Dieter Gollmann · Atsuko Miyaji
Hiroaki Kikuchi (Eds.)

LNCS 10355

Applied Cryptography and Network Security

15th International Conference, ACNS 2017
Kanazawa, Japan, July 10–12, 2017
Proceedings

Springer

Lecture Notes in Computer Science 10355

More information about this series at http://www.springer.com/series/7410

Dieter Gollmann · Atsuko Miyaji
Hiroaki Kikuchi (Eds.)

Applied Cryptography and Network Security

15th International Conference, ACNS 2017
Kanazawa, Japan, July 10–12, 2017
Proceedings

Springer

Editors
Dieter Gollmann
Hamburg University of Technology
Hamburg
Germany

Hiroaki Kikuchi
Department of Frontier Media Science
Meiji University
Tokyo
Japan

Atsuko Miyaji
Graduate School of Engineering
Osaka University
Suita, Osaka
Japan

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-61203-4 ISBN 978-3-319-61204-1 (eBook)
DOI 10.1007/978-3-319-61204-1

Library of Congress Control Number: 2017944358

LNCS Sublibrary: SL4 – Security and Cryptology

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The 15th International Conference on Applied Cryptography and Network Security (ACNS2017) was held in Kanazawa, Japan, during July 10–12, 2017. The previous conferences in the ACNS series were successfully held in Kunming, China (2003), Yellow Mountain, China (2004), New York, USA (2005), Singapore (2006), Zhuhai, China (2007), New York, USA (2008), Paris, France (2009), Beijing, China (2010), Malaga, Spain (2011), Singapore (2012), Banff, Canada (2013), Lausanne, Switzerland (2014), New York, USA (2015), and London, UK (2016).

ACNS is an annual conference focusing on innovative research and current developments that advance the areas of applied cryptography, cyber security, and privacy. Academic research with high relevance to real-world problems as well as developments in industrial and technical frontiers fall within the scope of the conference.

This year we have received 149 submissions from 34 different countries. Each submission was reviewed by 3.7 Program Committee members on average. Papers submitted by Program Committee members received on average 4.4 reviews. The committee decided to accept 34 regular papers. The broad range of areas covered by the high-quality papers accepted for ACNS 2107 attests very much to the fulfillment of the conference goals.

The program included two invited talks given by Dr. Karthikeyan Bhargavan (Inria Paris) and Prof. Doug Tygar (UC Berkeley).

The decisions of the best student paper award was based on a vote among the Program Committee members. To be eligible for selection, the primary author of the paper has to be a full-time student who is present at the conference. The winner was Carlos Aguilar-Melchor, Martin Albrecht, and Thomas Ricosset from Université de Toulouse, Toulouse, France, Royal Holloway, University of London, UK, and Thales Communications & Security, Gennevilliers, France. The title of the paper is "Sampling From Arbitrary Centered Discrete Gaussians For Lattice-Based Cryptography."

We are very grateful to our supporters and sponsors. The conference was co-organized by Osaka University, Japan Advanced Institute of Science and Technology (JAIST), and the Information-technology Promotion Agency (IPA); it was supported by the Committee on Information and Communication System Security (ICSS), IEICE, Japan, the Technical Committee on Information Security (ISEC), IEICE, Japan, and the Special Interest Group on Computer SECurity (CSEC) of IPSJ, Japan; it and was co-sponsored by the National Institute of Information and Communications Technology (NICT) International Exchange Program, Mitsubishi Electric Corporation, Support Center for Advanced Telecommunications Technology Research (SCAT), Foundation Microsoft Corporation, Fujitsu Hokuriku Systems Limited, Nippon Telegraph and Telephone Corporation (NTT), and Hokuriku Telecommunication Network Co., Inc.

We would like to thank the authors for submitting their papers to the conference. The selection of the papers was a challenging and dedicated task, and we are deeply grateful to the 48 Program Committee members and the external reviewers for their reviews and discussions. We also would like to thank EasyChair for providing a user-friendly interface for us to manage all submissions and proceedings files. Finally, we would like to thank the general chair, Prof. Hiroaki Kikuchi, and the members of the local Organizing Committee.

July 2017

Dieter Gollmann
Atsuko Miyaji

ACNS 2017

The 15th International Conference on Applied Cryptography and Network Security

Jointly organized by

Osaka University
and
Japan Advanced Institute of Science and Technology (JAIST)
and
Information-technology Promotion Agency (IPA)

General Chair

Hiroaki Kikuchi	Meiji University, Japan

Program Co-chairs

Dieter Gollmann	Hamburg University of Technology, Germany
Atsuko Miyaji	Osaka University / JAIST, Japan

Program Committee

Diego Aranha	University of Campinas, Brazil
Giuseppe Ateniese	Stevens Institute of Technology, USA
Man Ho Au	Hong Kong Polytechnic University, Hong Kong, SAR China
Carsten Baum	Bar-Ilan University, Israel
Rishiraj Bhattacharyya	NISER Bhubaneswar, India
Liqun Chen	University of Surrey, UK
Chen-Mou Chen	Osaka University, Japan
Céline Chevalier	Université Panthéon-Assas, France
Sherman S.M. Chow	Chinese University of Hong Kong, Hong Kong, SAR China
Mauro Conti	University of Padua, Italy
Alexandra Dmitrienko	ETH Zurich, Switzerland
Michael Franz	University of California, Irvine, USA
Georg Fuchsbauer	ENS, France
Sebastian Gajek	FUAS, Germany
Goichiro Hanaoka	AIST, Japan
Feng Hao	Newcastle University, UK

Swee-Huay Heng	Multimedia University, Malaysia
Francisco Rodrguez Henrquez	CINVESTAV-IPN, Mexico
Xinyi Huang	Fujian Normal University, China
Michael Huth	Imperial College London, UK
Tibor Jager	Paderborn University, Germany
Aniket Kate	Purdue University, USA
Stefan Katzenbeisser	TU Darmstadt, Germany
Kwangjo Kim	KAIST, Korea
Kwok-yan Lam	NTU, Singapore
Mark Manulis	University of Surrey, UK
Tarik Moataz	Brown University, USA
Ivan Martinovic	University of Oxford, UK
Jörn Müller-Quade	Karlsruhe Institute of Technology, Germany
David Naccache	École normale supérieure, France
Michael Naehrig	Microsoft Research Redmond, USA
Hamed Okhravi	MIT Lincoln Laboratory, USA
Panos Papadimitratos	KTH Royal Institute of Technology, Sweden
Jong Hwan Park	Sangmyung University, Korea
Thomas Peyrin	Nanyang Technological University, Singapore
Bertram Poettering	Ruhr-Universität Bochum, Germany
Christina Pöpper	NYU, United Arab Emirates
Bart Preneel	KU Leuven, Belgium
Thomas Schneider	TU Darmstadt, Germany
Michael Scott	Dublin City University, Ireland
Vanessa Teague	University of Melbourne, Australia
Somitra Kr. Sanadhya	Ashoka University, India
Mehdi Tibouchi	NTT Secure Platform Laboratories, Japan
Ivan Visconti	University of Salerno, Italy
Bo-Yin Yang	Academia Sinica, Taiwan
Kan Yasuda	NTT Secure Platform Laboratories, Japan
Fangguo Zhang	Sun Yat-sen University, China
Jianying Zhou	SUTD, Singapore

Organizing Committee

Local Arrangements

Akinori Kawachi	Tokushima University, Japan

Co-chairs

Kazumasa Omote	University of Tsukuba, Japan
Shoichi Hirose	University of Fukui, Japan
Kenji Yasunaga	Kanazawa University, Japan
Yuji Suga	IIJ, Japan

Finance Co-chairs

Masaki Fujikawa	Kogakuin University, Japan
Yuichi Futa	JAIST, Japan
Natsume Matsuzaki	University of Nagasaki, Japan
Takumi Yamamoto	Mitsubishi Electric, Japan

Publicity Co-chairs

Noritaka Inagaki	IPA, Japan
Masaki Hashimoto	IISEC, Japan
Naoto Yanai	Osaka University, Japan
Kaitai Liang	Manchester Metropolitan University, UK

Liaison Co-chairs

Keita Emura	NICT, Japan
Eiji Takimoto	Ritsumeikan University, Japan
Toru Nakamura	KDDI Research, Japan

System Co-chairs

Atsuo Inomata	Tokyo Denki University/NAIST, Japan
Masaaki Shirase	Future University Hakodate, Japan
Minoru Kuribayashi	Okayama University, Japan
Toshihiro Yamauchi	Okayama University, Japan
Shinya Okumura	Osaka University, Japan

Publication Co-chairs

Takeshi Okamoto	Tsukuba University of Technology, Japan
Takashi Nishide	University of Tsukuba, Japan
Ryo Kikuchi	NTT, Japan
Satoru Tanaka	JAIST, Japan

Registration Co-chairs

Hideyuki Miyake	Toshiba, Japan
Dai Watanabe	Hitachi, Japan
Chunhua Su	Osaka University, Japan

Additional Reviewers

Alesiani, Francesco
Aminanto, Muhamad Erza
Andaló, Fernanda
Armknecht, Frederik
Ashur, Tomer
Auerbach, Benedikt
Azad, Muhammad Ajmal
Bai, Shi

Barrera, David
Bauer, Balthazar
Beierle, Christof
Beunardeau, Marc
Blazy, Olivier
Bost, Raphael
Bourse, Florian
Broadnax, Brandon
Chakraborti, Avik
Chi-Domínguez, Jesús Javier
Chin, Ji-Jian
Choi, Rakyong
Choi, Suri
Ciampi, Michele
Connolly, Aisling
Coon, Ralph A.C.
Costello, Craig
Couteau, Geoffroy
Crane, Stephen
Culnane, Chris
Dargahi, Tooska
Datta, Nilanjan
Davies, Gareth T.
Del Pino, Rafael
Demmler, Daniel
Dirksen, Alexandra
Dominguez Perez, Luis J.
Dong, Xinshu
Dowling, Benjamin
Eom, Jieun
Faust, Sebastian
Ferradi, Houda
Frederiksen, Tore
Gay, Romain
Geraud, Remi
Germouty, Paul
Gochhayat, Sarada Prasad
Hartung, Gunnar
Herzberg, Amir
Huang, Yi
Iovino, Vincenzo
Jap, Dirmanto
Jati, Arpan
Jiang, Jiaojiao
Kairallah, Mustafa
Kamath, Chethan
Karvelas, Nikolaos
Keller, Marcel
Kim, Hyoseung
Kim, Jonghyun
Kim, Joonsik
Kim, Taechan
Kiss, Ágnes
Kitagawa, Fuyuki
Kohls, Katharina
Kuo, Po-Chun
Kurek, Rafael
Lai, Junzuo
Lai, Russell W.F.
Lain, Daniele
Lal, Chhagan
Lee, Kwangsu
Lee, Youngkyung
Li, Huige
Li, Wen-Ding
Li, Yan
Liebchen, Christopher
Liu, Jianghua
Liu, Yunwen
Longa, Patrick
Lu, Jingyang
Lu, Jiqiang
Luykx, Atul
Lyubashevsky, Vadim
Ma, Jack P.K.
Mainka, Christian
Mancillas-López, Cuauhtemoc
Masucci, Barbara
Matsuda, Takahiro
Mazaheri, Sogol
Mechler, Jeremias
Meier, Willi
Meng, Weizhi
Mohamad, Moesfa Soeheila
Moonsamy, Veelasha
Nagel, Matthias
Nielsen, Michael
Nishimaki, Ryo
O’Neill, Adam
Ochoa-Jiménez, José Eduardo
Oliveira, Thomaz
Peeters, Roel

Pereira, Hilder Vitor Lima
Perrin, Léo
Poh, Geong Sen
Puddu, Ivan
Ramanna, Somindu C.
Ramchen, Kim
Renes, Joost
Reparaz, Oscar
Resende, Amanda
Rill, Jochen
Roy, Arnab
Ruffing, Tim
Rupp, Andy
Sakai, Yusuke
Sasaki, Yu
Schuldt, Jacob
Sen Gupta, Sourav
Seo, Hwajeong
Seo, Minhye
Shahandashti, Siamak
Shin, Seonghan
Siniscalchi, Luisa
Spolaor, Riccardo
Stebila, Douglas
Su, Chunhua
Tai, Raymond K.H.
Tan, Syhyuan
Thillard, Adrian
Tosh, Deepak
Vannet, Thomas
Vergnaud, Damien
Volckaert, Stijn
Wang, Ding
Wang, Jiafan
Wang, Xiuhua
Weinert, Christian
Wong, Harry W.H.
Xagawa, Keita
Xie, Shaohao
Yamada, Shota
Yamakawa, Takashi
Yang, Rupeng
Yang, Shaojun
Yang, Xu
Yu, Zuoxia
Zaverucha, Greg
Zhang, Huang
Zhang, Tao
Zhang, Yuexin
Zhang, Zheng
Zhao, Yongjun
Zhou, Peng

Contents

Applied Cryptography

Data Protection and Mobile Security

Security Analysis

Cryptographic Primitives

Side Channel Attack

Cryptographic Protocol

Data and Server Security

Applied Cryptography

Sampling from Arbitrary Centered Discrete Gaussians for Lattice-Based Cryptography

Carlos Aguilar-Melchor[1], Martin R. Albrecht[2], and Thomas Ricosset[1,3](✉)

[1] INP ENSEEIHT, IRIT-CNRS, Université de Toulouse, Toulouse, France
{carlos.aguilar,thomas.ricosset}@enseeiht.fr
[2] Information Security Group, Royal Holloway, University of London, London, UK
martin.albrecht@royalholloway.ac.uk
[3] Thales Communications & Security, Gennevilliers, France

Abstract. Non-Centered Discrete Gaussian sampling is a fundamental building block in many lattice-based constructions in cryptography, such as signature and identity-based encryption schemes. On the one hand, the center-dependent approaches, e.g. cumulative distribution tables (CDT), Knuth-Yao, the alias method, discrete Zigurat and their variants, are the fastest known algorithms to sample from a discrete Gaussian distribution. However, they use a relatively large precomputed table for each possible real center in $[0, 1)$ making them impracticable for non-centered discrete Gaussian sampling. On the other hand, rejection sampling allows to sample from a discrete Gaussian distribution for all real centers without prohibitive precomputation cost but needs costly floating-point arithmetic and several trials per sample. In this work, we study how to reduce the number of centers for which we have to precompute tables and propose a non-centered CDT algorithm with practicable size of precomputed tables as fast as its centered variant. Finally, we provide some experimental results for our open-source C++ implementation indicating that our sampler increases the rate of Peikert's algorithm for sampling from arbitrary lattices (and cosets) by a factor 3 with precomputation storage up to 6.2 MB.

1 Introduction

Lattice-based cryptography has generated considerable interest in the last decade due to many attractive features, including conjectured security against quantum attacks, strong security guarantees from worst-case hardness and constructions of fully homomorphic encryption (FHE) schemes (see the survey [33]). Moreover, lattice-based cryptographic schemes are often algorithmically simple and efficient, manipulating essentially vectors and matrices or polynomials modulo relatively small integers, and in some cases outperform traditional systems.

M.R. Albrecht—The research of this author was supported by EPSRC grant "Bit Security of Learning with Errors for Post-Quantum Cryptography and Fully Homomorphic Encryption" (EP/P009417/1) and the EPSRC grant "Multilinear Maps in Cryptography" (EP/L018543/1).

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 3–19, 2017.
DOI: 10.1007/978-3-319-61204-1_1

Modern lattice-based cryptosystems are built upon two main average-case problems over general lattices: Short Integer Solution (SIS) [1] and Learning With Errors (LWE) [35], and their analogues over ideal lattices, ring-SIS [29] and ring-LWE [27]. The hardness of these problems can be related to the one of their worst-case counterpart, if the instances follow specific distributions and parameters are choosen appropriately [1,27,29,35].

In particular, discrete Gaussian distributions play a central role in lattice-based cryptography. A natural set of examples to illustrate the importance of Gaussian sampling are lattice-based signature and identity-based encryption (IBE) schemes [16]. The most iconic example is the signature algorithm proposed in [16] (hereafter GPV), as a secure alternative to the well-known (and broken) GGH signature scheme [18]. In this paper, the authors use the Klein/GPV algorithm [21], a randomized variant of Babai's nearest plane algorithm [4]. In this algorithm, the rounding step is replaced by randomized rounding according to a discrete Gaussian distribution to return a lattice point (almost) independent of a hidden basis. The GPV signature scheme has also been combined with LWE to obtain the first identity-based encryption (IBE) scheme [16] conjectured to be secure against quantum attacks. Later, a new Gaussian sampling algorithm for arbitrary lattices was presented in [32]. It is a randomized variant of Babai's rounding-off algorithm, is more efficient and parallelizable, but it outputs longer vectors than Klein/GPV's algorithm.

Alternatively to the above trapdoor technique, lattice-based signatures [11,23–26] were also constructed by applying the Fiat-Shamir heuristic [14]. Note that in contrast to the algorithms outlined above which sample from a discrete Gaussian distribution for any real center not known in advance, the schemes developed in [11,25] only need to sample from a discrete Gaussian centered at zero.

1.1 Our Contributions

We develop techniques to speed-up discrete Gaussian sampling when the center is not known in advance, obtaining a flexible time-memory trade-off comparing favorably to rejection sampling. We start with the cumulative distribution table (CDT) suggested in [32] and lower the computational cost of the precomputation phase and the global memory required when sampling from a non-centered discrete Gaussian by precomputing the CDT for a relatively small number of centers, in $\mathcal{O}(\lambda^3)$, and by computing the cdf when needed, i.e. when for a given uniform random input, the values returned by the CDTs for the two closest precomputed centers differ. Second, we present an adaptation of the lazy technique described in [12] to compute most of the cdf in double IEEE standard double precision, thus decreasing the number of precomputed CDTs. Finally, we propose a more flexible approach which takes advantage of the information already present in the precomputed CDTs. For this we use a Taylor expansion around the precomputed centers and values instead of this lazy technique, thus enabling to reduce the number of precomputed CDTs to a $\omega(\lambda)$.

We stress, though, that our construction is not constant time, which limits its utility. We consider addressing this issue important future work.

1.2 Related Work

Many discrete Gaussian samplers over the Integers have been proposed for lattice-based cryptography. Rejection Sampling [12,17], Inversion Sampling with a Cumulative Distribution Table (CDT) [32], Knuth-Yao [13], Discrete Ziggurat [7], Bernoulli Sampling [11], Kahn-Karney [20] and Binary Arithmetic Coding [36].

The optimal method will of course depend on the setting in which it is used. In this work, we focus on what can be done on a modern computer, with a comfortable amount of memery and hardwired integer and floating-point operations. This is in contrast to the works [11,13] which focus on circuits or embedded devices. We consider exploring the limits of the usual memory and hardwired operations in commodity hardware as much an interesting question as it is to consider what is feasible in more constrained settings.

Rejection Sampling and Variants. Straightforward rejection sampling [37] is a classical method to sample from any distribution by sampling from a uniform distribution and accept the value with a probability equal to its probability in the target distribution. This method does not use pre-computed data but needs floating-point arithmetic and several trials by sample. Bernoulli sampling [11] introduces an exponential bias from Bernoulli variables, which can be efficiently sampled specially in circuits. The bias is then corrected in a rejection phase based on another Bernouilli variable. This approach is particularly suited for embedded devices for the simplicity of the computation and the near-optimal entropy consumption. Kahn-Karney sampling is another variant of rejection sampling to sample from a discrete Gaussian distribution which does not use floating-point arithmetic. It is based on the von Neumann algorithm to sample from the exponential distribution [31], requires no precomputed tables and consumes a smaller amount of random bits than Bernoulli sampling, though it is slower. Currently the fastest approach in the computer setting uses a straightforward rejection sampling approach with "lazy" floating-point computations [12] using IEEE standard double precision floating-point numbers in most cases.

Note that none of these methods requires precomputation depending on the distribution's center c. In all the alternative approaches we present hereafter, there is some center-dependent precomputation. When the center is not know this can result in prohibitive costs and handling these becomes a major issue around which most of our work is focused.

Center-Dependent Approaches. The cumulative distribution table algorithm is based on the inversion method [9]. All non-negligible cumulative probabilities are stored in a table and at sampling time one generates a cumulative probability in $[0, 1)$ uniformly at random, performs a binary search through the table and returns the corresponding value. Several alternatives to straightforward CDT are possible. Of special interest are: the alias method [38] which encodes CDTs in a more involved but more efficient approach; BAC Sampling [36] which uses arithmetic coding tables to sample with an optimal consumption of random bits; and Discrete Ziggurat [7] which adapts the Ziggurat method [28] for a flexible time-memory trade-off. Knuth-Yao sampling [22] uses a random bit generator to traverse a binary tree formed from the bit representation of the probability of each possible sample, the terminal node is labeled by the corresponding sample. The main advantage of this method is that it consumes a near-optimal amount of random bits. A block variant and other practical improvements are suggested in [13]. This method is center-dependent but clearly designed for circuits and on a computer setting it is surpassed by other approaches.

Our main contribution is to show how to get rid of the known-center constraint with reasonable memory usage for center-dependent approaches. As a consequence, we obtain a performance gain with respect to rejection sampling approaches. Alternatively, any of the methods discussed above could have replaced our straightforward CDT approach. This, however, would have made our algorithms, proofs, and implementations more involved. On the other hand, further performance improvements could perhaps be achieved this way. This is an interesting problem for future work.

2 Preliminaries

Throughout this work, we denote the set of real numbers by $\mathbb{R}$ and the Integers by $\mathbb{Z}$. We extend any real function $f(\cdot)$ to a countable set A by defining $f(A) = \sum_{x\in A} f(x)$. We denote also by U_I the uniform distribution on I.

2.1 Discrete Gaussian Distributions on $\mathbb{Z}$

The discrete Gaussian distribution on $\mathbb{Z}$ is defined as the probability distribution whose unnormalized density function is

$$\begin{aligned} \rho : \mathbb{Z} &\rightarrow [0,1) \\ x &\rightarrow e^{\frac{-x^2}{2}} \end{aligned}$$

If $s \in \mathbb{R}^+$ and $c \in \mathbb{R}$, then we extend this definition to

$$\rho_{s,c}(x) := \rho\left(\frac{x-c}{s}\right)$$

and denote $\rho_{s,0}(x)$ by $\rho_s(x)$. For any mean $c \in \mathbb{R}$ and parameter $s \in \mathbb{R}^+$ we can now define the discrete Gaussian distribution $D_{s,c}$ as

$$\forall x \in \mathbb{Z},\ D_{s,c}(x) := \frac{\rho_{s,c}(x)}{\rho_{s,c}(\mathbb{Z})}$$

Note that the standard deviation of this distribution is $\sigma = s/\sqrt{2\pi}$. We also define $\mathrm{cdf}_{s,c}$ as the cumulative distribution function (cdf) of $D_{s,c}$

$$\forall x \in \mathbb{Z},\ \mathrm{cdf}_{s,c}(x) := \sum_{i=-\infty}^{x} D_{s,c}(i)$$

Smoothing Parameter. The smoothing parameter $\eta_\epsilon(\Lambda)$ quantifies the minimal discrete Gaussian parameter s required to obtain a given level of smoothness on the lattice Λ. Intuitively, if one picks a noise vector over a lattice from a discrete Gaussian distribution with radius at least as large as the smoothing parameter, and reduces this modulo the fundamental parallelepiped of the lattice, then the resulting distribution is very close to uniform (for details and formal definition see [30]).

Gaussian Measure. An interesting property of discrete Gaussian distributions with a parameter s greater than the smoothing parameter is that the Gaussian measure, i.e. $\rho_{s,c}(\mathbb{Z})$ for $D_{s,c}$, is essentially the same for all centers.

Lemma 1 (From the proof of [30, Lemma 4.4]**).** *For any* $\epsilon \in (0,1)$*,* $s > \eta_\epsilon(\mathbb{Z})$ *and* $c \in \mathbb{R}$ *we have*

$$\Delta_{\mathsf{measure}} := \frac{\rho_{s,c}(\mathbb{Z})}{\rho_{s,0}(\mathbb{Z})} \in \left[\frac{1-\epsilon}{1+\epsilon}, 1\right]$$

Tailcut Parameter. To deal with the infinite domain of Gaussian distributions, algorithms usually take advantage of their rapid decay to sample from a finite domain. The next lemma is useful in determining the tailcut parameter τ.

Lemma 2 ([17, Lemma 4.2]**).** *For any* $\epsilon > 0$*,* $s > \eta_\epsilon(\mathbb{Z})$ *and* $\tau > 0$*, we have*

$$E_{\mathsf{tailcut}} := \Pr_{X \sim D_{\mathbb{Z},s,c}} [|X - c| > \tau s] < 2e^{-\pi\tau^2} \cdot \frac{1+\epsilon}{1-\epsilon}$$

2.2 Floating-Point Arithmetic

We recall some facts from [12] about floating-point arithmetic (FPA) with m bits of mantissa, which we denote by $\mathbb{FP}_m$. A floating-point number is a triplet $\bar{x} = (s, e, v)$ where $s \in \{0,1\}$, $e \in \mathbb{Z}$ and $v \in \mathbb{N}_{2^m-1}$ which represents the real number $\bar{x} = (-1)^s \cdot 2^{e-m} \cdot v$. Denote by $\epsilon = 2^{1-m}$ the floating-point precision. Every FPA-operation $\bar{\circ} \in \{\bar{+}, \bar{-}, \bar{\times}, \bar{/}\}$ and its respective arithmetic operation on $\mathbb{R}$, $\circ \in \{+,-,\times,/\}$ verify

$$\forall \bar{x}, \bar{y} \in \mathbb{FP}_m, |(\bar{x} \,\bar{\circ}\, \bar{y}) - (\bar{x} \circ \bar{y})| \leq (x \circ y)\epsilon$$

Moreover, we assume that the floating-point implementation of the exponential function $\bar{\exp}(\cdot)$ verifies

$$\forall \bar{x} \in \mathbb{FP}_m, |\bar{\exp}(\bar{x}) - \exp(\bar{x})| \leq \epsilon.$$

2.3 Taylor Expansion

Taylor's theorem provides a polynomial approximation around a given point for any function sufficiently differentiable.

Theorem 1 (Taylor's theorem). *Let $d \in \mathbb{Z}^+$ and let the function $f : \mathbb{R} \rightarrow \mathbb{R}$ be d times differentiable in some neighborhood U of $a \in \mathbb{R}$. Then for any $x \in U$*

$$f(x) = \mathcal{T}_{d,f,a}(x) + \mathcal{R}_{d,f,a}(x)$$

where

$$\mathcal{T}_{d,f,a}(x) = \sum_{i=0}^{d} \frac{f^{(i)}(a)}{i!}(x-a)^i$$

and

$$\mathcal{R}_{d,f,a}(x) = \int_a^x \frac{f^{(d+1)}(t)}{d!}(x-t)^d dt$$

3 Variable-Center with Polynomial Number of CDTs

We consider the case in which the mean is variable, i.e. the center is not know before the online phase, as it is the case for lattice-based hash-and-sign signatures. The center can be any real number, but without loss of generality we will only consider centers in $[0, 1)$. Because CDTs are center-dependent, a first naive option would be to precompute a CDT for each possible real center in $[0, 1)$ in accordance with the desired accuracy. Obviously, this first option has the same time complexity than the classical CDT algorithm, i.e. $\mathcal{O}(\lambda \log s\lambda)$ for λ the security parameter. However, it is completely impractical with 2^λ precomputed CDTs of size $\mathcal{O}(s\lambda^{1.5})$. An opposite trade-off is to compute the CDT on-the-fly, avoiding any precomputation storage, which increase the computational cost to $\mathcal{O}(s\lambda^{3.5})$ assuming that the computation of the exponential function run in $\mathcal{O}(\lambda^3)$ (see Sect. 3.2 for a justification of this assumption).

An interesting question is can we keep the time complexity of the classical CDT algorithm with a polynomial number of precomputed CDTs. To answer this question, we start by fixing the number n of equally spaced centers in $[0, 1)$ and precompute the CDTs for each of these. Then, we apply the CDT algorithm to the two precomputed centers closest to the desired center for the same cumulative probability uniformly draw. Assuming that the number of precomputed CDTs is sufficient, the values returned from both CDTs will be equal most of the time, in this case we can conclude, thanks to a simple monotonic argument, that the returned value would have been the same for the CDT at the desired center and return it as a valid sample. Otherwise, the largest value will immediately follow the smallest and we will then have to compute the cdf at the smallest value for the desired center in order to know if the cumulative probability is lower or higher than this cdf. If it is lower then the smaller value will be returned as sample, else it will be the largest.

