'s instructions for that party.

The simulated execution starts upon $\mathcal{S}$ receiving $(\textsc{SendLeak}, sid, \hat{m})$ from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ in round ρ for $sid = (P_s, P_r, sid')$, where $\hat{m} \in \{m, l(m)\}$ and m is the message to be sent. Now, $\mathcal{S}$ simulates the first three phases of the protocol honestly, by simulating sending random bitstrings from P_s to P_r through the n wire-parties (i.e., by simulating leakage from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ to $\mathcal{A}$, and responding to corruption and influence requests directed from $\mathcal{A}$ to $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$) and by simulating sending a message from P_s to P_r or vice versa over the public channel (by appropriately playing the role of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$ for $\mathcal{A}$). In the fourth phase, $\mathcal{S}$ chooses random m_i's to be encoded (rather than m_i's such that $m = m_1 \oplus \cdots \oplus m_s$) if P_s and P_r are still honest; if P_s or P_r is corrupted by $\mathcal{A}$, then $\mathcal{S}$ learns m via leakage from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$.

Next, we describe how $\mathcal{S}$ simulates P_r's response to a $\textsc{Fetch}$ input from $\mathcal{Z}$ in the real world. If P_r is corrupted by $\mathcal{A}$, then $\mathcal{S}$ can wait to receive $(\textsc{FetchLeak}, sid)$ from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, upon which it possibly leaks the fetch to $\mathcal{A}$ and then sends $\textsc{InflSend}$ and $\textsc{Output}$ messages to $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ as appropriate. Otherwise, if P_s is corrupted by $\mathcal{A}$, then $\mathcal{S}$ needs to constantly influence $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ so that the dummy P_r fetches the correct value. Finally, if neither P_s nor P_r is corrupted, then $\mathcal{S}$ simply lets the dummy P_r fetch from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ when instructed by $\mathcal{Z}$. In this case, the real-world P_r outputs m except with the error probability.

An important issue is that when P_s or P_r is corrupted only after $\mathcal{S}$ has already decided on the random m_i's to be encoded in the fourth phase, $\mathcal{A}$ may be able to recover some m' from its view of the bitstrings sent in the first phase, but m' may not equal m and this could allow $\mathcal{Z}$ to distinguish between the real and ideal worlds. However, $\mathcal{S}$ can handle this case by faking what was sent in the first phase. In particular, at least one bitstring (corresponding to an uncorrupted wire-party) sent in the first phase is not visible to $\mathcal{A}$, so $\mathcal{S}$ can redefine it to be consistent with m (which $\mathcal{S}$ learns from leakage from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$).

The simulation is not perfect as there is an error probability, but $\mathcal{Z}$ still cannot distinguish between the two worlds. In particular, when P_s and P_r are not corrupted by $\mathcal{A}$, the assumption that at most all but one of the wire-parties are corrupted implies that the random bitstring sent on at least one wire in the first phase will mask the value of m from $\mathcal{A}$. ◀

Proof of Theorem 16. We first prove that $\mathscr{D}'$ is AE-monotone. Suppose that $(T_i, D_i) \in \mathscr{D}'$ and $T_i \subseteq T_j$ for $T_j \in \mathcal{T}'$. This means that $D_i = D \cup A$ for some D, A such that $D \subseteq D(T_i)$ and $(T_i \cup D(T_i), A) \in \mathscr{D}$. We want to show that $(T_j, D_i) \in \mathscr{D}'$, and it suffices to show that $D \subseteq D(T_j)$ and $(T_j \cup D(T_j), A) \in \mathscr{D}$. Since $D(T_i) \subseteq D(T_j)$ (using the fact that $\mathscr{D}_1, \ldots, \mathscr{D}_m$ are all AE-monotone), it follows that $D \subseteq D(T_j)$. On the other hand, since $T_i \cup D(T_i) \subseteq T_j \cup D(T_j)$, it follows that $(T_j \cup D(T_j), A) \in \mathscr{D}$ (using the fact that $\mathscr{D}$ is AE-monotone and that $T_j \cup D(T_j) \in \mathcal{T}$). We now prove the security of the compiled protocol.

Let $\mathcal{S}$ be a simulator (guaranteed to exist by the security of Π) such that no environment $\mathcal{Z}$ can distinguish whether it is interacting with Π and the dummy adversary $\mathcal{D}$, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ and $\mathcal{S}$. We use $\mathcal{S}$ to construct a simulator $\mathcal{S}'$ such that no environment $\mathcal{Z}'$ can distinguish whether it is interacting with $\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi)$ and $\mathcal{D}$, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F})$ and $\mathcal{S}'$.

$\mathcal{S}'$ internally runs $\mathcal{S}$ and plays the role of the environment and $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ for it. Inputs from $\mathcal{Z}'$ are forwarded to $\mathcal{S}$, with some additional processing. When $\mathcal{Z}'$ sends a corruption request directed to a party (i.e., telling $\mathcal{D}$ to corrupt a party directly), this is forwarded without modification. However, when $\mathcal{Z}'$ sends message delivery requests directed to an instance of $\mathcal{W}_{\text{AE}}^{\mathscr{D}_i}(\mathcal{F}_i)$ for some $i \in [m]$ (e.g., telling $\mathcal{D}$ to send a CORRUPT or INFLUENCE message to that functionality), $\mathcal{S}'$ sends message delivery requests directed to a corresponding instance of $\mathcal{F}_i$, with the following exception: a request to deliver a DOOM message is replaced by a request to deliver a CORRUPT message if $\mathscr{D}_i$ would accept it, and is dropped otherwise.

Similarly, outputs from $\mathcal{S}$ are forwarded to $\mathcal{Z}'$, with some additional processing. Assuming that Π uses instances of $\mathcal{F}_1, \ldots, \mathcal{F}_m$ to handle all inter-party communication, these outputs should take the form of reports of incoming messages directed from either a party or an instance of an aiding functionality $\mathcal{F}_i$ to the dummy adversary for Π; thus, the processing done by $\mathcal{S}'$ is that reported messages from an instance of $\mathcal{F}_i$ are replaced by reported messages from an instance of $\mathcal{W}_{\text{AE}}^{\mathscr{D}_i}(\mathcal{F}_i)$. Finally, $\mathcal{S}'$ plays the role of $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ by simply forwarding messages from $\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F})$ to $\mathcal{S}$ as if coming from $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$, and forwarding messages directed to $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ (from $\mathcal{S}$) to $\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F})$, except that CORRUPT messages for doomed parties (i.e., parties that $\mathcal{Z}'$ did not request to corrupt) are replaced by DOOM messages.

Suppose for a contradiction that there is an environment $\mathcal{Z}'$ such that $\text{IDEAL}_{\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F}),\mathcal{S}',\mathcal{Z}'} \not\equiv \text{EXEC}_{\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi),\mathcal{D},\mathcal{Z}'}$. Then, we construct an environment $\mathcal{Z}$ such that $\text{IDEAL}_{\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F}),\mathcal{S},\mathcal{Z}} \not\equiv \text{EXEC}_{\Pi,\mathcal{D},\mathcal{Z}}$. The environment $\mathcal{Z}$ will simulate an interaction between $\mathcal{Z}'$ and $\mathcal{D}$, and output whatever $\mathcal{Z}'$ outputs, as well as do some additional processing. Whenever $\mathcal{Z}'$ instructs its dummy adversary to deliver a message to an instance of $\mathcal{W}_{\text{AE}}^{\mathscr{D}_i}(\mathcal{F}_i)$, this is translated by $\mathcal{Z}$ into a delivery request for a corresponding instance of $\mathcal{F}_i$ and forwarded to the external adversary (either $\mathcal{S}$ or $\mathcal{D}$), except that a request to deliver a DOOM message is converted into a request to deliver a CORRUPT message if allowed by $\mathscr{D}_i$ and dropped otherwise. Corruption requests directed to parties are forwarded to the external adversary unmodified.

Next, whenever $\mathcal{Z}$ receives subroutine output from the external adversary, this is forwarded to $\mathcal{Z}'$, except that reported messages from instances of $\mathcal{W}_{\text{AE}}^{\mathscr{D}_i}(\mathcal{F}_i)$ are translated into reported messages from corresponding instances of $\mathcal{F}_i$. Finally, $\mathcal{Z}$ simply relays inputs and outputs between $\mathcal{Z}'$ and parties. We conclude by claiming that $\text{IDEAL}_{\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F}),\mathcal{S}',\mathcal{Z}'} \equiv \text{IDEAL}_{\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F}),\mathcal{S},\mathcal{Z}}$ and $\text{EXEC}_{\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi),\mathcal{D},\mathcal{Z}'} \equiv \text{EXEC}_{\Pi,\mathcal{D},\mathcal{Z}}$. Indeed, if $\mathcal{Z}$ interacts with $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ and $\mathcal{S}$, then the view of the simulated $\mathcal{Z}'$ within $\mathcal{Z}$ is identical to the view of $\mathcal{Z}'$ when interacting with $\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F})$ and $\mathcal{S}'$, and similarly if $\mathcal{Z}$ interacts with Π and $\mathcal{D}$, then the view of the simulated $\mathcal{Z}'$ within $\mathcal{Z}$ is identical to the view of $\mathcal{Z}'$ when interacting with $\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi)$ and $\mathcal{D}$. ◀

Static vs. Adaptive Security in Perfect MPC: A Separation and the Adaptive Security of BGW

Gilad Asharov
Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel

Ran Cohen
Efi Arazi School of Computer Science, Reichman University, Herzliya, Israel

Oren Shochat
Department of Computer Science, Bar-Ilan University, Ramat-Gan Israel

Abstract

Adaptive security is a highly desirable property in the design of secure protocols. It tolerates adversaries that corrupt parties as the protocol proceeds, as opposed to static security where the adversary corrupts the parties at the onset of the execution. The well-accepted folklore is that static and adaptive securities are *equivalent* for perfectly secure protocols. Indeed, this folklore is backed up with a transformation by Canetti et al. (EUROCRYPT'01), showing that any perfectly secure protocol that is statically secure and satisfies some basic requirements is also adaptively secure. Yet, the transformation results in an adaptively secure protocol with *inefficient simulation* (i.e., where the simulator might run in super-polynomial time even if the adversary runs just in polynomial time). Inefficient simulation is problematic when using the protocol as a sub-routine in the computational setting.

Our main question is whether an alternative *efficient* transformation from static to adaptive security exists. We show an inherent difficulty in achieving this goal generically. In contrast to the folklore, we present a protocol that is perfectly secure with efficient static simulation (therefore also adaptively secure with inefficient simulation), but for which efficient adaptive simulation does not exist (assuming the existence of one-way permutations).

In addition, we prove that the seminal protocol of Ben-Or, Goldwasser and Wigderson (STOC'88) is secure against adaptive, semi-honest corruptions with *efficient* simulation. Previously, adaptive security of the protocol, as is, was only known either for a restricted class of circuits, or for all circuits but with inefficient simulation.

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

Keywords and phrases secure multiparty computation, perfect security, adaptive security, BGW protocol

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.15

Related Version *Full Version*: `https://eprint.iacr.org/2022/758.pdf`

Funding *Gilad Asharov*: Sponsored by the Israel Science Foundation (grant No. 2439/20), by JPM Faculty Research Award, by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office, and by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 891234.
Ran Cohen: Research partially supported by NSF grant no. 2055568.
Oren Shochat: Sponsored by the Israel Science Foundation (grant No. 2439/20).

1 Introduction

Secure multiparty computation (MPC) [31, 20] enables a set of mutually distrustful parties to compute a joint function while keeping the privacy of their inputs and the correctness of their outputs. The seminal results from the '80s [31, 20, 5, 11, 30] as well as the vast majority

of MPC protocols in the literature were proven secure with respect to a *static* adversary; that is, security is guaranteed as long as the adversary decides which parties to corrupt at the onset of the execution. A more realistic setting, first considered by Beaver and Haber [4] and by Canetti et al. [10], considers an *adaptive* adversary, who can dynamically decide which parties to corrupt during the course of the protocol. Adaptive security is known to be strictly stronger than static security with many impossibility results separating the two notions, e.g., [10, 9].

In this work we focus on the well-studied setting of *perfect security*, where all existing separations from the literature no longer hold. Perfectly secure protocols guarantee security facing computationally unbounded adversaries without any error probability. This is a highly desirable property that in some sense provides the strongest security notion for MPC. For example, Kushilevitz et al. [24] showed that any perfectly secure protocol that is proven secure in the standalone model using a straight-line and black-box simulation[1] automatically guarantees security under universal composition (UC) [8].[2] Most relevant to our work, Canetti et al. [9] showed that *any* perfect statically secure protocol, satisfying basic requirements, remains secure also in the presence of adaptive adversaries. This result, however, comes with a caveat, since the transformation from static to adaptive security does not preserve the efficiency of the simulation.

Roughly speaking, a protocol is deemed secure if any attack on an execution of the protocol in the real world can be simulated also in an ideal world where a trusted party receives the inputs from all the parties and computes the function on their behalf. It is desirable that the simulator, i.e., the adversary who simulates the attack in the ideal world, will use roughly the same resources as the adversary who carries out the attack on the real-world protocol. In particular, we would like the simulator to run in polynomial time with respect to the running time of the real-world adversary; if so, we say that the simulation is *efficient*.

Unfortunately, the adaptive simulator in [9] does *not* run in polynomial time with respect to the real-world adversary. This means that given a perfectly secure protocol against static adversaries with an efficient simulator, the transformation in [9] guarantees adaptive security, albeit with an inefficient simulator. This leads us to the following fundamental question:

Does perfect, static security with efficient simulation imply perfect, adaptive security with efficient simulation?

Stated differently, is there another generic transformation from static to adaptive security, other than [9], that preserves the efficiency of the simulation? Are there other assumptions on the structure of the protocol and/or on the static simulation that might lead to such an efficient transformation? Or, perhaps, do there exist protocols that are statically secure but for which *efficient* adaptive simulation simply does not exist?

Efficient vs. inefficient simulation. One may ask whether the efficiency loss in the simulation makes a difference when considering perfect security: If security is anyway guaranteed against computationally unbounded adversaries, does it matter if the simulator is inefficient? Indeed, inefficient simulation is a weaker-yet-acceptable security notion when considering information-theoretic security.

---

[1] A simulator is called *straight-line* if it does not rewind the adversary, and is called *black-box* if it does not rely on the code of the adversary.

[2] We note that Backes et al. [3] showed that this transformation no longer holds if the simulator is not straight-line.

It turns out that inefficient simulation has an undesirable impact when considering *composition* of secure protocols. For example, consider a perfectly secure protocol π for computing a function f that is defined over secure communication channels, and consider a computationally secure realization of secure channels over authenticated channels (e.g., using non-committing encryption [10]). If π is secure with efficient simulation, then a composition theorem (e.g., from [7, 8]) can be used to derive a computationally secure protocol for f over authenticated channels. However, if the simulation is *inefficient* then the composition will not go through since it will result with a super-polynomial time adversary that can break the cryptographic assumptions used to realize secure channels.

A case study: on the adaptive security of the BGW protocol. The seminal results of Ben-Or, Goldwasser, and Wigderson [5] and of Chaum, Crépeau, and Damgård [11] show that any function f can be computed by an n-party protocol with perfect security as long as $t < n/2$ of the parties are corrupted in the semi-honest setting, and as long as $t < n/3$ are corrupted in the malicious setting. A full proof of security for the BGW protocol was given in 2011 by Asharov and Lindell [1, 2]. The proof was specified in the static, standalone setting, and universally composable security and security against adaptive adversaries were derived using the transformations of Kushilevitz et al. [24] and Canetti et al. [9], respectively. However, as discussed above, adaptive security is obtained with an inefficient simulation.

This issue was revisited by Damgård and Nielsen [15], who showed that the semi-honest version of the BGW protocol achieves adaptive security with efficient simulation. However, the result of [15] holds only for circuits in which each output wire is a direct output of a multiplication gate. Obviously, one can manually add such "multiplications with 1" to each output wire. While this seems suffice, there are two reasons why we revisit this problem: First, for linear functions, the semi-honest version of the BGW protocol can tolerate up to n corruptions, whereas the requirement that each output wire is a direct output of a multiplication gate reduces the corruption threshold to $t < n/2$. Second, this subtlety has been neglected by prior works that relied on [15], e.g., Lin et al. [26, Lem. 6.2] claimed that any degree-2 function can be computed by the BGW protocol in two rounds with adaptive security and efficient simulation. Yet, when adding multiplication gates the round complexity increases, which implies a gap in the literature as there is no proof for the claim mentioned in [26].

Alternatively, one can interpret this additional restriction on the circuit as adding some re-randomization step at the very end of the protocol before the parties reconstruct their output. Is this step essential to achieve adaptive security for all circuits? Can one prove the adaptive security of the original protocol directly, without the additional communication round?

1.1 Our Results

Our work revisits the question of static versus adaptive in perfectly secure multiparty computation. We show that in contrast to the "weaker" definition of adaptive security (i.e., inefficient simulation), perfect static security no longer implies perfect adaptive security when demanding the simulation to be efficient, even when merely considering semi-honest adversaries. Complementarily, we show that the BGW protocol is adaptively secure with efficient simulation even without changing the underlying circuit; this answers an open question posed by Damgård and Nielsen [15]. We focus on semi-honest security, which is

enough to highlight the subtleties that arise; indeed, the gap in the literature discussed above already appears in this setting. We conjecture that the analysis extends to the malicious case using standard secret-sharing techniques.

We proceed to describe our results in more details.

Separating perfect adaptive security from perfect static security. Our first result shows that under some cryptographic assumptions, there is no hope of finding an alternative (efficient) transformation to that of Canetti et al. [9], and that inefficiency of the adaptive simulation is inherent. More precisely, that there exist protocols that admit perfect, static security with efficient simulation but for which an efficient adaptive simulation does not exist.

▶ **Theorem 1.** *Assume the existence of a one-way permutation. Then, there exists an n-party functionality f and a protocol that securely computes f with efficient perfect static simulation, but for which efficient perfect adaptive simulation does not exist.*

The theorem is proven by showing a protocol for which all the additional requirements of [9] hold and therefore (inefficient) adaptive simulation *does* exist for this protocol, but an efficient adaptive simulator can be used to invert the one-way permutation with inverse-polynomial probability. Interestingly, this implies that the protocol is regarded as adaptively secure in the perfect setting, but the exact same protocol is not adaptively secure in the *computational* setting, where both the adversary and the simulator should run in polynomial time in the security parameter. We therefore derive the following somewhat counter-intuitive corollary.

▶ **Corollary 2.** *Assume the existence of a one-way permutation. Then, there exists an n-party functionality f and a protocol that securely computes f with perfect adaptive security, but that does not securely compute f with computational adaptive security.*

Revisiting adaptive security in the standalone setting. The above does not rule out the possibility of finding additional requirements from the protocol that would imply efficient adaptive simulation. This is exactly the approach taken by Damgård and Nielsen [15], who showed that under additional requirements of the protocol, an efficient adaptive simulation exists.

The transformation of [15] is directly proved in the UC framework with its full generality, capturing reactive functionalities and concurrency issues at once. However, the strong guarantees do not come without a price, since the requirements from the static simulator must capture multiple input phases (as required for reactive computations) and deal with the technical overhead needed for concurrent composition, e.g., incorporating an "online" environment to the definition. As a small side contribution we simplify this transformation.

Specifically, in addition to proving perfect static security, the transformation of [15] requires an additional (efficient!) algorithm, called "Patch," for sampling randomness that explains the simulated protocol whenever a corruption occurs. The adaptive simulator invokes the static simulator on a dynamically growing set of corrupted parties (initially empty). At any point, the simulation of the protocol towards the adaptive adversary is done by forwarding messages from the adaptive adversary to the static simulator, and vice-versa. Upon a corruption of a new party, say of P_i, the Patch algorithm receives the state of the static simulator until this point together with the input and output of P_i, and outputs a new state for the static simulator that allows the continuation of the simulation as if P_i was statically corrupted from the beginning.

We then propose an alternative recipe for proving universal composability and (efficient) adaptive security in the perfect setting:

1. Prove that the protocol is (efficiently) statically secure in the perfect *standalone* setting and that the protocol satisfies some natural requirements (similar to those in [9]).
2. Show the existence of an efficient "Patch" algorithm corresponding to the static simulator.
3. We show a transformation (which is essentially a distilled version of [15]) that the protocol is perfectly adaptively secure with an efficient simulator in the standalone setting.
4. Using [24], the protocol is also secure in the perfect adaptive setting with efficient simulation and with universal composability.

We remark that this result is not technically novel and is inspired by [15]. We hope that providing an alternative definition in the standalone setting would simplify proofs of efficient adaptive security in the future, as the designer of the protocol can focus on the standalone setting.

Adaptive security of the BGW protocol. Finally, we follow our recipe proposed above and show that the (semi-honest) BGW protocol is efficiently adaptively secure. No additional step to the protocol is needed, and the proof works for any circuit (as opposed to the proof of [15]). This result solves an open problem raised by [15], whether the assumption on the circuit (that each output wire is a direct output of a multiplication gate) is necessary. This reaffirms the security of the BGW protocol [5], and closes a gap in the proofs of [1, 15], providing sound foundations for cryptography.

▶ **Theorem 3.** *Let f be a deterministic n-party functionality. The BGW protocol securely computes f with perfect adaptive security and efficient simulation facing a semi-honest adversary corrupting $t < n/2$ parties.*

Further, if f is a linear function, the BGW protocol securely computes f with perfect adaptive security and efficient simulation facing a semi-honest adversary corrupting $t < n$ parties.

1.2 Related Work

Adaptive security is known to be strictly stronger than static security in many settings, with many impossibility results separating the two notions. Below, we compare our separation to existing separations between static and adaptive security from the literature.

The first separation was presented by [10] and relied on a positive error probability of the statically secure protocol. They considered a dealer that secret shares a value to a random set of parties and later announces their identities; an adaptive adversary can corrupt the members of the set and learn the value while a static adversary can only guess this set ahead of time and succeed with negligible (yet positive) probability. In this setting, the seminal protocol of Rabin and Ben-Or [30] that guarantees statistical information-theoretic security is not secure in the adaptive setting, as illustrated by Cramer et al. [14], who gave an adaptively secure variant of the protocol. Separations based on statistical security are different than ours as we consider *perfect* protocols that have zero error probability.

In the computational-security setting, many statically secure primitives do not remain secure under adaptive corruptions. For example, Nielsen [29] showed that public-key encryption for unbounded messages requires a programmable random oracle. Canetti et al. [9] separated adaptively secure commitments from statically secure ones. Lindell and Zarosim [27] showed that achieving adaptively secure oblivious transfer (OT) requires stronger assumptions than statically secure OT. Katz et al. [23] ruled out adaptively secure fully homomorphic

encryption; the latter result was generalized in [13]. Separations based on computational assumptions are different than ours as we consider *information-theoretic* protocols that remain secure against computationally unbounded adversaries.