3.1 Twin-CDT Algorithm

As discussed above, to decrease the memory required by the CDT algorithm when the distribution center is determined during the online phase, we can precompute CDTs for a number n of centers equally spaced in $[0, 1)$ and compute the cdf when necessary. Algorithm 1 resp. 2 describes the offline resp. online phase of the *Twin-CDT* algorithm. Algorithm 1 precomputes CDTs, up to a precision m that guarantees the λ most significant bits of each cdf, and store them with λ-bits of precision as a matrix $\mathbf{T}$, where the i-th line is the CDT corresponding to the i-th precomputed center i/n. To sample from $D_{s,c}$, Algorithm 2 searches the preimages by the cdf of a cumulative probability p, draw from the uniform distribution on $[0,1) \cap \mathbb{FP}_\lambda$, in both CDTs corresponding to the center $\lfloor n(c - \lfloor c \rfloor) \rfloor / n$ (respectively $\lceil n(c - \lfloor c \rfloor) \rceil / n$) which return a value v_1 (resp. v_2). If the same value is returned from the both CDTs (i.e. $v_1 = v_2$), then this value added the desired center integer part is a valid sample, else it computes $\text{cdf}_{s,c-\lfloor c \rfloor}(v_1)$ and returns $v_1 + \lfloor c \rfloor$ if $p < \text{cdf}_{s,c}(v_1)$ and $v_2 + \lfloor c \rfloor$ else.

Algorithm 1. Twin-CDT Algorithm: Offline Phase

```
Input: a Gaussian parameter $s$ and a number of centers $n$
Output: a precomputed matrix $\mathbf{T}$
initialize an empty matrix $\mathbf{T} \in \mathbb{FP}_\lambda^{n \times 2\lceil \tau s \rceil + 3}$
for $i \leftarrow 0, \ldots, n-1$ do
    for $j \leftarrow 0, \ldots, 2\lceil \tau s \rceil + 2$ do
        $\mathbf{T}_{i,j} \leftarrow \mathbb{FP}_m : \text{cdf}_{s,i/n}(j - \lceil \tau s \rceil - 1)$
```

Algorithm 2. Twin-CDT Algorithm: Online Phase

```
Input: a center $c$ and a precomputed matrix $\mathbf{T}$
Output: a sample $x$ that follows $D_{s,c}$
$p \leftarrow U_{[0,1) \cap \mathbb{FP}_\lambda}$
$v_1 \leftarrow i - \lceil \tau s \rceil - 1$ s.t. $\mathbf{T}_{\lfloor n(c-\lfloor c \rfloor) \rfloor, i-1} \leq p < \mathbf{T}_{\lfloor n(c-\lfloor c \rfloor) \rfloor, i}$
$v_2 \leftarrow j - \lceil \tau s \rceil - 1$ s.t. $\mathbf{T}_{\lceil n(c-\lfloor c \rfloor) \rceil, j-1} \leq p < \mathbf{T}_{\lceil n(c-\lfloor c \rfloor) \rceil, j}$
if $v_1 = v_2$ then
    return $v_1 + \lfloor c \rfloor$
else
    if $p < \mathbb{FP}_m : \text{cdf}_{s,c-\lfloor c \rfloor}(v_1)$ then
        return $v_1 + \lfloor c \rfloor$
    else
        return $v_2 + \lfloor c \rfloor$
```

Correctness. We establish correctness in the lemma below.

Lemma 3. *Assuming that m is large enough to ensure λ correct bits during the cdf computation, the statistical distance between the output distribution of Algorithm 2 instantiated to sample from $D_{\mathbb{Z}^m,\sigma,c}$ and $D_{\mathbb{Z}^m,\sigma,c}$ is bounded by $2^{-\lambda}$.*

Proof. First note that from the discrete nature of the considered distribution we have $D_{s,c} = D_{s,c-\lfloor c \rfloor} + \lfloor c \rfloor$. Now recall that the probability integral transform states that if X is a continuous random variable with cumulative distribution function cdf, then $\mathrm{cdf}(X)$ has a uniform distribution on $[0,1]$. Hence the inversion method: $\mathrm{cdf}^{-1}(U_{[0,1]})$ has the same distribution as X. Finally by noting that for all $s, p \in \mathbb{R}$, $\mathrm{cdf}_{s,c}(p)$ is monotonic in c, if $\mathrm{cdf}^{-1}_{s,c_1}(p) = \mathrm{cdf}^{-1}_{s,c_2}(p) := v$, then $\mathrm{cdf}^{-1}_{s,c}(p) = v$ for all $c \in [c_1, c_2]$, and as a consequence, for all $v \in [-\lceil \tau s \rceil - 1, \lceil \tau s \rceil + 1]$, the probability of outputting v is equal to $\mathbb{FP}_m : \mathrm{cdf}_{s,c}(v) - \mathbb{FP}_m : \mathrm{cdf}_{s,c}(v-1)$ which is $2^{-\lambda}$-close to $D_{s,c}(v)$. □

The remaining issue in the correctness analysis of Algorithm 2 is to determine the error occurring during the m-precision cdf computation. Indeed, this error allows us to learn what precision m is needed to correctly compute the λ most significant bits of the cdf. This error is characterized in Lemma 4.

Lemma 4. *Let $m \in \mathbb{Z}$ be a positive integer and $\varepsilon = 2^{1-m}$. Let $\bar{c}, \bar{s}, \bar{h} \in \mathbb{FP}_m$ be at distance respectively at most δ_c, δ_c and δ_h from $c, s, h \in \mathbb{R}$ and $h = 1/\rho_{s,c}(\mathbb{Z})$. Let $\Delta f(x) := |\mathbb{FP}_m : f(x) - f(x)|$. We also assume that the following inequalities hold: $s \geq 4$, $\tau \geq 10$, $s\delta_s \leq 0.01$, $\delta_c \leq 0.01$, $s^2\varepsilon \leq 0.01$, $(\tau s + 1)\varepsilon \leq 1/2$. We have the following error bound on $\Delta \mathrm{cdf}_{s,c}(x)$ for any integer x such that $|x| \leq \tau s + 2$*

$$\Delta \mathrm{cdf}_{s,c}(x) \leq 3.5\tau^3 s^2 \varepsilon$$

Proof. We derive the following bounds using [10, Facts 6.12, 6.14, 6.22]:

$$\Delta \mathrm{cdf}_{s,c}(x) \leq \Delta \left[\sum_{i=-\lceil \tau s \rceil - 1}^{\lceil \tau s \rceil + 1} \rho_{s,c}(i) \right] \left(\frac{1}{s} + 3.6 s \varepsilon \right) + 3.6 s \varepsilon$$

$$\Delta \left[\sum_{i=-\lceil \tau s \rceil - 1}^{\lceil \tau s \rceil + 1} \rho_{s,c}(i) \right] \leq 3.2 \tau^3 s^3 \varepsilon$$

□

For the sake of readability the FPA error bound of Lemma 4 is fully simplified and is therefore not tight. For practical implementation, one can derive a better bound using an ad-hoc approach such as done in [34].

Efficiency. On average, the evaluation of the cdf requires $\lceil \tau s \rceil + 1.5$ evaluations of the exponential function. For the sake of clarity, we assume that the exponential function is computed using a direct power series evaluation with schoolbook multiplication, so its time complexity is $\mathcal{O}(\lambda^3)$. We refer the reader to [6] for a discussion of different ways to compute the exponential function in high-precision.

Lemma 5 establishes that the time complexity of Algorithm 2 is $\mathcal{O}(\lambda \log s\lambda + \lambda^4/n)$, so with $n = \mathcal{O}(\lambda^3)$ it has asymptotically the same computational cost than the classical CDT algorithm.

Lemma 5. *Let $\mathsf{P}_{\mathsf{cdf}}$ be the probability of computing the* cdf *during the execution of Algorithm 2, assuming that $\tau s \geq 10$, we have*

$$\mathsf{P}_{\mathsf{cdf}} \leq 2.2\tau s \left(1 - e^{-\frac{1.25\tau}{sn}} \Delta_{\mathsf{measure}} \right)$$

Proof.

$$\mathsf{P}_{\mathsf{cdf}} \leq \max_{c\in[0,1)} \left(\sum_{i=-\lceil \tau s \rceil -1}^{\lceil \tau s \rceil +1} \left| \mathrm{cdf}_{s,c}(i) - cdf_{s,c+\frac{1}{n}}(i) \right| \right)$$

Assuming that $\tau s \geq 10$, we have

$$e^{-\frac{1.25\tau}{sn}} \Delta_{\mathsf{measure}}\, \mathrm{cdf}_{s,c}(i) \leq \mathrm{cdf}_{s,c+\frac{1}{n}}(i) \leq \mathrm{cdf}_{s,c}(i)$$

Hence the upper bound. □

On the other hand, the precomputation matrix generated by Algorithm 1 take n times the size of one CDT, hence the space complexity is $\mathcal{O}(ns\lambda^{1.5})$. Note that for n sufficiently big to make the cdf computational cost negligible, the memory space required by this algorithm is about 1 GB for the parameters considered in cryptography and thus prohibitively expensive for practical use.

3.2 Lazy-CDT Algorithm

A first idea to decrease the number of precomputed CDTs is to avoid costly cdf evaluations by using the same lazy trick as in [12] for rejection sampling. Indeed, a careful analysis of Algorithm 2 shows most of the time many of the computed cdf bits are not used. This gives us to a new strategy which consists of computing the bits of $\mathrm{cdf}_{s,c}(v_1)$ lazily. When the values corresponding to the generated probability for the two closest centers are different, the *Lazy-CDT* algorithm first only computes the cdf at a precision m' to ensure $k < \lambda$ correct bits. If the comparison is decided with those k bits, it returns the sample. Otherwise, it recomputes the cdf at a precision m to ensure λ correct bits.

Correctness. In addition to the choice of m, discussed in Sect. 3.1, to achieve λ bits of precision, the correctness of Algorithm 3 also requires to know k which is the number of correct bits after the floating-point computation of the cdf with m' bits of mantissa. For this purpose, given m' Lemma 4 provides a theoretical lower bound on k.

Efficiency. As explained in [12] the precision used for floating-point arithmetic has non-negligible impact, because fp-operation become much expensive when the precision goes over the hardware precision. For instance, modern processors typically provide floating-point arithmetic following the double IEEE standard double precision ($m = 53$), but quad-float FPA ($m = 113$) is usually about 10–20 times slower for basic operations, and the overhead is much more for multiprecision FPA. Therefore the maximal hardware precision is a natural choice for m'. However this choice for m' in Algorithm 3 is a strong constraint for cryptographic applications, where the error occurring during the floating-point cdf computation is usually greater than 10 bits, making the time-memory tradeoff of Algorithm 3 inflexible. Note that the probability of triggering high precision in Algorithm 3 given that $v_1 \neq v_2$ is about $2^{q-k}\mathsf{P}_{\mathsf{cdf}}$, where q is the number of

Algorithm 3. Lazy-CDT Algorithm: Online Phase

```
Input: a center $c$ and a precomputed matrix $\mathbf{T}$
Output: a sample $x$ that follows $D_{s,c}$
$p \leftarrow U_{[0,1)\cap\mathbb{FP}_\lambda}$
$v_1 \leftarrow i - \lceil \tau s \rceil - 1$ s.t. $\mathbf{T}_{\lfloor n(c-\lfloor c \rfloor)\rfloor, i-1} \leq p < \mathbf{T}_{\lfloor n(c-\lfloor c \rfloor)\rfloor, i}$
$v_2 \leftarrow j - \lceil \tau s \rceil - 1$ s.t. $\mathbf{T}_{\lceil n(c-\lfloor c \rfloor)\rceil, j-1} \leq p < \mathbf{T}_{\lceil n(c-\lfloor c \rfloor)\rceil, j}$
if $v_1 = v_2$ then
    return $v_1 + \lfloor c \rfloor$
else
    if $\mathbb{FP}_k : p < \mathbb{FP}_{m'} : \mathrm{cdf}_{s,c-\lfloor c \rfloor}(v_1)$ then
        return $v_1 + \lfloor c \rfloor$
    else
        if $\mathbb{FP}_k : p > \mathbb{FP}_{m'} : \mathrm{cdf}_{s,c-\lfloor c \rfloor}(v_1)$ then
            return $v_2 + \lfloor c \rfloor$
        else
            if $p > \mathbb{FP}_m : \mathrm{cdf}_{s,c-\lfloor c \rfloor}(v_1)$ then
                return $v_1 + \lfloor c \rfloor$
            else
                return $v_2 + \lfloor c \rfloor$
```

common leading bits of $\mathrm{cdf}_{s,\lfloor n(c-\lfloor c \rfloor)\rfloor/n}(v_1)$ and $\mathrm{cdf}_{s,\lceil n(c-\lfloor c \rfloor)\rceil/n}(v_2)$. By using this lazy trick in addition to lookup tables as described in Sect. 5 with parameters considered in cryptography, we achieve a computational cost lower than the classical centered CDT algorithm with a memory requirement in the order of 1 megabyte.

4 A More Flexible Time-Memory Tradeoff

In view of limitations of the lazy approach described above, a natural question is if we can find a better solution to approximate the cdf. The major advantage of this lazy trick is that it does not require additional memory. However, in our context the CDTs are precomputed and rather than approximate the cdf from scratch it would be interesting to reuse the information contained in these precomputations. Consider the cdf as a function of the center and note that each precomputed cdf is zero degree term of the Taylor expansion of the cdf around a precomputed center. Hence, we may approximate the cdf by its Taylor expansions by precomputing some higher degree terms.

At a first glance, this seems to increase the memory requirements of the sampling algorithm, but we will show that this approach allows to drastically reduce the number of precomputed to a $\omega(\lambda)$ centers thanks to a probability which decreases rapidly with the degree of the Taylor expansion. Moreover, this approximation is faster than the cdf lazy computation and it has no strong constraints related to the maximal hardware precision. As a result, we obtain a flexible time-memory tradeoff which reaches, in particular, the same time complexity as the CDT algorithm for centered discrete Gaussians with a practical memory requirements for cryptographic parameters.

4.1 Taylor-CDT Algorithm

Our *Taylor-CDT* algorithm is similar to the *Lazy-CDT* algorithm (Algorithm 3) described above, except that the lazy computation of the cdf is replaced by the Taylor expansion of the cdf, viewed as a function of the Gaussian center, around each precomputed centers for all possible values. The zero-degree term of each of these Taylor expansions is present in the corresponding CDT element $\mathbf{T}_{i,j}$ and the d higher-degree terms are stored as an element $\mathbf{E}_{i,j}$ of another matrix $\mathbf{E}$. As for the other approaches, these precomputations shall be performed at a sufficient precision m to ensure λ correct bits. During the online phase, Algorithm 5 proceed as follow. Draw p from the uniform distribution over $[0,1) \cap \mathbb{FP}_\lambda$ and search p in the CDTs of the two closest precomputed centers to the desired center decimal part. If the two values found are equal, add the desired center integer part to this value and return it as a valid sample. Otherwise, select the closest precomputed center to the desired center decimal part and evaluate, at the desired center decimal part, the Taylor expansion corresponding to this center and the value found in its CDT. If p is smaller or bigger than this evaluation with respect for the error approximation upper bound $\mathsf{E}_{\mathsf{expansion}}$, characterized in Lemma 6, add the desired center integer part to the corresponding value and return it as a valid sample. Otherwise, it is necessary to compute the full cdf to decide which value to return.

Algorithm 4. Taylor-CDT Algorithm: Offline Phase

```
Input: a Gaussian parameter $s$, a number of centers $n$, a Taylor expansion degree $d$
Output: two precomputed matrices $\mathbf{T}$ and $\mathbf{E}$
initialize two empty matrices $\mathbf{T} \in \mathbb{FP}_\lambda^{n \times 2\lceil \tau s \rceil + 3}$ and $\mathbf{E} \in (\mathbb{FP}_\lambda^d)^{n \times 2\lceil \tau s \rceil + 3}$
for $i \leftarrow 0, \ldots, n-1$ do
    for $j \leftarrow 0, \ldots, 2\lceil \tau s \rceil + 2$ do
        $\mathbf{T}_{i,j} \leftarrow \mathbb{FP}_m : \mathrm{cdf}_{s,i/n}(j - \lceil \tau s \rceil - 1)$
        $\mathbf{E}_{i,j} \leftarrow \mathbb{FP}_m : \mathcal{T}_{d,\mathrm{cdf}_{s,x}(j-\lceil \tau s \rceil - 1),i/n}(x) - \mathbf{T}_{i,j}$
```

Efficiency. Algorithm 5 performs two binary searches on CDTs in $\mathcal{O}(\lambda \log s\lambda)$, d additions and multiplications on $\mathbb{FP}_m$ in $\mathcal{O}(m^2)$ with probability $\mathsf{P}_{\mathsf{cdf}} \approx 3\lambda/n$ (see Lemma 5) and a cdf computation on $\mathbb{FP}_m$ in $\mathcal{O}(s\lambda^{3.5})$ with probability close to $2^{q+1}\mathsf{P}_{\mathsf{cdf}}\mathsf{E}_{\mathsf{expansion}}$, where q is the number of common leading bits of $\mathrm{cdf}_{s,\lfloor n(c-\lfloor c \rfloor)\rfloor/n}(v_1)$ and $\mathrm{cdf}_{s,\lceil n(c-\lfloor c \rfloor)\rceil/n}(v_2)$ and $\mathsf{E}_{\mathsf{expansion}}$ is the Taylor expansion approximation error bound described in Lemma 6.

Lemma 6. *Let $E_{expansion}$ be the maximal Euclidean distance between $\mathrm{cdf}_{s,x}(v)$ and $\mathcal{T}_{d,\mathrm{cdf}_{s,x}(v),c}(x)$, its Taylor expansion around c, for all $v \in [-\lceil \tau s \rceil - 1, \lceil \tau s \rceil + 1]$, $c \in [0,1)$ and $x \in [c, c + 1/2n]$, assuming that $\tau \geq 2.5$, $s \geq 4$, we have*

$$E_{expansion} < \frac{4\tau^{d+2}}{n^{d+1} s^{\frac{d+1}{2}}}$$

Algorithm 5. Taylor-CDT Algorithm: Online Phase

```
Input: a center $c$ and two precomputed matrices $\mathbf{T}$ and $\mathbf{E}$
Output: a sample $x$ that follows $D_{s,c}$
$p \leftarrow U_{[0,1)\cap\mathbb{FP}_\lambda}$
$v_1 \leftarrow i - \lceil \tau s \rceil - 1$ s.t. $\mathbf{T}_{\lfloor n(c-\lfloor c \rfloor)\rfloor, i-1} \leq p < \mathbf{T}_{\lfloor n(c-\lfloor c \rfloor)\rfloor, i}$
$v_2 \leftarrow j - \lceil \tau s \rceil - 1$ s.t. $\mathbf{T}_{\lceil n(c-\lfloor c \rfloor)\rceil, j-1} \leq p < \mathbf{T}_{\lceil n(c-\lfloor c \rfloor)\rceil, j}$
if $v_1 = v_2$ then
    return $v_1 + \lfloor c \rfloor$
else
    if $|c - \lfloor n(c - \lfloor c \rfloor)\rfloor| < |c - \lceil n(c - \lfloor c \rfloor)\rceil|$ then
        $c' \leftarrow \lfloor n(c - \lfloor c \rfloor)\rfloor$
    else
        $c' \leftarrow \lceil n(c - \lfloor c \rfloor)\rceil$
        $i \leftarrow j$
    if $p < \mathbf{T}_{c',i} + \mathbf{E}_{c',i}(c - \lfloor c \rfloor) - \mathsf{E}_{\mathsf{expansion}}$ then
        return $v_1 + \lfloor c \rfloor$
    else
        if $p > \mathbf{T}_{c',i} + \mathbf{E}_{c',i}(c - \lfloor c \rfloor) + \mathsf{E}_{\mathsf{expansion}}$ then
            return $v_2 + \lfloor c \rfloor$
        else
            if $p > \mathbb{FP}_m : \mathrm{cdf}_{s,c-\lfloor c \rfloor}(v_1)$ then
                return $v_1 + \lfloor c \rfloor$
            else
                return $v_2 + \lfloor c \rfloor$
```

Proof. From Theorem 1 we have

$$\mathsf{E}_{\mathsf{expansion}} = \max_{\substack{c\in[0,1)\\ x\in[c,c+1/2n]\\ v\in[-\lceil \tau s\rceil-1,\lceil \tau s\rceil+1]}} \left(\sum_{i=-\lceil \tau s \rceil -1}^{v} \int_c^x \frac{\rho_{s,t}^{(d+1)}(i)}{d!\,\rho_{s,t}(\mathbb{Z})} \left(c + \frac{1}{2n} - t\right)^d dt \right)$$

By using well-known series-integral comparison we obtain $\rho_{s,t}(\mathbb{Z}) \geq s\sqrt{2\pi} - 1$ and since $\left|\rho_{s,t}^{(d)}(i)\right| < \frac{d(1.3\tau)^d 2^d}{s^{d/2}}$ for $s \geq 4$ and $\tau \geq 2.5$, it follows that

$$\mathsf{E}_{\mathsf{expansion}} \leq \frac{(d+1)(1.3)^{d+1}\tau^{d+2}}{d!\,n^{d+1}s^{\frac{d+1}{2}}}$$

□

A careful analysis of this technique show that with $d = 4$ we achieve the same asymptotic computational cost as the classical CDT algorithm with $n = \omega(\lambda)$, where the hidden factor is less than 1/4, therefore for this degree the space complexity of Algorithms 4 and 5 is only λ times bigger than for centered sampling, showing that these algorithms can achieve a memory requirement as low as 1 MB. Finally, note that taking care to add the floating-point computation error to the error of approximation, one can compute the Taylor expansion evaluation at the maximal hardware precision to reduce its computational cost.

5 Lookup Tables

We shall now show how to use partial lookup tables to avoid the binary search in most cases when using CDT algorithms, this technique is the CDT analogue of the Knuth-Yao algorithm improvement described in [8]. Note that this strategy is particularly fitting for discrete Gaussian distributions with relatively small expected values. The basic idea is to subdivide the uniform distribution $U_{[0,1)}$ into ℓ uniform distributions on subsets of the same size $U_{[i/\ell,(i+1)/\ell)}$, with ℓ a power of two. We then precompute a partial lookup table on these subsets which allows to return the sample at once when the subset considered does not include a cdf image. We note that instead of subdividing the uniform range into stripes of the same size, we can also recursively subdivide only some stripes of the previous subdivision. However, for the sake of clarity and ease of exposure, this improvement is not included in this paper and we will describe this technique for the classical centered CDT algorithm.

First, we initialize a lookup table of size $\ell = 2^l$ where the i-th entry corresponds to a subinterval $[i/\ell, (i+1)/\ell)$ of $[0, 1)$. Second, after precomputing the CDT, we mark all the entries for which there is at least one CDT element in their corresponding subinterval $[i/\ell, (i+1)/\ell)$ with $\perp$, and all remaining entries with $\top$. Each entry marked with $\top$ allows to return a sample without the need to perform a binary search in the CDT, because only one value corresponds to this subinterval which is the first CDT element greater or equal to $(i + 1)/\ell$.

Efficiency. The efficiency of this technique is directly related to the number of entries, marked with $\top$, whose subintervals do not contain a CDT element. We denote the probability of performing binary search by $\mathsf{P}_{\mathsf{binsrch}}$, obviously the probability to return the sample immediately after choosing i, which is a part of p, is $1 - \mathsf{P}_{\mathsf{binsrch}}$. Lemma 7 gives a lower bound of $\mathsf{P}_{\mathsf{binsrch}}$.

Lemma 7. *For any $\ell \geq 2^8$ and $s \geq \eta_{\frac{1}{2}}(\mathbb{Z})$. Let $\mathsf{P}_{\mathit{binsrch}}$ be the probability of performing binary search during the execution of the CDT algorithm implemented with the lookup table trick described above, we have*

$$\mathsf{P}_{\mathit{binsrch}} < 1.2s\sqrt{\log_2 \ell}/\ell$$

Proof.

$$\mathsf{P}_{\mathsf{binsrch}} = \frac{\ell - \sum_{i=\lfloor c-\tau s\rfloor}^{\lceil c+\tau s\rceil} \lfloor \ell \operatorname{cdf}_{s,c}(i)\rfloor - \lfloor \ell \operatorname{cdf}_{s,c}(i-1)\rfloor}{\ell}$$

From Lemma 2 we have

$$\left\lfloor \ell \operatorname{cdf}_{s,c}\left(\left\lfloor c - 0.6s\sqrt{\log_2 \ell}\right\rfloor\right)\right\rfloor = 0$$

$$\left\lfloor \ell\left(1 - \operatorname{cdf}_{s,c}\left(\left\lceil c + 0.6s\sqrt{\log_2 \ell}\right\rceil\right)\right)\right\rfloor = 0$$

□

6 Experimental Results

In this section, we present experimental results of our C++ implementation[1] distributed under the terms of the GNU General Public License version 3 or later (GPLv3+) which uses the MPFR [15] and GMP [19] libraries as well as Salsa20 [5] as the pseudorandom number generator. Our non-centered discrete Gaussian sampler was implemented with a binary search executed byte by byte if $\ell = 2^8$ and 2-bytes by 2-bytes if $\ell = 2^{16}$ without recursive subdivision of $U_{[0,1)}$, therefore $[0, 1)$ is subdivided in ℓ intervals of the same size and $\mathrm{cdf}(x)$ is stored for all $x \in [-\lceil\tau\sigma\rceil - 1, \lceil\tau\sigma\rceil + 1]$. The implementation of our non-centered discrete Gaussian sampler uses a fixed number of precomputed centers $n = 2^8$ with a lookup table of size $\ell = 2^8$ and includes the lazy cdf evaluation optimization.

We tested the performance of our non-centered discrete Gaussian sampler by using it as a subroutine for Peikert's sampler [32] for sampling from $D_{(g),\sigma',0}$ with $g \in \mathbb{Z}[x]/(x^N + 1)$ for N a power of two. To this end, we adapted the implementation of this sampler from [3] where we swap out the sampler from

Table 1. Performance of sampling from $D_{(g),\sigma'}$ as implemented in [3] and with our non-centered discrete Gaussian sampler with $\ell = n = 2^8$. The column $D_{(g),\sigma'}/s$ gives the number of samples returned per second, the column "memory" the maximum amount of memory consumed by the process. All timings are on a Intel(R) Xeon(R) CPU E5-2667 (strombenzin). Precomputation uses 2 cores, the online phase uses one core.

[3]					
N	$\log\sigma'$	precomp	time	$D_{(g),\sigma'}/s$	memory
256	38.2	0.08 s	8.46 ms	118.17	11,556 kB
512	42.0	0.17 s	16.96 ms	58.95	11,340 kB
1024	45.8	0.32 s	38.05 ms	26.28	21,424 kB
2048	49.6	0.93 s	78.17 ms	12.79	41,960 kB
4096	53.3	2.26 s	157.53 ms	6.35	86,640 kB
8192	57.0	6.08 s	337.32 ms	2.96	192,520 kB
16384	60.7	13.36 s	700.75 ms	1.43	301,200 kB
This work					
N	$\log\sigma'$	precomp	time	$D_{(g),\sigma'}/s$	memory
256	38.2	0.31 s	2.91 ms	343.16	17,080 kB
512	42.0	0.39 s	5.99 ms	166.88	21,276 kB
1024	45.8	0.65 s	11.89 ms	84.12	38,280 kB
2048	49.6	1.04 s	25.07 ms	39.89	74,668 kB
4096	53.3	2.35 s	48.63 ms	20.56	148,936 kB
8192	57.0	7.27 s	96.67 ms	10.34	302,616 kB
16384	60.7	14.41 s	205.35 ms	4.87	618,448 kB

[1] The implementation is available at https://github.com/tricosset/FGN.

the dgs library [2] (implementing rejection sampling and [11]) used in [3] with our sampler for sampling for $D_{\mathbb{Z},\sigma,c}$. Note that sampling from $D_{(g),\sigma',0}$ is more involved and thus slower than sampling from $D_{\mathbb{Z}^N,\sigma',0}$. That is, to sample from $D_{(g),\sigma',0}$, [3] first computes an approximate square root of $\Sigma_2 = \sigma'^2 \cdot g^{-T} \cdot g^{-1} - r^2$ with $r = 2 \cdot \lceil \sqrt{\log N}\, \rceil$. Then, given an approximation $\sqrt{\Sigma_2}'$ of $\sqrt{\Sigma_2}$ it samples a vector $x \leftarrow_{\$} \mathbb{R}^N$ from a standard normal distribution and interpret it as a polynomial in $\mathbb{Q}[X]/(x^N + 1)$; computes $y = \sqrt{\Sigma_2}' \cdot x$ in $\mathbb{Q}[X]/(x^N + 1)$ and returns $g \cdot (\lfloor y \rceil_r)$, where $\lfloor y \rceil_r$ denotes sampling a vector in $\mathbb{Z}^N$ where the i-th component follows $D_{\mathbb{Z},r,y_i}$. Thus, implementing Peikert's sampler requires sampling from $D_{\mathbb{Z},r,y_i}$ for changing centers y_i and sampling from a standard normal distribution. We give experimental results in Table 1, indicating that our sampler increases the rate by a factor ≈ 3.

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC 1996, NY, USA, pp. 99–108. ACM, New York (1996)
2. Albrecht, M.R.: dgs – discrete gaussians over the integers (2014). https://bitbucket.org/malb/dgs
3. Albrecht, M.R., Cocis, C., Laguillaumie, F., Langlois, A.: Implementing candidate graded encoding schemes from ideal lattices. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 752–775. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48800-3_31
4. Babai, L.: On Lovász' lattice reduction and the nearest lattice point problem. In: Mehlhorn, K. (ed.) STACS 1985. LNCS, vol. 182, pp. 13–20. Springer, Heidelberg (1985). doi:10.1007/BFb0023990
5. Bernstein, D.J.: The salsa20 family of stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs: The eSTREAM Finalists, pp. 84–97. Springer, Heidelberg (2008)
6. Brent, R.P., et al.: Fast algorithms for high-precision computation of elementary functions. In: Proceedings of 7th Conference on Real Numbers and Computers (RNC 7), pp. 7–8 (2006)
7. Buchmann, J., Cabarcas, D., Göpfert, F., Hülsing, A., Weiden, P.: Discrete Ziggurat: a time-memory trade-off for sampling from a Gaussian distribution over the integers. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 402–417. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43414-7_20
8. de Clercq, R., Roy, S.S., Vercauteren, F., Verbauwhede, I.: Efficient software implementation of ring-LWE encryption. In: 2015 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 339–344 (2015)
9. Devroye, L.: Non-Uniform Random Variate Generation. Springer, Heidelberg (1986)
10. Ducas, L.: Lattice based signatures: attacks, analysis and optimization. Ph.D. thesis (2013)
11. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 40–56. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_3

12. Ducas, L., Nguyen, P.Q.: Faster Gaussian lattice sampling using lazy floating-point arithmetic. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 415–432. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34961-4_26
13. Dwarakanath, N.C., Galbraith, S.D.: Sampling from discrete Gaussians for lattice-based cryptography on a constrained device. Appl. Algebra Eng. Commun. Comput. **25**(3), 159–180 (2014)
14. Fiat, A., Shamir, A.: How To prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). doi:10.1007/3-540-47721-7_12
15. Fousse, L., Hanrot, G., Lefèvre, V., Pélissier, P., Zimmermann, P.: MPFR: a multiple-precision binary floating-point library with correct rounding. ACM Trans. Math. Softw. **33**(2) (2007)
16. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, STOC 2008, pp. 197–206. ACM, New York (2008)
17. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, pp. 197–206. ACM Press, Victoria, 17–20 May 2008
18. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski, B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (1997). doi:10.1007/BFb0052231
19. Granlund, T.: The GMP development team: GNU MP: The GNU Multiple Precision Arithmetic Library, 6.0.1 edn. (2015). http://gmplib.org/
20. Karney, C.F.F.: Sampling exactly from the normal distribution. ACM Trans. Math. Softw. **42**(1), 3:1–3:14 (2016)
21. Klein, P.: Finding the closest lattice vector when it's unusually close. In: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2000, pp. 937–941. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2000)
22. Knuth, D.E., Yao, A.C.: The complexity of nonuniform random number generation. In: Traub, J.F. (ed.) Algorithms and Complexity: New Directions and Recent Results. Academic Press, New York (1976)
23. Lyubashevsky, V.: Lattice-based identification schemes secure under active attacks. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 162–179. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78440-1_10
24. Lyubashevsky, V.: Fiat-Shamir with aborts: applications to lattice and factoring-based signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 598–616. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10366-7_35
25. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29011-4_43
26. Lyubashevsky, V., Micciancio, D.: Asymptotically efficient lattice-based digital signatures. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 37–54. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78524-8_3
27. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_1
28. Marsaglia, G., Tsang, W.W.: A fast, easily implemented method for sampling from decreasing or symmetric unimodal density functions. SIAM J. Sci. Stat. Comput. **5**, 349–359 (1984)

29. Micciancio, D.: Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. Comput. Complex. **16**(4), 365–411 (2007)
30. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. SIAM J. Comput. **37**(1), 267–302 (2007)
31. von Neumann, J.: Various techniques used in connection with random digits. J. Res. Nat. Bur. Stand. **12**, 36–38 (1951)
32. Peikert, C.: An efficient and parallel Gaussian sampler for lattices. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 80–97. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14623-7_5
33. Peikert, C.: A decade of lattice cryptography. Found. Trends Theor. Comput. Sci. **10**(4), 283–424 (2016)
34. Pujol, X., Stehlé, D.: Rigorous and efficient short lattice vectors enumeration. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 390–405. Springer, Heidelberg (2008). doi:10.1007/978-3-540-89255-7_24
35. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, STOC 2005, NY, USA, pp. 84–93. ACM, New York (2005)
36. Saarinen, M.J.O.: Arithmetic coding and blinding countermeasures for lattice signatures. J. Cryptographic Eng. 1–14 (2017)
37. Von Neumann, J.: The general and logical theory of automata. Cerebral Mech. Behav. **1**(41), 1–2 (1951)
38. Walker, A.J.: New fast method for generating discrete random numbers with arbitrary frequency distributions. Electron. Lett. **10**, 127–128 (1974)

Simple Security Definitions for and Constructions of 0-RTT Key Exchange

Britta Hale[1](✉), Tibor Jager[2], Sebastian Lauer[3], and Jörg Schwenk[3]

[1] NTNU, Norwegian University of Science and Technology, Trondheim, Norway
britta.hale@ntnu.no
[2] Paderborn University, Paderborn, Germany
tibor.jager@upb.de
[3] Horst Görtz Institute, Ruhr-University Bochum, Bochum, Germany
{sebastian.lauer,joerg.schwenk}@rub.de

Abstract. Zero Round-Trip Time (0-RTT) key exchange protocols allow for the transmission of cryptographically protected payload data without requiring the prior exchange of messages of a cryptographic key exchange protocol. The 0-RTT KE concept was first realized by Google in the QUIC Crypto protocol, and a 0-RTT mode has been intensively discussed for inclusion in TLS 1.3.

In 0-RTT KE two keys are generated, typically using a Diffie-Hellman key exchange. The first key is a combination of an ephemeral client share and a long-lived server share. The second key is computed using an ephemeral server share and the same ephemeral client share.

In this paper, we propose simple security models, which catch the intuition behind known 0-RTT KE protocols; namely that the first (resp. second) key should remain indistinguishable from a random value, even if the second (resp. first) key is revealed. We call this property *strong key independence*. We also give the first constructions of 0-RTT KE which are provably secure in these models, based on the generic assumption that secure non-interactive key exchange (NIKE) exists (This work was partially supported by a STSM Grant from COST Action IC1306).

Keywords: Foundations · Low-latency key exchange · 0-RTT protocols · Authenticated key exchange · Non-interactive key exchange · QUIC · TLS 1.3.

1 Introduction

Efficiency, in terms of messages to be exchanged before a key is established, is a growing consideration for internet protocols today. Basically, the first generation of internet key exchange protocols did not care too much about efficiency, since secure connections were considered to be the exception rather than the rule: SSL (versions 2.0 and 3.0) and TLS (versions 1.0, 1.1, and 1.2) require 2 round-trip times (RTT) for key establishment before the first cryptographically-protected payload data can be sent. With the increased use of encryption,[1] efficiency is

[1] For example, initiatives like Let's Encrypt (https://letsencrypt.org/).

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 20–38, 2017.
DOI: 10.1007/978-3-319-61204-1_2

of escalating importance for protocols like TLS. Similarly, the older IPSec IKE version v1 needs between 3 RTT (aggressive mode + quick mode) and 4.5 RTT (main mode + quick mode). This was soon realized to be problematic, and in IKEv2 the number of RTTs was reduced to 2.

The QUIC Protocol. Fundamentally, the discussion on low-latency key exchange (aka. LLKE, zero-RTT or 0-RTT key exchange) was opened when Google proposed the QUIC protocol.[2] QUIC (cf. Fig. 1) achieves low-latency by caching a signed server configuration file on the client side, which contains a medium-lived Diffie-Hellman (DH) share $Y_0 = g^{y_0}$.[3]

When a client wishes to establish a connection with a server and possesses a valid configuration file of that server, it chooses a fresh ephemeral DH share $X = g^x$ and computes a temporal key k_1 from $g^{y_0 x}$. Using this key k_1, the client can encrypt and authenticate data to be sent to the server, together with X. In response, the server sends a fresh DH share $Y = g^y$ and computes a session key k_2 from g^{xy}, which is used for all subsequent data exchanges.

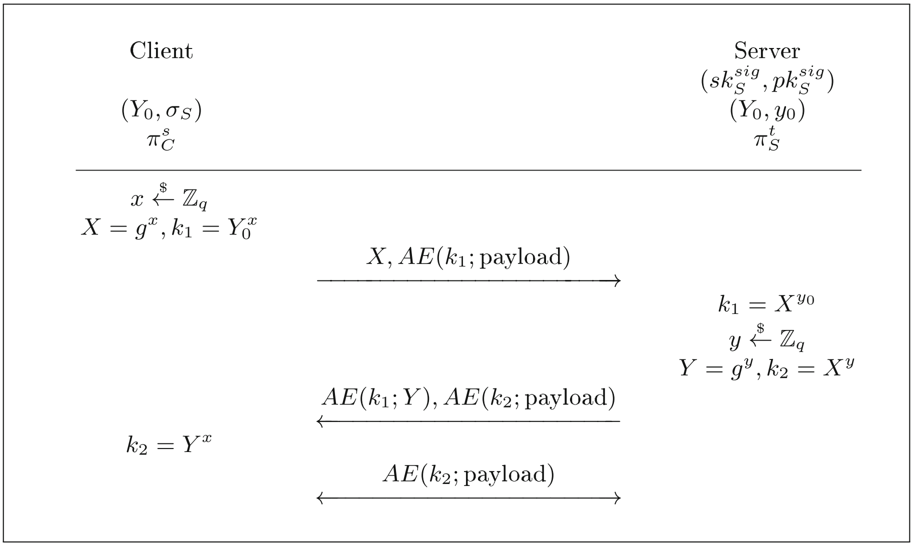

Client
(Y_0, σ_S)
π_C^s
Server
(sk_S^{sig}, pk_S^{sig})
(Y_0, y_0)
π_S^t
$x \xleftarrow{\$} \mathbb{Z}_q$
$X = g^x, k_1 = Y_0^x$
$X, AE(k_1; \text{payload})$
$k_1 = X^{y_0}$
$y \xleftarrow{\$} \mathbb{Z}_q$
$Y = g^y, k_2 = X^y$
$AE(k_1; Y), AE(k_2; \text{payload})$
$k_2 = Y^x$
$AE(k_2; \text{payload})$

Fig. 1. Google's QUIC protocol (simplified) with cached server key configuration file (Y_0, σ_S). AE denotes a symmetric authenticated encryption algorithm (e.g., AES-GCM), (sk_S^{sig}, pk_S^{sig}) denotes the server's long-term signing keys, and π_S^t (resp. π_C^s) denotes the oracle at server S executing the single t-th instance of the protocol (resp. for client).

[2] See https://www.chromium.org/quic.

[3] If the client does not have a valid file, it has to be requested from the server, which increases the number of RTTs by 1, but may then be re-used for future sessions.

TLS 1.3. Early TLS 1.3 drafts, e.g. `draft-ietf-tls-tls13-08` [25], contained a 0-RTT key exchange mode where a QUIC-like `ServerConfiguration` message is cached by the client. The current version `draft-ietf-tls-tls13-18` [26] follows a different approach, where the initial key establishment between a client and a server is never 0-RTT. Instead, it defines a method to establish a new session based on the secret key of a previous session. Even though this is also called "0-RTT" in the current TLS 1.3 specification, it is rather a "0-RTT session resumption" protocol, but does not allow for 0-RTT key establishment. Most importantly, the major difference between the approach of the current TLS 1.3 draft in comparison to a "real" 0-RTT key exchange protocol is that the former requires storing of *secret* key information on the client between sessions. In contrast, a 0-RTT key establishment protocol does not require secret information to be stored between sessions.

Facebook's Zero Protocol. Very recently, the social network Facebook announced that it is currently experimenting with a 0-RTT KE protocol called *Zero*.[4] Zero is very similar to QUIC, except that it uses another nonce and encryption of the ServerHello message. It is noteworthy that the main difference between Zero and QUIC was introduced in order to prevent an attack discovered by Facebook, which has been reported to Google and meanwhile been fixed in QUIC, too. We believe that this is a good example that shows the demand of simple security definitions and provably-secure constructions for such protocols.

Security Goals. 0-RTT KE protocols like QUIC have ad-hoc designs that aim at achieving three goals: (1) 0-RTT encryption, where ciphertext data can already be sent together with the first handshake message; (2) perfect forward secrecy (PFS), where all ciphertexts exchanged after the second handshake message will remain secure even after the (static or semi-static) private keys of the server have been leaked, and (3) key independence, where "knowledge" about one of the two symmetric keys generated should not endanger the security of the other key.

Strong Key Independence. Intuitively, a 0-RTT KE protocol should achieve *strong key independence* between k_1 and k_2; if any one of the two keys is leaked at any time, the other key should still be indistinguishable from a random value. In all known security models, this intuition would be formalized as follows: if the adversary $\mathcal{A}$ asks a Reveal query for k_1, he is still allowed to ask a Test query for k_2, and vice versa. If the two keys are computationally independent from each other (which also includes computations on the different protocol messages), then the adversary should have only a negligible advantage in answering the Test query correctly.

Ultimately this leads to the following research questions: *Do existing examples of 0-RTT KE protocols have strong key independence? Can we describe a generic way to construct 0-RTT KE protocols that provably achieve strong key independence?*

[4] See https://code.facebook.com/posts/608854979307125/building-zero-protocol-for-fast-secure-mobile-connections/.

QUIC Does Not Provide Strong Key Independence. If an attacker $\mathcal{A}$ is allowed to learn k_1 by a Reveal-query, then he is able to decrypt $AE(k_1; Y)$ and re-encrypt its own value $Y^* := g^{y^*}$ as $AE(k_1; Y^*)$. Furthermore, he can then compute the same $k_2 = X^{y^*}$ as the client oracle, and can thus distinguish between the "real" key and a "random" key chosen by the Test query. See [11] for more details on key dependency in QUIC.

Note that this theoretical attack does not imply that QUIC is insecure. It only shows that the authenticity of the server's Diffie-Hellman share, which is sent in QUIC to establish k_2, depends strongly on the security of key k_1. Therefore QUIC does not provide strong key independence in the sense sketched above.

Previous Work on 0-RTT Key Exchange. The concept of 0-RTT key exchange was not developed in academia, but in industry – motivated by concrete practical demands of distributed applications. All previous works on 0-RTT KE [11,23] conducted *a-posteriori* security analyses of the QUIC protocol, with tailored models. There are no foundational constructions as yet, and the relation to other cryptographic protocols and primitives is not yet well-understood.

At ACM CCS 2014, Fischlin and Günther [11] provided a formal definition of *multi-stage* key exchange protocols and used it to analyze the security of QUIC. Lychev *et al.* [23] gave an alternate analysis of QUIC, which considers both efficiency and security. They describe a security model which is bespoke to QUIC, adopting the complex, monolithic security model of [17] to the protocol's requirements. Zhao [31] considers identity-concealed 0-RTT protocols, where user privacy is protected by hiding identities of users in a setting with mutual cryptographic authentication of both communicating parties. Günther *et al.* [14] extended the "puncturable encryption"-approach of Green and Miers [13] to show that even 0-RTT KE with full forward secrecy is possible, by evolving the secret key after each decryption. However, their construction is currently mainly of conceptual interest, as it is not yet efficient enough to be deployed at large scale in practice.

Security Model. In this paper, we use a variant of the Canetti-Krawczyk [7] security model. This family of security models is especially suited to protocols with only two message exchanges, with *one-round* key exchange protocols constituting the most important subclass. Popular examples of such protocols are MQV [22], HMQV [18], SMQV [27], KEA [21,24], and NAXOS [20]. A comparison of different variants of the Canetti-Krawczyk model can be found in [9,29].

The Importance of Simplicity of Security Models. Security models for key exchange protocols have to consider *active* adversaries that may modify, replay, inject, drop, etc., any message transmitted between communicating parties. They also need to capture *parallel* executions of multiple protocol sessions, potential reveals of earlier session keys, and adaptive corruptions of long-term secrets of parties. This makes even standard security models for key exchange *extremely* complex (in comparison to most other standard cryptographic primitives, like digital signatures or public-key encryption, for example).

Naturally, the novel primitive of 0-RTT KE requires formal security definitions. There are different ways to create such a model. One approach is to focus on *generality* of the model. Fischlin and Günther [11] followed this path, by defining *multi-stage* key exchange protocols, a generalization of 0-RTT KE. This approach has the advantage that it lays the foundation for the study of a very general class of interesting and novel primitives. However, its drawback is that this generality inherently also brings a huge *complexity* to the model. Clearly, the more complex the security model, the more difficult it becomes to devise new, simple, efficient, and provably-secure constructions. Moreover, proofs in complex models tend to be error-prone and less intuitive, because central technical ideas may be concealed in formal details that are required to handle the generality of the model.

Another approach is to devise a model which is tailored to the analysis of *one specific protocol*. For example, the complex, monolithic ACCE security model was developed in [17] to provide an *a posteriori* security analysis of TLS.[5] A similar approach was followed by Lychev *et al.* [23], who adopted this model for an *a posteriori* analysis of QUIC, by defining the so-called Q-ACCE model. The notable drawback of this approach is that such tailor-made models tend to capture only the properties achieved by existing protocols, but not necessarily all properties that we would expect from a "good" 0-RTT KE protocol. In general, such tailor-made models do not, therefore, form a useful foundation for the creation of new protocols.

In this paper, we follow a different approach. We propose novel "bare-bone" security models for 0-RTT KE, which aim at capturing *all* (strong key independence and forward secrecy), but also *only* the properties intuitively expected from "good" 0-RTT KE protocols. We propose two different models. One considers the practically-relevant case of *server-only* authentication (where the client may or may not authenticate later over the established communication channel, similar in spirit to the server-only-authenticated ACCE model of [19]). The other considers traditional *mutual* cryptographic authentication of a client and server.

The reduced generality of our definitions – in comparison to the very general multi-stage security model of [11] – is intended. A model which captures *only*, but also *all* the properties expected from a "good" 0-RTT KE protocol allows us to devise relatively simple, foundational, and generic constructions of 0-RTT KE protocols with as-clean-as-possible security analyses.

Importance of Foundational Generic Constructions. Following [3], we use non-interactive key exchange (NIKE) [8,12] in combination with digital signatures as a main building block.[6] This yields the first examples of 0-RTT KE protocols with strong key independence, as well as the first constructions of 0-RTT KE from generic complexity assumptions. There are many advantages of such generic constructions:

[5] A more modular approach was later proposed in [4].

[6] Recall that digital signatures are implied by one-way functions, which in turn are implied by NIKE. Thus, essentially we only assume the existence of NIKE as a building block.

1. Generic constructions provide a better understanding of the structure of protocols. Since the primitives we use have abstract security properties, we can see precisely which abstract security requirements are needed to implement 0-RTT KE protocols.
2. They clarify the relations and implications between different types of cryptographic primitives.
3. They can be generically instantiated with building blocks based on different complexity assumptions. For example, if "post-quantum" security is needed, one can directly obtain a concrete protocol by using only post-quantum secure building blocks in the generic construction.

Usually generic constructions tend to involve more computational overhead than ad-hoc constructions. However, we note that our 0-RTT KE protocols can be instantiated relatively efficiently, given the efficient NIKE schemes of [12], for example.

Contributions. Contributions in this paper can be summarized as follows:

- *Simple security models.* We provide simple security models, which capture all properties that we expect from a "good" 0-RTT KE protocol, but only these properties. We consider both the "practical" setting with server-only authentication and the classical setting with mutual authentication.
- *First generic constructions.* We give intuitive, relatively simple, and efficient constructions of 0-RTT KE protocols in both settings.
- *First Non-DH instantiation.* Both QUIC and TLS 1.3 are based on DH key exchange. Our generic construction yields the first 0-RTT KE protocol which is not based on Diffie-Hellman (e.g., by instantiating the generic construction with the factoring-based NIKE scheme of Freire *et al.* [12]).
- *First 0-RTT KE with strong key independence.* Our 0-RTT KE protocols are the first to achieve strong key independence in the sense described above.
- *Well-established, general assumptions.* The construction is based on general assumptions, namely the existence of secure NIKE and digital signature schemes. For all building blocks we require only standard security properties.
- *Security in the Standard Model.* The security analysis is completely in the standard model, i.e. it is performed without resorting to the Random Oracle heuristic [1] and without relying on non-standard complexity assumptions.
- *Efficient instantiability.* Despite the fact that our constructions are generic, the resulting protocols can be instantiated relatively efficiently.

Full Version of this Paper. Due to space limitations, we have to defer several results to the full version of this paper [15]. This includes the full proof of Theorem 1, the Definition and Security Model for a 0-RTT protocol under mutual authentication ($\mathsf{0\text{-}RTT\text{-}M}$), a construction of a $\mathsf{0\text{-}RTT\text{-}M}$ protocol along with its security model and its security proof.

2 Preliminaries

For our construction in Sect. 4, we need signature schemes and non-interactive key exchange (NIKE) protocols. Here we summarize the definitions of these two primitives and their security from the literature.

2.1 Digital Signatures

A digital signature scheme consists of three polynomial-time algorithm $\mathsf{SIG} = (\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Vfy})$. The key generation algorithm $(sk, pk) \xleftarrow{\$} \mathsf{SIG.Gen}(1^\lambda)$ generates a public verification key pk and a secret signing key sk on input of security parameter λ. Signing algorithm $\sigma \xleftarrow{\$} \mathsf{SIG.Sign}(sk, m)$ generates a signature for message m. Verification algorithm $\mathsf{SIG.Vfy}(pk, \sigma, m)$ returns 1 if σ is a valid signature for m under key pk, and 0 otherwise.

Consider the following security experiment played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger generates a public/secret key pair $(sk, pk) \xleftarrow{\$} \mathsf{SIG.Gen}(1^\lambda)$, the adversary receives pk as input.
2. The adversary may query arbitrary messages m_i to the challenger. The challenger replies to each query with a signature $\sigma_i = \mathsf{SIG.Sign}(sk, m_i)$. Here i is an index, ranging between $1 \leq i \leq q$ for some $q \in \mathbb{N}$. Queries can be made adaptively.
3. Eventually, the adversary outputs a message/signature pair (m, σ).

Definition 1. *We define the advantage on an adversary $\mathcal{A}$ in this game as*

$$\mathtt{Adv}^{sEUF\text{-}CMA}_{\mathsf{SIG},\mathcal{A}}(\lambda) := \Pr\left[(m,\sigma) \xleftarrow{\$} \mathcal{A}^{\mathcal{C}(\lambda)}(pk) : \begin{array}{c} \mathsf{SIG.Vfy}(pk,\sigma,m) = 1, \\ (m,\sigma) \neq (m_i,\sigma_i)\ \forall i \end{array}\right].$$

SIG *is* strongly secure *against* existential forgeries under adaptive chosen-message attacks *(sEUF-CMA), if* $\mathtt{Adv}^{sEUF\text{-}CMA}_{\mathsf{SIG},\mathcal{A}}(\lambda)$ *is a negligible function in* λ *for all probabilistic polynomial-time adversaries* $\mathcal{A}$*.*

Remark 1. Signatures with sEUF-CMA security can be constructed generically from any EUF-CMA-secure signature scheme and chameleon hash functions [6,28].

2.2 Secure Non-interactive Key Exchange

Definition 2. *A non-interactive key exchange (NIKE) scheme consists of two deterministic algorithms* $(\mathsf{NIKE.Gen}, \mathsf{NIKE.Key})$*.*

$\mathsf{NIKE.Gen}(1^\lambda, r)$ *takes a security parameter* λ *and randomness* $r \in \{0,1\}^\lambda$*. It outputs a key pair* (pk, sk)*. We write* $(pk, sk) \xleftarrow{\$} \mathsf{NIKE.Gen}(1^\lambda)$ *to denote that* $\mathsf{NIKE.Gen}(1^\lambda, r)$ *is executed with uniformly random* $r \xleftarrow{\$} \{0,1\}^\lambda$*.*

$\mathsf{NIKE.Key}(sk_i, pk_j)$ *is a* deterministic *algorithm which takes as input a secret key sk_i and a public key pk_j, and outputs a key $k_{i,j}$.*

We say that a NIKE scheme is correct*, if for all $(pk_i, sk_i) \xleftarrow{\$} \mathsf{NIKE.Gen}(1^\lambda)$ and $(pk_j, sk_j) \xleftarrow{\$} \mathsf{NIKE.Gen}(1^\lambda)$ holds that $\mathsf{NIKE.Key}(sk_i, pk_j) = \mathsf{NIKE.Key}(sk_j, pk_i)$.*

A NIKE scheme is used by d parties $P_1, \ldots, P_d$ as follows. Each party P_i generates a key pair $(pk_i, sk_i) \leftarrow \mathsf{NIKE.Gen}(1^\lambda)$ and publishes pk_i. In order to compute the key shared by P_i and P_j, party P_i computes $k_{i,j} = \mathsf{NIKE.Key}(sk_i, pk_j)$. Similarly, party P_j computes $k_{j,i} = \mathsf{NIKE.Key}(sk_j, pk_i)$. Correctness of the NIKE scheme guarantees that $k_{i,j} = k_{j,i}$.

CKS-Light Security. The *CKS-light* security model for NIKE protocols is relatively simplistic and compact. We choose this model because other (more complex) NIKE security models like *CKS*, *CKS-heavy*, and *m-CKS-heavy* are polynomial-time equivalent to *CKS-light*. See [12] for more details.

Security of a NIKE protocol NIKE is defined by a game **NIKE** played between an adversary $\mathcal{A}$ and a challenger. The challenger takes a security parameter λ and a random bit b as input and answers all queries of $\mathcal{A}$ until she outputs a bit b'. The challenger answers the following queries for $\mathcal{A}$:

- $\mathsf{RegisterHonest}(i)$. $\mathcal{A}$ supplies an index i. The challenger runs $\mathsf{NIKE.Gen}(1^\lambda)$ to generate a key pair (pk_i, sk_i) and records the tuple $(\texttt{honest}, pk_i, sk_i)$ for later and returns pk_i to $\mathcal{A}$. This query may be asked *at most twice* by $\mathcal{A}$.
- $\mathsf{RegisterCorrupt}(pk_i)$. With this query $\mathcal{A}$ supplies a public key pk_i. The challenger records the tuple $(\texttt{Corrupt}, pk_i)$ for later.
- $\mathsf{GetCorruptKey}(i, j)$. $\mathcal{A}$ supplies two indices i and j where pk_i was registered as corrupt and pk_j as honest. The challenger runs $k \leftarrow \mathsf{NIKE.Key}(sk_j, pk_i)$ and returns k to $\mathcal{A}$.
- $\mathsf{Test}(i, j)$. The adversary supplies two indices i and j that were registered honestly. Now the challenger uses bit b: if $b = 0$, then the challenger runs $k_{i,j} \leftarrow \mathsf{NIKE.Key}(pk_i, sk_j)$ and returns the key $k_{i,j}$. If $b = 1$, then the challenger samples a random element from the key space, records it for later, and returns the key to $\mathcal{A}$.

The game **NIKE** outputs 1, denoted by $\mathbf{NIKE}^{\mathcal{A}}_{\mathsf{NIKE}}(\lambda) = 1$, if $b = b'$ and 0 otherwise. We say $\mathcal{A}$ wins the game if $\mathbf{NIKE}^{\mathcal{A}}_{\mathsf{NIKE}}(\lambda) = 1$.

Definition 3. *For any adversary $\mathcal{A}$ playing the above* **NIKE** *game against a NIKE scheme* NIKE*, we define the advantage of winning the game* **NIKE** *as*

$$\mathtt{Adv}^{\mathit{CKS\text{-}light}}_{\mathsf{NIKE},\mathcal{A}}(\lambda) = \left| 2 \cdot \Pr\left[\mathbf{NIKE}^{\mathcal{A}}_{\mathsf{NIKE}}(\lambda) = 1\right] - 1 \right|.$$

Let λ be a security parameter, NIKE *be a NIKE protocol and $\mathcal{A}$ an adversary. We say* NIKE *is a* CKS-light-secure *NIKE protocol, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the function $\mathtt{Adv}^{\mathit{CKS\text{-}light}}_{\mathsf{NIKE},\mathcal{A}}(\lambda)$ is a negligible function in λ.*

3 0-RTT Key Exchange Protocols: Syntax and Security with Server-Only Authentication

In the model presented in this section, we give formal definitions for 0-RTT KE with strong key independence and main-key forward secrecy. We start with the case of server-only authentication, as it is the more important case in practice (in particular, server-only authentication will be the main operating mode of both QUIC and TLS 1.3).