Canetti et al. [9] showed a separation based on the ability of the adversary to corrupt a party and change its input to the protocol based on messages that have already been transmitted. Similar separations were illustrated also for broadcast protocols [22, 12]. Canetti et al. [9] showed that such separations no longer hold when considering protocols that have a *committal round* [28], i.e., a fixed round in which all inputs to the protocol are committed. These separations hold only for malicious adversaries that can deviate from the protocol; our separation applies for semi-honest adversaries that cannot deviate from the protocol in any way.

Other separations are known when considering restricted interaction patterns, as was shown by Garay et al. [17] for protocols with sublinear communication, and by Boyle et al. [6] for protocols whose communication graph admits a sublinear cut. Our separation relies on the BGW protocol that induces a complete communication graph.

Finally, Garg and Sahai [18] showed that constant-round MPC with black-box simulation in the plain model cannot tolerate corruption of all of the parties. Our result holds irrespective of the number of rounds and does not require corrupting all the parties.

Organization of the paper. In Section 2 we present a technical overview of our results, in Section 3 the preliminaries, and in Section 4 we prove our separation result, showing the existence of a protocol that has inefficient adaptive simulation but no efficient adaptive simulation (assuming one-way permutations exist). Due to space limitation we omit the definition of secure multiparty computation and the proof of adaptive security of the BGW protocol, and refer the reader to the full version of this paper.

2 Technical Overview

We provide a technical overview of our results. In Section 2.1 we review our separation result, showing the existence of a protocol that has inefficient adaptive simulation but no efficient adaptive simulation (under some cryptographic assumptions). In Section 2.2, we review the adaptive security of the BGW protocol.

2.1 Static Security Does Not Imply Adaptive Security

Definition of adaptive security. We first recall the definition of adaptive security. We remark that since the transformation of [9] assumes some additional properties from the statically secure protocol and its simulation (specifically, it assumes that the protocol has a straight-line, black-box simulation), our description here incorporates those properties in the informal definition.

Given a protocol π, an adaptive adversary might corrupt parties on the fly as the protocol proceeds. Upon corruption, the adversary sees the corrupted party's random tape, the input it used, and all the messages it received so far. From that point on, the adversary completely controls the behavior of P_i. As the protocol proceeds, the adversary might decide to corrupt additional parties, as a function of whatever it saw so far.

We follow the ideal/real simulation paradigm and say that a protocol is adaptively secure if for every such an adversary in the real world, there exists a simulator in the ideal world that simulates its behavior. In the ideal world, the simulator invokes the real-world adversary and simulates the honest parties sending their message to the corrupted parties (without knowing

the inputs of the honest parties). At every round, the adversary might ask to corrupt some party P_i in the simulated protocol, and the simulator corrupts the corresponding party in the ideal world. Upon corruption, the simulator learns the input of P_i (and its output, if it was already computed by the trusted party), and it has to provide the adversary the input *together with randomness* that explains the messages sent by this party in the simulation until this point (known also as "equivocality" in the literature).

This is the challenging part of the proof as the simulator has to first simulate P_i without knowing its input, "commit" to some messages on its behalf, and later upon corruption find a randomness r_i that explains all the messages that were sent so far according to the input x_i. Note that the simulator is not allowed to rewind the adversary at any point of the execution, i.e., the simulation is "straight-line."

First attempt. The transformation of [9] shows that for any perfectly secure protocol, randomness r_i describing the behavior of P_i so far, exists. This implies that the simulator can *always* find it; however, finding it might not be an efficient procedure. Our first attempt is adding some cryptographic hardness while targeting exactly this procedure of finding the matching randomness. That is, our goal is making the finding of the randomness a computationally hard problem.

It is intuitive to simply take the BGW protocol, even just for semi-honest security and for computing a linear function, with one modification: Before a party starts the protocol, it takes its random tape r, and uses $\mathsf{OWP}(r)$ as its randomness in the protocol, where OWP is a one-way permutation. The intuition is that whenever the adversary asks to corrupt some party, the simulator would effectively have to invert the one-way permutation, which is computationally infeasible. The construction is still statically secure since the static simulator only goes "forward"; that is, it chooses some randomness r and then uses $\mathsf{OWP}(r)$ as the randomness of the simulated honest party. Moreover, an efficient simulation exists since r exists; however, it seems that an adaptive adversary will have to move "backward" and invert the one-way permutation.

To elaborate further, let n be the number of parties, and consider the function $f(a_1, \ldots, a_n) = \sum_{i=1}^{n} a_i$; that is, all parties receive the same output, which is the sum of their inputs. We consider some finite field $\mathbb{F}$ with $|\mathbb{F}| > n$. The protocol is as follows:

- **Input:** P_i has $a_i \in \mathbb{F}$ as input, and randomness r_i.
- **The protocol:**
  1. P_i computes $(r_{i,1}, \ldots, r_{i,n-1}) = \mathsf{OWP}(r)$, where each $r_{i,j} \in \mathbb{F}$.
  2. P_i secret shares its input a_i using Shamir's secret sharing scheme with degree $n-1$, using the polynomial $h_i(x) = a_i + r_{i,1}x + \ldots + r_{i,n-1}x^{n-1}$. It gives to each party P_j privately a point $h_i(\alpha_j)$.
  3. Upon receiving $h_1(\alpha_i), \ldots, h_n(\alpha_i)$ from all the parties, the party P_i computes $\beta_i = \sum_{i=j}^{n} h_j(\alpha_i)$ and sends β_i to all other parties.
  4. Upon receiving $\beta_1, \ldots, \beta_n$, each party reconstructs the unique polynomial $H(x)$ for which it holds for every α_j that $H(\alpha_j) = \beta_j$, and outputs $H(0)$.

Simulating this protocol. The above protocol can be simulated for every $t < n$ parties. Consider an adaptive simulator, and assume that the adversary already corrupted $n-2$ parties on the onset of the execution, say $P_3, \ldots, P_n$. The simulator then simulates just two honest parties: P_1 and P_2. As it does not know their inputs it just gives the adversary $2 \cdot (n-2)$ independent random points as the messages of the two honest parties it is simulating. For concreteness, assume that the messages the simulated P_1 sent are $(\gamma_3, \ldots, \gamma_n)$, where γ_i

is supposed to be delivered to P_i. Now, assume that the adversary asks to corrupt P_1 and that the simulator receives its input a_1. The simulator can now find a random polynomial $h_1(x)$ of degree $n-1$ under the constraints that

$$h_1(0) = a_1, \qquad \text{and} \qquad h_1(\alpha_3) = \gamma_3 \quad \dots \quad h_1(\alpha_n) = \gamma_n \ .$$

Note that these constraints define overall $n-1$ points. Since $h_1(x)$ has n coefficients, there are $|\mathbb{F}|$ different polynomials that satisfy these $n-1$ constraints. The simulator can pick one more point at random, and then uniquely determine the polynomial $h_1(x)$ using interpolation. For that particular polynomial, the simulator has to invert the one-way permutation and find the corresponding randomness.

The problem – the reduction to OWP. It is hard to use the above adaptive simulator to construct an inverter for the one-way permutation on some challenge point $y^* \in |\mathbb{F}|^{n-1}$, corresponding to the $n-1$ leading coefficients, say $y^* = (r_{1,1}, \dots, r_{1,n-1})$. The problem is that once the adaptive simulator "commits" to the values $(\gamma_3, \dots, \gamma_n)$, it might be the case that there is *no* input a_1 for which the polynomial $h_1(x) = a_1 + r_{1,1}x + \ldots + r_{1,n-1}x^{n-1}$ satisfies the constraints $h_1(\alpha_3) = \gamma_3$, …, $h_1(\alpha_n) = \gamma_n$. That is, there is no input a_1 in the support that agrees with the challenge y^* and with the simulated view, simultaneously.

Second attempt. To solve the above technicality, we change the functionality and the protocol. Instead of having the input of each party P_i be just $a_i \in \mathbb{F}$, it is augmented to be a polynomial $g_i(x)$ of degree $n-1$ over $\mathbb{F}$. The functionality is then defined as:

$$f\Big(g_1(x), \dots, g_n(x)\Big) = \sum_{i=1}^{n} g_i(0).$$

Note that the parties input polynomials, while all the leading coefficients do not affect the output of the computation. In the protocol, each party P_i then invokes $(r_{i,1}, \dots, r_{i,n-1}) = \mathsf{OWP}(\boldsymbol{\rho}_i)$ where $\boldsymbol{\rho}_i$ is its random tape, and then defines the polynomial $h_i(x) := g_i(x) + r_{i,1}x + \ldots + r_{i,n-1}x^{n-1}$. It then sends to each other party P_j the point $h_i(\alpha_j)$, just as in Step 2 in the previous protocol.

The difference is that now, no matter what points $(\gamma_3, \dots, \gamma_n)$ the adversary commits to as the messages P_1 had sent, for any challenge $y^* = (r_1^*, \dots, r_{n-1}^*)$, the inverter of the OWP can choose an input $g_1(x)$ such that the polynomial $g_1(x) + r_1^*x + \ldots + r_{n-1}^*x^{n-1}$ is in the support of the polynomials that the adaptive simulator might choose. To see that, given a challenge $y^* = (r_1^*, \dots, r_{n-1}^*)$ to the inverter, and given $(\gamma_3, \dots, \gamma_n)$ that were chosen by the simulator as the messages P_1 had sent in the first round, the inverter chooses two additional points γ_1 and γ_2 at random. It then interpolates the unique polynomial $h(x)$ such that for every $i \in [n]$ it holds that $h(\alpha_i) = \gamma_i$. It computes the polynomial $g_1(x) = h(x) - (r_1^*x + \ldots + r_{n-1}^*x^{n-1})$, and gives $g_1(x)$ to the adaptive simulator as the input of P_1.

The simulator now has to reply with some randomness $\boldsymbol{\rho}'$ for which $(r'_{1,1}, \dots, r'_{1,n-1}) = \mathsf{OWP}(\boldsymbol{\rho}')$ such that $h'_1(x) = g_1(x) + \sum_{k=1}^{n-1} r'_{1,k}x^k$ and it holds that $h'_1(\alpha_3) = \gamma_3, \dots, h'_n(\alpha_n) = \gamma_n$. Since $h'_1(\alpha_1)$ and $h'_2(\alpha_2)$ are not determined, the simulator essentially has $|\mathbb{F}|^2$ different polynomials to chose from. However, *one of them* corresponds to the challenge y^*. Moreover, y^* is distributed uniformly in the support of all valid solutions for the simulator. Since the adaptive simulator simulates with perfect security, the inverter succeeds in inverting y^* with probability $1/|\mathbb{F}|^2$. By tuning the parameters (such that $|\mathbb{F}|$ is polynomial in the security parameter), this is an inverse-polynomial advantage. Assuming that OWP is a one-way permutation, this implies that the adaptive simulator cannot run in polynomial time.

2.2 Adaptive Security of the BGW Protocol

The BGW Protocol. We briefly recall the semi-honest version of the BGW protocol [5] that is secure for $t < n/2$ corruptions; see [1] for more details.

Input sharing: Initially, each party P_i secret shares its input x_i using a $(t+1)$-out-of-n Shamir secret sharing (t is the privacy threshold and $t+1$ are needed for reconstruction), and sends the j^{th} share to P_j. Specifically, P_i chooses a random polynomial q_i of degree (at most) t with constant term x_i, and sends to P_j the share $q_i(\alpha_j)$.

Circuit emulation: The parties evaluate the circuit gate by gate on their shares. Addition and scalar-multiplication gates are evaluated locally; that is, each party performs the operation on its shares. Multiplication requires a dedicated sub-protocol in which each party inputs shares for two values a and b, and obtains a share for the product ab as output. For simplicity of the technical overview, we consider that this operation is done using the help of a trusted party (i.e., we work in the f_{mult}-hybrid model).

Output reconstruction: Upon conclusion of the circuit emulation, each party holds one share for each output wire. For simplicity, assume that output wire o_i is the private output for P_i. Then, each party P_j sends the relevant share $\beta_{j\to i}$ for P_i, who can reconstruct the polynomial g_i interpolating the points $(\alpha_1, \beta_{1\to i}), \ldots, (\alpha_n, \beta_{n\to i})$, and obtain $g_i(0)$ as its output.

Static simulation. To simulate the view of the adversary for the set of corrupted parties I of cardinality at most t, the static simulator simulates the input-sharing phase by choosing randomness for the corrupted parties (which, in turn, determines the polynomials they use in the input-sharing phase). Moreover, for the honest parties, the simulator just chooses $|I|$ random elements for each honest party and simulates the honest parties sending their shares on their inputs to the adversary. The simulator can then locally compute the shares of the corrupted parties on each internal wire of the circuit, while for simulating an invocation of f_{mult}, the simulator again just provides the adversary with $|I|$ random shares. In the output-reconstruction phase, the simulator has to provide all shares on the output wires of corrupted parties. It knows the constant term of each such wire, and it knows $|I|$ shares on each wire, and therefore it is possible to interpolate a random polynomial that passes through the $|I| \leq t$ shares and the known constant term, and simulate the honest parties sending shares on this polynomial to the adversary. When $|I| = t$ this is a deterministic process as we already have $t+1$ shares that are determined on the output wires; however, when $|I| < t$, the simulator needs to generate some new shares on that output wires.

The assumption on the circuit. Damgård and Neilsen assumed that the output wires are direct outputs of multiplication gates. As a result, the shares on each output wire are independent of each other – and the simulator can just choose additional $|I| - t$ random shares on the output wire to interpolate the polynomial on that wire. When considering arbitrary circuits, output wires that are not a result of a multiplication gate may have some linear relation between them. To illustrate the challenge, consider output wires o_1, o_2, and o_3, corresponding to three corrupted parties P_1, P_2, and P_3, respectively, such that $o_3 = o_1 + o_2$. Denote the output values by y_1, y_2, and y_3, respectively. The shares that the parties hold on the output wires define polynomials $g_1(x), g_2(x)$ and $g_3(x)$, respectively. In the real execution it holds that $g_3(x) = g_1(x) + g_2(x)$, and therefore the simulator must choose the shares on the output wires wisely to guarantee the same dependency, which makes the simulation more challenging.

Our static simulator. To guarantee such consistencies, our static simulator generates t random shares on each input wire and each output of a multiplication wire. That is, while the adversary corrupt some set I, the simulator, "in its head," chooses an arbitrary set $\hat{I} \supseteq I$ of cardinality exactly t and simulates the shares for all parties in $\hat{I}$, while giving the adversary only shares for the parties in I. As a result, we have no random choice when simulating the output wires. The simulator holds t shares on each output wire, and also knows the constant term on the output wires of the corrupted parties. It can deterministically interpolate the polynomials on the output wire, and simulate the honest parties sending their shares on those wires. This guarantees that if there is any dependency between output wires, then the simulator provides consistent shares.

Our adaptive simulator. Assume that the adaptive adversary corrupts some party P_{i^*}. If $i^* \in \hat{I} \setminus I$, then providing the view of that party is easy. The simulator has already sampled all the shares that P_{i^*} is supposed to receive. On the other hand, if $i^* \notin \hat{I} \setminus I$, then we face a new obstacle. This is because the simulator has already defined t shares on each wire, and defining an additional share on each wire would completely determine each polynomial. Let $j^* \in \hat{I} \setminus I$. The key idea of our adaptive simulator is to "replace" the sampling of shares for party P_{j^*} with sampling shares for party P_{i^*}. Specifically, we show that each random choice made for P_{j^*} that leads to the current view of the adversary, can also be interpreted by a random choice made for P_{i^*} that leads to the exact same view. This procedure then changes the set of the simulator and "forgets" all the random choices made for P_{j^*}, but instead samples matching choices for P_{i^*}. This is essentially sampling random shares for P_{i^*} on the input wires of all honest parties and the outputs of f_{mult}, under the constraints posed by the shares of P_{j^*} on the output wires. Such sampling can be performed efficiently by solving a linear set of equations. Then, we are back to the previous case, where the corruption is made on some $i^* \in \hat{I} \setminus I$.

3 Preliminaries

Our results hold in any natural model that captures perfect adaptive security, for example, [10, 7, 16, 8, 25, 21]. For concreteness, we will state our results using the modular (non-concurrent) composability framework of Canetti [7]. Indeed, the separation in this limited setting extends to any of the models listed above, whereas our positive results translate to the universal-composability setting via the transformation in [24]. Before describing the security model, we give basic notation and define the cryptographic primitive used in our separation.

Notation. We denote by λ the security parameter. For $n \in \mathbb{N}$, let $[n] = \{1, \ldots, n\}$. Let poly denote the set of all positive polynomials and let PPT denote a probabilistic algorithm that runs in *strictly* polynomial time. A function $\nu \colon \mathbb{N} \to [0, 1]$ is *negligible* if $\nu(\lambda) < 1/p(\lambda)$ for every $p \in \mathsf{poly}$ and large enough λ. Given a random variable X, we write $x \leftarrow X$ to indicate that x is selected according to X, and given a set $\mathcal{X}$ we write $x \leftarrow \mathcal{X}$ to indicate that x is selected uniformly at random from $\mathcal{X}$.

One-way permutations. Our separation in Section 4 will rely on the existence of a one-way permutation (OWP); that is a one-way function which is length preserving and one-to-one; we refer the reader to [19] for more details.

▶ **Definition 4** (OWP). *A polynomial-time function $f : \{0,1\}^* \to \{0,1\}^*$ is a one-way permutation if the following conditions are satisfied.*

1. *For every $\lambda \in \mathbb{N}$, f induces a permutation over $\{0,1\}^\lambda$, i.e., $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is one-to-one and onto.*
2. *There exists a deterministic polynomial-time algorithm A such that on input $x \in \{0,1\}^*$ the algorithm A outputs $f(x)$.*
3. *For every PPT algorithm $\mathcal{A}$, every positive polynomial $p(\cdot)$, and all sufficiently large λ's, it holds that*

$$\Pr_{x \leftarrow \{0,1\}^\lambda}\left[\mathcal{A}(f(x)) = x\right] < \frac{1}{p(\lambda)}.$$

4 Static Security Does Not Imply Adaptive Security

In this section we prove Theorem 1. We show a functionality together with a protocol that privately computes it with perfect static security and efficient simulation. Further, we show that if the protocol privately computes the functionality with perfect adaptive security and efficient simulation, then one-way permutations do not exist.

We start by defining the functionality and the protocol. Next, in Lemma 6 we prove static security and in Lemma 9 we prove that adaptive security with efficient simulation cannot be achieved assuming OWP. Combined, this proves Theorem 1.

4.1 The Functionality

The n-party functionality f_{sum} is parametrized by a finite binary field $\mathbb{F}_{2^\ell}$ such that $2^\ell > n$. Looking ahead, having a binary field will enable using a one-way permutation $\mathsf{OWP} : \{0,1\}^* \to \{0,1\}^*$ that is applied on a vector of field elements in a clean way. During the proof below, we will denote the field by $\mathbb{F} := \mathbb{F}_{2^\ell}$. The private input of every party is a polynomial of degree (at most) $n-1$ over $\mathbb{F}$, and the common output is the sum of the free coefficients.

- **Input:** The input of party P_i is a polynomial $g_i(x)$ of degree $n-1$, that is define by n coefficients $a_{i,0}, \ldots, a_{i,n-1} \in \mathbb{F}$ such that $g_i(x) = \sum_{k=0}^{n-1} a_{i,k} x^k$.
- **Output:** The output of every party is $\sum_{i=1}^{n} g_i(0) = \sum_{i=1}^{n} a_{i,0}$.

4.2 The Protocol

The n-party protocol π_{sum} is parametrized by the field $\mathbb{F}$ and assumes the existence of a one-way permutation $\mathsf{OWP} : \mathbb{F}^{n-1} \to \mathbb{F}^{n-1}$, i.e., $\mathsf{OWP} : \{0,1\}^{\ell(n-1)} \to \{0,1\}^{\ell(n-1)}$.

Protocol 5: Separating Protocol π_{sum}.

- **Private input:** The input of party P_i is a polynomial $g_i(x)$ of degree $n-1$ over $\mathbb{F}$.
- **Randomness:** The random tape of party P_i is $\boldsymbol{\rho}_i \leftarrow \mathbb{F}^{n-1}$.
- **Common inputs:** n distinct non-zero elements $\alpha_1, \ldots, \alpha_n \in \mathbb{F}$.
- **The protocol: code for party P_i:**
  1. Compute $(r_{i,1}, \ldots, r_{i,n-1}) = \mathsf{OWP}(\boldsymbol{\rho}_i)$.
  2. Consider the polynomial

$$\begin{aligned} h_i(x) &:= g_i(x) + r_{i,1}x + \ldots + r_{i,n-1}x^{n-1} \\ &= a_{i,0} + (a_{i,1} + r_{i,1})x + (a_{i,2} + r_{i,2})x^2 + \ldots + (a_{i,n-1} + r_{i,n-1})x^{n-1} \ . \end{aligned}$$

3. Send to every P_j its share $\gamma_{i\to j} := h_i(\alpha_j)$.
4. Having received all shares $\gamma_{1\to i}, \ldots, \gamma_{n\to i}$, party P_i locally computes $\beta_i = \sum_{j=1}^{n} \gamma_{j\to i}$ and sends β_i to all other parties.
5. Having received $\beta_1, \ldots, \beta_n$, party P_i finds the unique degree-$(n-1)$ polynomial $H(x)$ satisfying $H(\alpha_j) = \beta_j$ for every $j \in [n]$.

- **Output:** $H(0)$.

4.3 Static Security

We start by proving static security of π_{sum}. We note that static security holds even if OWP is simply a permutation that is not necessarily one way and can be inverted efficiently.