3.1 Syntax and Correctness

Definition 4. *A 0-RTT key exchange scheme with server-only authentication consists of deterministic algorithms* $(\mathsf{Gen}^{\mathsf{server}}, \mathsf{KE}^{\mathsf{client}}_{\mathsf{init}}, \mathsf{KE}^{\mathsf{client}}_{\mathsf{refresh}}, \mathsf{KE}^{\mathsf{server}}_{\mathsf{refresh}})$.

- $\mathsf{Gen}^{\mathsf{server}}(1^\lambda, r) \rightarrow (pk, sk)$*: A key generation algorithm that takes as input a security parameter* λ *and randomness* $r \in \{0,1\}^\lambda$ *and outputs a key pair* (pk, sk)*. We write* $(pk, sk) \xleftarrow{\$} \mathsf{Gen}^{\mathsf{server}}(1^\lambda)$ *to denote that a pair* (pk, sk) *is the output of* $\mathsf{Gen}^{\mathsf{server}}$ *when executed with uniformly random* $r \xleftarrow{\$} \{0,1\}^\lambda$.
- $\mathsf{KE}^{\mathsf{client}}_{\mathsf{init}}(pk_j, r_i) \rightarrow (k^{i,j}_{\mathtt{tmp}}, m_i)$*: An algorithm that takes as input a public key* pk_j *and randomness* $r_i \in \{0,1\}^\lambda$*, and outputs a temporary key* $k^{i,j}_{\mathtt{tmp}}$ *and a message* m_i.
- $\mathsf{KE}^{\mathsf{server}}_{\mathsf{refresh}}(sk_j, r_j, m_i) \rightarrow (k^{j,i}_{\mathtt{main}}, k^{j,i}_{\mathtt{tmp}}, m_j)$*: An algorithm that takes as input a secret key* sk_j*, randomness* r_j *and a message* m_i*, and outputs a key* $k^{j,i}_{\mathtt{main}}$*, a temporary key* $k^{j,i}_{\mathtt{tmp}}$ *and a message* m_j.
- $\mathsf{KE}^{\mathsf{client}}_{\mathsf{refresh}}(pk_j, r_i, m_j) \rightarrow k^{i,j}_{\mathtt{main}}$*: An algorithm that takes as input a public key* pk_j*, randomness* r_i*, and message* m_j*, and outputs a key* $k^{i,j}_{\mathtt{main}}$.

We say that a 0-RTT key exchange scheme is correct*, if for all* $(pk_j, sk_j), \xleftarrow{\$} \mathsf{Gen}^{\mathsf{server}}(1^\lambda)$ *and for all* $r_i, r_j \xleftarrow{\$} \{0,1\}^\lambda$ *holds that*

$$\Pr[k^{i,j}_{\mathtt{tmp}} \neq k^{j,i}_{\mathtt{tmp}} \text{ or } k^{i,j}_{\mathtt{main}} \neq k^{j,i}_{\mathtt{main}}] \leq \mathsf{negl}(\lambda) \ ,$$

where $(k^{j,i}_{\mathtt{tmp}}, m_i) \leftarrow \mathsf{KE}^{\mathsf{client}}_{\mathsf{init}}(pk_j, r_i)$*,* $(k^{i,j}_{\mathtt{main}}, k^{i,j}_{\mathtt{tmp}}, m_j) \leftarrow \mathsf{KE}^{\mathsf{server}}_{\mathsf{refresh}}(sk_j, r_j, m_i)$*, and* $k^{j,i}_{\mathtt{main}} \leftarrow \mathsf{KE}^{\mathsf{client}}_{\mathsf{refresh}}(pk_j, r_i, m_j)$.

A 0-RTT KE scheme is used by a set parties which are either clients C or servers S (cf. Fig. 2). Each server S_p has a generated key pair $(sk_p, pk_p) \xleftarrow{\$} \mathsf{Gen}^{\mathsf{server}}(1^\lambda, j)$ with published pk_p. The protocol is executed as follows:

1. The client oracle C_i chooses $r_i \in \{0,1\}^\lambda$ and selects the public key of the intended partner S_j (which must be a server, otherwise this value is undefined). Then it computes $(k^{i,j}_{\mathtt{tmp}}, m_i) \leftarrow \mathsf{KE}^{\mathsf{client}}_{\mathsf{init}}(pk_j, r_i)$, and sends m_i to S_j. Additionally, C_i can use $k^{i,j}_{\mathtt{tmp}}$ to encrypt some data M_i.

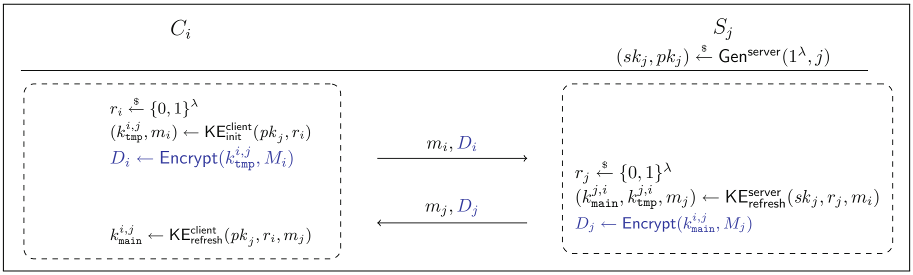

C_i
S_j
$(sk_j, pk_j) \xleftarrow{\$} \mathsf{Gen}^{\mathsf{server}}(1^\lambda, j)$
$r_i \xleftarrow{\$} \{0,1\}^\lambda$
$(k^{i,j}_{\mathtt{tmp}}, m_i) \leftarrow \mathsf{KE}^{\mathsf{client}}_{\mathsf{init}}(pk_j, r_i)$
$D_i \leftarrow \mathsf{Encrypt}(k^{i,j}_{\mathtt{tmp}}, M_i)$
m_i, D_i
$r_j \xleftarrow{\$} \{0,1\}^\lambda$
$(k^{j,i}_{\mathtt{main}}, k^{j,i}_{\mathtt{tmp}}, m_j) \leftarrow \mathsf{KE}^{\mathsf{server}}_{\mathsf{refresh}}(sk_j, r_j, m_i)$
$D_j \leftarrow \mathsf{Encrypt}(k^{i,j}_{\mathtt{main}}, M_j)$
m_j, D_j
$k^{i,j}_{\mathtt{main}} \leftarrow \mathsf{KE}^{\mathsf{client}}_{\mathsf{refresh}}(pk_j, r_i, m_j)$

Fig. 2. Execution of a 0-RTT KE Protocol with Server-Only Authentication in Parallel to Encrypted Application Data. Note that the messages D_i and D_j correspond to the symmetric encryption protocol used to encrypt payload data, and are therefore *not* part of the 0-RTT KE protocol, but a separate protocol. These messages are only displayed here only to illustrate the basic, parallel application message flow to that of a 0-RTT KE protocol. While it would in principle be possible to define the symmetric encryption directly as part of the protocol, this would require a significantly more complex "ACCE-style" [17] security model, which we avoid for sake of simplicity.

2. Upon reception of message m_i, S_j initializes a new oracle $\mathsf{S}_{j,t}$. This oracle chooses $r_j \in \{0,1\}^\lambda$ and computes $(k^{j,i}_{\mathtt{main}}, k^{j,i}_{\mathtt{tmp}}, m_j) \leftarrow \mathsf{KE}^{\mathsf{server}}_{\mathsf{refresh}}(sk_j, r_j, m_i)$. The server may use the ephemeral key $k^{j,i}_{\mathtt{tmp}}$ to decrypt D_i. Then, the server sends m_j and optionally some data M_j encrypted with the key $k^{j,i}_{\mathtt{main}}$ to the client.
3. C_i computes $k^{i,j}_{\mathtt{main}} \leftarrow \mathsf{KE}^{\mathsf{client}}_{\mathsf{refresh}}(pk_j, r_i, m_j)$ and can optionally decrypt D_j. Correctness of the 0-RTT KE scheme guarantees that $k^{i,j}_{\mathtt{main}} = k^{j,i}_{\mathtt{main}}$.

3.2 Execution Environment

We provide an adversary $\mathcal{A}$ against a 0-RTT KE protocol with the following *execution environment*. Clients, which are not in possession of a long-term secret are represented by oracles $\mathsf{C}_1, \ldots, \mathsf{C}_d$ (without any particular "identity"). We consider ℓ servers, each server has a long-term key pair (sk_j, pk_j)[7], $j \in \{1, \ldots, \ell\}$, and each client has access to all public keys $pk_1, \ldots, pk_\ell$. Each server is represented by a collection of k oracles $\mathsf{S}_{j,1}, \ldots, \mathsf{S}_{j,k}$, where each oracle represents a process that executes one single instance of the protocol.

We use the following variables to maintain the internal state of oracles.

Clients. Each client oracle C_i, $i \in [d]$, maintains

- two variables $\mathsf{k}^{\mathsf{tmp}}_i$ and $\mathsf{k}^{\mathsf{main}}_i$ to store the temporal and main keys of a session,

[7] We do not distinguish between static (i.e. long-lived) and semi-static (i.e. medium lived) key pairs. Thus the long-lived keys in this model correspond to the server configuration file keys of QUIC and TLS 1.3.

- a variable $\mathsf{Partner}_i$, which contains the identity of the intended communication partner, and
- variables $\mathcal{M}_i^{\mathbf{in}}$ and $\mathcal{M}_i^{\mathbf{out}}$ containing messages sent and received by the oracle.

The internal state of a client oracle is initialized to $(\mathsf{k}_i^{\mathsf{tmp}}, \mathsf{k}_i^{\mathsf{main}}, \mathsf{Partner}_i, \mathcal{M}_i^{\mathbf{in}}, \mathcal{M}_i^{\mathbf{out}}) := (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$.

Servers. Each server oracle $\mathsf{S}_{j,t}$, $(j,t) \in [\ell] \times [k]$, maintains:

- two variables $\mathsf{k}_i^{\mathsf{tmp}}$ and $\mathsf{k}_i^{\mathsf{main}}$ to store the temporal and main keys of a session, and
- variables $\mathcal{M}_{j,t}^{\mathbf{in}}$ and $\mathcal{M}_{j,t}^{\mathbf{out}}$ containing messages sent and received by the server.

The internal state of a server oracle is initialized to $(\mathsf{k}_{j,t}^{\mathsf{tmp}}, \mathsf{k}_{j,t}^{\mathsf{main}}, \mathcal{M}_{j,t}^{\mathbf{in}}, \mathcal{M}_{j,t}^{\mathbf{out}}) := (\emptyset, \emptyset, \emptyset, \emptyset)$.

We say that an oracle has *accepted the temporal key* if $\mathsf{k}^{\mathsf{tmp}} \neq \emptyset$, and *accepted the main key* if $\mathsf{k}^{\mathsf{main}} \neq \emptyset$.

In the security experiment, the adversary is able to interact with the oracles by issuing the following queries.

$\mathtt{Send}(\mathsf{C}_i/\mathsf{S}_{j,t}, m)$. The adversary sends a message m to the requested oracle. The oracle processes m according to the protocol specification. Any response generated by the oracle according to the protocol specification is returned to the adversary.

If a client oracle C_i receives m as the first message, then the oracle checks if m consists of a special initialization message $(m = (\mathtt{init}, j))$. If true, then the oracle responds with the first protocol message generated for intended partner $\mathsf{S}_{j,}$, else it outputs $\perp$.

$\mathtt{Reveal}(\mathsf{C}_i/\mathsf{S}_{j,t}, \mathtt{tmp/main})$. This query returns the key of the given stage if it already has been computed, or $\perp$ otherwise.

$\mathtt{Corrupt}(j)$. On input of a server identity j, this query returns the long-term private key of the server. If $\mathtt{Corrupt}(j)$ is the τ-th query issued by $\mathcal{A}$, we say a party is *τ-corrupted.* For parties that are not corrupted we define $\tau := \infty$.

$\mathtt{Test}(\mathsf{C}_i/\mathsf{S}_{j,t}, \mathtt{tmp/main})$. This query is used to test a key and is only asked once. It is answered as follows: If the variable of the requested key is not empty, a random $b \xleftarrow{\$} \{0,1\}$ is selected, and

- if $b = 0$ then the requested key is returned, else
- if $b = 1$ then a random key, according to the probability distribution of keys generated by the protocol, is returned.

Otherwise $\perp$ is returned.

Security Model Security Game $\mathcal{G}_{\mathcal{A}}^{0\mathcal{RTT}-sa}$. After receiving a security parameter λ the challenger $\mathcal{C}$ simulates the protocol and keeps track of all variables of the execution environment: he generates the long-lived key pairs of all server parties and answers faithfully to all queries by the adversary.

The adversary receives all public keys $pk_1, \ldots, pk_\ell$ and can interact with the challenger by issuing any combination of the queries $\mathtt{Send}()$, $\mathtt{Corrupt}()$, and

`Reveal`(). At some point the adversary queries `Test`() to an oracle and receives a key, which is either the requested key or a random value. The adversary may continue asking `Send`(), `Corrupt`(), and `Reveal`()-queries after receiving the key and finally outputs some bit b'.

Definition 5 (0-RTT KE-Security with Server-Only Authentication). *Let an adversary $\mathcal{A}$ interact with the challenger in game $\mathcal{G}_{\mathcal{A}}^{0\mathcal{RTT}-sa}$ as it is described above. We say the challenger outputs 1, denoted by $\mathcal{G}_{\mathcal{A}}^{0\mathcal{RTT}-sa}(\lambda) = 1$, if $b = b'$ and the following conditions hold:*

- *if $\mathcal{A}$ issues* $\texttt{Test}(\mathsf{C}_i, \texttt{tmp})$ *all of the following hold:*
  - $\texttt{Reveal}(\mathsf{C}_i, \texttt{tmp})$ *was never queried by $\mathcal{A}$*
  - $\texttt{Reveal}(\mathsf{S}_{j,t}, \texttt{tmp})$ *was never queried by $\mathcal{A}$ for any oracle $\mathsf{S}_{j,t}$ such that* $\mathsf{Partner}_i = j$ *and* $\mathcal{M}_{j,t}^{\boldsymbol{in}} = \mathcal{M}_i^{\boldsymbol{out}}$
  - *the communication partner* $\mathsf{Partner}_i = j$, *if it exists, is not* τ-corrupted *with* $\tau < \infty$
- *if $\mathcal{A}$ issues* $\texttt{Test}(\mathsf{C}_i, \texttt{main})$ *all of the following hold:*
  - $\texttt{Reveal}(\mathsf{C}_i, \texttt{main})$ *was never queried by $\mathcal{A}$*
  - $\texttt{Reveal}(\mathsf{S}_{j,t}, \texttt{main})$ *was never queried by $\mathcal{A}$, where* $\mathsf{Partner}_i = j$, $\mathcal{M}_{j,t}^{\boldsymbol{in}} = \mathcal{M}_i^{\boldsymbol{out}}$, *and* $\mathcal{M}_i^{\boldsymbol{in}} = \mathcal{M}_{j,t}^{\boldsymbol{out}}$
  - *the communication* $\mathsf{Partner}_i = j$ *is not* τ-corrupted *with* $\tau < \tau_0$, *where* $\texttt{Test}(\mathsf{C}_i, \texttt{main})$ *is the* τ_0*-th query issued by $\mathcal{A}$*
- *if $\mathcal{A}$ issues* $\texttt{Test}(\mathsf{S}_{j,t}, \texttt{tmp})$ *all of the following hold:*
  - $\texttt{Reveal}(\mathsf{S}_{j,t}, \texttt{tmp})$ *was never queried by $\mathcal{A}$*
  - *there exists an oracle C_i with* $\mathcal{M}_i^{\boldsymbol{out}} = \mathcal{M}_{j,t}^{\boldsymbol{in}}$
  - $\texttt{Reveal}(\mathsf{C}_i, \texttt{tmp})$ *was never queried by $\mathcal{A}$ to any oracle C_i with* $\mathcal{M}_i^{\boldsymbol{out}} = \mathcal{M}_{j,t}^{\boldsymbol{in}}$
  - $\texttt{Reveal}(\mathsf{S}_{j,t'}, \texttt{tmp})$ *was never queried by $\mathcal{A}$ for any oracle $\mathsf{S}_{j,t'}$ with* $\mathcal{M}_{j,t}^{\boldsymbol{in}} = \mathcal{M}_{j,t'}^{\boldsymbol{in}}$
  - *j is not* τ-corrupted *with* $\tau < \infty$
- *if $\mathcal{A}$ issues* $\texttt{Test}(\mathsf{S}_{j,t}, \texttt{main})$ *all of the following hold:*
  - $\texttt{Reveal}(\mathsf{S}_{j,t}, \texttt{main})$ *was never queried by $\mathcal{A}$*
  - *there exists an oracle C_i with* $\mathcal{M}_i^{\boldsymbol{out}} = \mathcal{M}_{j,t}^{\boldsymbol{in}}$
  - $\texttt{Reveal}(\mathsf{C}_i, \texttt{main})$ *was never queried by $\mathcal{A}$, if* $\mathcal{M}_i^{\boldsymbol{in}} = \mathcal{M}_{j,t}^{\boldsymbol{out}}$

else the game outputs a random bit. We define the advantage of $\mathcal{A}$ in the game $\mathcal{G}_{\mathcal{A}}^{0\mathcal{RTT}-sa}(\lambda)$ by

$$\mathsf{Adv}_{\mathcal{A}}^{0\mathcal{RTT}-sa}(\lambda) := \left|2 \cdot \Pr[\mathcal{G}_{\mathcal{A}}^{0\mathcal{RTT}-sa}(\lambda) = 1] - 1\right|.$$

Definition 6. *We say that a 0-RTT key exchange protocol is* `test`*-secure, if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all PPT adversaries $\mathcal{A}$ interacting according to the security game $\mathcal{G}_{\mathcal{A}}^{0\mathcal{RTT}-sa}(\lambda)$ it holds that*

$$\mathsf{Adv}_{\mathcal{A}}^{0\mathcal{RTT}-sa}(\lambda) \leq \mathsf{negl}(\lambda).$$

Remark 2. Our security model captures forward secrecy for the $\mathtt{main}$-key, because key indistinguishability is required to hold even if the adversary is able to corrupt the communication partner of the $\mathtt{test}$-oracle (but only after the $\mathtt{test}$-oracle has accepted, of course, in order to avoid trivial attacks).

Moreover, strong key independence is modeled by the fact that an adversary which attempts to distinguish a $\mathtt{tmp}$-key from random (i.e., an adversary which asks $\mathsf{Test}(X, \mathtt{tmp})$ for $X \in \{\mathsf{C}_i, \mathsf{S}_{j,t}$ for some $i, j, t\})$ is allowed to learn the $\mathtt{main}$-key of X. Similarly, an adversary which tries to distinguish a $\mathtt{main}$-key from random by asking $\mathsf{Test}(X, \mathtt{main})$ is allowed to learn the $\mathtt{tmp}$-key of X as well. Security in this sense guarantees that the $\mathtt{tmp}$-key and the $\mathtt{main}$-key look independent to a computationally-bounded adversary.

Remark 3. Note that the requirements of $\mathcal{M}_i^{\mathbf{out}} = \mathcal{M}_{j,t}^{\mathbf{in}}$ etc. in the above security model essentially adopt the notion of *matching conversations*, defined by Bellare and Rogaway [2] for general, multi-message key exchange protocols, to the special case of 0-RTT KE.

3.3 Composing a 0-RTT KE Protocol with Symmetric Encryption

The security model described above considers only the 0-RTT KE protocol, *without* symmetric encryption of payload data (that is, without the messages D_i and D_j displayed in Fig. 2). A protocol secure in this sense guarantees the indistinguishability of keys in a hypothetical setting, where the key is not used for symmetric encryption of payload messages potentially known to the adversary. One may think that this is not sufficient for 0-RTT KE, because the key *will* be used to encrypt payload data, and this will enable an adversary to trivially distinguish a "real" key from a "random" key (this holds for both the "temporal" key $k_{\mathtt{tmp}}^{i,j}$ and the actual "main" session key $k_{\mathtt{main}}^{i,j}$). Note that this argument applies not only to the above 0-RTT KE security model, but actually to any security model for (authenticated) key exchange which is based on the indistinguishability of keys, such as the classical model of Bellare and Rogaway and many similar models [2,5,7,10,20,27]. In practice, this key will usually be used in a cryptographic protocol, e.g. to encrypt messages, and therefore trivially allow for distinguishing "real" from "random" keys. The security of the composition of a protocol secure in the sense of [2,5,7,10,20,27] with a symmetric encryption protocol follows from a standard two-step hybrid argument, which essentially proceeds as follows:

1. In the original security experiment, the adversary interacts with a composed protocol, where the KE protocol is first used to derive a key k, which is then used to encrypt payload data with the symmetric encryption protocol.
2. In the next hybrid experiment, the adversary interacts with a composed protocol, where the symmetric encryption does not use the key k computed by the KE protocol, but an independent random key. Note that an adversary that distinguishes this hybrid from the original game can be used to distinguish a "real" key of the KE protocol from a "random" one.

Now the adversary interacts with an encryption protocol that uses a key which is *independent* of the KE protocol. This allows for a reduction of the security of the composed protocol to the security of the symmetric protocol.

A similarly straightforward hybrid argument applies to the composition of 0-RTT KE with symmetric encryption, which works as follows:

1. In the original security experiment, the adversary interacts with a composed protocol, where the 0-RTT KE protocol is first used to derive a key $k_{\mathtt{tmp}}^{i,j}$, which is then used to encrypt the payload data sent along with the first protocol message. Then the 0-RTT KE protocol is used to derive the main key $k_{\mathtt{main}}^{i,j}$, which in turn is used to encrypt all further payload data.
2. In the first hybrid experiment, the adversary interacts with a composed protocol, where only $k_{\mathtt{tmp}}^{i,j}$ is replaced with an independent random value. An adversary that distinguishes this hybrid from the original game can be used to distinguish a "real" $k_{\mathtt{tmp}}^{i,j}$ from a "random" one.
   Now the adversary interacts with an encryption protocol that encrypts the first payload message with a key which is *independent* of the 0-RTT KE protocol. This allows for a reduction of the security of the first payload message to the security of the symmetric protocol.
3. In the second hybrid experiment, the adversary interacts with a composed protocol, where $k_{\mathtt{main}}^{i,j}$ is now also replaced with an independent random value. An adversary that distinguishes this hybrid from the previous one can be used to distinguish a "real" $k_{\mathtt{main}}^{i,j}$ from a "random" one. This allows for a reduction of the security of all further payload messages to the security of the symmetric protocol.

Following the long tradition of previous works on indistinguishability-based key exchange security models [2,5,7,10,20,27], we can thus consider an indistinguishability-based security model for 0-RTT KE even though in practice key exchange messages will be interleaved with messages of the symmetric encryption protocol. This allows for simple security models, and enables a modular analysis of the building blocks of a composed protocol.

4 Generic Construction of 0-RTT KE from NIKE

Now we are ready to describe our generic NIKE-based 0-RTT KE protocol and its security analysis.

4.1 Generic Construction

Let $\mathsf{NIKE} = (\mathsf{NIKE.Gen}, \mathsf{NIKE.Key})$ be a NIKE scheme according to Definition 2 and let $\mathsf{SIG} = (\mathsf{SIG.Gen}, \mathsf{SIG.Sign}, \mathsf{SIG.Vfy})$ be a signature scheme. Then we construct a 0-RTT KE scheme $\mathsf{0\text{-}RTT} = (\mathsf{Gen}^{\mathsf{server}}, \mathsf{KE}_{\mathsf{init}}^{\mathsf{client}}, \mathsf{KE}_{\mathsf{refresh}}^{\mathsf{client}}, \mathsf{KE}_{\mathsf{refresh}}^{\mathsf{server}})$, per Definition 4, in the following way (cf. Fig. 3).

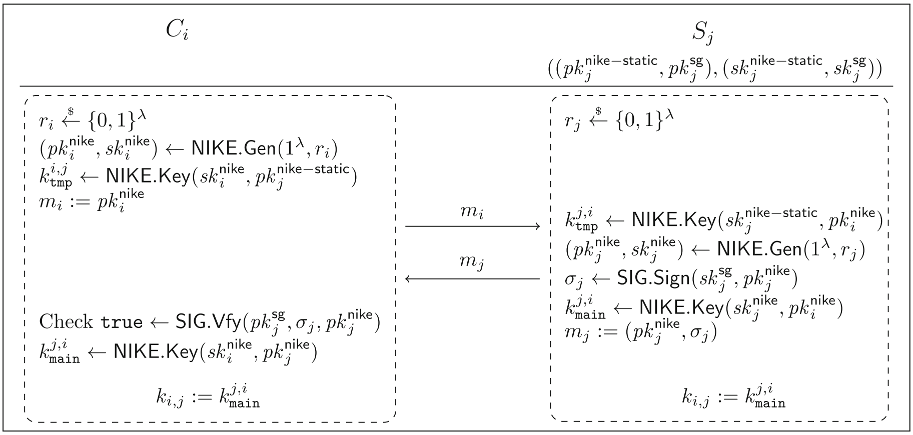

C_i
S_j
$((pk_j^{\mathsf{nike-static}}, pk_j^{\mathsf{sg}}), (sk_j^{\mathsf{nike-static}}, sk_j^{\mathsf{sg}}))$
$r_i \xleftarrow{\$} \{0,1\}^\lambda$
$(pk_i^{\mathsf{nike}}, sk_i^{\mathsf{nike}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda, r_i)$
$k_{\mathtt{tmp}}^{i,j} \leftarrow \mathsf{NIKE.Key}(sk_i^{\mathsf{nike}}, pk_j^{\mathsf{nike-static}})$
$m_i := pk_i^{\mathsf{nike}}$
$r_j \xleftarrow{\$} \{0,1\}^\lambda$
m_i
$k_{\mathtt{tmp}}^{j,i} \leftarrow \mathsf{NIKE.Key}(sk_j^{\mathsf{nike-static}}, pk_i^{\mathsf{nike}})$
$(pk_j^{\mathsf{nike}}, sk_j^{\mathsf{nike}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda, r_j)$
m_j
$\sigma_j \leftarrow \mathsf{SIG.Sign}(sk_j^{\mathsf{sg}}, pk_j^{\mathsf{nike}})$
$k_{\mathtt{main}}^{j,i} \leftarrow \mathsf{NIKE.Key}(sk_j^{\mathsf{nike}}, pk_i^{\mathsf{nike}})$
$m_j := (pk_j^{\mathsf{nike}}, \sigma_j)$
Check $\mathtt{true} \leftarrow \mathsf{SIG.Vfy}(pk_j^{\mathsf{sg}}, \sigma_j, pk_j^{\mathsf{nike}})$
$k_{\mathtt{main}}^{j,i} \leftarrow \mathsf{NIKE.Key}(sk_i^{\mathsf{nike}}, pk_j^{\mathsf{nike}})$
$k_{i,j} := k_{\mathtt{main}}^{j,i}$
$k_{i,j} := k_{\mathtt{main}}^{j,i}$

Fig. 3. 0-RTT KE from NIKE. Again, it is possible to include the parallel execution of a symmetric encryption protocol which would behave as in Fig. 2 for encrypted application data. As such a protocol is *not* part of the 0-RTT KE protocol, we omit it here for simplicity.

- $\mathsf{Gen}^{\mathsf{server}}(1^\lambda, r)$ computes key pairs using the NIKE key generation algorithm $(pk^{\mathsf{nike-static}}, sk^{\mathsf{nike-static}}) \xleftarrow{\$} \mathsf{NIKE.Gen}(1^\lambda)$ and signature keys using the SIG algorithm $(pk^{\mathsf{sg}}, sk^{\mathsf{sg}}) \xleftarrow{\$} \mathsf{SIG.Gen}$, and outputs
$$(pk, sk) := ((pk^{\mathsf{nike-static}}, pk^{\mathsf{sg}}), (sk^{\mathsf{nike-static}}, sk^{\mathsf{sg}})).$$
- $\mathsf{KE}_{\mathsf{init}}^{\mathsf{client}}(pk_j, r_i)$ samples $r_i \xleftarrow{\$} \{0,1\}^\lambda$, parses $pk_j = (pk_j^{\mathsf{nike-static}}, pk_j^{\mathsf{sg}})$, runs $(pk_i^{\mathsf{nike}}, sk_i^{\mathsf{nike}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda, r_i)$ and $k_{i,j}^{\mathsf{nike}} \leftarrow \mathsf{NIKE.Key}(sk_i^{\mathsf{nike}}, pk_j^{\mathsf{nike-static}})$, and outputs
$$(k_{\mathtt{tmp}}^{i,j}, m_i) := (k_{i,j}^{\mathsf{nike}}, pk_i^{\mathsf{nike}}).$$
- $\mathsf{KE}_{\mathsf{refresh}}^{\mathsf{server}}(sk_j, r_j, m_i)$ takes in $m_i = pk_i^{\mathsf{nike}}$, parses $sk_j = (sk_j^{\mathsf{nike-static}}, sk_j^{\mathsf{sg}})$, and samples $r_j \xleftarrow{\$} \{0,1\}^\lambda$. It then computes $k_{i,j}^{\mathsf{nike}} \leftarrow \mathsf{NIKE.Key}(sk_j^{\mathsf{nike-static}}, pk_i^{\mathsf{nike}})$, $(pk_j^{\mathsf{nike}}, sk_j^{\mathsf{nike}}) \leftarrow \mathsf{NIKE.Gen}(1^\lambda, r_j)$, and $\sigma_j \leftarrow \mathsf{SIG.Sign}(sk_j^{\mathsf{sg}}, pk_j^{\mathsf{nike}})$. If $m_i = pk_j^{\mathsf{nike-static}}$ then it samples $k_{\mathtt{main}}^{\mathsf{nike}}$ uniformly random, else it computes $k_{\mathtt{main}}^{\mathsf{nike}} \leftarrow \mathsf{NIKE.Key}(sk_j^{\mathsf{nike}}, pk_i^{\mathsf{nike}})$, outputting
$$(k_{\mathtt{main}}^{j,i}, k_{\mathtt{tmp}}^{j,i}, m_j) := (k_{\mathtt{main}}^{\mathsf{nike}}, k_{i,j}^{\mathsf{nike}}, (pk_j^{\mathsf{nike}}, \sigma_j)).$$
- $\mathsf{KE}_{\mathsf{refresh}}^{\mathsf{client}}(pk_j, r_i, m_j)$ parses $pk_j = (pk_j^{\mathsf{nike-static}}, pk_j^{\mathsf{sg}})$ and $m_j = (pk_j^{\mathsf{nike}}, \sigma_j)$. It then checks $\mathtt{true} \leftarrow \mathsf{SIG.Vfy}(pk_j^{\mathsf{sg}}, \sigma_j, pk_j^{\mathsf{nike}})$ and computes $k_{\mathtt{main}}^{\mathsf{nike}} \leftarrow \mathsf{NIKE.Key}(sk_i^{\mathsf{nike}}, pk_j^{\mathsf{nike}})$, outputting $k_{\mathtt{main}}^{i,j} := k_{\mathtt{main}}^{\mathsf{nike}}$.

Ultimately, the construction follows by applying the NIKE $\mathsf{NIKE.Gen}$ algorithm and the signature $\mathsf{SIG.Gen}$ algorithm to generate a server configuration file which is comprised of the server public key and a server public signature key which a client can then employ for generating the first protocol flow. In order for the 0-RTT KE construction to abstract the security guarantees of the underlying NIKE, the appropriate client $(pk_i^{\mathsf{nike}}, sk_i^{\mathsf{nike}})$ must be available for use in the $\mathsf{NIKE.Key}$ algorithm. Consequently, the $(pk_i^{\mathsf{nike}}, sk_i^{\mathsf{nike}})$ values are generated locally by the client, with pk_i^{nike} passed to the server as a message. Note that this construction naturally forgoes client-side authentication. Figure 3 demonstrates the construction.

Remark 4. One may wonder why we define $\mathsf{KE}^{\mathsf{server}}_{\mathsf{refresh}}(sk_j, r_j, m_i)$ such that it samples a random key when it takes as input a client message m_i which is equal to its own static NIKE key, that is, if $m_i = pk_j^{\mathsf{nike-static}}$. We note that this is necessary for the security the constructed 0-RTT KE scheme to be reducible to that of the NIKE scheme, because in some cases we will not be able to simulate the key computed by a server oracle that receives as input a message which is equal to the "static" NIKE public key contained in its 0-RTT KE public key. Note that this incurs a negligible correctness error. However, it is straightforward to verify the correctness of the protocol according to Definition 4.

Theorem 1. *Let* $\mathsf{0\text{-}RTT}$ *be executed with d clients, ℓ servers with long-term keys, and k server oracles modeling each server. From each attacker $\mathcal{A}$, we can construct attackers $\mathcal{B}_{sig}$, according to Definition 1, and $\mathcal{B}_{nike}$, according to Definition 3, such that*

$$\begin{aligned}\mathtt{Adv}_{\mathcal{A}}^{\mathcal{0RTT}-sa}(\lambda) \leq{} & 2kd\ell \cdot \Big(\mathtt{Adv}_{\mathsf{NIKE},\mathcal{B}_{nike}}^{CKS\text{-}light}(\lambda) + \mathtt{Adv}_{\mathsf{SIG},\mathcal{B}_{sig}}^{sEUF\text{-}CMA}(\lambda)\Big) \\ & + d\ell \cdot \Big(k \cdot \mathtt{Adv}_{\mathsf{NIKE},\mathcal{B}_{nike}}^{CKS\text{-}light}(\lambda) + \mathtt{Adv}_{\mathsf{SIG},\mathcal{B}_{sig}}^{sEUF\text{-}CMA}(\lambda)\Big) \\ & + d\ell \cdot \Big(\mathtt{Adv}_{\mathsf{NIKE},\mathcal{B}_{nike}}^{CKS\text{-}light}(\lambda) + \mathtt{Adv}_{\mathsf{SIG},\mathcal{B}_{sig}}^{sEUF\text{-}CMA}(\lambda)\Big) + 4 \cdot \mathtt{Adv}_{\mathsf{NIKE},\mathcal{B}_{nike}}^{CKS\text{-}light}(\lambda) .\end{aligned}$$

The running time of $\mathcal{B}_{sig}$ and $\mathcal{B}_{nike}$ is approximately equal to the time required to execute the security experiment with $\mathcal{A}$ once.

Intuition for the Proof of Theorem 1. In order to prove Theorem 1, we distinguish between four types of attackers:

- adversary $\mathcal{A}_1$ asks $\mathtt{Test}()$ to a client oracle and the temporary key (*CT-attacker*)
- adversary $\mathcal{A}_2$ asks $\mathtt{Test}()$ to a client oracle and the main key (*CM-attacker*)
- adversary $\mathcal{A}_3$ asks $\mathtt{Test}()$ to a server oracle and the temporary key (*ST-attacker*)
- adversary $\mathcal{A}_4$ asks $\mathtt{Test}()$ to a server oracle and the main key (*SM-attacker*).

Let us give some intuition why this classification of attackers will be useful for the security proof. In the 0-RTT KE scheme $\mathsf{0\text{-}RTT}$ each party computes 2 different keys $k^{\text{'}}_{\mathtt{tmp}}$ and $k^{\text{'}}_{\mathtt{main}}$, where $k^{\text{'}}_{\mathtt{tmp}}$ depends on the ephemeral keys of the client and the static keys of the server, and $k_{\mathtt{main}}^{\prime}$ depends on the ephemeral keys of both parties. In our proof we want to be able to reduce the indistinguishability of the 0-RTT-key to the indistinguishability of the NIKE-key.

In the NIKE security experiment the attacker receives two challenge public keys $\{pk_i^{\mathsf{nike}}, pk_j^{\mathsf{nike}}\}$. In the reduction, we want to embed these keys in the 0-RTT security experiment, according to Sect. 3.2, and still be able to answer all `Reveal()`- and `Corrupt()`-queries correctly. In the case of adversaries that test the temporary key of the client or the server we can embed the NIKE-keys as $pk_j^{\mathsf{nike-static}} = pk_j^{\mathsf{nike}}$ and $m_i = pk_i^{\mathsf{nike}}$. However, this does not work for adversaries against the main key, because $k_{\mathtt{main}}^{\prime}$ depends on the ephemeral keys of the parties. In this case we have to embed the keys as $m_i = pk_i^{\mathsf{nike}}$ and $m_j = pk_j^{\mathsf{nike}}$. The `Test()`-query of the attacker in the 0-RTT experiment can then be answered with the challenge the attacker in the NIKE experiment receives.

References

1. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, 3–5 November 1993, Fairfax, Virginia, USA, pp. 62–73. ACM Press (1993)
2. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994). doi:10.1007/3-540-48329-2_21
3. Bergsma, F., Jager, T., Schwenk, J.: One-round key exchange with strong security: an efficient and generic construction in the standard model. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 477–494. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46447-2_21
4. Bhargavan, K., Fournet, C., Kohlweiss, M., Pironti, A., Strub, P.-Y., Zanella-Béguelin, S.: Proving the TLS handshake secure (as it is). In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 235–255. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44381-1_14
5. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In: Darnell, M. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 30–45. Springer, Heidelberg (1997). doi:10.1007/BFb0024447
6. Boneh, D., Shen, E., Waters, B.: Strongly unforgeable signatures based on computational Diffie-Hellman. In: Yung et al. [30], pp. 229–240
7. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001). doi:10.1007/3-540-44987-6_28
8. Cash, D., Kiltz, E., Shoup, V.: The twin Diffie-Hellman problem and applications. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_8
9. Cremers, C.: Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In: Cheung, B.S.N., Hui, L.C.K., Sandhu, R.S., Wong, D.S. (eds.) ASIACCS 2011, 22–24 March 2011, Hong Kong, China, pp. 80–91. ACM Press (2011)
10. Cremers, C.J.F.: Session-state reveal is stronger than ephemeral key reveal: attacking the NAXOS authenticated key exchange protocol. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 20–33. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01957-9_2

11. Fischlin, M., Günther, F.: Multi-stage key exchange and the case of Google's QUIC protocol. In: Ahn, G.-J., Yung, M., Li, N. (eds.) ACM CCS 2014, 3–7 November 2014, Scottsdale, AZ, USA, pp. 1193–1204. ACM Press (2014)
12. Freire, E.S.V., Hofheinz, D., Kiltz, E., Paterson, K.G.: Non-interactive key exchange. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 254–271. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36362-7_17
13. Green, M.D., Miers, I.: Forward secure asynchronous messaging from puncturable encryption. In: IEEE S&P 2015 [16], pp. 305–320
14. Günther, F., Hale, B., Jager, T., Lauer, S.: 0-RTT key exchange with full forward secrecy. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 519–548. Springer, Cham (2017). doi:10.1007/978-3-319-56617-7_18
15. Hale, B., Jager, T., Lauer, S., Schwenk, J.: Simple security definitions for and constructions of 0-RTT key exchange. Cryptology ePrint Archive, Report 2015/1214 (2015). http://eprint.iacr.org/2015/1214
16. IEEE Symposium on Security and Privacy, 17–21 May 2015, San Jose, CA, USA. IEEE Computer Society Press (2015)
17. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 273–293. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_17
18. Krawczyk, H.: HMQV: a high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005). doi:10.1007/11535218_33
19. Krawczyk, H., Paterson, K.G., Wee, H.: On the security of the TLS protocol: a systematic analysis. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 429–448. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_24
20. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75670-5_1
21. Lauter, K., Mityagin, A.: Security analysis of KEA authenticated key exchange protocol. In: Yung et al. [30], pp. 378–394
22. Law, L., Menezes, A., Minghua, Q., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. Des. Codes Crypt. **28**(2), 119–134 (2003)
23. Lychev, R., Jero, S., Boldyreva, A., Nita-Rotaru, C.: How secure and quick is QUIC? Provable security and performance analyses. In: IEEE S&P 2015 [16], pp. 214–231
24. NIST: SKIPJACK and KEA algorithm specifications (1998). http://csrc.nist.gov/groups/STM/cavp/documents/skipjack/skipjack.pdf
25. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3: draft-ietf-tls-tls13-08. Technical report, August 2015. Expires 29 Feb 2016
26. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3: draft-ietf-tls-tls13-18. Technical report, October 2016. Expires 29 April 2017
27. Sarr, A.P., Elbaz-Vincent, P., Bajard, J.-C.: A new security model for authenticated key agreement. In: Garay, J.A., Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 219–234. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15317-4_15
28. Steinfeld, R., Pieprzyk, J., Wang, H.: How to strengthen any weakly unforgeable signature into a strongly unforgeable signature. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 357–371. Springer, Heidelberg (2006). doi:10.1007/11967668_23

29. Yoneyama, K., Zhao, Y.: Taxonomical security consideration of authenticated key exchange resilient to intermediate computation leakage. In: Boyen, X., Chen, X. (eds.) ProvSec 2011. LNCS, vol. 6980, pp. 348–365. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24316-5_25
30. Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.): PKC 2006. LNCS, vol. 3958. Springer, Heidelberg (2006)
31. Zhao, Y.: Identity-concealed authenticated encryption and key exchange. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, 24–28 October 2016, Vienna, Austria, pp. 1464–1479. ACM Press (2016)

TOPPSS: Cost-Minimal Password-Protected Secret Sharing Based on Threshold OPRF

Stanisław Jarecki[1(✉)], Aggelos Kiayias[2], Hugo Krawczyk[3], and Jiayu Xu[1]

[1] University of California, Irvine, USA
stasio@ics.uci.edu, jiayux@uci.edu
[2] University of Edinburgh, Edinburgh, UK
aggelos@kiayias.com
[3] IBM Research, New York City, USA
hugo@ee.technion.ac.il

Abstract. We present TOPPSS, the most efficient Password-Protected Secret Sharing (PPSS) scheme to date. A (t, n)-threshold PPSS, introduced by Bagherzandi et al. [4], allows a user to share a secret among n servers so that the secret can later be reconstructed by the user from any subset of $t+1$ servers with the sole knowledge of a password. It is guaranteed that any coalition of up to t corrupt servers learns nothing about the secret (or the password). In addition to providing strong protection to secrets stored online, PPSS schemes give rise to efficient Threshold PAKE (T-PAKE) protocols that armor single-server password authentication against the inherent vulnerability to offline dictionary attacks in case of server compromise.

TOPPSS is *password-only*, i.e. it does not rely on public keys in reconstruction, and enjoys remarkable efficiency: A single communication round, a single exponentiation per server and just two exponentiations per client regardless of the number of servers. TOPPSS satisfies threshold security under the (Gap) One-More Diffie-Hellman (OMDH) assumption in the random-oracle model as in prior efficient realizations of PPSS/T-PAKE [18,19]. Moreover, we show that TOPPSS realizes the *Universally Composable* PPSS notion of [19] under a generalization of OMDH, the *Threshold* One-More Diffie-Hellman (T-OMDH) assumption. We show that the T-OMDH and OMDH assumptions are both hard in the generic group model.

The key technical tool we introduce is a universally composable *Threshold Oblivious PRF* which is of independent interest and applicability.

1 Introduction

Passwords have well-known weaknesses as authentication tokens, foremost because of their vulnerability to offline dictionary attacks in case of the

S. Jarecki—Supported by NSF grant CNS-1547435.

A. Kiayias—Supported by ERC project CODAMODA and H2020 Project Panoramix, #653497.

H. Krawczyk—Supported by ONR Contract N00014-14-C-0113.

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 39–58, 2017.
DOI: 10.1007/978-3-319-61204-1_3

all-too-common leakage of the database of password hashes stored by the authentication server (see e.g., [1]). Worse still, most people re-use their passwords across multiple services, hence a break-in into one service effectively breaks the security of others. Yet, because of their convenience, passwords are a dominant form of authentication, and the amount and value of information protected using passwords keeps growing. Defenses such as the use of secondary authentication factors (e.g., a PIN generated by a personal device or a USB dongle) increase protection against on-line attacks but not against offline attacks upon server compromise. Techniques such as Password Authenticated Key Exchange (PAKE) [6,8] improve on today's de-facto standard of "password over TLS" authentication by eliminating the reliance on a Public Key Infrastructure (PKI), but they do not help against offline attacks after server compromise.

T-PAKE and PPSS. To address the threat of offline dictionary attacks on the server, Mackenzie et al. [26] introduced (t, n)-Threshold PAKE (T-PAKE), which replaces a single authentication server with a group of n servers and leaks no information on passwords even if up to t servers are corrupted. Bagherzandi et al. [4] proposed a related notion of Password-Protected Secret Sharing (PPSS) which simplifies the notion of T-PAKE by reducing the goal of key exchange between user and servers to that of the user retrieving a single secret previously shared with the servers. Specifically, a (t, n)-PPSS scheme, as formulated in the PKI-free setting by [18], allows a user to share a random secret s among n servers under the protection of her password pw s.t. (1) a reconstruction protocol involving at least $t+1$ honest servers recovers s if the user inputs the (correct) password pw; (2) the compromise of up to t servers leaks no information about either s or pw; (3) an adversary who corrupts $t' \leq t$ servers and has q_U interactions with the user and q_S interactions with the uncorrupted servers can test at most $\frac{q_S}{t-t'+1} + q_U$ passwords. (In the PKI setting one can set $q_U = 0$.)

The PPSS notion is useful in the design of efficient T-PAKE's because of the low-overhead generic PPSS-to-TPAKE compiler [4,18]. It is also an important primitive in its own right, allowing for online storage of sensitive information like keys, credentials, or personal records, with availability and privacy protection. The only token needed for retrieving stored information is a *single* password, and both information and password remain private if no more than t servers are compromised (and if the adversary does not guess or learn the password).

In this paper we present TOPPSS, the most efficient PPSS scheme to date – and using the PPSS-to-TPAKE compiler of [18] also the most efficient T-PAKE – with a hard-to-beat complexity as detailed below. Our work builds on the works of Jarecki et al. [18,19] who constructed PPSS protocols based on *Oblivious Pseudorandom Functions (OPRF)*, formulated as a universally composable (UC) functionality. The works of [18,19] define UC OPRF differently, but each instantiates its OPRF notion using the *blinded Diffie-Hellman* technique, following Ford and Kaliski [15], under the so-called (Gap) *One-More Diffie-Hellman* (OMDH) assumption [5,22] in the Random Oracle Model (ROM). Using one OPRF construction, [18] showed a PPSS whose reconstruction phase takes a single round between a user and $t+1$ servers, with 2 (multi)exponentiations per server and $2t+3$ for the user. The PPSS of [19] uses a simplified OPRF scheme secure under the same assumptions, with 1 exponentiation per server and $t+2$ for the user. In addition to improving on [18] in efficiency, the latter scheme satisfies a stronger PPSS notion formulated as a UC functionality, which we adopt here.

Our Contributions. We present TOPPSS, a simple PPSS protocol with remarkable and hard to beat performance. The reconstruction procedure requires *just one exponentiation per server and a total of two exponentiations for the user* (independent of the number of servers), plus $O(t)$ modular multiplications by each party. Communication is also optimal: The user sends a single group element to a subset of $t+1$ servers and gets one group element from each server. Furthermore, we show that this "minimal cost" (and PKI-free) PPSS satisfies the strong UC notion of PPSS from [19]. This contribution is based on the observation that a more efficient PPSS can result from replacing the OPRF used in the protocols of [18,19] with its *threshold* (or multi-party) counterpart which we define as *Threshold OPRF (T-OPRF).* We provide a UC definition of T-OPRF as a functionality that allows a group of servers to secret-share a key k for PRF f with a shared PRF evaluation protocol which lets the user compute $f_k(x)$ on her input x, s.t. both x and k are secret if no more than t of n servers are corrupted. T-OPRF is an input-oblivious strengthening of Distributed PRF (DPRF) of Naor et al. [27], hence in particular T-OPRF can replace DPRF in all its applications, e.g. for corruption-resilient Key Distribution Center, and long-term information protection (see [27]).

Using this strong notion of T-OPRF security we show a compiler which transforms UC T-OPRF into UC PPSS at negligible additional cost (in ROM). In particular, TOPPSS is obtained by designing a T-OPRF protocol, denoted 2HashTDH, with the efficiency parameters stated above. This T-OPRF protocol is essentially a "threshold exponentiation" protocol, where each server computes m^{k_i} on input m where k_i is the server's secret-share of the PRF key k. We prove that TOPPSS realizes UC T-OPRF under the following assumptions in ROM. Let $t' \leq t$ denote the number of parties actually controlled by an attacker. First, our results imply that in the so-called *full corruption* case, i.e. if $t' = t$, the same (Gap) OMDH assumption used in [18,19] implies that the attacker must query one uncorrupted party per each input on which the attacker wants to obtain the function value. Since this is the case when the attacker controls the full threshold t of servers it is also the case for any $t' < t$. In the application to PPSS this means that the attacker can test up to $q_S + q_U$ passwords, which matches the $\frac{q_S}{t-t'+1} + q_U$ bound for $t' = t$. Since many existing works on T-PAKE, e.g. [2,9,14,23,26,31], implicitly assume the $t' = t$ case by defining security using the simplified $q_S + q_U$ bound on the number of passwords the adversary can test, we call this level of security a *standard threshold security* for T-PAKE/PPSS.

Secondly, for the general case of $t' \leq t$, we show that TOPPSS achieves the stronger $\frac{q_S}{t-t'+1} + q_U$ bound assuming a generalization of the OMDH assumption which we call (Gap) *Threshold One-More Diffie-Hellman (T-OMDH).* As a sanity check for the T-OMDH assumption we show that the T-OMDH problem is hard in the generic group model. Since OMDH is a special case of T-OMDH, to the best of our knowledge this is also the first generic group analysis of OMDH. The stricter bound implies that an adversary controlling $t' \leq t$ servers must contact $t - t' + 1$ uncorrupted servers for each input on which it wants to compute the function, which coincides with the *standard threshold security* notion when $t' = t$, but it is stronger for $t' < t$. For example, it means that the default network adversary who does corrupt any party but runs q sessions with each server, can test up to $qn/(t+1)$ passwords, whereas the *standard threshold security* would in this case upper-bound the number of tested passwords only by qn.

As a point of comparison in the full version of this paper [20] we consider a generic compiler from *any* OPRF to T-OPRF. This compiler performs multiparty computation of the server code in the underlying OPRF protocol, but in the case of the OPRF of [19] such MPC protocol has the same low computational cost as the customized T-OPRF protocol 2HashTDH discussed above, i.e. 1 exponentiation per server and 2 for the user, with the only drawback of adding an additional communication round to enforce an agreement between the servers on the client's input to the MPC protocol. On the other hand, since the security depends only on the base OPRF, the resultant two-round T-OPRF protocol achieves the $\frac{q_S}{t-t'+1} + q_U$ bound based solely on OMDH for all $t' \leq t$.

Other Applications. Oblivious PRFs have found multiple applications which can also enjoy the benefits of a threshold version, particularly given the remarkable efficiency of our schemes. Examples of such applications include search on encrypted data [13,17], set intersection [22], and multiple-server DE-PAKE (device enhanced PAKE) [21].

Related Work. The first (t, n)-Threshold PAKE (T-PAKE) by Mackenzie et al. [26] required ROM in the security analysis and relied on PKI, namely, it assumed that the client can validate the public keys of the servers *during the reconstruction phase*.[1] Gennaro and Raimondo [14] dispensed with ROM and PKI (in authentication) but increased protocol costs. Abdalla et al. [2] showed a PKI-free T-PAKE in ROM with fewer communication rounds than T-PAKE of [26] but the client establishes a key with only one designated *gateway* server. Yi et al. [31] showed a similar round-reduction without ROM. The case of $n = 2$ servers, known as 2-PAKE, received special attention starting with Brainard et al. [9,29] on 2-PAKE in ROM and PKI, and several works [7,23–25] addressed the non-PKI and no-ROM case. Still, each of these T-PAKE schemes requires server-to-server communication. If communication is mediated by the client then the lowest round complexity is 3 for $n > 2$ [2] and 2 for $n = 2$ [7,25].

Bagherzandi et al. [4] introduced the notion of Password-Protected Secret Sharing (PPSS) with the goal of simplifying T-PAKE protocols. Specifically,

[1] When we say that PPSS/T-PAKE assumes PKI we mean that it relies on it for the security of the reconstruction/authentication phase. By contrast, the *initialization phase* of any PPSS/T-PAKE solution must assume some trust infrastructure, e.g. PKI, or otherwise each party could be initializing the scheme with an impostor.

they showed a PPSS protocol in ROM assuming PKI, with 2 rounds, constant-sized messages, and $8(t+1)$ (multi) exponentiations per client, and a low-cost PKI-model compiler from PPSS to T-PAKE. Camenisch et al. [10] constructed another PPSS scheme, called T-PASS, for *Threshold Password-Authenticated Secret Sharing*, without assuming PKI but with $14n$ exponentiations for the client, 7 exponentiations per server, and 5 rounds of communication.

Jarecki et al. [18,19] showed significantly faster PPSS protocols, also without assuming PKI (in reconstruction): The PPSS of [18] takes a single round (two messages) between a user and each server, and uses 2 (multi) exponentiations per server and $2t+3$ (multi) exponentiations for the client, secure under (Gap) OMDH in ROM. (They also show a 4-message non-ROM PPSS with $O(n \cdot |\mathsf{pw}|)$ exponentiations using Paillier encryption.) The PPSS of [19] improves upon this with a single-round PPSS with 1 exponentiation per server and $t+2$ exponentiations for the client, also under OMDH in ROM. In related works, [11] showed a single-round *proactive* PPSS in the PKI setting for the case of $t = n$, and [3] showed general methods for ensuring robustness in PPSS reconstruction, and a non-ROM PPSS using $O(|\mathsf{pw}|)$ exponentiations in a prime-order group.

Another important aspect of these PPSS solutions is the type of security notion they achieve. Both the PKI-model PPSS notion of [4] and the PKI-free PPSS notion of [18] were indistinguishability-based, while [10,19] provided Universally Composable (UC) definitions of the PPSS functionality. The essence of the UC PPSS definition of [19], which we adopt here, is that the only attack the adversary can stage is the inevitable one, namely, an online dictionary attack where validating a single password guess requires interaction with either $t+1$ instances of the servers or with the user. The UC definitions have further advantages for a password-based notion like PPSS, e.g. they imply security in the presence of non-uniformly distributed passwords, correlated passwords used for different services, and password mistyping.

Organization. In Sect. 2 we define the fundamental tool TOPPSS relies on, namely T-OPRF, as a UC functionality. In Sect. 3 we show a single-round, 1exp/server + 2exp/client realization of T-OPRF, protocol 2HashTDH, secure in ROM under the Threshold OMDH assumption we introduce in that section. In Sect. 4 we show a low-cost compiler from T-OPRF to PPSS, which we exemplify in Sect. 5 with a concrete instantiation using 2HashTDH.

2 Universally Composable Threshold OPRF

Notation. We use "$:=$" for deterministic assignment, "$\leftarrow$" for randomized assignment, and "$\leftarrow_R$" for uniform sampling from some set.

The T-OPRF Functionality. In the introduction we gave an informal overview of the notion of Threshold Oblivious PRF (T-OPRF) and its applicability, e.g. to PPSS schemes. Here we provide a formal definition of this notion as a secure realization of the UC functionality $\mathcal{F}_{\mathrm{TOPRF}}$ shown in Fig. 1 which generalizes the single-server (non-verifiable) OPRF functionality of [19] to the multi-party setting. In the $\mathcal{F}_{\text{TOPRF}}$ setting, the PRF key is effectively controlled by a collection of n servers and it remains secret as long as no more than a threshold t of these servers are corrupted. Such (t, n)-threshold "collective control" over a functionality can be realized as we show in our 2HashTDH realization in Sect. 3. We chose to base the T-OPRF notion on the *non-verifiable OPRF* notion of [19] rather than the *verifiable OPRF* notion of [18] because the former was shown to have a more efficient realization under the same assumptions, and because this form of OPRF suffices in the key application of interest to us, namely, Password-Protected Secret-Sharing.

Assume $\mathsf{tx}(p, S)$ and $T(p, x)$ are undefined for all p, x, S.

Initialization

- On message $(\textsc{Init}, sid, \mathcal{SI})$ from S, ignore it if $|\mathcal{SI}| \neq n$ or S is active. Otherwise mark S as "active" and if no record $\langle sid, [...] \rangle$ exists, pick any previously unused label p and record $\langle sid, \mathcal{SI}, p \rangle$. Send $(\textsc{Init}, sid, S, \mathcal{SI}, p)$ to $\mathcal{A}^*$.
- On message $(\textsc{Init}, sid, \mathcal{A}^*, p)$ from $\mathcal{A}^*$, check that p is a label that has not been used before, record $\langle \mathcal{A}^*, p \rangle$ and return $(\textsc{Init}, sid, \mathcal{A}^*, p)$ to $\mathcal{A}^*$.
- On message $(\textsc{InitComplete}, sid, S)$ from $\mathcal{A}^*$, retrieve tuple $\langle sid, \mathcal{SI}, p \rangle$. Ignore the message if there is no such tuple or $S \notin \mathcal{SI}$ or not all servers in $\mathcal{SI}$ are active. Otherwise, send $(\textsc{InitComplete}, sid)$ to S and mark S as "initialized."