▶ **Lemma 6.** *Protocol π_{sum} statically $(n-1)$-privately computes f_{sum} with perfect semi-honest security and efficient simulation.*

Proof. Let $\mathcal{A}$ be a static semi-honest adversary and let $I \subset [n]$ of size $|I| < n$ denote the set of corrupted parties' indices. We construct a simulator $\mathcal{S}$ as follows:

1. The simulator initially receives auxiliary information z and the inputs of the corrupted parties $(g_i(x))_{i\in I}$. First, $\mathcal{S}$ sends $(g_i(x))_{i\in I}$ to the trusted party and receives back the output value y. Next, $\mathcal{S}$ invokes $\mathcal{A}$ on z and $(g_i(x))_{i\in I}$.
2. To simulate the first round, for every $j \notin I$, the simulator chooses a random $\boldsymbol{\rho}_j \leftarrow \mathbb{F}^{n-1}$ and computes $(r_{j,1}, \ldots, r_{j,n-1}) = \mathsf{OWP}(\boldsymbol{\rho}_j)$. Next, $\mathcal{S}$ chooses arbitrary polynomials $(\tilde{g}_j(x))_{j\notin I}$, each of degree at most $n-1$, under the constraint that

   $$\sum_{j\notin I} \tilde{g}_j(0) = y - \sum_{i\in I} g_i(0),$$

   and defines for every $j \notin I$

   $$h_j(x) = \tilde{g}_j(x) + r_{j,1}x + \ldots + r_{j,n-1}x^{n-1}.$$

   Finally, for every $j \notin I$ and $i \in I$, the simulator sends $\gamma_{j\to i} := h_j(\alpha_i)$ to $\mathcal{A}$ as the message from an honest P_j to a corrupted P_i, and receives the messages $\gamma_{i\to j}$ from $\mathcal{A}$ as the message from a corrupted P_i to an honest P_j.
3. To simulate the second round, for every $j \notin I$, the simulator computes $\beta_j := \sum_{k=1}^{n} \gamma_{k\to j}$ and sends β_j as the message from P_j. Next, $\mathcal{S}$ receives the message β_i from $\mathcal{A}$ on behalf of every corrupted P_i.
4. Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs, and halts.

Note that by construction, the simulator $\mathcal{S}$ runs in polynomial time in the running time of $\mathcal{A}$.

▷ Claim 7. $\left\{\mathrm{REAL}_{\pi_{\mathsf{sum}},I,\mathcal{A}}(\mathbf{x},z)\right\}_{(\mathbf{x},z)\in(\{0,1\}^*)^{n+1}} \equiv \left\{\mathrm{IDEAL}_{f_{\mathsf{sum}},I,\mathcal{S}}(\mathbf{x},z)\right\}_{(\mathbf{x},z)\in(\{0,1\}^*)^{n+1}}$.

Proof. Note that the only difference between the real execution of the protocol and the simulated execution in the ideal world is the construction of the constant terms of the polynomials $(h_j(x))_{j\notin I}$:

- In the real protocol, these constant terms are the constant terms of the original inputs of the honest parties $(g_j(x))_{j\notin I}$. In this case, by the linearity of Shamir's secret sharing and by the correctness of the interpolation, it holds that the output equals $\sum_{i=1}^{n} g_i(0)$.

- In the simulated execution, these constant terms are the constant terms of the polynomials $(\tilde{g}_j(x))_{j\notin I}$. These polynomials are generated under the constraint that $\sum_{j\notin I}\tilde{g}_j(0) = y - \sum_{i\in I} g_i(0)$, where y is computed by the trusted party to be $\sum_{i=1}^{n} g_i(0)$. It follows that

$$\sum_{j\notin I}\tilde{g}_j(0) = \sum_{j\notin I} g_j(0).$$

The claim now follows by the perfect privacy of Shamir's secret sharing. ◁

This concludes the proof of Lemma 6. ◀

4.4 Inefficient Adaptive Security

By the construction of the static simulator it is clear that the simulation is straight-line and black-box; further, since we consider a semi-honest adversary that in particular does not change the corrupted parties' inputs, "round zero" (the beginning of the protocol) can be set as the committal round. Therefore, we can use [9] to derive the following corollary.

▶ **Corollary 8.** *Protocol π_{sum} adaptively $(n-1)$-privately computes f_{sum} with perfect semi-honest security.*

4.5 No Efficient Adaptive Simulator

We proceed to prove that an efficient adaptive simulator can be used to invert the one-way permutation.

▶ **Lemma 9.** *Assume that* OWP *is a one-way permutation. Then, Protocol π_{sum} does* not *adaptively $(n-1)$-privately computes f_{sum} with perfect semi-honest security and efficient simulation.*

Proof. Let λ denote the security parameter, i.e., the one-way permutation is defined as $\mathsf{OWP} : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$. For simplicity, assume that $\lambda = (n-1)\log(n+1)$ for some n such that $n+1$ is a power of 2 (this holds without loss of generality, since the security of the OWP holds for all sufficiently large λ's). We consider the n-party functionality f_{sum} that is defined with respect to the field $\mathbb{F} = \mathbb{F}_{2^\ell}$ where $\ell = \log(n+1)$; then, indeed, $|\mathbb{F}| > n$ as required and OWP induces a permutation over $\mathbb{F}^{n-1}$.

Defining the adversary and the environment. Consider the following adaptive, semi-honest, $(n-1)$-limited adversary $\mathcal{A}$ and the environment $\mathcal{Z}$ for π_{sum}.

1. The environment does not send any auxiliary information to the adversary at the beginning.
2. The adversary $\mathcal{A}$ starts by corrupting the parties $P_3,\ldots,P_n$ and learns their inputs $(g_3(x),\ldots,g_n(x))$ and random tapes $(\boldsymbol{\rho}_3,\ldots,\boldsymbol{\rho}_n)$. The environment does not send any auxiliary information to the adversary for these corruptions.
3. Next, $\mathcal{A}$ receives first-round messages $\gamma_{1\to 3},\ldots,\gamma_{1\to n}$ from P_1 and $\gamma_{2\to 3},\ldots,\gamma_{2\to n}$ from P_2.
4. The adversary completes the execution honestly and outputs its view.
5. The environment corrupts P_1 in the PEC phase and learns (amongst other things) its input $g_1(x)$, randomness $\boldsymbol{\rho}_1$, and the values $(r_{1,1},\ldots,r_{1,n-1})$ computed in Step 1 of π_{sum}.
6. The environment checks whether $(r_{1,1},\ldots,r_{1,n-1}) = \mathsf{OWP}(\boldsymbol{\rho}_1)$; if so it outputs real and otherwise ideal.

The corresponding adaptive simulator. By the assumed security of π_{sum}, there exists an efficient adaptive simulator $\mathcal{S}$ for $\mathcal{A}$ and $\mathcal{Z}$. Note that by construction, the adversary $\mathcal{A}$ runs in polynomial time with respect to the parameter λ; hence, $\mathcal{S}$ is PPT with respect to λ. We will use $\mathcal{S}$ to construct a PPT inverter $\mathcal{D}$ for the OWP.

Constructing the inverter from the simulator. The inverter $\mathcal{D}$ receives as input the challenge $y^* \in \{0,1\}^\lambda$. We will consider y^* as element of $\mathbb{F}^{n-1}$, i.e., $y^* = (r_1^*, \ldots, r_{n-1}^*) \in \mathbb{F}^{n-1}$. The inverter $\mathcal{D}$ proceeds as follows:

1. $\mathcal{D}$ invokes the simulator $\mathcal{S}$ and emulates the ideal computation of f_{sum} toward $\mathcal{S}$; initially, $\mathcal{D}$ sends nothing as the auxiliary information to $\mathcal{S}$.
2. When $\mathcal{S}$ corrupts the parties $P_3, \ldots, P_n$ in the emulated ideal computation, $\mathcal{D}$ chooses arbitrary polynomials $(g_3(x), \ldots, g_n(x))$ of degree at most $n-1$ over $\mathbb{F}$ and hands $g_i(x)$ to $\mathcal{S}$ as the input value of P_i; $\mathcal{D}$ sends nothing as the auxiliary information to $\mathcal{S}$.
3. When $\mathcal{S}$ sends inputs to the trusted party on behalf of the corrupted parties, $\mathcal{D}$ responds with an arbitrary output value $y \in \mathbb{F}$.
4. Once $\mathcal{S}$ generates its output, containing the view of $\mathcal{A}$, the inverter $\mathcal{D}$ extracts the first-round messages that each corrupted party received from the honest party P_1, denoted $\gamma_{1\to 3}, \ldots, \gamma_{1\to n}$.
5. $\mathcal{D}$ samples two field elements $\gamma_1, \gamma_2 \leftarrow \mathbb{F}$ and interpolates the unique polynomial h of degree $n-1$ satisfying the constraints:

   $$h(\alpha_1) = \gamma_1, \qquad h(\alpha_2) = \gamma_2, \qquad h(\alpha_i) = \gamma_{1\to i} \text{ for } i \in \{3, \ldots, n\}.$$

   Next, $\mathcal{D}$ computes $g_1(x) = h(x) - \sum_{k=1}^{n-1} r_k^* x^k$.
6. $\mathcal{D}$ sends a "corrupt P_1" request to $\mathcal{S}$ during the PEC phase. When $\mathcal{S}$ corrupts P_1 in the emulated ideal computation, $\mathcal{D}$ sets the input of P_1 to be $g_1(x)$. Next, $\mathcal{S}$ responds with the view of P_1, containing the content of its random tape $\boldsymbol{\rho}_1$.
7. $\mathcal{D}$ outputs $\boldsymbol{\rho}_1$.

Efficiently inverting the OWP. First, notice that by construction, the running time of the inverter $\mathcal{D}$ is polynomial with respect to its input y^* and to the running time of $\mathcal{S}$; therefore, $\mathcal{D}$ is PPT with respect to the parameter λ. We proceed to show that the success probability of $\mathcal{D}$ in inverting a random challenge $y^* \leftarrow \{0,1\}^\lambda$ is non-negligible.

▷ Claim 10. $\Pr_{y^* \leftarrow \mathbb{F}^{n-1}}\left[\mathsf{OWP}(\mathcal{D}(y^*)) = y^*\right] = \frac{1}{|\mathbb{F}|^2}$.

Proof. In our proof, we do not assume any specific *behavior* of the adaptive simulator $\mathcal{S}$ (i.e., we do not know *how* the simulator generates its output messages). However, based on the assumed perfect security of the protocol, the adaptive simulator $\mathcal{S}$ operates according to the following interface:

1. $\mathcal{S}$ corrupts parties $P_3, \ldots, P_n$ and learns their inputs $(g_3(x), \ldots, g_n(x))$.
2. $\mathcal{S}$ sends the inputs $(g_3(x), \ldots, g_n(x))$ to the trusted party and obtains the output value y.
3. $\mathcal{S}$ generates the simulated view of the adversary, which in particular contains the messages $\gamma_{1\to 3}, \ldots, \gamma_{1\to n}$ from P_1 to parties $P_3, \ldots, P_n$.
4. $\mathcal{S}$ corrupts party P_1 and learns its input $g_1(x)$.
5. $\mathcal{S}$ outputs the view of P_1, including its random tape $\boldsymbol{\rho}_1$ and values $(r_{1,1}, \ldots, r_{1,n-1}) = \mathsf{OWP}(\boldsymbol{\rho}_1)$, such that the polynomial $h_1(x) = g_1(x) + \sum_{k=1}^{n-1} r_{1,k} x^k$ satisfies $h_1(\alpha_i) = \gamma_{1\to i}$ for $i \in \{3, \ldots, n\}$.

Recall that in Step 5 of the construction of the inverter, $\mathcal{D}$ samples $\gamma_1, \gamma_2 \leftarrow \mathbb{F}$ and interpolates the polynomial $h(x)$ that satisfies the following constraints:

$$h(\alpha_1) = \gamma_1, \qquad h(\alpha_2) = \gamma_2, \qquad h(\alpha_i) = \gamma_{1 \to i} \text{ for } i \in \{3, \ldots, n\} \ .$$

First, for every choice of $\gamma_1, \gamma_2 \in \mathbb{F}$ there exists a unique polynomial of degree $n-1$ over $\mathbb{F}$ satisfying these constraints. In the other direction, every polynomial $h'(x)$ of degree at most $n-1$ over $\mathbb{F}$, satisfying $h'(\alpha_i) = \gamma_{1 \to i}$ for $i \in \{3, \ldots, n\}$, induces two points $\gamma_1 = h'(\alpha_1)$ and $\gamma_2 = h'(\alpha_2)$ in $\mathbb{F}$.

It follows that the inverter succeeds in inverting y^* if and only if $h_1(\alpha_1) = \gamma_1$ and $h_1(\alpha_2) = \gamma_2$ (where $h_1(x)$ is the polynomial defined by $\mathcal{S}$). Indeed, in this case $h_1(x) = h(x)$, which means that $h_1(x) - g_1(x) = h(x) - g_1(x)$, i.e., $\sum_{k=1}^{n-1} r_{1,k} x^k = \sum_{k=1}^{n-1} r_k^* x^k$, and finally $(r_{1,1}, \ldots, r_{1,n-1}) = (r_1^*, \ldots, r_{n-1}^*)$. Since γ_1 and γ_2 are sampled uniformly at random from $\mathbb{F}$, this happens with probability $1/|\mathbb{F}|^2$. ◁

Finally, note that $y^* \in \mathbb{F}^{n-1}$, i.e., $y^* \in \{0,1\}^{(n-1)\log(n+1)} = \{0,1\}^{\lambda}$. However, the inverting probability is $1/|\mathbb{F}|^2 = 1/2^{2\log(n+1)} = 1/(n+1)^2$, which is non-negligible in λ. We conclude that $\mathcal{D}$ is PPT in λ and succeeds in inverting OWP with non-negligible probability. This contradicts the assumption that OWP is a one-way permutation. ◀

References

1. Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly-secure multiparty computation. IACR Cryptol. ePrint Arch., 2011. URL: `http://eprint.iacr.org/2011/136`.
2. Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 30(1):58–151, 2017.
3. Michael Backes, Jörn Müller-Quade, and Dominique Unruh. On the necessity of rewinding in secure multiparty computation. In *Proceedings of the Fourth Theory of Cryptography Conference, TCC 2007*, pages 157–173, 2007.
4. Donald Beaver and Stuart Haber. Cryptographic protocols provably secure against dynamic adversaries. In *Advances in Cryptology – EUROCRYPT 1992*, pages 307–323, 1992.
5. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 1988.
6. Elette Boyle, Ran Cohen, Deepesh Data, and Pavel Hubáček. Must the communication graph of MPC protocols be an expander? In *Advances in Cryptology – CRYPTO 2018, part III*, pages 243–272, 2018.
7. Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
8. Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.
9. Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. Adaptive versus non-adaptive security of multi-party protocols. *Journal of Cryptology*, 17(3):153–207, 2004.
10. Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 639–648, 1996.
11. David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19, 1988.

12 Ran Cohen, Juan A. Garay, and Vassilis Zikas. Completeness theorems for adaptively secure broadcast. IACR Cryptol. ePrint Arch., 2021. URL: http://eprint.iacr.org/2021/775.

13 Ran Cohen, Abhi Shelat, and Daniel Wichs. Adaptively secure MPC with sublinear communication complexity. In *Advances in Cryptology – CRYPTO 2019, part II*, pages 30–60, 2019.

14 Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology – EUROCRYPT 1999*, pages 311–326, 1999.

15 Ivan Damgård and Jesper Buus Nielsen. Adaptive versus static security in the UC model. In *Proceedings of the 8th International Conference on Provable Security (ProvSec)*, Lecture Notes in Computer Science, pages 10–28, 2014.

16 Yevgeniy Dodis and Silvio Micali. Parallel reducibility for information-theoretically secure computation. In *Advances in Cryptology – CRYPTO 2000*, pages 74–92, 2000.

17 Juan A. Garay, Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. The price of low communication in secure multi-party computation. In *Advances in Cryptology – CRYPTO 2017, part I*, pages 420–446, 2017.

18 Sanjam Garg and Amit Sahai. Adaptively secure multi-party computation with dishonest majority. In *Advances in Cryptology – CRYPTO 2012*, pages 105–123, 2012.

19 Oded Goldreich. *Foundations of Cryptography – VOLUME 2: Basic Applications*. Cambridge University Press, 2004.

20 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 218–229, 1987.

21 Martin Hirt, Chen-Da Liu-Zhang, and Ueli Maurer. Adaptive security of multi-party protocols, revisited. In *Proceedings of the 19th Theory of Cryptography Conference, TCC 2021, part I*, pages 686–716, 2021.

22 Martin Hirt and Vassilis Zikas. Adaptively secure broadcast. In *Advances in Cryptology – EUROCRYPT 2010*, pages 466–485, 2010.

23 Jonathan Katz, Aishwarya Thiruvengadam, and Hong-Sheng Zhou. Feasibility and infeasibility of adaptively secure fully homomorphic encryption. In *Proceedings of the 16th International Conference on the Theory and Practice of Public-Key Cryptography (PKC)*, pages 14–31, 2013.

24 Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. Information-theoretically secure protocols and security under composition. *SIAM Journal on Computing*, 39(5):2090–2112, 2010.

25 Ralf Küsters, Max Tuengerthal, and Daniel Rausch. The IITM model: A simple and expressive model for universal composability. *Journal of Cryptology*, 33(4):1461–1584, 2020.

26 Huijia Lin, Tianren Liu, and Hoeteck Wee. Information-theoretic 2-round MPC without round collapsing: Adaptive security, and more. In *Proceedings of the 18th Theory of Cryptography Conference, TCC 2020, part II*, pages 502–531, 2020.

27 Yehuda Lindell and Hila Zarosim. Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. *Journal of Cryptology*, 24(4):761–799, 2011.

28 Silvio Micali and Phillip Rogaway. Secure computation (abstract). In *Advances in Cryptology – CRYPTO 1991*, pages 392–404, 1991.

29 Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology – CRYPTO 2002*, pages 111–126, 2002.

30 Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 73–85, 1989.

31 Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, 1982.

Tight Estimate of the Local Leakage Resilience of the Additive Secret-Sharing Scheme & Its Consequences

Hemanta K. Maji
Department of Computer Science,
Purdue University, West Lafayette, IN, USA

Hai H. Nguyen
Department of Computer Science,
Purdue University, West Lafayette, IN, USA

Anat Paskin-Cherniavsky
Department of Computer Science,
Ariel University, Israel

Tom Suad
Department of Computer Science,
Ariel University, Israel

Mingyuan Wang
Department of EECS,
University of California Berkeley, CA, USA

Xiuyu Ye
Department of Computer Science,
Purdue University, West Lafayette, IN, USA

Albert Yu
Department of Computer Science,
Purdue University, West Lafayette, IN, USA

Abstract

Innovative side-channel attacks have repeatedly exposed the secrets of cryptosystems. Benhamouda, Degwekar, Ishai, and Rabin (CRYPTO–2018) introduced local leakage resilience of secret-sharing schemes to study some of these vulnerabilities. In this framework, the objective is to characterize the unintended information revelation about the secret by obtaining independent leakage from each secret share. This work accurately quantifies the vulnerability of the additive secret-sharing scheme to local leakage attacks and its consequences for other secret-sharing schemes.

Consider the additive secret-sharing scheme over a prime field among k parties, where the secret shares are stored in their natural binary representation, requiring λ bits – the security parameter. We prove that the reconstruction threshold $k = \omega(\log \lambda)$ is necessary to protect against local physical-bit probing attacks, improving the previous $\omega(\log \lambda / \log\log \lambda)$ lower bound. This result is a consequence of accurately determining the distinguishing advantage of the "parity-of-parity" physical-bit local leakage attack proposed by Maji, Nguyen, Paskin-Cherniavsky, Suad, and Wang (EUROCRYPT–2021). Our lower bound is optimal because the additive secret-sharing scheme is perfectly secure against any $(k-1)$-bit (global) leakage and (statistically) secure against (arbitrary) one-bit local leakage attacks when $k = \omega(\log \lambda)$.

Any physical-bit local leakage attack extends to (1) physical-bit local leakage attacks on the Shamir secret-sharing scheme with adversarially-chosen evaluation places, and (2) local leakage attacks on the Massey secret-sharing scheme corresponding to any linear code. In particular, for Shamir's secret-sharing scheme, the reconstruction threshold $k = \omega(\log \lambda)$ is necessary when the number of parties is $n = \mathcal{O}(\lambda \log \lambda)$. Our analysis of the "parity-of-parity" attack's distinguishing advantage establishes it as the best-known local leakage attack in these scenarios.

Our work employs Fourier-analytic techniques to analyze the "parity-of-parity" attack on the additive secret-sharing scheme. We accurately estimate an exponential sum that captures the vulnerability of this secret-sharing scheme to the parity-of-parity attack, a quantity that is also closely related to the "discrepancy" of the Irwin-Hall probability distribution.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives; Security and privacy → Cryptanalysis and other attacks

Keywords and phrases leakage resilience, additive secret-sharing, Shamir's secret-sharing, physical-bit probing leakage attacks, Fourier analysis

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.16

Funding Hemanta K. Maji, Hai H. Nguyen, Mingyuan Wang, Xiuyu Ye, and Albert Yu are supported in part by an NSF CRII Award CNS–1566499, NSF SMALL Awards CNS–1618822 and CNS–2055605, the IARPA HECTOR project, MITRE Innovation Program Academic Cybersecurity Research Awards (2019–2020, 2020–2021), a Ross-Lynn Research Scholars Grant (2021–2022), a Purdue Research Foundation (PRF) Award (2017–2018), and The Center for Science of Information, an NSF Science and Technology Center, Cooperative Agreement CCF–0939370. Anat Paskin-Cherniavsky and Tom Suad are supported by the Ariel Cyber Innovation Center in conjunction with the Israel National Cyber directorate in the Prime Minister's Office. Mingyuan Wang is also supported in part by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation, and Visa Inc. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

1 Introduction

Innovative and sophisticated side-channel attacks, beginning with [13, 14], have repetitively exposed the secrets of cryptosystems. Over the last few decades, there have been extensive studies on the security and efficiency of cryptosystems against various models of potential attacks (refer to the excellent survey [12]).

Benhamouda, Degwekar, Ishai, and Rabin [2] recently introduced local leakage resilience of secret-sharing schemes to investigate some of these vulnerabilities (this primitive is also implicitly studied by Goyal and Kumar [6]). Leakage-resilient cryptography aims to provide provable security in the presence of known attacks and even unforeseen attacks. Secret-sharing schemes are crucial building blocks for nearly all threshold cryptography. In leakage-resilient secret-sharing, the objective is to characterize the unintended information revelation about the secret by obtaining independent leakage from each secret share. The secret-sharing scheme is *locally leakage-resilient* if the joint distribution of the leakage from every secret share is (statistically) independent of the secret.

Interestingly, the local leakage resilience of Shamir's secret-sharing scheme is closely related to the problem of repairing Reed-Solomon codes [8, 9, 21, 7, 3, 17]. To break the leakage-resilience of a secret-sharing scheme, the adversary does not need to reconstruct the whole secret; obtaining partial information to distinguish any two secrets is sufficient. For example, in a linear secret-sharing scheme over characteristic-two fields, a suitable one-bit leakage from each share determines the "least significant bit" of the secret. The adversary's objective is to leak as small and simple a leakage as possible to achieve as significant a distinguishing advantage as possible.

The *physical-bit leakage model* is a realistic (and analytically-tractable) leakage model where an adversary probes physical bits in the memory hardware [11, 10, 4]. In the context of local leakage resilience of secret-sharing schemes, parties store their secret shares in their natural *binary representation*. The adversary chooses a bounded number of positions to probe the memory hardware storing these secret shares. The adversary's objective is to use this leakage to obtain some partial information about the secret. If the adversary's view is statistically independent of the secret, the secret-sharing scheme is secure against the local leakage; an *indistinguishability-based definition* captures this intuition [2].

This work characterizes the vulnerability of the additive secret-sharing scheme to the "parity-of-parity" physical-bit local leakage attack proposed by [15]. Next, we explore the consequences of this result to the leakage resilience of other linear secret-sharing schemes (in particular, Shamir's secret-sharing scheme).

Summary of known attacks

Consider the *additive secret-sharing scheme* among k parties over a prime field. Benhamouda et al. [2] proposed a one-bit local leakage attack with a distinguishing advantage of $\geqslant 1/k^k$.[1] Recently, Maji et al. [15] proposed the "parity-of-parity" attack, where the secret shares are stored in their natural binary representation, and the attacker leaks the least significant bit from every secret share. Adams et al. [1] proved that the "parity-of-parity" attack has a *distinguishing advantage* $\geqslant (1/2^k \cdot k!) \approx (\mathrm{e}/2)^k/k^k$. Therefore, the threshold k must be $\omega(\log \lambda/\log\log \lambda)$ for the additive secret-sharing scheme to be secure, where λ is the security parameter. Since the physical-bit probing attack is a significantly weak leakage attack, their result poses a pressing threat to the secret-sharing scheme's security. Furthermore, a local leakage attack on the additive secret-sharing scheme extends to Shamir's secret-sharing schemes for adversarially-chosen evaluation places [15].

Using a probabilistic argument, Nielsen and Simkin [19] presented a leakage attack on Shamir's secret-sharing scheme. They showed the existence of a leakage function and a secret such that the leakage is consistent with the secret with a probability of at least $1/2$. Their attack requires $m \geqslant \frac{k \log p}{n-k}$ bits of leakage from *each* secret share, where n is the number of parties and k is the reconstruction threshold. This result is not applicable when, for example, the number of parties $n = k$, the reconstruction threshold.

Summary of our results

This work presents a tight analysis of the parity-of-parity attack (Figure 1). We prove that this attack has a distinguishing advantage of $\geqslant \frac{1}{2} \cdot (2/\pi)^k$, which, in turn, implies that the threshold k must be $\omega(\log \lambda)$ for the additive secret-sharing scheme to be secure. Observe that our result *qualitatively improves* the lower bounds of [2] and [1] while relying only on *physical-bit* local leakage.

Our result shows that the simplistic parity-of-parity physical-bit probing attack is asymptotically optimal. The distinguishing advantage of any local leakage attack (possibly performing more sophisticated leakages) cannot be significantly higher because Benhamouda et al. [2] proved that the distinguishing advantage of *any* one-bit local leakage attack on the additive secret-sharing scheme is $\leqslant 2.47 \cdot (2/\pi)^k$. Furthermore, due to the $(k-1)$ independence of the additive secret-sharing scheme, any (global) $(k-1)$ bits of leakage has *no advantage* in distinguishing any two secrets.

Maji et al. [15] and Adams et al. [1] showed that any physical-bit local leakage attack extends to a physical-bit leakage attack on Shamir's secret-sharing scheme with adversarially-chosen evaluation places. Previously, the best-known distinguishing advantage was $\geqslant 1/(2^k \cdot k!)$ [1], where k is the reconstruction threshold of Shamir's secret-sharing scheme. Our work improves this lower bound to $\geqslant \frac{1}{2} \cdot (2/\pi)^k$, which implies that $k = \omega(\log \lambda)$ is necessary for security against physical-bit local leakage attacks.

This attack also translates into a local leakage attack on the Massey secret-sharing scheme corresponding to any linear code (refer to Appendix C for a definition); for example, Shamir's secret-sharing scheme with arbitrary evaluation places. Before our work, to ensure local leakage-resilience, the lower bound on the reconstruction threshold of Shamir's secret-sharing

[1] This attack performs a computation on the entire secret share and leaks one bit of information from it. We emphasize that this attack is *not* a physical-bit attack.

scheme was (1) $k = \omega(\log \lambda / \log\log \lambda)$, if $n = \mathcal{O}(\lambda \log \lambda / \log\log \lambda)$ [1], and (2) $k \geqslant n/(\lambda+1)$, if $n = \omega(\lambda \log \lambda / \log\log \lambda)$ [19]. Our results improve the lower bound to $k = \omega(\log \lambda)$ when the number of parties $n = \mathcal{O}(\lambda \log \lambda)$.

Technically, we obtain our lower bound through a Fourier-analytic approach and an accurate estimation of an appropriate exponential sum. As a consequence of our result, we also improve the bound on the "discrepancy" of the Irwin-Hall probability distribution, a fundamental property of any real-valued probability distribution proposed in [15].

2 Our Contribution

We begin with some notation to facilitate an overview of our results.

Secret-sharing schemes and local leakage resilience

Fix a prime field F of order p. The elements of F are naturally represented as λ-bit binary strings corresponding to the elements $\{0, 1, \ldots, p-1\}$, where $2^{\lambda-1} \leqslant p < 2^{\lambda}$. Fix a linear secret-sharing scheme over F among n parties with a reconstruction threshold k. Note that the secret and the secret shares are all elements of F. The number of bits in the representation of the secret and the secret shares is the security parameter λ.

Our work considers a (static) adversary who obtains $m = 1$ physical-bit leakage from each secret share. A one-bit physical-bit leakage function $\tau = (\tau_1, \tau_2, \ldots, \tau_n)$ is a collection of functions $\tau_i \colon F \to \{0,1\}$ such that, on input $x \in F$, function τ_i outputs the ℓ_i-th physical-bit of x for some $1 \leqslant \ell_i \leqslant \lambda$, for all $1 \leqslant i \leqslant n$. For instance, $\ell_i = 1$ refers to the least significant bit and $\ell_i = \lambda$ refers to the most significant bit. Let $\tau(s)$ be the joint distribution of the leakage function τ over the sample space $\{0,1\}^n$ defined by the experiment: (1) sample random secret shares $(s_1, s_2, \ldots, s_n) \in F^n$ for the secret $s \in F$ and (2) output the leakage $(\tau_1(s_1), \tau_2(s_2), \ldots, \tau_n(s_n))$.

A secret-sharing scheme is *ε-locally leakage-resilient* against one physical-bit probing attacks, if, for any pair of secrets $s^{(0)}, s^{(1)} \in F$, the leakage distributions $\tau(s^{(0)})$ and $\tau(s^{(1)})$ have statistical distance $\leqslant \varepsilon$. As per convention, we want to ensure that the parameter ε decays faster than any inverse-polynomial in the security parameter λ, represented as $\varepsilon = \mathsf{negl}(\lambda)$.

Additive secret-sharing scheme

Consider the *additive secret-sharing scheme* with k parties over a finite field F (possibly of composite order). For a secret $s \in F$, this secret-sharing scheme chooses random secret shares $s_1, \ldots, s_k \in F$ such that $s_1 + \cdots + s_k = s$. We assume that if F is a prime field, parties store the secret shares $s_1, \ldots, s_k$ in their natural binary representation. However, if F is a composite order field of characteristic p, then the secret shares are stored as a vector of F_p elements, where every F_p element is represented in its natural binary representation.[2]

Parity-of-parity attack

Maji et al. [15] introduced the *parity-of-parity* local physical-bit leakage attack on the additive secret sharing scheme over fields of arbitrary characteristic. If F is a prime field (of an odd order), then the attacker leaks the least significant bit of each secret share, i.e., the leaked bit

[2] The degree-a extension of the field F_p, i.e. the finite field F_{p^a}, is isomorphic to $F_p[X]/\pi(X)$, where $\pi(X)$ is a degree-a irreducible polynomial. Therefore, every element $s \in F_{p^a}$ has a natural $(s_1, \ldots, s_a) \in F_p^a$ representation, each element in turn has a λ-bit binary representation.

indicates whether the secret share $s_i \in \{0, 2, \ldots, |F| - 1\}$ or $s_i \in \{1, 3, \ldots, |F| - 2\}$. Finally, the attack predicts the parity of the secret using the parity of these leaked parities. If F is a degree-a extension of the prime field F_p, then every secret share $s_i \in F$ has equivalent representation $(s_{i,1}, \ldots, s_{i,a}) \in F_p^a$. For some fixed index $j \in \{1, 2, \ldots, a\}$, the attacker leaks the parity of the element $s_{i,j}$ from the i-th secret share. Over extension fields, this attack predicts the parity of s_j, where the secret $s = (s_1, \ldots, s_a) \in F^a$.

For example, if F has characteristic 2, observe that the parity of the j-th coordinate of all the secret shares (as vectors in F_2) yields the j-th coordinate of the secret, which completely breaks the leakage-resilience of the additive secret-sharing scheme.

Adams et al. [1] proved that the advantage of this attacker is maximized when the secrets are $s^{(0)} = 0$ and $s^{(1)} = (p-1)/2$. Furthermore, they proved that the advantage of this attack is $\geqslant 1/(2^k \cdot k!)$.

Our results

Given ε and k, our objective is to identify whether there are two distinct secrets $s^{(0)}, s^{(1)} \in F$ such that the parity-of-parity attack has (at least) ε-advantage in distinguishing the secret shares that these secrets generate. Without loss of generality, assume that F is a prime field of order $\geqslant 2$, because the characteristic of the field determines the vulnerability of the additive secret-sharing scheme. We prove the following result.

▶ **Theorem 1.** *Consider the additive secret sharing scheme with k parties over the prime field F. There exist two secrets $s^{(0)}, s^{(1)} \in F$ such that the parity-of-parity attack has ε-advantage in distinguishing the secret shares of $s^{(0)}$ from the secret shares of $s^{(1)}$, where*

$$\varepsilon \geqslant \frac{1}{2} \cdot \left(\frac{2}{\pi}\right)^k.$$

▶ Remark 2. Our bound captures the intuition that, for a fixed k, with increasing p, the insecurity of the additive secret-sharing scheme reduces. As $p \to \infty$, the insecurity tends (from above) to the limit $\frac{1}{2}\left(\frac{2}{\pi}\right)^k$, a constant. Intuitively, the "most-secure additive secret-sharing scheme" corresponds to the case when the order of the finite field is an "infinitely large prime p." This phenomenon and an exponential lower bound in k were originally conjectured in [15] based on empirical evidence (refer to Figure 1). Recently, [1] made partial progress towards non-trivially lower-bounding the advantage of the parity-of-parity attack by proving $\varepsilon \geqslant 1/(2^k \cdot (k-1)!) - (3(k-1)^2 + 1)/p$.[3] However, this insecurity bound is increasing in p; thus, their work could not substantiate this conjecture. Our result substantiates the empirical evidence of [15] and positively resolves their conjecture.

Our lower bound is (asymptotically) *optimal* and also proves the optimality of the parity-of-parity attack in the following sense. Over prime fields, [2] proved that the additive secret-sharing scheme is $2.47 \cdot (2/\pi)^k$-secure against *any* local one-bit leakage attack (i.e., the leakage function $\tau_i : F \to \{0, 1\}$ is arbitrary and *need not be a physical-bit probing* leakage). Consequently, the reconstruction threshold of the additive secret-sharing scheme must satisfy $k = \omega(\log \lambda)$ to be leakage-resilient to one physical-bit leakage from every secret share. Our result improves the previous best-known lower bound of $k = \omega(\log \lambda / \log \log \lambda)$ for additive secret-sharing schemes using the leakage attack presented in [2, 1].

[3] This bound proves that the discrepancy of the Irwin-Hall distribution is non-zero and is an integer multiple of $1/2^k(k-1)!$. Next, it transfers this lower bound to the distinguishing advantage of the parity-of-parity attacker against the additive secret-sharing scheme over finite prime fields.

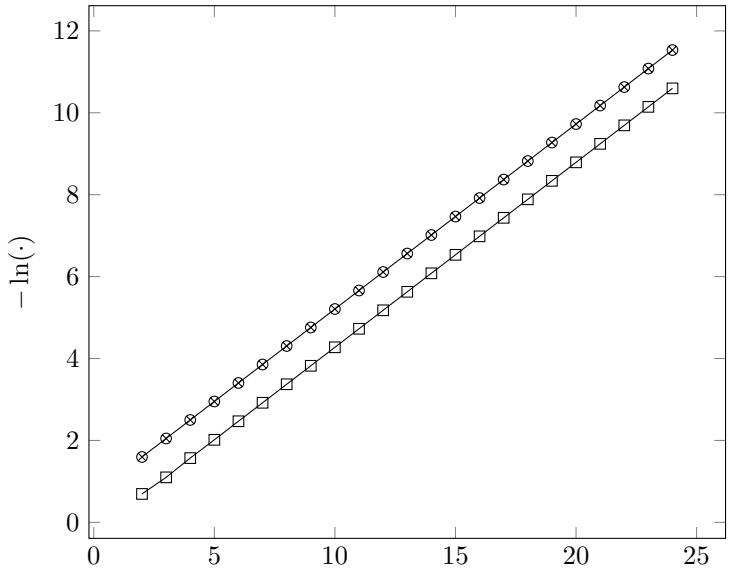

Figure 1 The horizontal axis represents the number of shares k in the additive secret-sharing scheme. The vertical axis represents the $-\ln(\cdot)$ of the distinguishing advantage of the parity-of-parity attack introduced by Maji et al. [15]. The squared points represent the empirically computed value for small k over a large enough field F as presented in [15] . The circled points represents the lower bound we prove in this work.

To better bound the effectiveness of the parity-of-parity attack, Maji et al. [15] proposed the notion: *discrepancy of the Irwin-Hall distribution.* The first Irwin-Hall distribution IH_1 is the uniform distribution over $[0, 1)$. The i-th Irwin-Hall distribution IH_i is the convolution of the $(i-1)$-th Irwin-Hall distribution IH_{i-1} with the uniform distribution over $[0, 1)$. The discrepancy of the k-th Irwin-Hall distribution $\mathsf{disc}(k)$ is defined as

$$\mathsf{disc}(k) := \sup_y \left| \int_{-\infty}^{\infty} (-1)^{\lceil x-y \rceil} \cdot \mathsf{IH}_k(x)\, \mathrm{d}x \right|. \tag{1}$$

Appendix A provides a pictorial illustration of this notion. We refer the readers to [1] for more discussion on why this measure represents the effectiveness of the parity-of-parity attack. In particular, they proved that $\mathsf{disc}(k-1)$ is $\Theta\big(k^2/p\big)$-close to the effectiveness of the parity-of-parity attack on additive secret-sharing among k parties over prime field F of order p. Consequently, our result implies that the discrepancy of the Irwin-Hall distribution is also exponential in k, improving upon the previous best lower bound $1/(2^k \cdot k!)$ [1].

▶ **Corollary 3.** *For $k \in \{1, 2, \dots\}$, let $\mathsf{disc}(k)$ represents the discrepancy of the k-th Irwin-Hall distribution. Then, it holds that* $\mathsf{disc}(k) = \Theta\left(\left(\frac{2}{\pi}\right)^k\right)$.

Finally, motivated by applications in leakage-resilient secure computation, observe that our result extends to a stronger adversary who obtains some secret shares in the clear and performs local leakage attacks on the remaining secret shares.[4] We have the following theorem for such insider attackers.

[4] For example, in secret-sharing based multi-party computation protocols [5], an adversary can corrupt some parties and get their entire secret shares in the clear. Additionally, the adversary may perform leakage attacks on the secret shares of the remaining honest parties.

▶ **Corollary 4.** *Consider the additive secret sharing scheme with k parties over the prime field F. Suppose a more general adversary obtains θ secret shares and gets the least significant bit from other shares. Then, there exist two secrets such that the adversary's advantage of distinguishing the two secrets is at least $\frac{1}{2} \cdot \left(\frac{2}{\pi}\right)^{k-\theta}$.*

Shamir's secret-sharing scheme. Let $\mathsf{ShamirSS}(n, k, \vec{X})$ represent Shamir's secret-sharing scheme among n parties, reconstruction threshold k, and evaluation places $\vec{X} = (X_1, \ldots, X_n)$. The evaluation places $X_1, \ldots, X_n$ are distinct elements of F^*. Let $s \in F$ be the secret. The secret-sharing scheme picks a random polynomial $f(Z) \in F[Z]/Z^k$ conditioned on the fact that $f(0) = s$. For $i \in \{1, \ldots, n\}$, the i-th secret share is $f(X_i)$.

Maji et al. [15] show a set of evaluation places such that one could perform the parity-of-parity attack on the first k secret shares to get the same advantage as the attack on the additive secret-sharing scheme. Hence, our result implies the following theorem.

▶ **Theorem 5.** *Let F be a prime field of order p such that $p = 1 \mod k$. Let $\alpha \in F^*$ be such that $\{\alpha, \alpha^2, \ldots, \alpha^k = 1\} \subseteq F^*$ is the set of k roots of the equation $Z^k - 1 = 0$. Suppose there exists $\beta \in F^*$ such that $\{\beta\alpha, \beta\alpha^2, \ldots, \beta\alpha^k = \beta\}$ is a subset of the evaluation places $\vec{X}$. One can perform the parity-of-parity attack on the secret shares corresponding to evaluation places $\{\beta\alpha, \beta\alpha^2, \ldots, \beta\alpha^k = \beta\}$ to get a distinguishing advantage of $\geqslant \frac{1}{2} \cdot (2/\pi)^k$. Therefore, if $\mathsf{ShamirSS}(n, k, \vec{X})$ is $\mathsf{negl}(\lambda)$-locally leakage-resilient secret-sharing scheme against one physical-bit leakage from each secret share, then it must be the case that $k = \omega(\log \lambda)$.*

Extension to arbitrary local leakage attacks. The following result extends the parity-of-parity attack to a local leakage attack to Massey secret-sharing scheme and Shamir's secret-sharing scheme. Given a linear code $C \subseteq F^{(n+1)}$, the Massey secret-sharing scheme [18] corresponding to a code C, is defined as follows. For a secret $s \in F$, one samples a random codeword $(s_0, s_1, \ldots, s_n) \in C$ such that $s_0 = s$. For $i \in \{1, 2, \ldots, n\}$, the i^{th} secret share is $s_i \in F$.

▶ **Theorem 6.** *Let F be a prime order field. For any Massey secret-sharing scheme corresponding to an $[n+1, k]_F$-linear code C or any $\mathsf{ShamirSS}(n, k, \vec{X})$ with arbitrary evaluation places $\vec{X}$ over F, there is a one-bit local leakage attack such that the distinguishing advantage is at least $\frac{1}{2} \cdot \left(\frac{2}{\pi}\right)^k$.*

To see why our results imply this theorem, assume the secret could be reconstructed from the first k shares as $s = \sum_{i=1}^{k} \alpha_i \cdot s_i$, where $\alpha_1, \ldots, \alpha_k$ are some fixed field elements (determined by the $[n+1, k]_F$ linear code). One can leak the least significant bit of $\alpha_i \cdot s_i$ from the i-th secret share s_i. It is easy to see that the advantage of this adversary is identical to the advantage of the parity-of-parity attack on the additive secret-sharing scheme.

However, we clarify that this leakage is *not* the physical-bit leakage because the local leakage involves field multiplication. As a consequence of Theorem 6, we obtain a similar lower bound for the reconstruction threshold against *arbitrary* local leakage.

▶ **Corollary 7.** *Fix $n, k \in \mathbb{N}$ and a prime order field F. If the Massey secret-sharing scheme corresponding to an $[n+1, k]_F$-linear code or $\mathsf{ShamirSS}(n, k, \vec{X})$ over F with arbitrary evaluation places $\vec{X}$ is $\mathsf{negl}(\lambda)$-locally leakage-resilient against one-bit local leakage, then it must hold that $k = \omega(\log(\lambda))$.*

We clarify that a physical-bit leakage analog for this result does not hold. [15] proved that with close-to-one-probability the $\mathsf{ShamirSS}(n, k, \vec{X})$ with *random evaluation places* $\vec{X}$ is $\mathsf{negl}(\lambda)$-locally leakage-resilient even for $k = 2$. Our result shows that the lower bound on the

reconstruction threshold of Shamir's secret-sharing scheme is $k = \omega(\log \lambda)$ when the number of parties is $n = \mathcal{O}(\lambda \log \lambda)$. Before our work, the lower bound was (1) $k = \omega(\log \lambda / \log\log \lambda)$, if $n = \mathcal{O}(\lambda \log \lambda / \log\log \lambda)$ [1], and (2) $k \geqslant n/(\lambda + 1)$, if $n = \omega(\lambda \log \lambda / \log\log \lambda)$ [19].

▶ Remark 8. Our analysis also extends to the thermal noise leakage model in which the adversary obtains a noisy version of the leakage bits as considered in [1]. In this model, instead of obtaining the leakage $\tau(s) = (\tau_1(s_1), \tau_2(s_2), \ldots, \tau_n(s_n))$, the adversary receives a noisy leakage $\tau'(s) = (\tau'_1(s_1), \tau'_2(s_2), \ldots, \tau'_n(s_n))$, where every $\tau'_i(s_i)$ is ρ_i-correlated with $\tau_i(s_i)$.[5] The distinguishing advantage is reduced by a (multiplicative) factor of $\rho = \rho_1 \rho_2 \cdots \rho_n \leqslant 1$. For instance, the distinguishing advantage of the parity-of-parity attack in the presence of $(\rho_1, \ldots, \rho_n)$ noise would be

$$\rho_1 \cdot \rho_2 \cdots \rho_n \cdot \frac{1}{2} \cdot \left(\frac{2}{\pi}\right)^n .$$

This observation follows from facts of convolution.

3 Technical Overview

This section presents an overview of our technical approach. Let F be a prime field of order p. Consider the additive secret-sharing scheme over F. Let τ be the leakage attack that leaks the least significant bit from every share.

We refer the readers to Section 4.1 for an introduction to Fourier analysis. By the Fourier-analytic approach from prior works [2, 15, 16], for any two secrets $s^{(0)}$ and $s^{(1)}$, we have

$$\mathsf{SD}\left(\tau\left(s^{(0)}\right), \tau\left(s^{(1)}\right)\right) = \frac{1}{2} \cdot \sum_{\ell \in \{0,1\}^n} \left| \sum_{\alpha \in F^*} \left(\prod_{i=1}^{n} \widehat{\mathbb{1}_{\ell_i}}(\alpha) \right) \left(\omega^{\alpha \cdot s^{(0)}} - \omega^{\alpha \cdot s^{(1)}} \right) \right|,$$

where $\omega = \exp(2\pi\imath/p)$ is the p^{th} root of unity. Furthermore, $\mathbb{1}_0$ is the indicator function for the set $S_0 := \{0, 2, \ldots, p-1\}$ and, similarly, $\mathbb{1}_1$ is the indicator function for the set $S_1 := \{1, 3, \ldots, p-2\}$. That is, S_b is the set of field elements whose least significant bit is b.