Evaluation

- On message $(\textsc{Eval}, sid, ssid, \mathcal{SE}, x)$ from $P \in \{U, \mathcal{A}^*\}$, retrieve $\langle sid, \mathcal{SI}, p \rangle$ if $P = U$ or $\langle \mathcal{A}^*, p \rangle$ if $P = \mathcal{A}^*$. Ignore this message if there is no such tuple, if $|\mathcal{SE}| \neq t{+}1$, or if tuple $\langle ssid, P, \cdot, \cdot, \cdot \rangle$ already exists. Otherwise record $\langle ssid, P, p, \mathcal{SE}, x \rangle$ and send $(\textsc{Eval}, sid, ssid, P, \mathcal{SE})$ to $\mathcal{A}^*$.
- On message $(\textsc{SndrComplete}, sid, ssid, S)$ from $\mathcal{A}^*$, retrieve tuple $\langle sid, \mathcal{SI}, p \rangle$. Ignore this message if there is no such tuple, or if $S \notin \mathcal{SI}$, or if S is not initialized. Otherwise, set $\mathsf{tx}(p, S){+}{+}$ (or set it to 1 if $\mathsf{tx}(p, S)$ is undefined), and send $(\textsc{SndrComplete}, sid, ssid)$ to S.
- On message $(\textsc{RcvComplete}, sid, ssid, P, p^*)$ from $\mathcal{A}^*$, retrieve $\langle ssid, P, p, \mathcal{SE}, x \rangle$. Ignore this message if there is no such tuple, or if any of the following conditions fails: (i) if $p^* = p$ then $|\{S \in \mathcal{SI} \mid \mathsf{tx}(p, S) > 0\}| > t$, (ii) if all servers in $\mathcal{SE}$ are honest then $p^* = p$. Otherwise, if $p^* = p$ then set $\mathsf{tx}(p, S){-}{-}$ for any $t + 1$ distinct $S \in \mathcal{SI}$ s.t. $\mathsf{tx}(p, S) > 0$, and if $T(p^*, x)$ is undefined then pick $\rho \leftarrow_{\mathrm{R}} \{0, 1\}^{\ell}$ and set $T(p^*, x) := \rho$. Finally, send $(\textsc{Eval}, sid, ssid, T(p^*, x))$ to P.

Fig. 1. Functionality $\mathcal{F}_{\text{TOPRF}}$ with parameters t, n.

The T-OPRF functionality of Fig. 1 has two stages, Initialization and Evaluation. The functionality enforces that the outputs of any such function are uniformly disributed, similarly to the single-server OPRF notion of [19], even in the case that the adversary controls the private key and/or its sharing among the

n servers. In more detail, in the initialization stage, a set of n servers, denoted $\mathcal{SI}$, are activated at the discretion of the adversary. The stage is complete when all servers become active. Note that the set may include adversarial servers, yet the functionality guarantees that all servers identified in $\mathcal{SI}$ become active by the end of the initialization stage. The initialization also specifies a parameter p used to identify a table $T(p, \cdot)$ of random values that defines the proper PRF values computed by the user when interacting with any subset of $t + 1$ *honest* servers from the set $\mathcal{SI}$. Additional parameters p^*, and corresponding tables $T(p^*, \cdot)$, can be specified by the adversary to represent rogue tables with values computed by the user in interaction with corrupted servers (see more on this below). The parameter p is also used to identify a counter $\mathsf{tx}(p, S)$ for each $S \in \mathcal{SI}$ as specified below.

In the evaluation stage, users connect to an arbitrary set of servers $\mathcal{SE}$ chosen by the adversary and which may arbitrarily overlap with $\mathcal{SI}$ (representing the fact that the user has no memory of who the servers in $\mathcal{SI}$ are). When, at the discretion of the adversary, a server $S \in \mathcal{SI}$ completes its interaction, the functionality increases the counter $\mathsf{tx}(p, S)$. Eventually, the adversary can trigger a response to the user which will be drawn from one of the tables maintained by the functionality. Recall that in addition to the proper table $T(p, \cdot)$ the adversary can register additional function tables $T(p^*, \cdot)$ and may connect an evaluation request from a user to any such table of its choice.

The security guarantees provided by the T-OPRF functionality are the following: (1) it enforces the use of the proper function table p whenever the set of servers $\mathcal{SE}$ selected for an evaluation are all honest; (2) it "charges" $t + 1$ server tickets for accessing the proper table p by decrementing (non-zero) ticket counters $\mathsf{tx}(p, S)$ for an arbitrary set of $t + 1$ servers in $\mathcal{SI}$; and (3) all tables T (the proper table p as well as any additional ones set by the adversary with $p^* \neq p$) are filled with random entries that are chosen on demand as the functionality responds back to the user. These guarantees ensure that at least $t + 1 - t'$ honest servers from $\mathcal{SI}$ need to be contacted for the proper function to be evaluated once. To see why this is the case observe that $t + 1$ tickets are "spent" (decremented) during evaluation which correspond to at least $t + 1 - t'$ tickets from honest ticketing counters. This implies that $t + 1$ servers from $\mathcal{SI}$ have registered a SndrComplete message as this is the only event that triggers a counter increment. In the real world this corresponds to the event that a server has completed its interaction with a user that attempts to perform an evaluation.

It is important to highlight that the functionality does not necessarily decrement the ticketing counters of the servers identified in the chosen evaluation set $\mathcal{SE}$; rather, it decrements an arbitrary set of $t + 1$ non-zero counters for servers in $\mathcal{SI}$. This reflects the fact that the functionality does not provide any guarantee about the identities of the responding servers. For instance, this means that we allow for an implementation of T-OPRF where an honest user U attempts to connect to a set of servers $\mathcal{SE}_1$ that are corrupted and its message is rerouted by the adversary so that, unbeknownst to U, an honest set of servers servers $\mathcal{SE}_2$ becomes the responder set.

Another important point regarding the T-OPRF functionality is that while it guarantees correct OPRF evaluation in case the user completes an undisturbed interaction with $t+1$ honest servers in $\mathcal{SI}$, the ideal world adversary may also maintain an arbitrary collection of random tables and connect a user to them, if desired, as long as the responder set is not composed of honest servers only. For instance, the adversary can assign to a subset of corrupted servers $\mathcal{SE}_1$ a certain function table, while it can assign a different function table to a different subset of corrupted servers $\mathcal{SE}_2$. While the two function tables will be independent, they are not under the control of the ideal world adversary completely: their contents will be populated by the ideal functionality with random values independently of each other. In practice this means that we allow for an implementation where two successive evaluation requests for the same x value result in a different (but still random) value to be produced, depending on which set of servers the user connects to. We stress that the secrecy of the input x is always preserved irrespectively of the subset of servers the user communicates with. At the same time, observe that the randomness requirement imposed for adversarial tables restricts our ability to implement the functionality to the random oracle setting.

3 Threshold OPRF Protocol from OMDH and T-OMDH

Here we present our Threshold Oblivious PRF protocol, called 2HashTDH, that instantiates the $\mathcal{F}_{\mathrm{TOPRF}}$ functionality defined in Sect. 2. Thus, 2HashTDH provides a secure T-OPRF for use in general applications and, in particular, as the basis for our PPSS scheme, TOPPSS, presented in Sect. 4. The 2HashTDH scheme is formally defined as a realization of $\mathcal{F}_{\mathrm{TOPRF}}$ in Fig. 3. In a nutshell, it is a threshold version of the 2HashDH OPRF from [19], recalled in Fig. 2. The underlying PRF, $f_k(x) = H_2(x, (H_1(x))^k)$, remains unchanged, but the key k is shared using Shamir secret-sharing across n servers, where server S_i stores the key share k_i. The initialization of such secret-sharing can be done via a Distributed Key Generation (DKG) for discrete-log-based systems, e.g. [16], and in Fig. 2 we assume it is done with a UC functionality $\mathcal{F}_{\mathrm{DKG}}$ which we discuss further below. For evaluation, given any subset $\mathcal{SE}$ of $t+1$ servers, the user U sends to each of them the *same* message $a = (H'(x))^r$ for random r, exactly as

PRF Definition
G: a group of prime order m; H, H': hash functions with resp. ranges $\{0,1\}^\ell$ and G.
PRF f from $\{0,1\}^*$ to $\{0,1\}^\ell$: For a key $k \leftarrow_{\mathrm{R}} \mathbb{Z}_m$, define $f_k(x) = H(x, (H'(x))^k)$.

Oblivious Computation of PRF $\boldsymbol{f_k(x)}$ between user $\boldsymbol{U}$ and server $\boldsymbol{S}$

1. On input x, U chooses $r \leftarrow_{\mathrm{R}} \mathbb{Z}_m$; sends $a = (H'(x))^r$ to S.
2. S verifies that the received a is in group G and if so it responds with $b = a^k$.
3. U outputs $f_k(x) = H(x, b^{1/r})$.

Fig. 2. The 2HashDH OPRF [19]

in the single-server OPRF protocol 2HashDH. If each server S_i in $\mathcal{SE}$ returned $b_i = a^{k_i}$ then U could reconstruct the value a^k using standard Lagrange interpolation in the exponent, i.e. $a^k = \prod_{i \in \mathcal{SE}} b_i^{\lambda_i}$ with the Lagrange coefficients λ_i computed using the indexes of servers in $\mathcal{SE}$. After computing a^k, the value of $f_k(x)$ is computed by U by deblinding a^k exactly as in the case of protocol 2HashDH. Note that this takes a single exponentiation for each server and two exponentiations for the user (to compute a and to deblind a^k) plus one multi-exponentiation by U to compute the Lagrange interpolation on the b_i values. We optimize this function evaluation by having each server S_i compute $b_i = a^{\lambda_i \cdot k_i}$, which costs one exponentiation and $O(t)$ multiplications and divisions in $\mathbb{Z}_m$ to compute λ_i. (Note that S_i must know set $\mathcal{SE}$ to compute λ_i.) This way U can compute a^k using only t multiplications instead of a multi-exponentiation, and the total costs are 1 exps for each S_i and 2 exps for U.

Protocol 2HashTDH can be also be seen as a simplification of a protocol which results from a generic transformation of any OPRF to T-TOPRF using multi-party secure computation of the server code, and then applying this transformation to the 2HashDH OPRF of [19]. The server in 2HashDH computes a^k on input a, and the MPC protocol for it is exactly the *threshold exponentiation* protocol described above, except that this generic OPRF to T-OPRF transformation must assure that the servers perform the MPC subprotocol on the same input a, and this involves an additional round of server-to-server interaction, which the 2HashTDH protocol avoids. We refer to the full version of this paper [20] for the specification of this general OPRF to T-OPRF compiler.

Roadmap. In Sect. 3.1 we show protocol 2HashTDH and explain the assumptions taken in its specification. In Sect. 3.2 we introduce the T-OMDH assumption, a generalization of OMDH, and we show that it is equivalent to OMDH in several cases, including the *full corruption* case $t' = t$ discussed in the introduction. In Sect. 3.3 we show that protocol 2HashTDH realizes the Threshold OPRF functionality $\mathcal{F}_{\text{TOPRF}}$ under the T-OMDH assumption in ROM for any threshold parameters (t, n) and any number $t' < t$ of corrupted servers. It follows that protocol 2HashTDH achieves the *standard threshold security* property, which corresponds to the full corruption case, under just OMDH in ROM. Note that the non-threshold OPRF 2HashDH of [19] also relies on OMDH.

3.1 T-OPRF Protocol Based on T-OMDH Assumption

The 2HashTDH T-OPRF protocol is shown in Fig. 3, relying on realizations of functionalities $\mathcal{F}_{\text{DKG}}, \mathcal{F}_{\text{AUTH}}$ and $\mathcal{F}_{\text{SEC}}$, which model, respectively, the distributed key generation, authenticated channel, and secure channel. Assuming these functionalities, the 2HashTDH protocol realizes the UC T-OPRF functionality defined in Sect. 2, under the T-OMDH assumption in ROM. As we argue in Sect. 3.2, this implies security under OMDH in ROM in several cases, including the *full corruption* case, where the adversary corrupts $t' = t$ servers, and the *additive sharing* case, where $t = n - 1$. Functionalities $\mathcal{F}_{\text{DKG}}, \mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{SEC}}$ all have well-known efficient realizations in ROM under the Diffie-Hellman assumption which is implied by OMDH, and hence also by T-OMDH.

Let $\mathcal{F}_{\text{DKG}}, \mathcal{F}_{\text{AUTH}}$ and $\mathcal{F}_{\text{SEC}}$ be, respectively, the distributed key generation, authenticated channel, and secure channel functionalities; Let G be a cyclic group of prime order m; Let H_1, H_2 be hash functions with resp. ranges G and $\{0,1\}^{\ell}$.

Initialization

S1: On input $(\textsc{Init}, \mathit{sid}, \mathcal{SI}{=}(S_1, \ldots, S_n))$, server S forwards this input to $\mathcal{F}_{\text{DKG}}$.
S2: On message $(\textsc{InitComplete}, \mathit{sid}, y, k_i)$ from $\mathcal{F}_{\text{DKG}}$, server S records $(\mathit{sid}, \mathcal{SI}, y, i, k_i)$, marks itself active, and outputs $(\textsc{InitComplete}, \mathit{sid})$.

Evaluation

U1: On input $(\textsc{Eval}, \mathit{sid}, \mathit{ssid}, \mathcal{SE}, x)$, U picks $r \leftarrow_{\text{R}} \mathbb{Z}_m$, computes $a := H_1(x)^r$, and sends $(\textsc{Send}, (\mathit{sid}, \mathit{ssid}, 1), S, (\mathcal{SE}, a))$ to $\mathcal{F}_{\text{AUTH}}$ for all $S \in \mathcal{SE}$.
S1: On message $(\textsc{Sent}, (\mathit{sid}, \mathit{ssid}, 1), U, (\mathcal{SE}, a))$ from $\mathcal{F}_{\text{AUTH}}$, server S, provided it is active, computes $b_i := a^{\lambda_i \cdot k_i}$ where λ_i is a Lagrange interpolation coefficient for index i and index set $\mathcal{SE}$, sends $(\textsc{Send}, (\mathit{sid}, \mathit{ssid}, 2), U, b_i)$ to $\mathcal{F}_{\text{AUTH}}$, and outputs $(\textsc{SndrComplete}, \mathit{sid}, \mathit{ssid})$.
U2: When U receives $(\textsc{Sent}, (\mathit{sid}, \mathit{ssid}, 2), S_i, b_i)$ from $\mathcal{F}_{\text{AUTH}}$ for all $S_i \in \mathcal{SE}$, it outputs $(\textsc{Eval}, \mathit{sid}, \mathit{ssid}, H_2(x, (\prod_{S_i \in \mathcal{SE}} b_i)^{1/r}))$.

Fig. 3. Protocol 2HashTDH realizing $\mathcal{F}_{\text{TOPRF}}$ assuming $\mathcal{F}_{\text{DKG}}, \mathcal{F}_{\text{SEC}}, \mathcal{F}_{\text{AUTH}}$.

Note on Authentic and Secret Channels. In Fig. 3 protocol 2HashTDH is presented in the $(\mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{SEC}}, \mathcal{F}_{\text{DKG}})$-hybrid world, i.e., assuming that there are both authenticated and secure (i.e. authenticated and secret) channels between protocol participants. We refer to [12] for the UC models of authenticated and secret channels, but simply speaking, what the authenticated and secure channel functionalities model is that if party P_1 sends message m to party P_2 using $\mathcal{F}_{\text{AUTH}}$ command $(\textsc{Send}, \mathit{sid}, P_2, m)$, then P_2 will be able to authenticate m as originated from P_1, i.e. if P_2 receives command $(\textsc{Sent}, \mathit{sid}, P_1, m')$, it is guaranteed that $m' = m$, and if P_1 sends m to P_2 using $\mathcal{F}_{\text{SEC}}$ command $(\textsc{Send}, \mathit{sid}, P_2, m)$, then P_2 can verify authenticity of P_1's message as above, but in addition m will be hidden to the adversary unless P_2 is corrupted.

We note that using ideal functionalities such $\mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{SEC}}$ in the hybrid world, does not determine their implementation when the UC protocol is deployed in the real world. This is because they only describe how the adversarial model against the protocol is envisioned. For instance, $\mathcal{F}_{\text{AUTH}}$ *may* be realized using a PKI involving all connected participants, or it may be simply substituted by unauthenticated TCP/IP communication in case it is deemed that modifying message contents is not a relevant threat in the protocol deployment. Indeed, this will also be the case in our setting since we allow the (adversarial) environment to choose the servers that a user connects in the evaluation stage of the protocol in a way that is independent from the initialization servers; in this way, any man-in-the-middle scenario can be simulated by the adversary without violating the $\mathcal{F}_{\text{AUTH}}$ constraints. Similarly, $\mathcal{F}_{\text{SEC}}$ *may* be implemented by TLS, but may also be achieved in other ways, e.g., physically transferring private state between the parties engaged in the protocol.

A second important point is that if a user U initializes a T-OPRF instance with a server set $\mathcal{SI} = \{S_1, ..., S_n\}$ such that some subset B of $\mathcal{SI}$ is made of corrupt entities (which models both the fact that some $\mathcal{SI}$ members are corrupt and the fact that U can execute T-OPRF initialization on an incorrect set of servers), then in this case command $(\textsc{Send}, sid, S_i, m)$ for $S_i \in B$ will leak m to the adversary, and if U receives $(\textsc{Sent}, sid, S_i, m)$ from $\mathcal{F}_{\text{AUTH}}$ for $S_i \in B$, we can assume that the adversary supplies message m. In other words, the $\mathcal{F}_{\text{AUTH}}$ and $\mathcal{F}_{\text{SEC}}$ channels implement authenticated and/or secret point-to-point message delivery only if they are executed for a proper and non-corrupt server. We note that we assume a secret channel $\mathcal{F}_{\text{SEC}}$ in addition to an authenticated channel $\mathcal{F}_{\text{AUTH}}$ solely to simplify the description of T-OPRF initialization. Indeed, the former can be built from the latter [12], e.g. by having each server S_i first send its encryption public key to U using the authenticated channel.

Note on Distributed Key Generation. Protocol 2HashTDH assumes that servers in $\mathcal{SI}$ establish a secret-sharing $(k_1, ..., k_n)$ of a random key k over authenticated channels via a Distributed Key Generation (DKG) functionality $\mathcal{F}_{\text{DKG}}$, shown in Fig. 4. The DKG sub-protocol for discrete-log based cryptosystems can be efficiently realized without user's involvement [16,30], but if the call to initialize a TOPRF instance is executed by an *honest user* U then the DKG subprotocol can be even simpler, because U can generate sharing $(k_1, ..., k_n)$ of k and then distribute the shares among the servers in $\mathcal{SI}$. Note that since our realizations of $\mathcal{F}_{\text{TOPRF}}$ pertains only to the *static* adversarial model, where the identity of corrupt parties is determined at the outset, we would not explicitly require that the parties erase the information used in the initialization, but any implementation should erase such information. In our specification of protocol 2HashTDH we rely on the $\mathcal{F}_{\text{DKG}}$ functionality to abstract from any specific DKG implementation, e.g. whether it is done by the server or by an honest user.

3.2 Threshold OMDH Assumption

Notation. If n is an integer, then $[n] = \{1, ..., n\}$. If D is a set, then $|D|$ is its cardinality. We use bold font to denote vectors, e.g. $\boldsymbol{a} = [a_1, ..., a_n]$. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are two vectors of the same dimension, then $\boldsymbol{a} \odot \boldsymbol{b}$ is their Hadamard (component-wise) product. If $|\boldsymbol{a}| = n$ and J is a sequence in $[n]$ then $\boldsymbol{a}_J$ denotes the components of $\boldsymbol{a}$ with indices in J, i.e. $[a_{i_1}, ..., a_{i_k}]^T$ if $J = (i_1, ..., i_k)$.

Let $\mathcal{I}_w$ be the set of w-element subsets of $[n]$, i.e. $\mathcal{I}_w = \{I \subseteq [n] \ s.t. \ |I| = w\}$. Let $W(\boldsymbol{a})$ be the hamming weight of $\boldsymbol{a}$. Let $\mathcal{V}_w$ be the set of n-bit binary vectors $\boldsymbol{q}$ s.t. $W(\boldsymbol{q}) = w$, i.e. $\mathcal{V}_w = \{\boldsymbol{v} \in \{0,1\}^n \ s.t. \ v_i = 1 \text{ iff } i \in \mathcal{I}_w\}$. For $\boldsymbol{q} = [q_1, \ldots, q_n]$ define $C_w(\boldsymbol{q})$ as the maximum integer m for which there exist $\boldsymbol{v}_1, ..., \boldsymbol{v}_m \in \mathcal{V}_w$ (not necessarily distinct) s.t. $\boldsymbol{v}_1 + ... + \boldsymbol{v}_m \leq \boldsymbol{q}$. In other words, $C_w(\boldsymbol{q})$ is the

Let g generate a cyclic group G of prime order m; Let t, n be integers s.t. $t < n$.

- On message $(\textsc{Init}, \mathit{sid}, \mathcal{SI})$ from S, ignore it if $|\mathcal{SI}| \neq n$ or S is active or $S \notin \mathcal{SI}$. Otherwise, if no record $\langle \mathit{sid}, [...]\rangle$ exists, let $\mathsf{Corrupted}$ be the subset of $\mathcal{SI}$ that is corrupted and set $t' = |\mathsf{Corrupted}|$. If $t' \leq t$ then pick $a_0, a_1, \ldots, a_{t-t'} \leftarrow_{\mathrm{R}} \mathbb{Z}_m$, record $\langle \mathit{sid}, \mathcal{SI}, a_0, a_1, \ldots, a_{t-t'}\rangle$ else record $\langle \mathit{sid}, \mathcal{SI}\rangle$. Irrespectively, mark S as "active", and send $(\textsc{Init}, \mathit{sid}, P, \mathcal{SI})$ to $\mathcal{A}^*$.
- On message $(\textsc{Init}, \mathit{sid}, \mathcal{SI}, s)$ from a corrupted $S \in \mathcal{SI}$, if the record $\langle \mathit{sid}, \mathcal{SI}, [...]\rangle$ exists, record $\langle \mathcal{A}^*, S, s\rangle$, mark S as active and send $(\textsc{Init}, \mathit{sid}, \mathcal{A}^*, S)$ to $\mathcal{A}^*$.
- On message $(\textsc{InitComplete}, \mathit{sid}, S_i)$ from $\mathcal{A}^*$, retrieve tuple $\langle \mathit{sid}, \mathcal{SI}, a_0, a_1, \ldots, a_{t-t'}\rangle$. Ignore the message, if there is no such tuple, or if $S_i \notin \mathcal{SI}$ or not all servers in $\mathcal{SI}$ are active. Otherwise, send $(\textsc{InitComplete}, \mathit{sid}, g^{a_0}, i, s_i)$ to S_i and $(\textsc{InitComplete}, \mathit{sid}, S_i, g^{a_0})$ to $\mathcal{A}^*$, where $s_i = p(i)$ and $p(x)$ is a polynomial whose first $t - t' + 1$ coefficients match $a_0, a_1, \ldots, a_{t-t'}$ and $p(j) = s_j$ for each j such that $S_j \in \mathsf{Corrupted}$ and $\langle \mathcal{A}^*, S_j, s_j\rangle$ has been previously recorded.

Fig. 4. Distributed key generation functionality $\mathcal{F}_{\mathrm{DKG}}$ [30].

maximum number of times one can subtract elements in $\mathcal{V}_w$ from $\boldsymbol{q}$ s.t. the result remains $\geq \boldsymbol{0}$. For example if and $\boldsymbol{q} = [3, 3, 4]$ then $C_2(\boldsymbol{q}) = 4$ because $\boldsymbol{q} = 2 \times [1, 0, 1] + [1, 1, 0] + 2 \times [0, 1, 1]$.

T-OMDH Intuition. Let $\langle g\rangle$ be a cyclic group of prime order $m > n$. The T-OMDH assumption considers the setting where a random exponent $k \in \mathbb{Z}_m$ is secret-shared using a random t-degree polynomial $p(\cdot)$, and the n trustees holding shares $k_1{=}p(1), \ldots, k_n{=}p(n)$ implement a "threshold exponentiation" protocol which computes a^k for any given $a \in \langle g\rangle$ and $k = p(0)$. Let $\mathsf{TOMDH}_p(\cdot, \cdot)$ be an oracle which on input $(i, a) \in [n] \times \langle g\rangle$ outputs $a^{p(i)}$. The standard way to implement threshold exponentiation is to choose a set $I \in \mathcal{I}_{t+1}$, compute $b_i = \mathsf{TOMDH}_p(i, a) = a^{k_i}$ for each i in I and derive a^k as $\prod_{i \in I} b_i^{\lambda_i}$ using Lagrange interpolation coefficients λ_i s.t. $k = \sum_{i \in I} \lambda_i{\cdot}k_i$. The T-OMDH assumption states that querying oracle $\mathsf{TOMDH}_p(\cdot, \cdot)$ on at least $t + 1$ different points $i \in [n]$ is *necessary* to compute $a^{p(0)}$ for a given random challenge a. More generally, T-OMDH considers an experiment where the attacker $\mathcal{A}$ receives a challenge set $R = \{g_1, ..., g_N\}$ of random elements in $\langle g\rangle$ and is given access to the $\mathsf{TOMDH}_p(\cdot, \cdot)$ oracle for random t-degree polynomial $p(\cdot)$. T-OMDH assumption states that $\mathcal{A}$ can compute g_j^k for $k = p(0)$ for no more than $C_{t+1}(q_1, \ldots, q_n)$ elements $g_j \in R$, where q_i is the number of $\mathcal{A}$'s queries to $\mathsf{TOMDH}_p(i, \cdot)$.

The above intuition and Definition 1 below correspond to the setting where the attacker does not control any of the trustees holding shares of p, hence it needs $t + 1$ queries to $\mathsf{TOMDH}_p(\cdot, \cdot)$ to compute $a^{p(0)}$ for each random challenge a. Later we extend this definition to the case where $\mathcal{A}$ controls a subset of trustees.

Definition 1. *The (t, n, N, Q)-Threshold One-More Diffie Hellman (T-OMDH) assumption holds in group $\langle g \rangle$ of prime order m if the probability of any polynomial-time adversary $\mathcal{A}$ winning the following game is negligible. $\mathcal{A}$ receives challenge set $R = \{g_1, \ldots, g_N\}$ where $g_i \leftarrow_R \langle g \rangle$ for $i \in [N]$, and is given access to an oracle $\mathsf{TOMDH}_p(\cdot, \cdot)$ for a random t-degree polynomial $p(\cdot)$ over $\mathbb{Z}_m$. $\mathcal{A}$ wins if it outputs g_j^k where $k = p(0)$ for $Q + 1$ different elements g_j in R, and if $C_{t+1}(q_1, \ldots, q_n) \leq Q$ where q_i is the number of $\mathcal{A}$'s queries to $\mathsf{TOMDH}_p(i, \cdot)$.*

Note that the (N, Q)-OMDH assumption [5,22] is the (t, n, N, Q)-T-OMDH assumption for $t = 0$ and any $n \geq 1$, because then $p(\cdot)$ is a constant polynomial and $C_1(\boldsymbol{q}) = W(\boldsymbol{q})$, i.e. the total number of $\mathcal{A}$'s $\mathsf{TOMDH}_p(\cdot, \cdot)$ queries.

T-OMDH: The General Case. In its general form, the T-OMDH assumption corresponds to computing g_j^k if some subset of $t' \leq t$ trustees holding shares $k_i = p(i)$ is corrupt, and hence the adversary can not only learn these shares but can also set them at will.

Definition 2. *The (t', t, n, N, Q)-T-OMDH assumption holds in group $\langle g \rangle$ of prime order m if for any $\mathsf{B} \subseteq [n]$ s.t. $|\mathsf{B}| = t' \leq t$, the probability of any polynomial-time adversary $\mathcal{A}$ winning the following game is negligible. On input a challenge set $R = \{g_1, \ldots, g_N\}$ where $g_i \leftarrow_R \langle g \rangle$ for $i \in [N]$, adversary $\mathcal{A}$ specifies a set of t' values $\{\alpha_j\}_{j \in \mathsf{B}}$ in $\mathbb{Z}_m$. A random t-degree polynomial $p(\cdot)$ over $\mathbb{Z}_m$ is then chosen subject to the constraint that $p(j) = \alpha_j$ for $j \in \mathsf{B}$, and the adversary $\mathcal{A}$ is given access to oracle $\mathsf{TOMDH}_p(\cdot, \cdot)$. We say that $\mathcal{A}$ wins if it outputs g_j^k where $k = p(0)$ for $Q + 1$ different elements g_j in R, and if $C_{t-t'+1}(q_1, \ldots, q_n) \leq Q$ where q_i for $i \notin \mathsf{B}$ is the number of $\mathcal{A}$'s queries to $\mathsf{TOMDH}_p(i, \cdot)$, and $q_i = 0$ for $i \in \mathsf{B}$.*

Note that (t', t, n, N, Q)-T-OMDH is identical to (t, n, N, Q)-T-OMDH for $t' = 0$.

Gap T-OMDH. In order to prove the security of T-OPRF, we need to extend the T-OMDH assumption stated in Definition 2 to its "gap" form, i.e. suppose $\langle g \rangle$ is a gap group where $\mathcal{A}$ is in addition given access to the DDH oracle in $\langle g \rangle$.