Note that the above expression is an *identity*. Our first observation is that, for any $\ell \in \{0,1\}^n$, the *magnitude* of the expression

$$U(\alpha) := \prod_{i=1}^{n} \widehat{\mathbb{1}_{\ell_i}}(\alpha)$$

is exponentially decaying as α goes from the central points $\frac{p-1}{2}$ and $\frac{p+1}{2}$ to the end points 1 and $p-1$ (refer to Figure 2). Informally, it holds that

$$|U(\alpha)| \approx \left(\frac{2}{\pi}\right)^n \cdot \left(\frac{1}{|2\alpha - p|}\right)^n .$$

For the magnitude of the other term

$$V(\alpha) := \left(\omega^{\alpha \cdot s^{(0)}} - \omega^{\alpha \cdot s^{(1)}}\right),$$

[5] For any $\rho \in [0,1]$, a bit b is ρ-correlated with another bit b' if $b = b'$ with probability ρ, and b is an independent and uniformly random bit with probability $1 - \rho$.

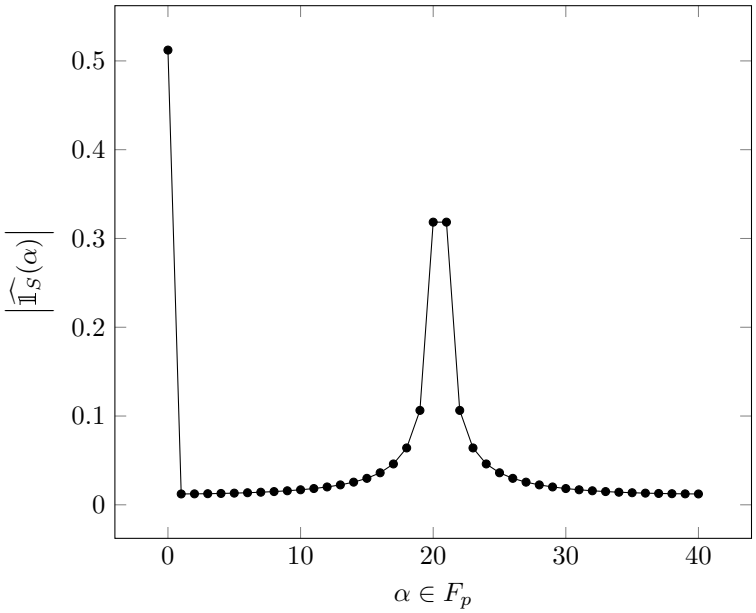

Figure 2 For the representative case of $p = 41$, the Fourier spectrum of the indicator function $\mathbb{1}_S$, where $S = \{0, 2, \ldots, 40\} \subset F_p$ is the subset of all "even elements".

we use the naive triangle inequality to upper bound it by 2 for the *non-central terms* (i.e., $\alpha \neq \frac{p-1}{2}, \frac{p+1}{2}$). And we argue that there exists two secrets $s^{(0)}$ and $s^{(1)}$ such that $V\left(\frac{p-1}{2}\right)$ and $V\left(\frac{p+1}{2}\right)$ are large (e.g., $\geqslant 3/2$).

Together, these observations enable us to lower bound the statistical distance by (approximately) the magnitude of the dominant term $U\left(\frac{p-1}{2}\right)$ and $U\left(\frac{p+1}{2}\right)$, which are $\Theta\left(\left(\frac{2}{\pi}\right)^n\right)$.

Finally, observe that the two distributions $\tau\left(s^{(0)}\right)$ and $\tau\left(s^{(1)}\right)$ are $(n-1)$-indistinguishable. That is, these two distributions restricted to any proper subset of their coordinates are identical. Therefore, by standard techniques, parity is the optimal distinguisher for these two distributions (we provide a formal discussion on this in Appendix B). Consequently, the parity-of-parity attack [15] has an distinguishing advantage of $\Theta\left(\left(\frac{2}{\pi}\right)^n\right)$.

▶ Remark 9. Due to the form of our lower bound expression, it is tempting to naïvely argue that the advantage of the parity-of-parity attack correctly predicting the secret's parity is (some form of a) "k-fold convolution of a $(2/\pi)$-biased predictor." This intuition is (seriously) technically flawed. The least significant bit of the first $(k-1)$ secret shares are each $1/p$-biased and independent of the secret.

4 Analysis of the Parity-of-parity Attack on Additive Secret-sharing Schemes

Maji et al. [15] proposed the following parity-of-parity attack. Suppose the field elements are stored in their natural binary representation. The adversary leaks the least significant bit (LSB) as the local leakage of every secret share. Finally, the adversary outputs the parity of the LSB from every secret share as the prediction of the secret. Adams et al. [1] proved that the distinguishing advantage of this adversary is at least $\Omega(1/n!)$. In this section, we shall present a tight analysis of this attack. In particular, we shall show that the distinguishing advantage is $\exp(-\mathcal{O}(n))$.

This lower bound we prove is tight up to a small constant, as Benhamouda et al. [2] prove that the distinguishing advantage of the adversary is upper-bounded by $\left(\frac{2}{\pi}\right)^{n-2}$. Note that the upper bound of [2] holds for any local leakage attack on the additive secret-sharing scheme. Therefore, our result also demonstrates that *the "parity-of-parity" attack is the optimal attack*.

Formally, let $\mathsf{AddSS}(s)$ represent the distribution of the additive secret shares of the secret s. That is, $\mathsf{AddSS}(s) = (s_1, \ldots, s_n)$ is sampled uniformly at random conditioned on that $s_1 + s_2 + \cdots + s_n = s$. For any $x \in F$, let $\mathsf{lsb}(x)$ represent the least significant bit of x.[6]

Let τ represent the local leakage function that leaks the LSB of every secret share. That is, $\tau(\mathsf{AddSS}(s)) := \Big(\mathsf{lsb}(s_1), \mathsf{lsb}(s_2), \ldots, \mathsf{lsb}(s_n)\Big)$. We prove the following theorem.

▶ **Theorem 10.** *There exists two secrets $s^{(0)}$ and $s^{(1)}$ such that*

$$\mathsf{SD}\left(\tau(\mathsf{AddSS}(s^{(0)}))\ ,\ \tau(\mathsf{AddSS}(s^{(1)}))\right) \geqslant \frac{1}{2} \cdot \left(\frac{2}{\pi}\right)^n.$$

In particular, to ensure that the adversary has a negligible distinguishing advantage $\mathsf{negl}(\lambda)$, it must hold that $n = \omega(\log \lambda)$.

▶ Remark 11 (On the characteristics of the field). We emphasize that our lower bound holds for arbitrarily large characteristics. Intuitively, as the characteristic of the field increases, one expects the advantage of the adversary to decrease. However, our result shows that the advantage of the adversary is guaranteed to be higher than $\frac{1}{2} \cdot \left(\frac{2}{\pi}\right)^n$even when the characteristic of the field tends to infinity.

Finally, observe that $\tau(\mathsf{AddSS}(s^{(0)}))$ and $\tau(\mathsf{AddSS}(s^{(1)}))$ are $(n-1)$-indistinguishable distributions since the additive secret sharing is $(n-1)$-private. By standard techniques in Fourier analysis, the parity of all the bits is the best distinguisher (up to a small constant) for any two $(n-1)$-indistinguishable distributions. For completeness, we provide formal proof of this in Appendix B. This observation, together with the theorem, implies the optimality of the parity-of-parity attack.

Surprisingly, our proof of Theorem 10 is based on Fourier analysis. Typically, Fourier analytic approach is employed to *upper bound* the distinguishing advantage of the adversary. However, we shall use it to prove a *lower bound* result.

We start by introducing some notations and basics of Fourier analysis that suffice for our purposes. Next, we present the proof of Theorem 10.

4.1 Preliminaries on Fourier Analysis

Let F be a prime field of order p. For any complex number $x \in \mathbb{C}$, let $\overline{x}$ represent its conjugate. For any two functions $f, g \colon F \to \mathbb{C}$, their *inner product* is

$$\langle f, g \rangle := \frac{1}{p} \cdot \sum_{x \in F} f(x) \cdot \overline{g(x)}.$$

[6] This section restricts our discussion to a field F of prime order. If E is an degree t extension field of the field F, then every element α of E can be seen as a polynomial $a_{t-1}X^{t-1} + \cdots + a_1 X + a_0$ in $F[X]$. We shall call a_0 the least significant symbol of α. Observe that, for an additive secret sharing of the secret s over E, the least significant symbol of every secret share forms an additive secret sharing of the least significant symbol of s over F. Therefore, the result for prime order fields naturally extends to composite order fields when the attacker leaks the LSB of the least significant symbol of every share.

Let $\omega = \exp(2\pi\imath/p)$ be the p^{th} root of unity. For all $\alpha \in F$, the function χ_α is defined to be

$$\chi_\alpha(x) := \omega^{\alpha \cdot x},$$

and the respective Fourier coefficient $\widehat{f}(\alpha)$ is defined as

$$\widehat{f}(\alpha) := \langle f, \chi_\alpha \rangle .$$

Our proof relies on the following lemma. We refer the readers to [2] for a proof.

▶ **Lemma 12** (Poisson Summation Formula). *Let $C \subseteq F^n$ be a linear code with dual code $C^\perp$. For all $i \in \{1, 2, \ldots, n\}$, let $f_i \colon F \to \mathbb{C}$ be an arbitrary function. It holds that*

$$\mathop{\mathrm{E}}_{\vec{x} \leftarrow C} \left[\prod_{i=1}^{n} f_i(x_i) \right] = \sum_{\vec{y} \in C^\perp} \left(\prod_{i=1}^{n} \widehat{f_i}(y_i) \right).$$

The following claims will also be useful, which follows directly from the definition.

▶ **Claim 13.** *Let $S, T \subseteq F$ be a partition of F. For all $\alpha \in F$,*

$$\widehat{\mathbb{1}_S}(\alpha) = -\widehat{\mathbb{1}_T}(\alpha).$$

▶ **Claim 14.** *For all $S \subseteq F$ and $x \in F$, it holds that*

$$\widehat{\mathbb{1}_{x+S}}(\alpha) = \widehat{\mathbb{1}_S}(\alpha) \cdot \omega^{-\alpha \cdot x}.$$

The *statistical distance* (a.k.a, total variation distance) between two distributions P and Q over a finite sample space Ω is defined as $\mathsf{SD}(P, Q) = \frac{1}{2} \sum_{x \in \Omega} |P(x) - Q(x)|$. For any code $C \subseteq F^n$ and any vector $x \in F^n$, we define $x + C := \{x + c \colon c \in C\}$.

4.2 Proof of Theorem 10

We start by introducing some notations and facts. Define a bipartition of F as

$$S_0 := \{0, 2, \ldots, p-1\} \qquad \text{and} \qquad S_1 := \{1, 3, \ldots, p-2\}.$$

That is, S_b is the set of field elements on which the LSB function will output b.

▶ **Claim 15.** *For $\alpha \in F^*$, it holds that*

$$\widehat{\mathbb{1}_{S_0}}(\alpha) = \frac{1}{2p} \cdot \frac{1}{\cos(\pi\alpha/p)} \cdot \omega^{\alpha/2}, \text{ and } \widehat{\mathbb{1}_{S_1}}(\alpha) = -\frac{1}{2p} \cdot \frac{1}{\cos(\pi\alpha/p)} \cdot \omega^{\alpha/2}.$$

Furthermore,

$$\left|\widehat{\mathbb{1}_{S_0}}(\alpha)\right| = \left|\widehat{\mathbb{1}_{S_1}}(\alpha)\right| = \frac{1}{2p} \cdot \frac{1}{|\cos(\pi\alpha/p)|}.$$

Proof of Claim 15. By definition, we have

$$\begin{aligned}\widehat{\mathbb{1}_{S_0}}(\alpha) = \langle \mathbb{1}_{S_0}, \chi_\alpha \rangle &= \frac{1}{p} \sum_{x \in S_0} \omega^{-\alpha \cdot x} = \frac{1}{p} \cdot \sum_{j=0}^{(p-1)/2} \omega^{-\alpha \cdot (2j)} \\ &= \frac{1}{p} \cdot \frac{1 - \omega^{-(2\alpha) \cdot (p+1)/2}}{1 - \omega^{-2\alpha}} = \frac{1}{p} \cdot \frac{1 - \omega^{-\alpha}}{1 - \omega^{-2\alpha}}.\end{aligned}$$

One could verify that $1-\omega^{-\alpha} = 2\sin(\pi\alpha/p)\cdot\omega^{\frac{p}{4}-\frac{\alpha}{2}}$. Hence,

$$\widehat{\mathbb{1}_{S_0}}(\alpha) = \frac{1}{p}\cdot\frac{2\sin(\pi\alpha/p)\cdot\omega^{\frac{p}{4}-\frac{\alpha}{2}}}{2\sin(\pi(2\alpha)/p)\cdot\omega^{\frac{p}{4}-\frac{2\alpha}{2}}} = \frac{1}{2p}\cdot\frac{1}{\cos(\pi\alpha/p)}\cdot\omega^{\alpha/2}.$$

By Claim 13, we have

$$\widehat{\mathbb{1}_{S_1}}(\alpha) = -\frac{1}{2p}\cdot\frac{1}{\cos(\pi\alpha/p)}\cdot\omega^{\alpha/2}.$$

Finally, since $\left|w^{\alpha/2}\right| = 1$, it is easy to see that

$$\left|\widehat{\mathbb{1}_{S_0}}(\alpha)\right| = \left|\widehat{\mathbb{1}_{S_1}}(\alpha)\right| = \frac{1}{2p}\cdot\frac{1}{|\cos(\pi\alpha/p)|},$$

which completes the proof. ◀

Let C be the parity code. That is, $(c_1,\ldots,c_n)\in C$ if $c_1+\cdots+c_n=0$. The secret shares of a secret s is uniformly distributed over the set $(s,0,\ldots,0)+C$; or equivalently, it is uniformly distributed over $(n^{-1}\cdot s,\ldots,n^{-1}\cdot s)+C$. For ease of presentation, we use the latter form. Additionally, the dual code of C, denoted by $C^\perp$, is simply the repetition code, i.e., $C^\perp = \{(\alpha,\ldots,\alpha)\colon \alpha\in F\}$.

We are ready to prove Theorem 10 as follows. We shall abuse notation and write $\mathbb{1}_b$ for $\mathbb{1}_{S_b}$. Observe that

$$\begin{aligned}
&\mathsf{SD}\left(\tau\left(\mathsf{AddSS}(s^{(0)})\right),\tau\left(\mathsf{AddSS}(s^{(1)})\right)\right)\\
=&\frac{1}{2}\cdot\sum_{\ell\in\{0,1\}^n}\left|\mathop{\mathrm{E}}_{\vec{x}\leftarrow C}\left[\prod_{i=1}^n\mathbb{1}_{\ell_i}(x_i+n^{-1}\cdot s^{(0)})\right]-\mathop{\mathrm{E}}_{\vec{x}\leftarrow C}\left[\prod_{i=1}^n\mathbb{1}_{\ell_i}(x_i+n^{-1}\cdot s^{(1)})\right]\right| && \text{(By definition of SD)}\\
=&\frac{1}{2}\cdot\sum_{\ell\in\{0,1\}^n}\left|\sum_{\vec{y}\in C^\perp}\left(\prod_{i=1}^n\widehat{\mathbb{1}_{\ell_i}}(y_i+n^{-1}\cdot s^{(0)})\right)-\sum_{\vec{y}\in C^\perp}\left(\prod_{i=1}^n\widehat{\mathbb{1}_{\ell_i}}(y_i+n^{-1}\cdot s^{(1)})\right)\right| && \text{(Lemma 12)}\\
=&\frac{1}{2}\cdot\sum_{\ell\in\{0,1\}^n}\left|\sum_{\alpha\in F}\left(\prod_{i=1}^n\widehat{\mathbb{1}_{\ell_i}}(\alpha+n^{-1}\cdot s^{(0)})\right)-\sum_{\alpha\in F}\left(\prod_{i=1}^n\widehat{\mathbb{1}_{\ell_i}}(\alpha+n^{-1}\cdot s^{(1)})\right)\right| && \text{(By the definition of } C^\perp)\\
=&\frac{1}{2}\cdot\sum_{\ell\in\{0,1\}^n}\left|\sum_{\alpha\in F}\left(\prod_{i=1}^n\widehat{\mathbb{1}_{\ell_i}}(\alpha)\right)\left(\omega^{\alpha\cdot s^{(0)}}-\omega^{\alpha\cdot s^{(1)}}\right)\right| && \text{(Claim 14)}\\
=&\frac{1}{2}\cdot\sum_{\ell\in\{0,1\}^n}\left|\sum_{\alpha\in F^*}\left(\prod_{i=1}^n\left((-1)^{\ell_i}\frac{1}{2p}\cdot\frac{1}{\cos(\pi\alpha/p)}\cdot\omega^{\alpha/2}\right)\right)\left(\omega^{\alpha\cdot s^{(0)}}-\omega^{\alpha\cdot s^{(1)}}\right)\right| && \text{(Claim 15)}\\
=&2^{n-1}\cdot\left|\sum_{\alpha\in F^*}\left(\frac{1}{2p}\cdot\frac{1}{\cos(\pi\alpha/p)}\cdot\omega^{\alpha/2}\right)^n\left(\omega^{\alpha\cdot s^{(0)}}-\omega^{\alpha\cdot s^{(1)}}\right)\right| && \text{(Identity transformation)}
\end{aligned}$$

Note that the proof so far has not used any inequalities. The expression above is identical to the statistical distance. For brevity, let us define

$$U(\alpha) := \left(\frac{1}{2p}\cdot\frac{1}{\cos(\pi\alpha/p)}\cdot\omega^{\alpha/2}\right)^n, \text{ and } V(\alpha) := \omega^{\alpha\cdot s^{(0)}}-\omega^{\alpha\cdot s^{(1)}}.$$

Additionally, let $W(\alpha) := U(\alpha) \cdot V(\alpha)$. Intuitively, we shall prove that the magnitude of $\sum_{\alpha \in F^*} W(\alpha)$ is approximately the magnitude of its leading term $W((p-1)/2)$ and $W((p+1)/2)$. In particular, we prove the following claims.

▶ **Claim 16.** *There exists a universal constant $\mu \geqslant 3/2$ and two secrets $s^{(0)}, s^{(1)} \in F$ such that*

$$\left| W\left(\frac{p-1}{2}\right) + W\left(\frac{p+1}{2}\right) \right| \geqslant \mu \cdot \pi^{-n}.$$

▶ **Claim 17.** *For all secrets $s^{(0)}, s^{(1)}$, we have*

$$\left| \sum_{\alpha \in F^* \setminus \{\frac{p-1}{2}, \frac{p+1}{2}\}} W(\alpha) \right| \leqslant \exp(-\Theta(n)) \cdot \pi^{-n}.$$

Using Claim 16 and Claim 17, the proof of the Theorem 10 follows from the fact that

$$\begin{aligned} \mathsf{SD}\left(\tau\left(\mathsf{AddSS}(s^{(0)})\right), \tau\left(\mathsf{AddSS}(s^{(1)})\right)\right) &\geqslant 2^{n-1} \cdot (\mu - \exp(-\Theta(n))) \cdot \pi^{-n}, \\ &\geqslant 2^{n-1} \cdot \left(\frac{3}{2} - \mathrm{o}(1)\right) \cdot \pi^{-n}, \\ &\geqslant \frac{1}{2} \cdot 1 \cdot \left(\frac{2}{\pi}\right)^n \qquad \text{(for large enough } n.\text{)} \end{aligned}$$

Consequently, it suffices to prove Claim 16 and Claim 17 to complete the proof of Theorem 10.

Proof of Claim 16. Observe that

$$\begin{aligned} W\left(\frac{p-1}{2}\right) &= \left(\frac{1}{2p} \cdot \frac{1}{\cos\left(\pi \cdot \frac{p-1}{2p}\right)} \cdot \omega^{\frac{p-1}{4}}\right)^n \cdot V\left(\frac{p-1}{2}\right) \\ &= \left(\frac{1}{2p} \cdot \frac{1}{\sin\left(\pi \cdot \frac{1}{2p}\right)}\right)^n \cdot \omega^{n \cdot \frac{p-1}{4}} \cdot V\left(\frac{p-1}{2}\right) \end{aligned}$$

and

$$\begin{aligned} W\left(\frac{p+1}{2}\right) &= \left(\frac{1}{2p} \cdot \frac{1}{\cos\left(\pi \cdot \frac{p+1}{2p}\right)} \cdot \omega^{\frac{p+1}{4}}\right)^n \cdot V\left(\frac{p+1}{2}\right) \\ &= \left(\frac{1}{2p} \cdot \frac{1}{\sin\left(\pi \cdot \frac{1}{2p}\right)}\right)^n \cdot (-1)^n \cdot \omega^{n \cdot \frac{p+1}{4}} \cdot V\left(\frac{p+1}{2}\right) \end{aligned}$$

Therefore,

$$\begin{aligned} &\left| W\left(\frac{p-1}{2}\right) + W\left(\frac{p+1}{2}\right) \right| \\ &= \left| \left(\frac{1}{2p} \cdot \frac{1}{\sin\left(\pi \cdot \frac{1}{2p}\right)}\right)^n \right| \cdot \left| \omega^{n \cdot \frac{p-1}{4}} \cdot V\left(\frac{p-1}{2}\right) + (-1)^n \cdot \omega^{n \cdot \frac{p+1}{4}} \cdot V\left(\frac{p+1}{2}\right) \right| \\ &= \left| \left(\frac{1}{2p} \cdot \frac{1}{\sin\left(\pi \cdot \frac{1}{2p}\right)}\right)^n \right| \cdot \left| V\left(\frac{p-1}{2}\right) + (-1)^n \cdot \omega^{\frac{n}{2}} \cdot V\left(\frac{p+1}{2}\right) \right| \end{aligned}$$

Note that $x \cdot \sin(1/x)$ is strictly increasing as x increases and tends to 1 as $x \to \infty$.[7] Therefore,

$$\left|W\left(\frac{p-1}{2}\right) + W\left(\frac{p+1}{2}\right)\right| \geqslant \pi^{-n} \cdot \left|V\left(\frac{p-1}{2}\right) + (-1)^n \cdot \omega^{\frac{n}{2}} \cdot V\left(\frac{p+1}{2}\right)\right|.$$

It remains to prove that there exist secrets $s^{(0)}$ and $s^{(1)}$ such that $V\left(\frac{p-1}{2}\right)$ and $(-1)^n \cdot \omega^{\frac{n}{2}} \cdot V\left(\frac{p+1}{2}\right)$ does not cancel each other to be too small. More formally, for any p and n, we shall show that there exist a universal constant μ and secrets $s^{(0)}$ and $s^{(1)}$ such that

$$\left|\left(\omega^{\frac{p-1}{2}\cdot s^{(0)}} - \omega^{\frac{p-1}{2}\cdot s^{(1)}}\right) + (-1)^n \cdot \omega^{\frac{n}{2}} \cdot \left(\omega^{\frac{p+1}{2}\cdot s^{(0)}} - \omega^{\frac{p+1}{2}\cdot s^{(1)}}\right)\right| \geqslant \mu.$$

Let $f(s^{(0)})$ (resp., $g(s^{(1)})$) denote the terms involving $s^{(0)}$ (resp., $s^{(1)}$) in the above expression. And we are interested in $\left|f(s^{(0)}) + g(s^{(1)})\right|$. Observe that

$$\sum_{s^{(1)} \in F} g(s^{(1)}) = 0.$$

Therefore, we have

$$\begin{aligned}\max_{s^{(1)}} \left|f(s^{(0)}) + g(s^{(1)})\right| &\geqslant \frac{1}{p} \sum_{s^{(1)} \in F} \left|f(s^{(0)}) + g(s^{(1)})\right| \\ &\geqslant \frac{1}{p} \left|\sum_{s^{(1)} \in F} \left(f(s^{(0)}) + g(s^{(1)})\right)\right| = \left|f(s^{(0)})\right|.\end{aligned}$$

Hence, it suffices to show that there exists an $s^{(0)}$ such that $\left|f(s^{(0)})\right|$ is sufficiently large. That is,

$$\max_{s^{(0)}} \left|\omega^{\frac{p-1}{2}\cdot s^{(0)}} + (-1)^n \cdot \omega^{\frac{n}{2}} \cdot \omega^{\frac{p+1}{2}\cdot s^{(0)}}\right| \geqslant \mu,$$

which is equivalent to

$$\max_{s^{(0)}} \left|1 + (-1)^n \cdot \omega^{\frac{n}{2}} \cdot \omega^{s^{(0)}}\right| \geqslant \mu.$$

It is easy to see that the phase of $\omega^{s^{(0)}}$ could be an arbitrary multiple of $2\pi/p$. Hence, there must exist an $s^{(0)}$ such that the above expression has magnitude $\geqslant 3/2$.[8] This completes the proof. ◀

[7] Intuitively, the advantage of the adversary decreases as the characteristic of the field increases.
[8] In fact, as p tends to infinity, the maximum gets arbitrarily close to 2.

Proof of Claim 17. By a simple triangle inequality, we have $|V(\alpha)| \leqslant 2$. Hence,

$$\begin{aligned}
&\left|\sum_{\alpha\in F^*\setminus\{\frac{p-1}{2},\frac{p+1}{2}\}} W(\alpha)\right| \\
&\leqslant \sum_{\alpha\in F^*\setminus\{\frac{p-1}{2},\frac{p+1}{2}\}} |W(\alpha)| && \text{(Triangle inequality)} \\
&\leqslant 2\cdot \sum_{\alpha\in F^*\setminus\{\frac{p-1}{2},\frac{p+1}{2}\}} |U(\alpha)| && \text{(Triangle inequality)} \\
&= 2\cdot \sum_{\alpha\in F^*\setminus\{\frac{p-1}{2},\frac{p+1}{2}\}} \left|\frac{1}{2p}\cdot\frac{1}{\cos(\pi\alpha/p)}\right|^n && \text{(Identity transformation)} \\
&= 4\cdot \sum_{j=1}^{(p-3)/2} \left(\frac{1}{2p}\cdot\frac{1}{\cos(\pi j/p)}\right)^n && \text{(Identity transformation)} \\
&= 4\cdot \sum_{j=1}^{(p-3)/2} \left(\frac{1}{2p}\cdot\frac{1}{\sin(\pi(p-2j)/(2p))}\right)^n && \text{(Identity transformation)}
\end{aligned}$$

Observe that $\sin(x) \geqslant x/2$ for every $x \in (0, \pi/2)$. Hence,

$$\begin{aligned}
&\left|\sum_{\alpha\in F^*\setminus\{\frac{p-1}{2},\frac{p+1}{2}\}} W(\alpha)\right| \\
&\leqslant 4\cdot \sum_{j=1}^{(p-3)/2} \left(\frac{1}{2p}\cdot\frac{2}{\pi(p-2j)/(2p)}\right)^n \\
&= \pi^{-n}\cdot 4\cdot \sum_{j=1}^{(p-3)/2} \left(\frac{2}{p-2j}\right)^n \\
&\leqslant \pi^{-n}\cdot 4\cdot \left(\left(\frac{2}{3}\right)^n + \int_3^\infty \left(\frac{1}{x}\right)^n \mathrm{d}x\right) \\
&= \pi^{-n}\cdot 4\cdot \left(\left(\frac{2}{3}\right)^n + \frac{1}{n+1}\left(\frac{1}{3}\right)^{n+1}\right) \\
&= \pi^{-n}\cdot \exp(-\Theta(n)).
\end{aligned}$$

This completes the proof. ◀

References

1 Donald Q. Adams, Hemanta K. Maji, Hai H. Nguyen, Minh L. Nguyen, Anat Paskin-Cherniavsky, Tom Suad, and Mingyuan Wang. Lower bounds for leakage-resilient secret sharing schemes against probing attacks. In *IEEE International Symposium on Information Theory ISIT 2021*, 2021. URL: `https://www.cs.purdue.edu/homes/hmaji/papers/AMNNPSW21.pdf`.

2 Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 531–561, Santa Barbara, CA, USA, August 19–23 2018. Springer, Heidelberg, Germany. `doi:10.1007/978-3-319-96884-1_18`.

3 Hoang Dau, Iwan M. Duursma, Han Mao Kiah, and Olgica Milenkovic. Repairing reed-solomon codes with multiple erasures. *IEEE Trans. Inf. Theory*, 64(10):6567–6582, 2018. doi:10.1109/TIT.2018.2827942.

4 Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440, Copenhagen, Denmark, May 11–15 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_24.

5 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27 1987. ACM Press. doi:10.1145/28395.28420.

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

7 Venkatesan Guruswami and Ankit Singh Rawat. MDS code constructions with small sub-packetization and near-optimal repair bandwidth. In Philip N. Klein, editor, *28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2109–2122, Barcelona, Spain, January 16–19 2017. ACM-SIAM. doi:10.1137/1.9781611974782.137.

8 Venkatesan Guruswami and Mary Wootters. Repairing reed-solomon codes. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 216–226, Cambridge, MA, USA, June 18–21 2016. ACM Press. doi:10.1145/2897518.2897525.

9 Venkatesan Guruswami and Mary Wootters. Repairing reed-solomon codes. *IEEE Trans. Inf. Theory*, 63(9):5684–5698, 2017. doi:10.1109/TIT.2017.2702660.

10 Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327, St. Petersburg, Russia, May 28 – June 1 2006. Springer, Heidelberg, Germany. doi:10.1007/11761679_19.

11 Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481, Santa Barbara, CA, USA, August 17–21 2003. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-45146-4_27.

12 Yael Tauman Kalai and Leonid Reyzin. A survey of leakage-resilient cryptography. Cryptology ePrint Archive, Report 2019/302, 2019. URL: https://eprint.iacr.org/2019/302.

13 Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Santa Barbara, CA, USA, August 18–22 1996. Springer, Heidelberg, Germany. doi:10.1007/3-540-68697-5_9.

14 Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Santa Barbara, CA, USA, August 15–19 1999. Springer, Heidelberg, Germany. doi:10.1007/3-540-48405-1_25.

15 Hemanta K. Maji, Hai H. Nguyen, Anat Paskin-Cherniavsky, Tom Suad, and Mingyuan Wang. Leakage-resilience of the shamir secret-sharing scheme against physical-bit leakages. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 344–374, Zagreb, Croatia, October 17–21 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-77886-6_12.

16 Hemanta K. Maji, Anat Paskin-Cherniavsky, Tom Suad, and Mingyuan Wang. Constructing locally leakage-resilient linear secret-sharing schemes. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 779–808, Virtual Event, August 16–20 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-84252-9_26.

17 Jay Mardia, Burak Bartan, and Mary Wootters. Repairing multiple failures for scalar MDS codes. *IEEE Trans. Inf. Theory*, 65(5):2661–2672, 2019. doi:10.1109/TIT.2018.2876542.

18 James L. Massey. Some applications of coding theory in cryptography. *Mat. Contemp*, 21(16):187–209, 2001.

19 Jesper Buus Nielsen and Mark Simkin. Lower bounds for leakage-resilient secret sharing. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 556–577, Zagreb, Croatia, May 10–14 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-45721-1_20.

20 Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

21 Itzhak Tamo, Min Ye, and Alexander Barg. Optimal repair of reed-solomon codes: Achieving the cut-set bound. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 216–227, Berkeley, CA, USA, October 15–17 2017. IEEE Computer Society Press. doi:10.1109/FOCS.2017.28.

A The Discrepancy of the Irwin-Hall Distribution

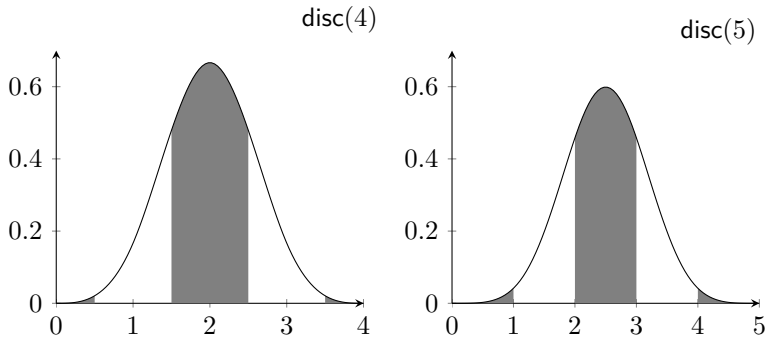

Figure 3 The plot of the fourth (left) and fifth (right) Irwin-Hall distribution. Intuitively, the discrepancy of the Irwin-Hall distribution is the difference between the total probability mass inside the black bands and the total probability mass outside the black bands. In particular, we are interested in the maximum difference as the black bands shift along the x-axis. Equation 1 provides a precise definition. This maximum difference is defined as the discrepancy of the k-th Irwin-Hall distribution, denoted by $\mathsf{disc}(k)$.

B On the Optimality of the Parity Distinguisher

Let $\mathcal{D}^{(0)}$ and $\mathcal{D}^{(1)}$ be two distributions over the universe $\{0,1\}^n$. Suppose $\mathcal{D}^{(0)}$ and $\mathcal{D}^{(1)}$ are $(n-1)$-indistinguishable.[9] That is, for any proper subset $S \subset \{1,2,\ldots,n\}$, we have

$$\mathsf{SD}\left(\left\{\begin{array}{l}\vec{x} \leftarrow \mathcal{D}^{(0)}\\ \text{Output } \vec{x}_S\end{array}\right\}, \left\{\begin{array}{l}\vec{x} \leftarrow \mathcal{D}^{(1)}\\ \text{Output } \vec{x}_S\end{array}\right\}\right) = 0.$$

[9] We do not use the term $(n-1)$-independent since the LSB of a uniformly random field element is not exactly uniform over $\{0,1\}$.

For a distribution $\mathcal{D}$ and any set $S \subseteq 1, 2, \ldots, n$, define the bias of $\mathcal{D}$ over S as

$$\mathsf{bias}(\mathcal{D}, S) := \mathop{\mathrm{E}}_{\vec{x} \leftarrow \mathcal{D}} \left[(-1)^{\sum_{i \in S} x_i} \right].$$

The following fact about the bias shall be useful. We refer the readers to [20] for a proof.

▶ **Lemma 18.**

$$\mathsf{SD}\left(\mathcal{D}^{(0)}, \mathcal{D}^{(1)}\right) \leqslant \frac{1}{2} \cdot \sqrt{\sum_{S \in \Omega} \left(\mathsf{bias}(\mathcal{D}^{(0)}, S) - \mathsf{bias}(\mathcal{D}^{(1)}, S)\right)^2},$$

where Ω *is the power set of* $\{1, 2, \ldots, n\}$.

Observe that $\mathcal{D}^{(0)}$ and $\mathcal{D}^{(1)}$ are $(n-1)$-indistinguishable implies that

$$\mathsf{bias}(\mathcal{D}^{(0)}, S) = \mathsf{bias}(\mathcal{D}^{(1)}, S)$$

for all proper subsets $S \subset \{1, 2, \ldots, n\}$.

Therefore, this lemma implies that

$$\mathsf{SD}\left(\mathcal{D}^{(0)}, \mathcal{D}^{(1)}\right) \leqslant \frac{1}{2} \cdot \left| \mathsf{bias}\left(\mathcal{D}^{(0)}, \{1, 2, \ldots, n\}\right) - \mathsf{bias}\left(\mathcal{D}^{(1)}, \{1, 2, \ldots, n\}\right) \right|.$$

This shows that the parity is the optimal distinguisher up to a constant as the right hand side is exactly the advantage of the parity distinguisher.

C Massey's Secret-sharing Schemes

For completeness, we recall Massey's Secret-sharing scheme. The following is taken verbatim from [16].

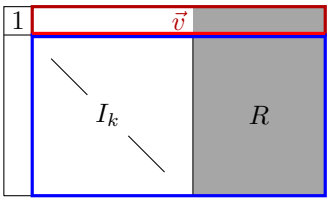

Figure 4 A pictorial summary of the generator matrix $G^+ = [I_{k+1} \mid P]$, where P is the shaded matrix. The indices of rows and columns of G^+ are $\{0, 1, \ldots, k\}$ and $\{0, 1, \ldots, n\}$, respectively. The (blue) matrix $G = [I_k \mid R]$ is a submatrix of G^+. In particular, the secret shares of secret $s = 0$ form the code $\langle G \rangle$. The (red) vector is $\vec{v}$. In particular, for any secret s, the secret shares of s form the affine subspace $s \cdot \vec{v} + \langle G \rangle$.

A *linear code* C (over the finite field F) of *length* $(n+1)$ and *rank* $(k+1)$ is a $(k+1)$-dimension vector subspace of F^{n+1}, referred to as an $[n+1, k+1]_F$-code. The *generator matrix* $G \in F^{(k+1)\times(n+1)}$ of an $[n+1, k+1]_F$ linear code C ensures that every element in C can be expressed as $\vec{x} \cdot G$, for an appropriate $\vec{x} \in F^{k+1}$. Given a generator matrix G, the row-span of G, i.e., the code generated by G, is represented by $\langle G \rangle$. A generator matrix G is in the *standard form* if $G = [I_{k+1} | P]$, where $I_{k+1} \in F^{(k+1)\times(k+1)}$ is the identity matrix and $P \in F^{(k+1)\times(n-k)}$ is the parity check matrix. In this work, we always assume that the generator matrices are in their standard form.

Massey Secret-sharing Schemes. Let $C \subseteq F^{n+1}$ be a linear code. Let $s \in F$ be a secret. The Massey secret-sharing scheme corresponding to C picks a random element $(s, s_1, \ldots, s_n) \in C$ to share the secret s. The secret shares of parties $1, \ldots, n$ are $s_1, \ldots, s_n$, respectively.

Recall that the set of all codewords of the linear code generated by the generator matrix $G^+ \in F^{(k+1)\times(n+1)}$ is

$$\left\{ \vec{y} : \vec{x} \in F^{k+1}, \vec{x} \cdot G^+ =: \vec{y} \right\} \subseteq F^{n+1}.$$

For such a generator matrix, its rows are indexed by $\{0, 1, \ldots, k\}$ and its columns are indexed by $\{0, 1, \ldots, n\}$. Let $s \in F$ be the secret. The secret-sharing scheme picks independent and uniformly random $r_1, \ldots, r_k \in F$. Let

$$(y_0, y_1, \ldots, y_n) := (s, r_1, \ldots, r_k) \cdot G^+.$$

Observe that $y_0 = s$ because the generator matrix G^+ is in the standard form. The secret shares for the parties $1, \ldots, n$ are $s_1 = y_1, s_2 = y_2, \ldots, s_n = y_n$, respectively. Observe that every party's secret share is an element of the field F. Of particular interest will be the set of all secret shares of the secret $s = 0$. Observe that the secret shares form an $[n, k]_F$-code that is $\langle G \rangle$, where $G = G^+_{\{1,\ldots,k\}\times\{1,\ldots,n\}}$. Note that the matrix G is also in the standard form. The secret shares of $s \in F^*$ form the affine space $s \cdot \vec{v} + \langle G \rangle$, where $\vec{v} = G^+_{0,\{1,\ldots,n\}}$. Refer to Figure 4 for a pictorial summary.

Suppose parties $i_1, \ldots, i_t \in \{1, \ldots, n\}$ come together to reconstruct the secret with their, respective, secret shares $s_{i_1}, \ldots, s_{i_t}$. Let $G^+_{*,i_1}, \ldots, G^+_{*,i_t} \in F^{(k+1)\times 1}$ represent the columns indexed by $i_1, \ldots, i_t \in \{1, \ldots, n\}$, respectively. If the column $G^+_{*,0} \in F^{(k+1)\times 1}$ lies in the span of $\left\{G^+_{*,i_1}, \ldots, G^+_{*,i_t}\right\}$ then these parties can reconstruct the secret s using a linear combination of their secret shares. If the column G^+_{*0} *does not* lie in the span of $\left\{G^+_{*,i_1}, \ldots, G^+_{*,i_t}\right\}$ then the secret remains *perfectly hidden* from these parties.

Information-Theoretic Distributed Point Functions

Elette Boyle
IDC Herzliya, Israel

Niv Gilboa
Ben-Gurion University of the Negev, Beer-Sheva, Israel

Yuval Ishai
Technion, Haifa, Israel

Victor I. Kolobov
Technion, Haifa, Israel

Abstract

A *distributed point function* (DPF) (Gilboa-Ishai, Eurocrypt 2014) is a cryptographic primitive that enables compressed additive secret-sharing of a secret weight-1 vector across two or more servers. DPFs support a wide range of cryptographic applications, including efficient private information retrieval, secure aggregation, and more. Up to now, the study of DPFs was restricted to the *computational* security setting, relying on one-way functions. This assumption is necessary in the case of a dishonest majority.

We present the first statistically private 3-server DPF for domain size N with subpolynomial key size $N^{o(1)}$. We also present a similar *perfectly* private 4-server DPF. Our constructions offer benefits over their computationally secure counterparts, beyond the superior security guarantee, including better computational complexity and better protocols for distributed key generation, all while having comparable communication complexity for moderate-sized parameters.

2012 ACM Subject Classification Theory of computation → Cryptographic primitives

Keywords and phrases Information-theoretic cryptography, homomorphic secret sharing, private information retrieval, secure multiparty computation

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.17

Funding *Elette Boyle*: Supported by AFOSR Award FA9550-21-1-0046, ERC Project HSS (852952), ERC Project NTSC (742754), and a Google Research Scholar Award.
Niv Gilboa: Supported by ISF grant 2951/20, ERC grant 876110, and a grant by the BGU Cyber Center.
Yuval Ishai: Supported by ERC Project NTSC (742754), BSF grant 2018393, and ISF grant 2774/20.
Victor I. Kolobov: Supported by ERC Project NTSC (742754) and ISF grant 2774/20.

1 Introduction

A *Distributed Point Function* (DPF) [28, 13] enables splitting any secret *point function* $f_{\alpha,\beta}$ (i.e., for which $f_{\alpha,\beta}(x) = \beta$ if $x = \alpha$, and 0 otherwise) into m succinctly described function shares f_i, that individually hide $f_{\alpha,\beta}$, and which support a simple additive per-input reconstruction $f_{\alpha,\beta}(x) = \sum_i f_i(x)$ over some fixed Abelian group. More concretely, each function share f_i is described by a *key* k_i such that with an appropriate evaluation algorithm Eval it holds that $\mathsf{Eval}(k_i, x) = f_i(x)$. In effect, this provides a compressed additive secret-sharing of a secret weight-1 vector across servers.

DPFs have a wide range of cryptographic applications, including Private Information Retrieval (PIR) [17, 16, 28], anonymous messaging systems [18, 35], secure aggregation and statistical analysis [13, 7], private set intersection [39, 22], secure computation for

RAM programs [23, 15] and programs with mixed-mode operations [14, 8], and recently pseudorandom correlation generators [9, 10, 11], with applications to secure computation and beyond.

As with many cryptographic notions, the security property of DPFs can be either *computational* (based on computational hardness assumptions), or *information theoretic*. The vast majority of attention to date has been placed in the two-server regime, where it is known that nontrivial DPFs require the existence of one-way functions [28, 12]. In turn, from one-way functions, efficient two-server DPF constructions have been demonstrated with small key size, which grows logarithmically with the domain size of $f_{\alpha,\beta}$ [28, 13].

However, as soon as one steps beyond two servers to an honest majority, the impossibility no longer holds, and the question of minimizing the key size of information-theoretic DPFs becomes wide open. Despite its neglect up to now, the regime of information-theoretically secure DPFs offers potential for application scenarios where information-theoretic security is desired (or required), as well as appealing potential for simplicity of constructions. Another important motivation for information-theoretic constructions is the possibility to avoid the limitations of current techniques for distributed key generation of computationally secure DPF [23].

1.1 Our Contribution

We initiate an investigation of information-theoretically secure DPFs (IT DPFs for short), focusing on the case of non-colluding servers (i.e., security threshold $t = 1$). While simple constructions based on Reed-Muller codes are implicit in the PIR literature [17], these have polynomial key size of $O(N^{1/(m-1)})$, where N is the domain size and m is the number of servers. In contrast, the new generation of PIR schemes [40, 25, 24, 4], which achieve sub-polynomial communication, do not directly give rise to standard DPFs. Instead, they imply a relaxed form of DPF in which the output is not shared additively. While this suffices for the PIR application, it does not suffice for most other applications of DPFs. Even in the PIR context, an additive representation is helpful for maximizing the download rate [26].