Definition 3. *The* Gap *(t', t, n, N, Q)-T-OMDH assumption is the T-OMDH assumption of Definition 2 except that $\mathcal{A}$ is also given access to the DDH oracle in group $\langle g \rangle$, which on input (a, b, c, d) outputs 1 if $\log_a b = \log_c d$ and 0 otherwise.*

In the full version of this paper [20] we show that the (Gap) (t, t', n, N, Q)-T-OMDH assumption holds in the generic group model for any (t', t, n). Specifically, the advantage of a T-OMDH adversary restricted to r generic group operations is upper-bounded by $O(Qr^2/m)$, assuming $r \geq Q \geq N$. This is larger by factor Q from the $O(r^2/m)$ upper-bounds on generic group attacks against many static problems related to discrete logarithm [28], and this weakening is caused by the presence of up to Q-degree polynomials of the "target" secret $k = p(0)$ in the representation of the group elements which the adversary can compute given access to $\mathsf{TOMDH}_p(\cdot, \cdot)$ using the query pattern $\boldsymbol{q} = [q_1, ..., q_n]$ s.t. $C_{t-t'+1}(\boldsymbol{q}) \leq Q$. Since (Q, N)-OMDH is identical to (t', t, n, N, Q)-T-OMDH for $(t', t) = (0, 0)$

and any n, the same upper-bound applies applies to OMDH, and to the best of our knowledge this is the first generic model security hardness argument for the OMDH (or Gap OMDH) assumption.

T-OMDH = OMDH in Full Corruption and Additive Sharing Cases. The T-OMDH and OMDH assumptions are equivalent in two important cases, namely the *full corruption* case of $t' = t$, for any (t, n), and in the *additive sharing* case of $t = n - 1$, for any t'. We refer to the full version of this paper [20] for (easy) proofs of above equivalences. Note also that whereas the question whether the T-OMDH and OMDH assumptions are equivalent for any $t' < t$ and $t + 1 < n$ remains open, in the full version [20] we also show the same generic group hardness bound for both problems.

3.3 Security Analysis of 2HashTDH

Protocol 2HashTDH protocol of Fig. 3 is secure under the T-OMDH assumption. As a corollary of the fact that in the full corruption case of $t' = t$ faults the T-OMDH and OMDH assumptions are equivalent, Theorem 1 implies that protocol 2HashTDH is secure under OMDH in ROM in the full corruption case of $t' = t$. The proof of Theorem 1 appears in the full version of this paper [20].

Theorem 1. *Protocol 2HashTDH realizes functionality $\mathcal{F}_{\mathrm{TOPRF}}$ with parameters t, n in the $(\mathcal{F}_{\mathrm{AUTH}}, \mathcal{F}_{\mathrm{SEC}}, \mathcal{F}_{\mathrm{DKG}})$-hybrid model, assuming static corruptions, hash functions $H_1(\cdot)$ and $H_2(\cdot, \cdot)$ modelled as Random Oracles, and the Gap (t', t, n, N, Q)-T-OMDH on group $\langle g \rangle$, where Q is the number of* Eval *messages sent by any user, $N = Q + q_1$ where q_1 is the number of $H_1(\cdot)$ queries the adversary makes, and $t' < t$ is the number of corrupted servers in $\mathcal{SI}$.*

Specifically, for any efficient adversary $\mathcal{A}$ against protocol 2HashTDH, there is a simulator SIM *s.t. no efficient environment $\mathcal{Z}$ can distinguish the view of $\mathcal{A}$ interacting with the real 2HashTDH protocol and the the view of* SIM *interacting with the ideal functionality $\mathcal{F}_{\mathrm{TOPRF}}$, with advantage better than $q_T \cdot \epsilon(N, Q) + N^2/m$, where q_T is the number of TOPRF instances, $\epsilon(N, Q)$ is the bound on the probability that any algorithm of the same cost violates the Gap (t', t, n, N, Q)-T-OMDH assumption, and $m = |\langle g \rangle|$.*

4 TOPPSS: A PPSS Scheme Based on T-OPRF

In Fig. 5 we show a compiler which converts a T-OPRF scheme which realizes the UC T-OPRF notion of Sect. 2 into a PPSS scheme, called TOPPSS, which realizes UC PPSS functionality of [19]. The terminology of the UC setting might obscure the amazing practicality of this construction, so in Sect. 5 we show a concrete implementation of this scheme with the $\mathcal{F}_{\mathrm{TOPRF}}$ functionality implemented using the T-OPRF instantiation 2HashTDH from Sect. 3.

TOPPSS Overview. To explain the mechanics of TOPPSS based on the T-OPRF functionality, it is instructive to compare it to the OPRF-based PPSS

Let $\mathcal{F}_{\text{AUTH}}$ and $\mathcal{F}_{\text{TOPRF}}$ be, respectively, the authenticated channel and the T-OPRF functionality; Let $H(\cdot)$ be a hash function with range $\{0,1\}^{\ell}$.

<u>Init for user U</u>:

1. On input $(\textsc{Init}, sid, \mathcal{SI}, \mathsf{pw})$, send $(\textsc{Send}, (sid, 0), S, \mathcal{SI})$ to $\mathcal{F}_{\text{AUTH}}$ for all S in $\mathcal{SI}$.
2. On $(\textsc{Sent}, (sid, 1), S, \textsc{done})$ from $\mathcal{F}_{\text{AUTH}}$ for all $S \in \mathcal{SI}$, send $(\textsc{Eval}, sid, 0, \mathcal{SE}, \mathsf{pw})$ to $\mathcal{F}_{\text{TOPRF}}$ for any $\mathcal{SE} \subseteq \mathcal{SI}$ such that $|\mathcal{SE}| = t + 1$.
3. On $\mathcal{F}_{\text{TOPRF}}$'s response $(\textsc{Eval}, sid, 0, v)$, parse $H(v)$ as $[C|K]$ and send $(\textsc{Send}, (sid, 2), S, C)$ to $\mathcal{F}_{\text{AUTH}}$ for every $S \in \mathcal{SI}$.
4. On $(\textsc{Sent}, (sid, 3), S, \textsc{ack})$ for all $S \in \mathcal{SI}$ from $\mathcal{F}_{\text{AUTH}}$, output $(\text{U}\textsc{Init}, sid, K)$.

<u>Init for server S</u>:

1. On $(\textsc{Sent}, (sid, 0), U, \mathcal{SI})$ from $\mathcal{F}_{\text{AUTH}}$, send $(\textsc{Init}, sid, \mathcal{SI})$ to $\mathcal{F}_{\text{TOPRF}}$.
2. On $(\textsc{InitComplete}, sid)$ from $\mathcal{F}_{\text{TOPRF}}$, send $(\textsc{Send}, (sid, 1), U, \textsc{done})$ to $\mathcal{F}_{\text{AUTH}}$.
3. On $(\textsc{Sent}, (sid, 2), U, C)$ from $\mathcal{F}_{\text{AUTH}}$, record (sid, C), send $(\textsc{Send}, (sid, 3), U, \textsc{ack})$ to $\mathcal{F}_{\text{AUTH}}$, and output $(\text{S}\textsc{Init}, sid)$.

<u>Rec for user U</u>:

1. On input $(\textsc{Rec}, sid, ssid, \mathcal{SR}, \mathsf{pw}')$ send $(\textsc{Eval}, sid, [1|ssid], \mathcal{SR}, \mathsf{pw}')$ to $\mathcal{F}_{\text{TOPRF}}$.
2. On $\mathcal{F}_{\text{TOPRF}}$'s response $(\textsc{Eval}, sid, [1|ssid], v')$ and $(\textsc{Sent}, (sid, ssid, 1), S, C')$ from $\mathcal{F}_{\text{AUTH}}$ for all $S \in \mathcal{SR}$, if each message contains C' s.t. $[C'|K'] = H(v')$, then set $\textsc{Res} := K'$, otherwise set $\textsc{Res} := \textsc{Fail}$. Output $(\text{U}\textsc{Rec}, sid, ssid, \textsc{Res})$.

<u>Rec for server S</u>:

1. On $(\textsc{SndrComplete}, sid, [1|ssid])$ from $\mathcal{F}_{\text{TOPRF}}$, if S holds record (sid, C), then send $(\textsc{Send}, (sid, ssid, 1), U, C)$ to $\mathcal{F}_{\text{AUTH}}$ and output $(\text{S}\textsc{Rec}, sid, ssid)$.

Fig. 5. The TOPPSS protocol

scheme of [19]. In that scheme each server holds its own *independently random* key k_i for an OPRF f. At initialization, the secret to be protected is processed with a (t, n) secret sharing scheme and each share is stored at one of n servers, where server S_i stores the i-th share encrypted under $f_{k_i}(\mathsf{pw})$. At reconstruction, the user receives the encrypted shares from $t+1$ servers which it decrypts using the values $f_{k_i}(\mathsf{pw})$ that it learns by running the OPRF on pw with each of these servers. By contrast, in our TOPPSS scheme, which is T-OPRF-based, the (random) secret to be protected is defined as a single PRF value $v = f_k(\mathsf{pw})$ where k is a key secret-shared as part of a T-OPRF scheme. This provides a significant performance gain by reducing the number of exponentiations performed by the user from $t+2$ to just 2. In the scheme of [19] implemented with 2HashDH, the user computes the OPRF sub-protocol with each server independently, which involves one blinding operation re-used across all servers, but requires one deblinding operation *per server* for a total of $t+2$ exponentiations. By contrast, in the T-OPRF protocol 2HashTDH of Sect. 3 the user performs a single blinding and de-blinding, hence just 2 exponentiations, regardless of the number of servers and threshold t.

Note that the T-OPRF functionality allows the user to evaluate function $f_k(\cdot)$ on the user's password pw, without leaking any information about pw, but it does not let the user verify whether the function is computed correctly. Indeed, following the rules of functionality $\mathcal{F}_{\mathrm{TOPRF}}$, either corrupt servers or a man-in-the-middle adversary could make the user compute $f_k(\mathsf{pw})$ on key k of their choice. If the dictionary D from which the user draws her password is small, the adversary can potentially pick k s.t. function $f_k(\cdot)$ behaves on domain D in some ways the adversary can exploit (e.g., reducing the number of possible outputs). However, since $\mathcal{F}_{\mathrm{TOPRF}}$ assures that $f_k(\cdot)$ behaves like a random function for all k's, even for k's chosen by the adversary, it suffices to include a commitment to the master secret $v = f_k(\mathsf{pw})$ in the information that the servers send to the user, so that the user can verify its correctness. The adversary can still pick k but if $f_k(\cdot)$ is pseudorandom for all k then the adversary cannot change either k or v without guessing pw. Note that the randomness for verifying this commitment must be derived from the committed plaintext $f_k(\mathsf{pw})$ itself as this is the only value the user can retrieve using its only input pw. Although this mechanism requires the commitment scheme to be deterministic, the hiding property of the commitment is still satisfied thanks to the pseudorandomness of the committed plaintext $v = f_k(\mathsf{pw})$ (and assuming no more than t corruptions).

Since our realizations of $\mathcal{F}_{\mathrm{TOPRF}}$, protocol 2HashTDH, requires the Random Oracle Model (ROM) for hash functions in the security analysis, we implement this commitment simply with another hash function modeled as a random oracle. Finally, since the user needs to verify the master-secret v as well as to derive a key K from it, we implement both operation using a single hash function call, i.e. we set $[C|K]$ to $H(v)$ where H hashes onto strings of length 2ℓ.

The proof of the following theorem is in the full version of this paper [20].

Theorem 2. *The TOPPSS scheme of Fig. 5 UC-realizes the PPSS functionality $\mathcal{F}_{\mathrm{PPSS}}$ assuming access to the T-OPRF functionality $\mathcal{F}_{\mathrm{TOPRF}}$ and to the authenticated message delivery functionality $\mathcal{F}_{\mathrm{AUTH}}$, and assuming that hash function H is a random oracle.*

5 Concrete Instantiation of TOPPSS Using 2HashTDH

For concreteness we show an instantiation of TOPPSS with the T-OPRF functionality realized by protocol 2HashTDH from Fig. 3 in Sect. 3. In this figure we realize the $\mathcal{F}_{\mathrm{DKG}}$ subprotocol assuming an honest user U, because in the context of a PPSS protocol, we only care about security for PPSS instances which were initialized with an honest user. Hence we simply have U create the sharing of the T-OPRF key and distributing it among the servers in $\mathcal{SI}$ (see a note on DKG in

Parameters: Security parameter ℓ, threshold parameters t, n, cyclic group $\langle g \rangle$ of prime order m, hash functions H_1, H_2, H_3 with ranges $\langle g \rangle$, $\{0,1\}^{2\ell}$ and $\{0,1\}^{2\ell}$.
Communication Setting: Communication between U and S_i's in Init goes over secure channels, e.g. TLS, communication in Rec not necessarily (see text).

Init for user U on input $(sid, \mathcal{SI}, \mathsf{pw})$ where $\mathcal{SI} = \{S_1, ..., S_n\}$:

1. Pick $k \leftarrow_{\mathrm{R}} \mathbb{Z}_m$, generate $(k_1, \ldots, k_n)$ as a (t,n)-Shamir's secret sharing of k over $\mathbb{Z}_m$, and send $(sid, \mathcal{SI}, i, k_i)$ to each $S_i \in \mathcal{SI}$.
2. After receiving Ack's from all servers in $\mathcal{SI}$, compute $v := H_2(\mathsf{pw}, H_1(\mathsf{pw})^k)$, parse $H_3(v)$ as $[C|K]$, send (sid, C) to all $S_i \in \mathcal{SI}$ and output K.

Init for server S_i:

1. On $(sid, \mathcal{SI}, i, k_i)$ from U, abort if sid not unique or $S_i \notin \mathcal{SI}$, otherwise record $(sid, \mathcal{SI}, i, k_i)$ and send (sid, Ack) to U.
2. On (sid, C) from U, append C to tuple $(sid, \mathcal{SI}, i, k_i)$.

Rec for user U on input $(sid, ssid, \mathcal{SR}, \mathsf{pw})$ where $\mathcal{SR} = \{S'_1, ..., S'_{t+1}\}$:

1. Pick $r \leftarrow_{\mathrm{R}} \mathbb{Z}_m$, set $a := H_1(x)^r$, send $((sid, ssid), (\mathcal{SR}, a))$ to all $S'_i \in \mathcal{SR}$.
2. After receiving message $((sid, ssid), b_i, C)$ from all servers in $\mathcal{SR}$, abort if not all C's are the same. Otherwise compute $b := \prod_{S'_i \in \mathcal{SR}} b_i$ and $v := H_2(x, b^{1/r})$.
3. Parse $H_3(v)$ as $[C'|K']$. If $C' = C$ then output K', otherwise output Fail.

Rec for server $S_i \in \mathcal{SR}$:

1. On message $((sid, ssid), (\mathcal{SR}, a))$ from U, recover stored tuple $(sid, \mathcal{SI}, i, k_i, C)$. (Abort if there is no such tuple or if $a \notin \langle g \rangle$.) Compute $b_i := a^{\lambda_i \cdot k_i}$ where λ_i is an interpolation coefficient for index i and the subset of t+1 indices defined by $\mathcal{SR} \subseteq \mathcal{SI}$, and send $((sid, ssid), b_i, C)$ to U.

Fig. 6. Concrete instantiation of TOPPSS based on 2HashTDH T-OPRF.

Sect. 3.1). Note that if we implement $\mathcal{F}_{\mathrm{DKG}}$ in this user-centric way then we do not have to execute T-OPRF evaluation for U to compute $v = f_k(\mathsf{pw})$ as part of the initialization: User U can just compute $v = f_k(\mathsf{pw})$ locally because U picked the TOPRF key k (Fig. 6).

On the Role of Secure Channels. The communication in such instantiation of TOPPSS must go over secure channels in the *initialization phase*, which in practice could be implemented using e.g. TLS.[2] In the *reconstruction phase*, the communication does not have to go over secure channels, because TOPPSS is

[2] Note that if the $\mathcal{F}_{\mathrm{DKG}}$ was instantiated with the distributed key generation then authenticated channels would suffice for the communication between the user and the servers because the TOPRF evaluation protocol does not need secure channels. However, the standard realization of $\mathcal{F}_{\mathrm{DKG}}$ [30] would require secure channels between the servers.

secure in the password-only, i.e. PKI-free, model. However using TLS would offer a security benefit against the network adversary as a *hedge* against any server-spoofing attacks due to which the user might be tricked to run the PPSS reconstruction with the wrong set of servers. To see the benefit of running a PPSS protocol over TLS channels, denote the set of server identities which U inputs in the reconstruction as $\mathcal{SR}$. In the case of running PPSS reconstruction over TLS these can be equated with the public keys the user would use in the TLS sessions with the $t+1$ servers in the reconstruction. Consider the following two cases, and refer to the specification of the UC PPSS functionality $\mathcal{F}_{\text{PPSS}}$ of [19], which we include in the full version of this paper [20].

Case I: Every server S' in set $\mathcal{SR}$ is either incorrect (i.e. $S' \notin \mathcal{SI}$) and w.l.o.g. represents a malicious entity, or it is correct (i.e. $S' \in \mathcal{SI}$) but it is corrupted. In this case, according to $\mathcal{F}_{\text{PPSS}}$ specifications (see line 3b of the reconstruction phase), the adversary can perform one on-line password guess on such session. In other words, if the user runs reconstruction with incorrect/corrupt servers, the security is as in a (password-only) PAKE, i.e. the adversary can attempt to authenticate to such user using a password guess pw^*, and test if $\mathsf{pw}^* = \mathsf{pw}$.

Case II: There are *some* servers S' in set $\mathcal{SR}$ which are both correct (i.e. $S' \in \mathcal{SI}$) and uncorrupted. In this case, according to $\mathcal{F}_{\text{PPSS}}$ specifications (lines 3a and 3b of $\mathcal{F}_{\text{PPSS}}$), the adversary cannot learn anything from such instance, and can only either let it execute (line 3a) in which case U reconstructs the (correct!) secret K, or interfere with the protocol (line 3c) and make U output Fail.
In short, if PPSS reconstruction is executed over insecure channels then the man-in-the-middle adversary could make every reconstruction instance fall into Case I. By contrast, executing it over TLS forces the reconstruction instances to fall into Case II, unless the adversary tricks U to execute the reconstruction for the set of servers $\mathcal{SR}$ which includes *only* corrupt entities, in which case such reconstruction instance (and only such instance) falls back into Case I.

Note on sid/ssid Monikers. As we explain above, it is not essential for security of reconstruction that the user remembers the servers in the initialization set $\mathcal{SI}$. It might also be helpful to clarify the potential security implications of sid/ssid monikers which we assume are inputs in the initialization and the reconstruction phase. String *sid* (which stands for "session ID" in the AKE and UC terminology) in the context of a PPSS scheme can be equated with a "user ID", because it is a string which servers in $\mathcal{SI}$ will use to disambiguate between multiple PPSS instances which they can potentially service. It is therefore sensible to require that U remembers this user ID string *sid* in addition to her password pw. On the other hand, string *ssid* could be a nonce, or some application-determined identifier of a unique PPSS reconstruction session.

References

1. Russian hackers amass over a billion internet passwords. New York Times, 08 June 2014. http://goo.gl/aXzqj8

2. Abdalla, M., Chevassut, O., Fouque, P.-A., Pointcheval, D.: A simple threshold authenticated key exchange from short secrets. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 566–584. Springer, Heidelberg (2005). doi:10.1007/11593447_31
3. Abdalla, M., Cornejo, M., Nitulescu, A., Pointcheval, D.: Robust password-protected secret sharing. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9879, pp. 61–79. Springer, Cham (2016). doi:10.1007/978-3-319-45741-3_4
4. Bagherzandi, A., Jarecki, S., Saxena, N., Lu, Y.: Password-protected secret sharing. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 433–444. ACM (2011)
5. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M., et al.: The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. J. Cryptol. **16**(3), 185–215 (2003)
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000). doi:10.1007/3-540-45539-6_11
7. Blazy, O., Chevalier, C., Vergnaud, D.: Mitigating server breaches in password-based authentication: secure and efficient solutions. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 3–18. Springer, Cham (2016). doi:10.1007/978-3-319-29485-8_1
8. Boyko, V., MacKenzie, P., Patel, S.: Provably secure password-authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000). doi:10.1007/3-540-45539-6_12
9. Brainard, J., Juels, A., Kaliski, B., Szydlo, M.: Nightingale: a new two-server approach for authentication with short secrets. In: 12th USENIX Security Symposium, pp. 201–213. IEEE Computer Society (2003)
10. Camenisch, J., Lehmann, A., Lysyanskaya, A., Neven, G.: Memento: how to reconstruct your secrets from a single password in a hostile environment. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 256–275. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44381-1_15
11. Camenisch, J., Lehmann, A., Neven, G.: Optimal distributed password verification. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 182–194. ACM (2015)
12. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, 2001, pp. 136–145. IEEE (2001)
13. Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 353–373. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_20
14. Raimondo, M., Gennaro, R.: Provably secure threshold password-authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 507–523. Springer, Heidelberg (2003). doi:10.1007/3-540-39200-9_32
15. Ford, W., Kaliski, B.S.: Server-assisted generation of a strong secret from a password. In: Proeedings of the IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises 2000, (WET ICE 2000), pp. 176–180. IEEE (2000)
16. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. **20**(1), 51–83 (2007)

17. Jarecki, S., Jutla, C., Krawczyk, H., Rosu, M., Steiner, M.: Outsourced symmetric private information retrieval. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 875–888. ACM (2013)
18. Jarecki, S., Kiayias, A., Krawczyk, H.: Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 233–253. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45608-8_13
19. Jarecki, S., Kiayias, A., Krawczyk, H., Xu, J.: Highly-efficient and composable password-protected secret sharing (or: how to protect your bitcoin wallet online). In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 276–291. IEEE (2016)
20. Jarecki, S., Kiayias, A., Krawczyk, H., Xu, J.: Threshold oblivious PRF and minimal-cost password-protected secret sharing. Cryptology ePrint Archive (2017). http://eprint.iacr.org/2017/[TBD]
21. Jarecki, S., Krawczyk, H., Shirvanian, M., Saxena, N.: Device-enhanced password protocols with optimal online-offline protection. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, pp. 177–188. ACM (2016)
22. Jarecki, S., Liu, X.: Fast secure computation of set intersection. In: Garay, J.A., Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 418–435. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15317-4_26
23. Katz, J., MacKenzie, P., Taban, G., Gligor, V.: Two-server password-only authenticated key exchange. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 1–16. Springer, Heidelberg (2005). doi:10.1007/11496137_1
24. Kiefer, F., Manulis, M.: Distributed smooth projective hashing and its application to two-server password authenticated key exchange. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 2014. LNCS, vol. 8479, pp. 199–216. Springer, Cham (2014). doi:10.1007/978-3-319-07536-5_13
25. Kiefer, F., Manulis, M.: Universally composable two-server PAKE. In: Bishop, M., Nascimento, A. (eds.) ISC 2016. LNCS, vol. 9866, pp. 147–166. Springer, Cham (2016). doi:10.1007/978-3-319-45871-7_10
26. MacKenzie, P., Shrimpton, T., Jakobsson, M.: Threshold password-authenticated key exchange. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 385–400. Springer, Heidelberg (2002). doi:10.1007/3-540-45708-9_25
27. Naor, M., Pinkas, B., Reingold, O.: Distributed pseudo-random functions and KDCs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 327–346. Springer, Heidelberg (1999). doi:10.1007/3-540-48910-X_23
28. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_18
29. Szydlo, M., Kaliski, B.: Proofs for two-server password authentication. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 227–244. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30574-3_16
30. Wikström, D.: Universally composable DKG with linear number of exponentiations. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 263–277. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30598-9_19
31. Yi, X., Hao, F., Chen, L., Liu, J.K.: Practical threshold password-authenticated secret sharing protocol. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 347–365. Springer, Cham (2015). doi:10.1007/978-3-319-24174-6_18

Secure and Efficient Pairing at 256-Bit Security Level

Yutaro Kiyomura[1](✉), Akiko Inoue[2], Yuto Kawahara[1], Masaya Yasuda[3], Tsuyoshi Takagi[3], and Tetsutaro Kobayashi[1]

[1] NTT Secure Platform Laboratories, Musashino, Japan
{kiyomura.yutaro,kawahara.yuto,kobayashi.tetsutaro}@lab.ntt.co.jp
[2] NEC Central Research Laboratories, Kawasaki, Japan
a-inoue@cj.jp.nec.com
[3] Kyushu University, Fukuoka, Japan
{yasuda,takagi}@imi.kyushu-u.ac.jp

Abstract. At CRYPTO 2016, Kim and Barbulescu proposed an efficient number field sieve (NFS) algorithm for the discrete logarithm problem (DLP) in a finite field. The security of pairing-based cryptography (PBC) is based on the difficulty in solving the DLP. Hence, it has become necessary to revise the bitlength that the DLP is computationally infeasible against the efficient NFS algorithms. The timing of the main operations of PBC (i.e. pairing, scalar multiplication on the elliptic curves, and exponentiation on the finite field) generally becomes slower as the bitlength becomes longer, so it has become increasingly important to compute the main operations of PBC more efficiently. To choose a suitable pairing-friendly curve from among various pairing-friendly curves is one of the factors that affect the efficiency of computing the main operations of PBC. We should implement the main operations of PBC and compare the timing among some pairing-friendly curves in order to choose the suitable pairing-friendly curve precisely. In this paper, we focus on the five candidate pairing-friendly curves from the Barreto-Lynn-Scott (BLS) and Kachisa-Schaefer-Scott (KSS) families as the 256-bit secure pairing-friendly curves and show the following two results; (1) the revised bitlength that the DLP is computationally infeasible against the efficient NFS algorithms for each candidate pairing-friendly curve, (2) the suitable pairing-friendly curve by comparing the timing of the main operations of PBC among the candidate pairing-friendly curves using the revised bitlength.

1 Introduction

Many pairing-based cryptography (PBC) have been proposed, e.g., ID-based encryption [8,40], attribute-based encryption [41], and functional encryption [38]. A pairing on the elliptic curve is a non-degenerate bilinear map

A. Inoue—This work was done while the author was the student of Kyushu University.

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 59–79, 2017.
DOI: 10.1007/978-3-319-61204-1_4

$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$, where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_3$ are a group with order r respectively. The security of the pairing is based on the difficulty in solving the discrete logarithm problem (DLP) in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_3$. The group $\mathbb{G}_1$, $\mathbb{G}_2$ are a subgroup on the elliptic curve, and the DLP on an elliptic curve (ECDLP) in $\mathbb{G}_1$, $\mathbb{G}_2$ must be computationally infeasible against the rho algorithm [16,39]. Hence, we should choose r with a secure bitlength against the rho algorithm. The group $\mathbb{G}_3$ is a subgroup on finite field $\mathbb{F}_{p^k}$, where p is a prime and $k \geq 1$ is an embedding degree, and the DLP on a finite field (FFDLP) in $\mathbb{G}_3$ must be computationally infeasible against the number field sieve (NFS) algorithms. There are various NFS algorithms (e.g. Classical-NFS [25], tower NFS (TNFS) [10,45], and special NFS (SNFS) [10,26]). We should choose p and k with a secure bitlength of p^k against the NFS algorithms.

The recommended bitlength of p^k of the pairing was discussed in the 2013 report of ENISA [15, Table 3.6], where the pairing and RSA have the same recommended bitlength. This was in accordance with a general belief stated, for example, by Lenstra: "An RSA modulus n and a finite field $\mathbb{F}_{p^k}$ therefore offer about the same level of security if n and p^k are of the same order of magnitude" [32, Sect. 5.1]. The recommended bitlength of RSA was derived from the complexity of the NFS algorithm for integer factorization [33]. In other words, the bitlength of p^k of the pairing was estimated considering the complexity of this NFS algorithm.

At CRYPTO 2016, Kim and Barbulescu proposed an efficient NFS algorithm called the extended tower number field sieve (exTNFS) algorithm [28]. This NFS algorithm greatly impacted the security of the mainstream pairing such as optimal ate pairing [47]. The complexity of the exTNFS algorithm was reduced from that of previous NFS algorithms by using the trivial equation $\mathbb{F}_{p^k} = \mathbb{F}_{p^{\eta\kappa}}$, where $\gcd(\eta, \kappa) = 1$. Kim and Barbulescu concluded that the bitlength of the pairing should increase roughly twice [28]. Therefore, we should revise to estimate the secure bitlength of p^k against the exTNFS in detail. Note that Menezes *et al.* estimated the bitlength of p^k for the pairing considering the exTNFS algorithm at 128- and 192-bit security levels [37].