Our primary technical contribution is in bridging this gap. We obtain the first statistically private 3-server DPF for domain size N with subpolynomial key size $N^{o(1)}$. We also present a similar *perfectly* private 4-server DPF. Our constructions offer benefits over their computationally secure counterparts, beyond the superior security guarantee, including better computational complexity and potential for "MPC friendliness" in the sense of efficient distributed key generation, all while having comparable key size[1] for moderate-sized parameters.

We obtain the following main results:

▶ **Theorem 1** (4-server perfectly secure IT DPF, informal). *Let $p \geq 3$ be a prime and $s \geq 1$ an integer. There exists a perfectly secure 4-server DPF, for point functions with output group $\mathbb{Z}_{p^s}$ and key size $O\left(s \log(p) \cdot 2^{2p\sqrt{\log N \log\log N}}\right)$.*

[1] This assumes that β is taken from a small output group, such as $\mathbb{Z}_2$, which suffices for many applications of DPF. While in this work we focus on asymptotic efficiency and do not attempt to optimize concrete efficiency, our techniques can be applied to concretely efficient variants of the "matching vector" based PIR schemes on which we rely (see Table 2 in the full version of [31]). Unlike the Reed-Muller based 3-server PIR, these variants can be practical even for (sparse, virtual) database of size $\approx 2^{60}$, which arise in private keyword search applications.

▶ **Theorem 2** (3-server statistically-secure IT DPF, informal). *Let $p \geq 2$ be a prime. There exists a $2^{-\lambda}$-statistically secure 3-server DPF, for point functions with output group $\mathbb{Z}_p$ and key size $O\left(\lambda \log(p) \cdot 2^{k(p)\sqrt{\log N \log\log N}}\right)$ where $k(2) = 6$, $k(3) = 10$, and $k(p) = 2p$ if $p \geq 5$.*

Due to the prime p appearing in the exponent in the key size, Theorem 1 permits only groups of the form $\mathbb{Z}_{p^s}$ for small prime p (or products of such groups via CRT). The same is true for Theorem 2, except that there we further have the restriction of $s = 1$. However, for many applications of DPF (including both "reading" and "writing") an output group of $\mathbb{Z}_2$ suffices, in which case Theorem 2 gives an efficient construction. Moreover, in applications of DPF that require a group of a large characteristic (e.g., for aggregation [7] or weighted private set intersection [22]), Theorem 1 gives an efficient construction over $\mathbb{Z}_{3^s}$ for a sufficiently large s.

We further explore advantages of our IT DPF constructions over existing computationally secure constructions, beyond their stronger security guarantees. We explicitly demonstrate one such benefit: simplicity of distributed key generation. This relates to the procedure of two or more clients jointly executing the DPF key generation algorithm for an input point function that is *secret shared* across servers (i.e., where no client individually knows the secret $f_{\alpha,\beta}$). This "Distributed Gen" procedure is a crucial and costly part of important DPF-based applications. Distributed Gen protocols in the computational setting currently fall into one of two categories. They use either generic MPC machinery, which requires non-black box secure computation of cryptographic primitives such as PRGs, or tailored protocols [23] requiring computation that is proportional to the size of the input domain and a number of communication rounds that is logarithmic in that size. Moreover, there is no known approach for distributing the key generation of 2-server DPF in the *malicious* security setting that makes black-box use of a PRG, regardless of round complexity.

In contrast, the simpler structure of keys in IT DPFs implies the following:

▶ **Theorem 3** (Distributed key generation, informal). *There exist protocols for distributed generation of the keys required in Theorems 1 and 2 that are information-theoretically secure for $m \geq 3$ servers with one malicious corruption, or for 2PC in the OT-hybrid model, have computation and communication cost $\tilde{O}(h)$ for required key size h, and $O(\log h)$ rounds. Alternatively, settling for computational security, there are such* constant-round *protocols that only make a* black-box *use of a PRG.*

1.2 Overview of Techniques

Our information-theoretic (IT) DPF constructions are based on a related primitive, IT *private information retrieval* (PIR) [17]. A PIR scheme allows a client to retrieve a single bit from a database D of N bits, by communicating with $m \geq 2$ servers, such that no server learns the client's bit index. Multi-server PIR served as an original driving motivation behind the introduction of DPFs, as an m-server DPF directly yields an m-server PIR protocol. In this work, however, we study this connection in the other direction: building DPF from PIR.

Assuming the m-server IT PIR scheme satisfies that each server responds with a single bit to the client query, and that the client's reconstruction is additive, then in fact we obtain an IT DPF for the point function $f_{\alpha,1}$, by having the client query for index α, and the servers considering the database D_x which has the value 1 at index x, and 0 at all other indices. As we will see, some existing classes of IT PIR schemes fit into this framework, and thus yield IT DPF with similar communication. However, other categories of IT PIR constructions will require more work.

Known IT PIR schemes can be roughly classified into three generations. The first-generation schemes, originating in the work of [17], are based on Reed-Muller codes, and achieve communication complexity $N^{1/\Theta(m)}$. As it turns out, for $m \geq 3$ these schemes imply IT DPF schemes with similar communication complexity. Hence, these constructions serve as our baseline.

▶ **Theorem 4** (Reed-Muller IT DPF - Informal, implicit in [17, 3]). *Let $p \geq 2$ be a prime and $m \geq 2$ an integer. There exists a perfectly secure m-server DPF, for point functions with output group $\mathbb{Z}_p$ and key size $O_m(\log(p) \cdot N^{1/(m-1)})$.*

Note that the above theorem is stated for prime sized cyclic groups, as the usual Reed-Muller code based constructions are based on extension fields. However, by employing a similar construction over extension rings, which are used in the MPC literature on secure computation over rings [20, 19], it is possible to extend the above result to any prime power sized cyclic groups, and hence to any Abelian group.

In the second-generation PIR scheme of [5] the exponent of N vanishes super-linearly with m (but is still a constant for any fixed m), and corresponding IT DPF constructions can be derived as well.

Finally, third-generation PIR schemes [40, 25, 24, 4] achieve $N^{o(1)}$ communication complexity with as low as 3 servers, or even 2 servers if we allow the servers to respond with $N^{o(1)}$-bit messages. These schemes are based on a nontrivial combinatorial object called a *matching vectors* (MV) family, based on the work of [30]. In addition to their superior asymptotic communication complexity, as was discussed in [4, 31], for moderate size parameters, these schemes can achieve superior concrete complexity as well, by employing an MV family based on the work of Frankl [27].

Unfortunately, unlike the first and second generation PIR schemes, MV-based PIR schemes do not readily imply a DPF. Indeed, for some specific output ring R, the schemes imply a form of "quasi-additive" DPF, where β can either be chosen to be zero or *some* invertible element ζ of R (which depends on the choice of α and the randomness of the key generation). Note that given such a quasi-additive DPF for m servers, it can be converted to a true DPF with $2m$ servers by replicating each quasi-additive DPF share among two servers, as well as secret sharing $\zeta = \zeta_1 + \zeta_2$ among them. Indeed, this principle can also be applied to *balanced* PIR schemes, where the output message of each server is a vector instead of a single element. By applying this to the 2-server "quasi-additive" DPF implicit in the 2-server PIR work of Dvir and Gopi [24] we obtain Theorem 1.

The above discussion leaves open the question of obtaining a 3-server IT DPF with communication complexity $N^{o(1)}$. We are able to construct such a DPF with statistical security. One subtle difficulty is that even though the nonzero payload β generated by the quasi-additive DPF depends on the randomness of the key generation, this entropy is eliminated when we condition on the view of a server. Our strategy is to repeat the quasi-additive DPF σ times, for point functions $f_{\alpha,\beta_1}, \ldots, f_{\alpha,\beta_\sigma}$, such that with probability $1/2$ we take $\beta_i = 0$ and take a nonzero β_i otherwise. This ensures that even when fixing α and conditioning on the view of a single server, the payload β has some entropy. Then, we provide each server with its respective σ keys. In addition, denoting by B the vector of payloads, such that each coordinate i takes the value β_i, we provide the servers with a random vector r satisfying $\langle r, B \rangle = \beta$ for the desired output value β.

By the perfect security of the PIR, the σ keys alone do not reveal any information. However, since r is correlated with them and with β, some information on the relation between α and β is revealed. To argue that the amount of information is a negligible function of σ, we first invoke the leftover hash lemma to argue that for a uniformly random r', the

distribution of $(r', \langle r', B\rangle)$ is statistically close to uniform. We then argue that if we condition this joint distribution on different values of $\langle r', B\rangle$, the distribution of r' cannot change much. By applying this principle to the PIR scheme of [4], we obtain Theorem 2.

2 Preliminaries

Notation. For $N \in \mathbb{N}$ we let $[N] = \{1, \ldots, N\}$. We denote the inner product of two vectors u and v of the same length by $\langle u, v\rangle = \sum_i u_i v_i$.

Probability. For two distributions D_1, D_2 we denote by $d(D_1, D_2) = \frac{1}{2}\sum_\omega |\Pr_{D_1}[\omega] - \Pr_{D_2}[\omega]|$ their total variation distance. We denote by U_ℓ uniformly distributed random strings of length ℓ.

Groups. We represent an Abelian group $\mathbb{G}$ of the form $\mathbb{G} = \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_\ell}$, for prime powers $q_1, \ldots, q_\ell$ by $\hat{\mathbb{G}} = (q_1, \ldots, q_\ell)$ and represent a group element of $\mathbb{G}$ by a sequence of ℓ non-negative integers.

Point functions. Given a domain size N and Abelian group $\mathbb{G}$, a *point function* $f_{\alpha,\beta} : [N] \to \mathbb{G}$ for $\alpha \in [N]$ and $\beta \in \mathbb{G}$ evaluates to β on input α and to $0 \in \mathbb{G}$ on all other inputs. We denote by $\hat{f}_{\alpha,\beta} = (N, \hat{\mathbb{G}}, \alpha, \beta)$ the representation of such a point function.

2.1 Distributed Point Functions

We begin with a formal definition of the cryptographic primitive of distributed point functions (DPFs).

▶ **Definition 5** (DPF [28, 13]). *A (1-private) m-server* distributed point function*, or m-DPF for short, is a tuple of algorithms $\Pi = (\mathsf{Gen}, \mathsf{Eval}_0, \ldots, \mathsf{Eval}_{m-1})$ with the following syntax:*

- $\mathsf{Gen}(1^\lambda, \hat{f}_{\alpha,\beta}) \to (k_0, \ldots, k_{m-1})$*: On input security parameter $\lambda \in \mathbb{N}$ and point function description $\hat{f}_{\alpha,\beta} = (N, \hat{\mathbb{G}}, \alpha, \beta)$, the (randomized) key generation algorithm* Gen *returns an m-tuple of keys $k_0, \ldots, k_{m-1} \in \{0,1\}^*$. We assume that N and $\mathbb{G}$ are determined by each key.*
- $\mathsf{Eval}_i(k_i, x) \to y_i$*: On input key $k_i \in \{0,1\}^*$ and input $x \in [N]$ the (deterministic) evaluation algorithm of server i,* Eval_i*, returns a group element $y_i \in \mathbb{G}$.*

We require Π to satisfy the following requirements:

- ***Correctness:*** *For every λ, $\hat{f}_{\alpha,\beta} = (N, \hat{\mathbb{G}}, \alpha, \beta)$ and $x \in [N]$, if $(k_0, \ldots, k_{m-1}) \leftarrow \mathsf{Gen}(1^\lambda, \hat{f}_{\alpha,\beta})$, then $\Pr\left[\sum_{i=0}^{m-1} \mathsf{Eval}_i(k_i, x) = f_{\alpha,\beta}(x)\right] = 1$.*
- ***Security:*** *Consider the following semantic security challenge experiment for a corrupted server $T \in \{0, \ldots, m-1\}$:*
  1. *The adversary gives challenge point function descriptions $(\hat{f}^1 = (N_1, \hat{\mathbb{G}}_1, \alpha_1, \beta_1), \hat{f}^2 = (N_2, \hat{\mathbb{G}}_2, \alpha_2, \beta_2)) \leftarrow \mathcal{A}(1^\lambda)$ with $N_1 = N_2$ and $\hat{\mathbb{G}}_1 = \hat{\mathbb{G}}_2$.*
  2. *The challenger samples $b \xleftarrow{\$} \{0,1\}$ and $(k_0, \ldots, k_{m-1}) \leftarrow \mathsf{Gen}(1^\lambda, \hat{f}^b)$.*
  3. *The adversary outputs a guess $b' \leftarrow \mathcal{A}(k_T)$.*

  Denote by $\mathrm{Adv}(1^\lambda, \mathcal{A}, T) := \Pr[b = b'] - 1/2$ the advantage of $\mathcal{A}$ in guessing b in the above experiment. For circuit size bound $S = S(\lambda)$ and advantage bound $\epsilon(\lambda)$, we say that Π is (S, ϵ)-secure if for all T, and all non-uniform adversaries $\mathcal{A}$ of size $S(\lambda)$, we have $\mathrm{Adv}(1^\lambda, \mathcal{A}, T) \le \epsilon(\lambda)$. We say that Π is:

- Computationally ϵ-secure *if it is* (S, ϵ)*-secure for all polynomials* S*.*
- Computationally secure *if it is* $(S, 1/S)$*-secure for all polynomials* S*.*
- Statistically ϵ-secure *if it is* (S, ϵ)*-secure for all* S*. When* ϵ *is omitted it is understood to be negligible in* λ*.*
- Perfectly secure *if it is statistically* 0*-secure.*

3 Constructions

In this section we give two main constructions of information-theoretic DPF. The first is perfectly secure but requires 4 servers. The second requires just 3 servers but only offers statistical security.

3.1 4-server MV-based DPF

Our first result is the following, based on 2-server "quasi-additive" DPF, implicit in [24]. In that quasi-additive DPF the response of each server is a vector, such that the result is constructed by taking an inner product of the sum of the servers' vectors with a reconstruction vector that is a function of the point α and therefore must not be part of the DPF key. However, the reconstruction vector can be readily secret-shared between two keys. Using four servers such that each pair of servers receives one of the original keys of the two-server DPF of [24] and secret shares of the reconstruction vector, results in a scheme in which each server returns a single element.

▶ **Theorem 6.** *Let* $p \geq 3$ *be a prime and* $s \geq 1$ *an integer. There exists a perfectly secure* 4*-DPF, for point functions with output group* $\mathbb{Z}_{p^s}$*,* $\beta \in \{0, 1\}$*, domain size* N*, and key size* $|k_i| = O\left(s \log(p) \cdot 2^{2p\sqrt{\log N \log\log N}}\right)$*,* $i \in \{0, 1, 2, 3\}$*.*

We prove the theorem in several steps. The following theorem is a generalization of the construction implicit in [24] to the case of matching vector families over moduli $m = 2p^s$, for a prime $p \geq 3$ and an integer s. Here, Share is a randomized algorithm that shares an input $\alpha \in [N]$ between two servers, Conv is a share conversion algorithm employed by the servers that maps the shares of α to shares of $f_{\alpha,\zeta}(x)$, for some nonzero ζ, and Rec is an algorithm that allows, given α, to recover $f_{\alpha,\zeta}(x)$.

▶ **Theorem 7** (Dvir Gopi share conversion [24], generalized)**.** *Let* $p \geq 3$ *be a prime,* $s \geq 1$ *an integer, and denote* $q = 2p^s$*. For every integer* $N \geq 1$ *there exist a randomized mapping* $\mathsf{Share} : [N] \to \mathbb{Z}_q^h \times \mathbb{Z}_q^h$*,* $h = O\left(\log(q) \cdot 2^{2p\sqrt{\log N \log\log N}}\right)$*, and deterministic mappings* $\mathsf{Conv} : \{0, 1\} \times \mathbb{Z}_q^h \times [N] \to \mathbb{Z}_{p^s}^h$ *and* $\mathsf{Rec} : [N] \to \mathbb{Z}_{p^s}^h$*, such that*

- *For every* $\alpha, x \in [N]$*,*

$$\Pr\left[(c_0, c_1) \leftarrow \mathsf{Share}(\alpha) : \left\langle \mathsf{Rec}(\alpha), \sum_{i=0}^{1} \mathsf{Conv}(i, c_i, x) \right\rangle \begin{cases} \in \{2, -2\}, & x = \alpha \\ = 0, & x \neq \alpha \end{cases}\right] = 1.$$

- *For every* $\alpha, \alpha' \in [N]$ *and* $i \in \{0, 1\}$*,*

$$[(c_0, c_1) \leftarrow \mathsf{Share}(\alpha); \mathit{Output}\ c_i] \equiv [(c_0, c_1) \leftarrow \mathsf{Share}(\alpha'); \mathit{Output}\ c_i]$$

- Share, Conv, Rec *are computable in time polynomial in their input and output size.*

We will first need the following result from [30].

Notation: Let $\mathsf{Share}, \mathsf{Conv}, \mathsf{Rec}$ be as in Theorem 7.

$\mathsf{Gen}(\hat{f}_{\alpha,\beta} = (N, \hat{\mathbb{G}} = \widehat{\mathbb{Z}_{p^s}}, \alpha, \beta \in \{0,1\}))$:
- Compute $(c_0, c_1) \leftarrow \mathsf{Share}(\alpha)$.
- Compute $r = \left[\left\langle \mathsf{Rec}(\alpha), \sum_{i=0}^{1} \mathsf{Conv}(i, c_i, \alpha)\right\rangle\right]^{-1} \mathsf{Rec}(\alpha)\beta$, and share it additively $r = r_0 + r_1$.
- Output $k_0 = (c_0, r_0), k_1 = (c_0, r_1), k_2 = (c_1, r_0), k_3 = (c_1, r_1)$.

$\mathsf{Eval}_i(k_i = (c_{j_1}, r_{j_2}), x)$:
- Compute and output $\langle r_{j_2}, \mathsf{Conv}(j_1, c_{j_1}, x)\rangle$.

Figure 1 4-server MV-based DPF.

▶ **Theorem 8** (Matching Vectors [30]). *For every integers N, s and prime p, there is $h = O\left(\log(q) 2^{2p\sqrt{\log N \log\log N}}\right)$ and a collection of vectors $(u_i, v_i)_{i\in[N]}$ in $\mathbb{Z}_q^h$ (called* matching vectors*), $q = 2p^s$, such that*
- *For every $i \in [N]$, $\langle u_i, v_i\rangle = 0$.*
- *For every $i \neq j$, $\langle u_i, v_j\rangle \in \{1, p^s, p^s + 1\}$.*

We are now ready to prove Theorem 7.

Proof of Theorem 7. Let h and (u_i, v_i) be as in Theorem 8. $\mathsf{Share}(\alpha)$ draws a random vector $w \xleftarrow{\$} \mathbb{Z}_q^h$ and outputs $(w, w + u_\alpha)$. $\mathsf{Conv}(i, w', x)$ outputs

$$((-1)^{\langle w', v_x\rangle}, (-1)^{\langle w', v_x\rangle} v_x) \mod p^s.$$

$\mathsf{Rec}(\alpha)$ outputs $(1, -u_\alpha) \mod p^s$. Efficiency and security are obvious. Correctness follows because the expression

$$\left\langle (1, -u_\alpha), ((-1)^{\langle w, v_x\rangle}, (-1)^{\langle w, v_x\rangle} v_x) + ((-1)^{\langle w+u_\alpha, v_x\rangle}, (-1)^{\langle w+u_\alpha, v_x\rangle} v_x)\right\rangle \mod p^s$$

equals $(-1)^{\langle w, v_x\rangle} \cdot (1 - \langle u_\alpha, v_x\rangle) \cdot \left(1 + (-1)^{\langle u_\alpha, v_x\rangle}\right) \mod p^s$ which is in $\{-2, 2\}$ if $x = \alpha$ and equals 0 if $x \neq \alpha$, because then $\langle u_\alpha, v_x\rangle \in \{1, p^s, p^s + 1\}$. ◀

Using the above result, we can construct a 4-server IT DPF. Below is a construction for the output group $\mathbb{Z}_{p^s}$ and $\beta \in \{0, 1\}$. An extension to general finite Abelian group $\mathbb{G}$ and any $\beta \in \mathbb{G}$ can be done by bit decomposition, which will incur a multiplicative factor of $\log|\mathbb{Z}_q| = \log q$ in privacy loss, computational cost, and key length. However, some groups might have large key size, due to p appearing as an exponent in the key size in Theorem 6.

Proof of Theorem 6. The construction is given in Figure 1.

Security and efficiency are obvious. Correctness follows because

$$\begin{aligned}
\sum_{j_1=0}^{1}\sum_{j_2=0}^{1} \langle r_{j_2}, \mathsf{Conv}(j_1, c_{j_1}, x)\rangle &= \left\langle \sum_{j_2=0}^{1} r_{j_2}, \sum_{j_1=0}^{1} \mathsf{Conv}(j_1, c_{j_1}, x)\right\rangle \\
&= \left\langle \left[\left\langle \mathsf{Rec}(\alpha), \sum_{i=0}^{1} \mathsf{Conv}(i, c_i, \alpha)\right\rangle\right]^{-1} \mathsf{Rec}(\alpha)\beta, \sum_{j_1=0}^{1} \mathsf{Conv}(j_1, c_{j_1}, x)\right\rangle \\
&= \beta \left[\left\langle \mathsf{Rec}(\alpha), \sum_{i=0}^{1} \mathsf{Conv}(i, c_i, \alpha)\right\rangle\right]^{-1} \left\langle \mathsf{Rec}(\alpha), \sum_{i=0}^{1} \mathsf{Conv}(i, c_i, x)\right\rangle,
\end{aligned}$$

which is either β or 0 depending on whether $x = \alpha$ or $x \neq \alpha$, respectively. ◀

3.2 3-server statistically-secure MV-based DPF

To construct a 3-server statistically-secure DPF, we need the following result from [4].

▶ **Theorem 9** ([4, Theorem 3.5]). *For every domain size $N \geq 1$ there exist a randomized mapping $\mathsf{Share} : [N] \to \mathbb{Z}_6^h \times \mathbb{Z}_6^h \times \mathbb{Z}_6^h$, $h = O\left(2^{6\sqrt{\log N \log\log N}}\right)$, and a deterministic mapping $\mathsf{Conv} : \mathbb{Z}_6^h \times \mathbb{Z}_6^h \times [N] \to \mathbb{Z}_2^2$, such that*

1. *For every $\alpha, x \in [N]$,*

$$\Pr\left[(c_0, c_1, c_2) \leftarrow \mathsf{Share}(\alpha) : \sum_{i=0}^{2} \mathsf{Conv}(c_i, c_{(i+1) \bmod 3}, x) \begin{cases} \neq \mathbf{0}, & x = \alpha \\ = \mathbf{0}, & x \neq \alpha \end{cases}\right] = 1.$$

2. *For every $\alpha, \alpha' \in [N]$ and $i, j \in \{0, 1, 2\}$,*

$$[(c_0, c_1, c_2) \leftarrow \mathsf{Share}(\alpha); \mathit{Output}\ (c_i, c_j)] \equiv [(c_0, c_1, c_2) \leftarrow \mathsf{Share}(\alpha'); \mathit{Output}\ (c_i, c_j)]$$

3. *$\mathsf{Share}, \mathsf{Conv}$ are computable in time polynomial in their input and output size.*

Below is the our main theorem for this section. Using the results of [37, 36], we also show how to extend this theorem to bigger payloads.

▶ **Theorem 10.** *Fix an integer $\lambda > 0$. The construction in Figure 2 is a statistically $\left(41 \cdot 2^{\frac{2-\lambda}{2}}\right)$-secure 3-DPF, for point functions with output group $\mathbb{G} = \mathbb{Z}_2^2$, domain size N, and key size $|k_i| = O\left(\lambda \cdot 2^{6\sqrt{\log N \log\log N}}\right)$, $i \in \{0, 1, 2\}$.*

Next, we will need an additional result.

▶ **Definition 11.** *Let X be a random variable. Then the* min-entropy *of X is*

$$H_\infty(X) = \min_x \log \frac{1}{\Pr[X = x]}$$

▶ **Lemma 12** (Leftover Hash Lemma). *Let $\mathbb{F}$ be a finite field. If X is a random variable over $\mathbb{F}^n$ with $H_\infty(X) \geq R$ and $Y \xleftarrow{\$} \mathbb{F}^n$ is drawn independently, then it holds that $d\left((Y, \langle Y, X\rangle), U_{(n+1)\log|\mathbb{F}|}\right) \leq 2^{\frac{\log|\mathbb{F}| - R}{2}}$.*

Proof of Theorem 10.

Efficiency. Follows by construction and Theorem 9.

Correctness. In fact, the Gen algorithm may not be defined if $y = 0$, as there might not be r such that $\langle r, y\rangle = \beta$. In that case we can just let Gen reveal α and β for a negligible privacy loss. When this does not happen, we need to show that $\sum_{i=1}^{3} \mathsf{Eval}_i(k_i, x) = \langle r, \sum_{i=1}^{3} y_i\rangle = f_{\alpha,\beta}(x)$. Indeed, when $x \neq \alpha$ we have that $\sum_{i=1}^{3} y_i = 0$ because every $(c_0^\ell, c_1^\ell, c_2^\ell)$ was produced by computing either $\mathsf{Share}(\alpha)$ or $\mathsf{Share}(N+1)$. When $x = \alpha$ we have that $\sum_{i=1}^{3} y_i^\ell = y^\ell$, which implies that $\langle r, \sum_{i=1}^{3} y_i\rangle = \langle r, y\rangle = \beta$.

Notation: Let $\mathsf{Share}, \mathsf{Conv}$ be as in Theorem 9 with domain size $N+1$.

$\mathsf{Gen}(1^\lambda, \hat{f}_{\alpha,\beta} = (N, \hat{\mathbb{G}} = \widehat{\mathbb{Z}_2^2}, \alpha, \beta))$:

- For $\ell = 1, \ldots, \lambda$ draw $\alpha_\ell^* \xleftarrow{\$} \{\alpha, N+1\}$ and compute $(c_0^\ell, c_1^\ell, c_2^\ell) \leftarrow \mathsf{Share}(\alpha_\ell^*)$.
- For $\ell = 1, \ldots, \lambda$ set

$$y^\ell = \begin{cases} \mathsf{Conv}(c_1^\ell, c_2^\ell, \alpha) + \mathsf{Conv}(c_2^\ell, c_0^\ell, \alpha) + \mathsf{Conv}(c_0^\ell, c_1^\ell, \alpha), & \alpha_\ell^* = \alpha \\ 0, & \alpha_\ell^* = N+1 \end{cases}$$

  Denote by $y \in \mathbb{F}_4^\lambda$ the vector of all y^ℓ values concatenated, where we naturally associate elements of $\mathbb{Z}_2^2$ with these of $\mathbb{F}_4$ (mapping zero to zero).
- Choose $r \in \mathbb{F}_4^\lambda$ at random under the constraint that $\langle r, y \rangle = \beta$.
- Output $k_0 = ((c_1^\ell, c_2^\ell)_{\ell=1}^\lambda, r), k_1 = ((c_2^\ell, c_0^\ell)_{\ell=1}^\lambda, r), k_2 = ((c_0^\ell, c_1^\ell)_{\ell=1}^\lambda, r)$.

$\mathsf{Eval}_i(k_i = ((c_a^\ell, c_b^\ell)_{\ell=1}^\lambda, r), x)$:

- For $\ell = 1, \ldots, \lambda$ set

$$y_i^\ell := \mathsf{Conv}(c_a^\ell, c_b^\ell, x),$$

  and denote by $y_i \in \mathbb{F}_4^\lambda$ the vector of all y_i^ℓ values concatenated.
- Compute and output $\langle r, y_i \rangle$.

Figure 2 3-server statistically secure MV-based DPF.

Security. Denote by $D_{\alpha,\beta}$ the distribution of k_0 as outputted by Gen on input λ and $\hat{f}_{\alpha,\beta}$. We will show that for $\alpha_1 \neq \alpha_2$ and β_1, β_2 the distributions $D_1 = D_{\alpha_1,\beta_1}$ and $D_2 = D_{\alpha_2,\beta_2}$ have statistical distance negligible in λ. The claim for k_1 and k_2 follows without loss of generality. It holds by part 2 of Theorem 9 that

$$\begin{aligned} d(D_1, D_2) &= \frac{1}{2} \sum_{c'} \sum_{r'} \left| \Pr_{D_1}[k_0 = c', r = r'] - \Pr_{D_2}[k_0 = c', r = r'] \right| \\ &= \frac{1}{2} \sum_{c'} \Pr_{D_1}[k_0 = c'] \sum_{r'} \left| \Pr_{D_1}[r = r' | k_0 = c'] - \Pr_{D_2}[r = r' | k_0 = c'] \right| \\ &\leq \frac{1}{2} \max_{c'} \sum_{r'} \left| \Pr_{D_1}[r = r' | k_0 = c'] - \Pr_{D_2}[r = r' | k_0 = c'] \right| \\ &\leq \max_{c'} d(D_1|_{k_0=c'}, D_2|_{k_0=c'}). \end{aligned}$$

Therefore, it is sufficient to upper bound the distance between the distributions D_1 and D_2 conditioned on $k_0 = c'$.

Let y be the vector depending on c' in the distribution D_i conditioned on $k_0 = c'$, which is a distribution over the set $\{(c', r') : r' \in \mathbb{F}_4^\lambda\}$. Then, by part 1 of Theorem 9, in this distribution, every y^ℓ attains two possible values with equal probability, either some nonzero value (depending on c' and α) if $\alpha_\ell^* = \alpha$ or zero if $\alpha_\ell^* = N+1$. Therefore, $H_\infty(y) = \lambda$. By applying Lemma 12 we deduce that when $\hat{r} \xleftarrow{\$} \mathbb{F}_4^\lambda$, the joint distribution $(\hat{r}, \langle \hat{r}, y \rangle)$ is $\epsilon := 2^{\frac{2-\lambda}{2}}$-close to $U_{2(\lambda+1)}$. In particular, $\left| \Pr[\langle \hat{r}, y \rangle = \beta_i] - \frac{1}{4} \right| \leq \epsilon$.

Conditioned on $k_0 = c'$, the distribution of r is exactly $\hat{r}|_{\langle \hat{r}, y\rangle = \beta_i}$. Hence, for a value $w := \Pr[\langle \hat{r}, y\rangle = \beta_i] - \frac{1}{4}$, $-\epsilon \le w \le \epsilon$, we arrive at

$$\begin{aligned}
d(\hat{r}|_{\langle \hat{r}, y\rangle = \beta_i}, U_{2\lambda}) &= \sup_{E \subseteq \{(r', \beta_i) : r' \in \mathbb{F}_4^\lambda\}} \left| \frac{\Pr[(\hat{r}, \langle \hat{r}, y\rangle) \in E]}{\frac{1}{4} + w} - \frac{|E|}{4^\lambda} \right| \\
&\le 4 \sup_{E \subseteq \mathbb{F}_4^{\lambda+1}} \left| \frac{\Pr[(\hat{r}, \langle \hat{r}, y\rangle) \in E]}{1 + 4w} - \frac{|E|}{4^{\lambda+1}} \right| \\
&\le 4 \sup_{E \subseteq \mathbb{F}_4^{\lambda+1}} \left| \Pr[(\hat{r}, \langle \hat{r}, y\rangle) \in E] - \frac{|E|}{4^{\lambda+1}} \right| + 16|w| + O(|w|^2) \\
&= 4d((\hat{r}, \langle \hat{r}, y\rangle), U_{2(\lambda+1)}) + 16\epsilon + O(\epsilon^2) \\
&= 20\epsilon + O(\epsilon^2),
\end{aligned}$$

which concludes the proof, because if $d(\hat{r}|_{\langle \hat{r}, y\rangle = \beta_i}, U_{2\lambda}) \le 20\epsilon + O(\epsilon^2)$ in both distributions, then also $d(D_1|_{k_0=c'}, D_2|_{k_0=c'}) \le 40\epsilon + O(\epsilon^2) \le 41\epsilon$ by the triangle inequality, and by choosing $\lambda \ge 10$. ◀

The construction from Theorem 10 can be generalized to any prime characteristic, due to the results of [37, 36], from which we get the following.

▶ **Theorem 13** ([37, 36]). *Let p and $p_1 < p_2$ be primes such that either*

- *$p_1, p_2 \ne 2$ and $p \in \{p_1, p_2\}$;*
- *$2 \in \{p_1, p_2\}$ and $p = 2$.*

Then, for $q = p_1 p_2$, there exists a randomized mapping $\mathsf{Share} : [N] \to \mathbb{Z}_q^h \times \mathbb{Z}_q^h \times \mathbb{Z}_q^h$, $h = O\left(\log(q) 2^{2p_2\sqrt{\log N \log\log N}}\right)$, and a deterministic mapping $\mathsf{Conv} : \mathbb{Z}_{2p_2}^h \times \mathbb{Z}_q^h \times [N] \to \mathbb{Z}_p^\ell$, for some constant $\ell = O(q^2)$, such that

1. *For every $\alpha, x \in [N]$,*

$$\Pr\left[(c_0, c_1, c_2) \leftarrow \mathsf{Share}(\alpha) : \sum_{i=0}^{2} \mathsf{Conv}(c_i, c_{(i+1) \bmod 3}, x) \begin{cases} \ne \mathbf{0}, & x = \alpha \\ = \mathbf{0}, & x \ne \alpha \end{cases}\right] = 1.$$

2. *For every $\alpha, \alpha' \in [N]$ and $i, j \in \{0, 1, 2\}$, the distributions of (c_i, c_j), produced by either $(c_0, c_1, c_2) \leftarrow \mathsf{Share}(\alpha)$ or $(c_0, c_1, c_2) \leftarrow \mathsf{Share}(\alpha')$, are identical.*
3. *Share, Conv are computable in time polynomial in their input and output length.*

Utilizing Theorem 13 in similar fashion to how Theorem 9 is used in the proof of Theorem 10, we deduce the following. Note that we require the group size p to be rather small, due to p_2 appearing in the exponent in the expression for h.

▶ **Theorem 14.** *Fix an integer $\lambda > 0$. There exists a statistically $2^{-\Omega(\lambda)}$-secure 3-DPF, for point functions with output group $\mathbb{G} = \mathbb{Z}_p$, where p is a prime, domain size N, and key size $|k_i| = O\left(\lambda \log(p) \cdot 2^{k(p)\sqrt{\log N \log\log N}}\right)$, $i \in \{0, 1, 2\}$, where $k(2) = 6$, $k(3) = 10$, and $k(p) = 2p$ if $p \ge 5$.*

4 Distributed Key Generation

In the standard model for DPF, a client accepts a point (α, β) as input and generates appropriate DPF keys. However, in certain applications of DPF, such as distributed computation of RAM program or MPC with preprocessing for mixed-mode computations, one needs to

accommodate an input (α, β) that is secret-shared among parties that jointly act as client to generate DPF keys, which can then be either locally evaluated or provided to external servers.

We discuss two different settings for distributed key generation: either two clients sharing an input and then jointly generating keys for $m \geq 3$ servers, or all m parties together secret-sharing the input with threshold $t = 1$ and then generating the keys. In either setting it is natural to consider both semi-honest and malicious adversaries.

The point α can be shared in the input in different ways, e.g. secret-sharing each bit separately, or sharing α as an integer value modulo a $N' \geq N$ for domain size N. Generic MPC protocols can be used to switch between these representations with security against malicious adversaries and in time and communication that is linear in the size of the input (for a constant number of servers). Since the input size is negligible in the key length and in the overall communication and computation for distributed key generation we ignore this cost in the rest of the section.

DPF schemes that are based on a family of Matching Vectors such as the schemes in Figures 1, 2 or the scheme in [4] that is based on Frankl's MV family [27] have Gen algorithms that use the following template. Associate the points in the input domain with subsets of a given size w out of a universe of k items. Each point x in the input domain is therefore associated with a binary vector v_x of length k and Hamming weight w, and it must hold that $\binom{k}{w} \geq N$. The vector v_x determines a second binary vector u_x in which each coordinate is a product of a fixed subset of the coordinates of v_x. On input point α the Gen algorithm returns as output u_α. By adapting the discussion in Appendix C of [2] we have that:

▶ **Proposition 15.** *Let $f_{\alpha,\beta} : [N] \to \mathbb{Z}_2$ be a point function and let Share and $h = O\left(2^{6\sqrt{\log N \log\log N}}\right)$ be as in Theorem 9. Choose w to be the smallest integer such that $\binom{w^2}{w} \geq N$, and set $k = w^2$. Then, there exists a Boolean circuit that computes Share with $O(\binom{w^2}{3\sqrt{w}} \cdot 3\sqrt{w} \cdot w^2) = \tilde{O}(h)$ gates and depth $O(\log h)$.*

A circuit to compute the mapping Share can be readily transformed into a circuit that computes Gen for the IT-DPF schemes that we presented. In the 3-server quasi-additive DPF from [4], Gen is identical to Share. In the 3-server statistically secure DPF scheme from Section 3.2, Share is repeated λ times for a statistical security parameter λ and each key is of twice the size of the key from [4] due to CNF sharing of each coordinate in the vector. Therefore, the circuit for Gen is 2λ times the size of the circuit for Share. Finally, in the 4-server scheme of Section 3.1 the circuit size is identical to the circuit size of the 3-server quasi-additive DPF from [4].

The next theorem describes the asymptotic features of using general MPC protocols to securely and distributively generate the keys in the presence of an adversary that corrupts at most one of the servers.

▶ **Theorem 16.** *Let $f_{\alpha,\beta} : [N] \to \mathbb{Z}_2$ be a point function and $h = O\left(2^{6\sqrt{\log N \log\log N}}\right)$. If α and β are secret-shared between $m \geq 2$ servers, for constant m, and the adversary controls at most one server then there exist protocols for distributed key generation for the protocols in Figure 1 and Figure 2 that have the following features:*

- *If $m \geq 3$ then the protocol has information-theoretic security against a malicious adversary using only secure point-to-point channels.*
- *If $m = 2$ then the protocol has information-theoretic security against a malicious adversary in the OT-hybrid model.*
- *The communication and computation costs of the protocol are $\tilde{O}(h)$.*

- *The round complexity is $O(\log h)$; alternatively, the protocols can have constant round complexity if we settle for computational security, while making only a* black-box *use of a pseudorandom generator.*

Proof. General MPC protocols for $m \geq 3$ parties communicating only by point-to-point channels that are information-theoretically secure against a semi-honest adversary that controls at most one party have first been proposed by [6]. Protocols in the same setting that are secure against a malicious adversary were given in [38]. Two-party protocols in the OT-hybrid model, i.e. that are information-theoretically secure in the OT-hybrid model were given in [29, 33, 32].

All of the above protocols have communication and computation cost $\tilde{O}(|C|)$ if the computed function can be realized by a circuit C with $|C|$ gates and their round complexity scales linearly with the circuit depth. Combining these results with the result of Proposition 15 on the size and depth of the circuit to compute key generation gives the information-theoretic variant of the protocol. For the computational case, we can use constant-round protocols based on garbled circuits that make a black-box use of a PRG [1, 21, 34]. ◀

5 Open Questions

We leave open the question of extending our results to general output groups. In particular:

1. Is there a *perfectly secure* 3-server DPF with key size $N^{o(1)}$?
2. Can our results be extended to general Abelian output groups? For the case of $\mathbb{Z}_p$ with an s-bit prime p, we do not know how to construct a DPF with key size $\mathrm{poly}(s) \cdot N^{o(1)}$, even if we allow an arbitrary constant number of servers and settle for statistical security.

We briefly explain the relevant barriers. For the first question, it is not clear how to construct a share conversion that improves upon the one in Theorem 9 by satisfying $\sum_{i=0}^{2} \mathsf{Conv}(c_i, c_{(i+1) \bmod 3}, x) = 1$ whenever $x = \alpha$, instead of just being nonzero. For the second question, the obstacle to obtaining a DPF over $\mathbb{Z}_p$ for a large prime p is that this necessitates the underlying share conversion to operate over characteristic p. For existing share conversion schemes, this requires matching vectors whose length grows super-polynomially with the bit-length of p.

References

1 Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513. ACM, 1990. doi:10.1145/100216.100287.

2 Amos Beimel, Yuval Ishai, Ranjit Kumaresan, and Eyal Kushilevitz. On the cryptographic complexity of the worst functions. In *Theory of Cryptography Conference*, pages 317–342. Springer, 2014.

3 Amos Beimel, Yuval Ishai, and Eyal Kushilevitz. General constructions for information-theoretic private information retrieval. *J. Comput. Syst. Sci.*, 71(2):213–247, 2005.

4 Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Ilan Orlov. Share conversion and private information retrieval. In *2012 IEEE 27th Conference on Computational Complexity*, pages 258–268. IEEE, 2012.

5 Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and J-F Raymond. Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval. In *FOCS 2002*, pages 261–270, 2002.

6 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In Oded Goldreich, editor, *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 351–371. ACM, 2019. doi:10.1145/3335741.3335756.

7 Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Lightweight techniques for private heavy hitters. In *42nd IEEE Symposium on Security and Privacy, SP 2021*, pages 762–776. IEEE, 2021.

8 Elette Boyle, Nishanth Chandran, Niv Gilboa, Divya Gupta, Yuval Ishai, Nishant Kumar, and Mayank Rathee. Function secret sharing for mixed-mode and fixed-point secure computation. In *EUROCRYPT 2021, Part II*, pages 871–900, 2021.

9 Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 896–912, 2018.

10 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO 2019*, 2019.

11 Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-LPN. In *Annual International Cryptology Conference*, pages 387–416. Springer, 2020.

12 Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *Eurocrypt 2015*, pages 337–367, 2015.

13 Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *CCS*, 2016.

14 Elette Boyle, Niv Gilboa, and Yuval Ishai. Secure computation with preprocessing via function secret sharing. In *Theory of Cryptography Conference*, pages 341–371, 2019.

15 Paul Bunn, Jonathan Katz, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient 3-party distributed oram. In *International Conference on Security and Cryptography for Networks*, pages 215–232. Springer, 2020.

16 Benny Chor and Niv Gilboa. Computationally private information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 304–313, 1997.

17 Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 41–50. IEEE, 1995.

18 Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *2015 IEEE Symposium on Security and Privacy*, pages 321–338. IEEE, 2015.

19 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 *Annual International Cryptology Conference*, pages 769–798. Springer, 2018.

20 Ronald Cramer, Serge Fehr, Yuval Ishai, and Eyal Kushilevitz. Efficient multi-party computation over rings. In *EUROCRYPT 2003*, pages 596–613, 2003.

21 Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 378–394. Springer, 2005. doi:10.1007/11535218_23.

22 Samuel Dittmer, Yuval Ishai, Steve Lu, Rafail Ostrovsky, Mohamed Elsabagh, Nikolaos Kiourtis, Brian Schulte, and Angelos Stavrou. Function secret sharing for psi-ca: With applications to private contact tracing. *arXiv preprint arXiv:2012.13053*, 2020.

23 Jack Doerner and Abhi Shelat. Scaling ORAM for secure computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 523–535, 2017.

24 Zeev Dvir and Sivakanth Gopi. 2-server PIR with sub-polynomial communication. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *STOC 2015*, pages 577–584. ACM, 2015.

25 Klim Efremenko. 3-query locally decodable codes of subexponential length. *SIAM Journal on Computing*, 41(6):1694–1703, 2012.

26 Ingerid Fosli, Yuval Ishai, Victor I. Kolobov, and Mary Wootters. On the download rate of homomorphic secret sharing. In *ITCS*, 2022.

27 Peter Frankl. Constructing finite sets with given intersections. *Combinatorial mathematics (Marseille-Luminy, 1981)*, pages 289–291, 1983.

28 Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In *EUROCRYPT*, 2014.

29 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In Oded Goldreich, editor, *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 307–328. ACM, 2019. doi:10.1145/3335741.3335755.

30 Vince Grolmusz. Superpolynomial size set-systems with restricted intersections mod 6 and explicit ramsey graphs. *Combinatorica*, 20(1):71–86, 2000.

31 Matthew M Hong, Yuval Ishai, Victor I Kolobov, and Russell WF Lai. On computational shortcuts for information-theoretic pir. In *Theory of Cryptography Conference*, pages 504–534. Springer, 2020.

32 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer–efficiently. In *Annual international cryptology conference*, pages 572–591. Springer, 2008.

33 Joe Kilian. Founding cryptography on oblivious transfer. In Janos Simon, editor, *STOC 1988*, pages 20–31, 1988.

34 Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*, pages 52–78. Springer, 2007. doi:10.1007/978-3-540-72540-4_4.

35 Zachary Newman, Sacha Servan-Schreiber, and Srinivas Devadas. Spectrum: High-bandwidth anonymous broadcast. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 229–248, 2022.

36 Anat Paskin-Cherniavsky and Olga Nissenbaum. New bounds and a generalization for share conversion for 3-server PIR. *Entropy*, 24(4), 2022. doi:10.3390/e24040497.

37 Anat Paskin-Cherniavsky and Leora Schmerler. On share conversions for private information retrieval. *Entropy*, 21(9):826, 2019.

38 Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 73–85, 1989.

39 Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. Epione: Lightweight contact tracing with strong privacy. *arXiv preprint arXiv:2004.13293*, 2020.

40 Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *Journal of the ACM (JACM)*, 55(1):1–16, 2008.