Generally, faster timing of the main operations of PBC (i.e. pairing, scalar multiplication on the elliptic curves, and exponentiation on the finite field) is preferred to implement the PBC. To choose a suitable pairing-friendly curve from among various pairing-friendly curves is one of the factors that affect the efficiency of computing the main operations of PBC. Among the studies conducted before the exTNFS algorithm was proposed, Scott [44] theoretically chose the suitable pairing-friendly curve at each security level based on the bitlength of r, p^k and ρ-value given in Freeman *et al.'s* taxonomy [16]. However, Aranha *et al.* [3] discussed the suitable pairing-friendly curve different from that chosen theoretically by comparing the timing of the pairing among several pairing-friendly curves at 192-bit security level. To choose a suitable pairing-friendly curve at a certain security level, it is important to not only choose theoretically but also compare the timing of the main operations of PBC.

Our Contributions. Our goal with this paper is to obtain a secure and efficient pairing at 256-bit security level. To achieve this, our contribution is to revise the estimation of the bitlength of p^k due to the efficient NFS algorithms (e.g. Special exTNFS, Special TNFS) and choose the suitable pairing-friendly curve for efficiently computing the main operations of PBC. We focus on the Barreto-Lynn-Scott (BLS) [11] and Kachisa-Schaefer-Scott (KSS) [29] families that have high embedding degree and are easy to implement the pairing. We specifically choose the following five candidate pairing-friendly curves at the 256-bit security level; the BLS-k with $k = 24, 42, 48$ and KSS-k with $k = 32, 36$. For these curves, we estimate the secure bitlength p^k in detail against the efficient NFS algorithms by comparing the upper bound of norms of these algorithms using the Kim and Barbulescu's estimation method [28]. Furthermore, based on the revised bitlength of p^k, we search for the specific parameter of each candidate pairing-friendly curve to implement the main operations of PBC, and then compare the timing of these operations among the five candidate pairing-friendly curves. Finally, we show the suitable pairing-friendly curve at 256-bit security level.

2 Overview of Pairing

2.1 Definition and Properties

Let p be a prime and E be an elliptic curve defined over the finite field $\mathbb{F}_p$. Let r be a prime with $\gcd(p, r) = 1$. An embedding degree k is the smallest positive integer with $r \mid p^k - 1$. Let $\mathbb{G}_1$, $\mathbb{G}_2$ be a subgroup on the elliptic curve with order r and $\mathbb{G}_3$ be a subgroup on the finite field $\mathbb{F}_{p^k}$ with order r. A pairing e is defined by $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_3$; $(P, Q) \longmapsto f_{r,P}(Q)^{(p^k-1)/r}$, where the rational function $f_{r,P}$ satisfies $\mathrm{div}(f_{r,P}) = r(P) - r(\mathcal{O})$ for the point at infinity $\mathcal{O}$. For $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ and $a \in \mathbb{Z}$, a pairing e has the following properties;

- bilinearity: $e(aP, Q) = e(P, aQ) = e(P, Q)^a$,
- non-degeneracy: for all $P \in \mathbb{G}_1$, $e(P, Q) = 1$ then $Q = \mathcal{O}$ and for all $Q \in \mathbb{G}_2$, $e(P, Q) = 1$ then $P = \mathcal{O}$,
- efficiently computable: $e(P, Q)$ can be efficiently computed.

2.2 Optimal Ate Pairing

The optimal ate pairing proposed by Vercauteren [47] is the most efficient method of computing the pairing e. There are many implementation results of the optimal ate pairing [3,9,13,36,44]. Let m be an integer such that $r \nmid m$. Let $\lambda = mr$ and write $\lambda = \sum_{i=0}^{\omega} \alpha_i p^i$ where $\omega = \lfloor \log_p \lambda \rfloor$. Let $E[r]$ be an r-torsion subgroup. Define $\mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi_p - [1]) = E(\mathbb{F}_p)[r]$, $\hat{\mathbb{G}}_2 = E[r] \cap \mathrm{Ker}(\pi_p - [p]) \subseteq E(\mathbb{F}_{p^k})[r]$ as the subgroup with r. Let E' be a twist of degree d of E with $\psi : E' \to E$ defined over $\mathbb{F}_{p^d}$, and define $\mathbb{G}_2 = \psi^{-1}(\hat{\mathbb{G}}_2)$. Note that d

depends on the pairing-friendly curve and is in {2, 3, 4, 6} [24]. An optimal ate pairing a_k is defined by

$$a_k : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_3,\ (P, Q) \longmapsto \left(\prod_{i=0}^{\omega} f_{\alpha_i,Q}^{p^i}(P) \cdot \prod_{i=0}^{\omega-1} \frac{\ell_{[\beta_{i+1}]Q,[\alpha_i p^i]Q}(P)}{v_{[\beta_i]Q}(P)} \right)^{\frac{p^k-1}{r}} \quad (1)$$

where $\beta_i = \sum_{j=i}^{\omega} \alpha_j p^j$, $\ell_{T,T'}$ is the line through T and T', and v_T is the vertical line through T, where T and T' are points on the elliptic curve.

3 Candidate Pairing-Friendly Curves at 256-Bit Security Level

In this section, we choose the five candidate pairing-friendly curves satisfying the security and efficiency from the BLS [11] and KSS [29] families to choose the suitable pairing-friendly curve at 256-bit security level. In this paper, we define len(x) as the bitlength of x.

3.1 How to Choose Candidate Pairing-Friendly Curves

We show the security against the ECDLP and FFDLP and the efficiency for implementation of the main operations of PBC. An embedding degree k is determined by the chosen pairing-friendly curve, and the primes r and p are represented by the polynomial of a positive integer x.

Security. The parameters r, p, and k should satisfy the complexity of solving the DLP in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ to achieve the $\mathcal{K}$-bit security level. The definition of ECDLP in $\mathbb{G}_1$ and $\mathbb{G}_2$ is as follows. Given points $G, Y \in \mathbb{G}_1$ (or $\mathbb{G}_2$), find $x \in \mathbb{Z}$ such that $Y = xG$. An efficient algorithm for solving the ECDLP is the rho algorithm [16,39], which has the complexity of $\mathcal{O}(\sqrt{r})$. Therefore, we should choose the bitlength of r with $\text{len}(r) \geq 2\mathcal{K}$. The definition of the FFDLP in $\mathbb{G}_3$ is as follows. Given points $g, y \in \mathbb{G}_3$, find $x \in \mathbb{Z}$ such that $y = g^x$. An efficient algorithm for solving the FFDLP are the STNFS [10,26] and SexTNFS [28] algorithms. We give the bitlength of p^k which the FFDLP is computationally infeasible against these NFS algorithms in Sect. 5.

Efficiency. To efficiently compute the main operations of PBC (i.e. pairing, scalar multiplication in $\mathbb{G}_1$ and $\mathbb{G}_2$, and exponentiation in $\mathbb{G}_3$) with the above security, we consider the following conditions as affecting the efficiency of these operations.

- $\text{len}(r)$ and $\text{len}(p^k)$ are as small as possible.
- The ρ-value is approximately 1 ($\rho = \log p / \log r$).
- Parameter x in polynomials (e.g. $p(x)$, $r(x)$) has a low Hamming weight.
- The embedding degree k has the form $k = 2^i \cdot 3^j$ ($i \in \mathbb{Z}_{\geq 1}, j \in \mathbb{Z}_{\geq 0}$).
- The twist of degree d is 6 ($d = 6$ is maximum of degree).

These conditions are theoretically efficient ones, then the effect of each condition is uncertain in the implementation.

Table 1. Parameters for the five candidate pairing-friendly curves

BLS-24 [16, Construction 6.6]	$k = 24,\ \rho = 1.250,\ \deg(p(x)) = 10,\ \varphi(k) = 8,$ $p(x) = (x-1)^2(x^8 - x^4 + 1)/3 + x,\ \ r(x) = x^8 - x^4 + 1,\ \ t(x) = x + 1$
KSS-32 [29, Example 4.4]	$k = 32,\ \rho = 1.063,\ \deg(p(x)) = 18,\ \varphi(k) = 16,$ $p(x) = (x^{18} - 6x^{17} + 13x^{16} + 57120x^{10} - 344632x^9 + 742560x^8$ $+815730721x^2 - 4948305594x + 10604499373)/2970292,$ $r(x) = (x^{16} + 57120x^8 + 815730721)/(2 \cdot 13^8 \cdot 239^2),$ $t(x) = (-2x^9 - 56403x + 3107)/3107$
KSS-36 [29, Example 4.5]	$k = 36,\ \rho = 1.167,\ \deg(p(x)) = 14,\ \varphi(k) = 12,$ $p(x) = (x^{14} - 4x^{13} + 7x^{12} + 683x^8 - 2510x^7 + 4781x^6 + 117649x^2$ $-386569x + 823543)/28749,$ $r(x) = (x^{12} + 683x^6 + 117649)/(7^6 \cdot 37^2),\ \ t(x) = (2x^7 + 757x + 259)/259$
BLS-42 [16, Construction 6.6]	$k = 42,\ \rho = 1.333,\ \deg(p(x)) = 16,\ \varphi(k) = 12,$ $p(x) = (x-1)^2(x^{14} - x^7 + 1)/3 + x,$ $r(x) = x^{12} + x^{11} - x^9 - x^8 + x^6 - x^4 - x^3 + x + 1,\ \ t(x) = x + 1$
BLS-48 [16, Construction 6.6]	$k = 48,\ \rho = 1.125,\ \deg(p(x)) = 18,\ \varphi(k) = 16,$ $p(x) = (x-1)^2(x^{16} - x^8 + 1)/3 + x,\ \ r(x) = x^{16} - x^8 + 1,\ \ t(x) = x + 1$

* deg(): degree of polynomial, $\varphi()$: Euler function

3.2 Selection of Candidate Pairing-Friendly Curves

In this subsection, we decide the candidate pairing-friendly curves from the BLS [11] and KSS [29] families to choose the suitable pairing-friendly curve at 256-bit security level based on Sect. 3.1. We focus on the BLS and KSS families that have high embedding degree and can be easy to construct the pairing. We specifically choose the following five candidate pairing-friendly curves at the 256-bit security level; the BLS-k with $k = 24, 42, 48$ and the KSS-k with $k = 32, 36$. In the case of the BLS-k, the small $\mathrm{len}(r)$ and $\mathrm{len}(p^k)$ in the BLS-42 and BLS-48 can be choose because these curve have high embedding degree k, and the implementation results in the BLS-24 exists [9,44]. In the case of the KSS-k, the KSS-36 has the small $\mathrm{len}(r)$, the KSS-32 has small ρ-value and simple tower construction for $\mathbb{F}_{p^k}$ since $k = 32 = 2^5$.

The detail of the five candidate pairing-friendly curves are as follows; the curves with $6 \mid k$ are defined by $E/\mathbb{F}_p : y^2 = x^3 + b$, and has the complex multiplication discriminant $D = 3$ and $d = 6$, the curves with $4 \mid k$ is defined by $E/\mathbb{F}_p : y^2 = x^3 + ax$, and has $D = 1$ and $d = 4$. Table 1 shows the parameters $p(x)$, $r(x)$, $t(x)$, k, ρ-value, $\deg(p(x))$, and Euler function $\varphi(k)$ for each curves. The parameters $n(x)$ and $f(x)$ satisfy $n(x) = p(x) + 1 - t(x)$ and $4p(x) - t(x)^2 = Df(x)^2$, respectively.

4 Overview of Number Field Sieve and Its Variants

In this section, we give an overview of the NFS algorithm and its variants to revise the bitlength of the five candidate pairing-friendly curves introduced in Sect. 3.2.

The FFDLP is classified into three cases by size of p: small, medium, or large. In medium and large cases, the NFS algorithms is the most efficient algorithm for solving the FFDLP. To accurately classify p, let $p = L_{p^k}(l_p, c_p)$, where $L_{p^k}(l_p, c_p) = \exp((c_p + o(1))(\log p^k)^{l_p}(\log\log p^k)^{1-l_p})$. $o(1)$ becomes 0 when

$p^k \to \infty$. The prime p is called medium if $1/3 < l_p < 2/3$, large if $2/3 < l_p < 1$, boundary if $l_p = 2/3$.

Note that the above L_{p^k}-notation is just an asymptotic value. If we fix the value of p^k, the L_{p^k}-notation has a constant value c such that $c \times \exp((c_p + o(1))(\log p^k)^{l_p}(\log\log p^k)^{1-l_p})$ and $o(1) \neq 0$. Therefore, when we substitute the concrete value for p^k in L_{p^k}-notation, it is important to evaluate c and $o(1)$.

The NFS algorithms for solving the FFDLP are classified into three types: Classical-NFS, TNFS, and exTNFS, according to their mathematical constructions. The Classical-NFS algorithm was proposed in 2006, and the complexities are $L_{p^k}(1/3, (128/9)^{1/3})$ and $L_{p^k}(1/3, (64/9)^{1/3})$ in the medium and large cases, respectively. The TNFS algorithm was proposed in 1999 and later applied to the large case in 2015, where the complexity of the TNFS algorithm is also $L_{p^k}(1/3, (64/9)^{1/3})$ in the large case. Finally, the exTNFS algorithm proposed in 2015 is the generalization of combining the Classical-NFS and TNFS algorithms, and its complexities in medium and large cases are $L_{p^k}(1/3, (64/9)^{1/3})$.

4.1 Extended TNFS and Special-NFS Algorithms

In this subsection, we explain the exTNFS algorithm [28], which is effective for solving the FFDLP. We then give an overview of the Special-NFS algorithm. The NFS algorithms (not specified for exTNFS) are divided into the following four steps: 1. polynomial selection, 2. relation collection, 3. linear algebra, and 4. individual logarithm.

exTNFS Algorithm. We can use the exTNFS algorithm when the extension degree k is composite. Let $k = \eta\kappa$. We select an irreducible polynomial $h(t) \in \mathbb{Z}[t]$ over $\mathbb{Q}$ and $\mathbb{F}_p$ whose degree is η. We construct $\mathbb{Q}(\iota) = \mathbb{Q}[t]/h(t)$ and put $R = \mathbb{Z}[t]/h(t) \subset \mathbb{Q}(\iota)$.

Note that the Classical-NFS algorithm [25] is the case in which $R = \mathbb{Z}$ in the exTNFS algorithm, and the TNFS algorithm [45] is the case in which $\deg h = n$ in the exTNFS algorithm.

Polynomial Selection. We select polynomials f_1 and $f_2 \in R[X]$ that satisfy the condition that $f_1 \mod p$ and $f_2 \mod p$ have a common factor $\varphi(X)$ of degree κ, which is irreducible over $\mathbb{F}_{p^\eta} = R/pR$. In this section, $i \in \{1, 2\}$. Let K_i be the number fields defined by f_i above the fraction field of R and $\mathcal{O}_i$ be the integer ring of K_i. We denote the roots of f_i in $\mathbb{C}$ by θ_i and the degree of f_i by d_i. We then obtain two maps from $R[X]$ to $(R/pR)[X]/\varphi(X) \cong \mathbb{F}_{p^k}$.

Relation Collection. We select smoothness bound $B \in \mathbb{N}$ and define factor base $\mathcal{F}_i$ as follows: $\mathcal{F}_i = \{(\mathfrak{q}, \theta_i - \gamma) : \mathfrak{q} : \text{prime in } \mathbb{Q}(\iota) \text{ lying over a prime } p \leq B, f_i(\gamma) \equiv 0 \mod \mathfrak{q}\}$. We then obtain $a - bX \in R[X]$ by selecting $(a(t), b(t)) \in R^2$. The coefficients of $a(t)$ and $b(t)$ are bounded by A. Let $E = A^\eta$ be the sieve parameter. The norm of $a - b\theta_i$ in K_i is expressed as follows.

$$\mathcal{N}_{K_i/\mathbb{Q}}(a - b\theta_i) = \left|\mathrm{Res}\left(h(t), \sum_{j\in[0,d_i]} f_{i,j} a(t)^j b(t)^{d_i-j}\right)\right|,$$

where $f_{i,j}$ is the coefficient of polynomial $f_i = \sum_{j=0}^{d_i} f_{i,j} X^j$. When $\mathcal{N}_{K_1/\mathbb{Q}}(a - b\theta_1)$ and $\mathcal{N}_{K_2/\mathbb{Q}}(a - b\theta_2)$ are B-smooth, the $(a(t), b(t))$ pair is called a double smooth pair (an integer is B-smooth if the largest prime factor is less than B). When $(a(t), b(t))$ is a double smooth pair, $(a-b\theta_1)$ and $(a-b\theta_2)$ can be factored into the prime ideal in $\mathcal{O}_1$ and $\mathcal{O}_2$ using only the elements of $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively. Therefore, we obtain the following notation: $(a - b\theta_i) = \prod_{\mathfrak{p}\in\mathcal{F}_i} \mathfrak{p}^{\mu_\mathfrak{p}}$ and the following relation up to units.

$$\begin{aligned}\phi_1((a - b\theta_1)) = \phi_2((a - b\theta_2)) \text{ in } \mathbb{F}_{p^k} \iff \quad & \phi_1(\prod_{\mathfrak{p}\in\mathcal{F}_1} \mathfrak{p}^{\mu_\mathfrak{p}}) = \phi_2(\prod_{\mathfrak{q}\in\mathcal{F}_2} \mathfrak{q}^{\mu_\mathfrak{q}}) \\ \iff \quad & \prod_{\mathfrak{p}\in\mathcal{F}_1} \phi_1(\mathfrak{p})^{\mu_\mathfrak{p}} = \prod_{\mathfrak{q}\in\mathcal{F}_2} \phi_2(\mathfrak{q})^{\mu_\mathfrak{q}}\end{aligned}$$

Thus, this leads to

$$\begin{aligned}&\sum_{\mathfrak{p}\in\mathcal{F}_1} \mu_\mathfrak{p} \log \phi_1(\mathfrak{p}) + \sum_j \lambda_{1,j} \log \varLambda_{1,j} \\ &= \sum_{\mathfrak{q}\in\mathcal{F}_2} \mu_\mathfrak{q} \log \phi_2(\mathfrak{q}) + \sum_j \lambda_{2,j} \log \varLambda_{2,j} \mod p^k - 1,\end{aligned} \tag{2}$$

where $\log \phi_1(\mathfrak{p})$, $\log \phi_2(\mathfrak{q})$, $\log \varLambda_{1,j}$ and $\log \varLambda_{2,j}$ are the unknowns called virtual logarithms [46], and $\lambda_{1,j}$ and $\lambda_{2,j}$ are computable values called character maps to distinguish the difference in units. Let N_λ be the number of character maps. When we collect more than $|\mathcal{F}_1| + |\mathcal{F}_2| + N_\lambda$ double smooth pairs $(a(t), b(t))$, we obtain the relations of (2).

In this section, we collect double smooth pairs (a, b), but it is possible to collect double smooth tupples $(a_1, a_2, \cdots, a_\tau)$. We call parameter τ a sieve dimension.

Linear Algebra. In collecting adequate relations in the previous step, we can construct and solve the simultaneous congruence. We obtain the values of the virtual logarithms $\log \phi_1(\mathfrak{p})$, $\log \phi_2(\mathfrak{q})$, $\log \varLambda_{1,j}$, and $\log \varLambda_{2,j}$.

Individual Logarithm. Finally, we compute the target logarithm x from the values of the virtual logarithms.

Special-NFS Algorithms. We collectively call three NFS algorithms (exTNFS, Classical-NFS, and TNFS) as the General NFS (GNFS) algorithms. The GNFS algorithms can be applied for special polynomial selection when p has a special form. The special cases of the GNFS algorithms are called Special-NFS (SNFS) algorithms. We consider the SNFS algorithms for solving the FFDLP to estimate the security of the pairing where p has a special form.

4.2 Larger Norm Implies Higher Complexity

In this section, we give an overview on the complexity of the NFS algorithms. The main steps to evaluate this complexity are as follows.

- We evaluate the upper bound of norms and probabilities in which the norms are B-smooth.

The smaller the upper bound of norms is, the higher the probability the norms are B-smooth. Therefore, we have to select polynomials so that the upper bounds of norms become small.

- We set the parameters appropriately so that we can collect adequate double smooth pairs from the sieve region.

The sieve region is the region to collect relations. When the sieve degree is τ, the sieve region is E^{τ}. We set the appropriate parameters to satisfy the inequality that $E^{\tau}\times$ (the probability of B-smooth of norm's upper bound) $\leq$ (the number of double smooth pairs we collect) $= B^{1+o(1)}$.

- Relation collection and linear algebra have the same complexities.

The whole complexity of NFS algorithms is the sum of the complexities of following two steps: relation collection and linear algebra. In the exTNFS algorithm, the complexity of relation collection is $O(E^2)$. We need to evaluate sizes of parameters because of the trade-off between relation collection and linear algebra.

When the norm is small, the probability that norms are B-smooth becomes high. We can obtain relations with a few trials. The complexity of relation collection becomes small, and the whole complexity becomes small. That is, the decrease in norms implies the reduction in the security of cryptography, which is based on the difficulty of the FFDLP. Therefore, we can estimate bitlengths by comparing the sizes of norms.

4.3 Comparing Norms of NFS Algorithms by Using Kim and Barbulescu's Estimation Method

We refer to the method of comparing norms in [28] to estimate and compare the norms of various NFS algorithms. The norms of each GNFS algorithm are listed in Table 2, and the norms of each SNFS algorithm are listed in Table 3 (part of Table 2 is omitted).

In Tables 2 and 3, E_G is the sieve parameter of the GNFS algorithms and E_S is the sieve parameter of the SNFS algorithms. In addition, d is the degree of polynomial selected in the step of polynomial selection. Note that d in SNFS algorithm is equal to the degree of p. The τ is the sieve dimension, and others are parameters used in each NFS algorithm. Sieve parameter E depends on the implementations. Kim and Barbulescu [28] used the formula

$$E_G = c_G L_{p^k}\left(\frac{1}{3}, \left(\frac{8}{9}\right)^{1/3}\right), E_S = c_S L_{p^k}\left(\frac{1}{3}, \left(\frac{4}{9}\right)^{1/3}\right).$$

Table 2. Norm sizes of GNFS algorithms

Algorithm	Norm product
NFS-JLSV$_1$	$E_G^{\frac{4k}{\tau}} (p^k)^{\frac{\tau-1}{k}}$
TNFS	$E_G^{\frac{2(d+1)}{\tau}} (p^k)^{\frac{2(\tau-1)}{d+1}}$
exTNFS-Conj	$E_G^{\frac{6\kappa}{\tau}} (p^k)^{\frac{\tau-1}{2\kappa}}$

Table 3. Norm sizes of SNFS algorithms

Algorithm	Norm product
STNFS	$E_S^{\frac{2(d+1)}{\tau}} (p^k)^{\frac{\tau-1}{d}}$
SNFS-JP	$E_S^{\frac{2k(d+1)}{\tau}} (p^k)^{\frac{\tau-1}{kd}}$
SexTNFS	$E_S^{\frac{2\kappa(d+1)}{\tau}} (p^k)^{\frac{\tau-1}{\kappa d}}$

They determined $\log_2 c_G \approx -4.30$ using the results of three implementations [6,7,14]. Similarly, they determined $\log_2 c_S \approx -4.27$ using the results of an implementation [1]. After the values of E_G and E_S are determined, other parameters must be determined. The parameters, except τ, are computed using the theoretical optimal values. Then τ is determined as the best value in their bitsize of the norm.

5 Revise the Bitlength for Candidate Pairing-Friendly Curves

In this section, we revise to estimate the bitlengths for the five pairing-friendly curves (i.e. the BLS-k with $k = 24, 42, 48$ and KSS-k with $k = 32, 36$) at 256-bit security level by using the norms of NFS algorithms in the previous section.

5.1 Revised Estimation of Bitlength for BLS-48

In this subsection, we revised to estimate the bitlength for the BLS-48. We compare the norms of NFS algorithms based on the constants c_G and c_S and estimate the bitlength based on the initial norm of the GNFS algorithms at the 256-bit security level. The estimations for other pairing-friendly curves (i.e. the BLS-24, KSS-32, KSS-36 and BLS-42) are described in the Appendix A.

Determining Constants c_G *and* c_S. Before plotting norms, the constant values of c_G and c_S must be evaluated. As previously mentioned, c_G and c_S are evaluated from the implementation results. We discuss these values by adding new implementation results. In Kim and Barbulescu's study [28], $\log_2 c_G \approx -4.30$ and $\log_2 c_S \approx -4.27$; however, we evaluate c_G and c_S by adding to new results. First, we evaluate c_G using the result from Kleinjung [31] who solved the DLP in $\mathbb{F}_p$. We extrapolate from the pair ($\log_2 p^k = 768$, $\log_2 E_G \approx 35$) Kleinjung used [31] and obtain $\log_2 c_G \approx -3.26$. The sieve parameter E_G using Kleinjung's result is larger than that Kim and Barbulescu evaluated. Because E_G by Kim and Barbulescu [28] is evaluated more strictly, we plot the norms of the GNFS algorithms using $\log_2 c_G \approx -4.30$ they used. Next, we evaluate c_S using the results by Fried *et al.* [17] and Guillevic *et al.* [22]. We extrapolate from the pair ($\log_2 p^k = 1024$, $\log_2 E_S \approx 31$) used by Fried *et al.* [17] and obtain $\log_2 c_S \approx -3.43$. We also extrapolate from the pair ($\log_2 p^k = 510$, $\log_2 E_S \approx 26$) used by Guillevic *et al.* [22] and obtain $\log_2 c_S \approx 0.67$. The sieve parameters E_S using the results from Fried *et al.* and Guillevic *et al.* are larger than those Kim and Barbulescu evaluated. As with E_G, because E_S from Kim and Barbulescu [28] is evaluated more strictly, we plot the norms of the SNFS algorithms using $\log_2 c_S \approx -4.27$ they used.

Initial Norm of General NFS Algorithms at 256-Bit Security Level. We define the initial norm as the norm of the GNFS algorithm for integer factorization, which corresponds to the bitlength at 256-bit security level. Let N be a composite number. The norm of this GNFS algorithm is $E_G^{d+1} N^{2/d+1}$, where d is the degree of the polynomial selected in the step of polynomial selection. The recommended bitlength of RSA at 256-bit security level is 15360-bit [5]. When N is 15360-bit, the optimal value of d is 15. We substitute the value of c_G, which we evaluated in the previous section, for the norm, and the initial norm is about 4006-bit.

Fastest Variant of NFS Algorithms for BLS-48. We concretely estimate the bitlengths of the PBC. We give details of the BLS-48, and the other curves are mentioned in the Appendix A. We fix the extension degree to $k = 48$ and examine the fastest NFS algorithm that solves the DLP in $\mathbb{F}_{p^{48}}$, where p is expressed by the BLS-48.

First, we plot the norms of three GNFS algorithms (i.e. NFS-JLSV$_1$, TNFS, and exTNFS-Conj) in Fig. 1. The E_G is as follows;

$$E_G = 2^{-4.3} L_{p^k}\left(\frac{1}{3}, \left(\frac{8}{9}\right)^{\frac{1}{3}}\right).$$

- NFS-JLSV$_1$. The norm of the NFS-JLSV$_1$ algorithm is expressed as ${E_G}^{\frac{4k}{\tau}} (p^k)^{\frac{\tau-1}{k}}$. The optimal value of τ is 9 when the norm is ${E_G}^{\frac{192}{\tau}} (p^k)^{\frac{\tau-1}{48}}$.
- TNFS. The norm of the TNFS algorithm is expressed as ${E_G}^{\frac{2(d+1)}{\tau}} (p^k)^{\frac{2(\tau-1)}{d+1}}$. The d is 15 using the formula $d = \sqrt[3]{3}\left(\log p^k / \log\log p^k\right)^{1/3}$. The optimal value of τ is 2 when the norm is ${E_G}^{\frac{32}{\tau}} (p^k)^{\frac{2(\tau-1)}{16}}$
- exTNFS-Conj. The norm of the exTNFS-Conj algorithm is expressed as ${E_G}^{\frac{6\kappa}{\tau}} (p^k)^{\frac{\tau-1}{2\kappa}}$. Kim and Barbulescu [28] used the case of $\gcd(\eta, \kappa) = 1$. However, Kim and Jeong proposed an algorithm allowing the choosing of η and κ freely from the co-primality condition, and their algorithm has the same complexities as when η and κ are co-prime [30]. Therefore, we use all cases of (η, κ). When $n = 48$, we can consider the cases of (η, κ) = (2, 24), (3, 16), (4, 12), (6, 8), (8, 6), (12, 4), (16, 3), (24, 2). When (η, κ) = (2, 24), (3, 16), (4, 12), (6, 8), (8, 6), (12, 4), (16, 3), (24, 2), the optimal value of τ is 7, 5, 4, 3, 2, 2, 2, 2, respectively.

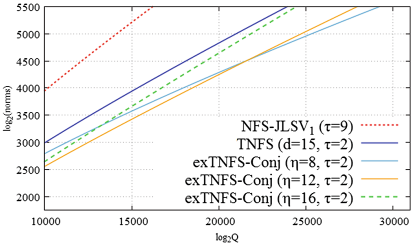


Fig. 1. Norms of GNFS algorithms in $\mathbb{F}_{p^{48}}$

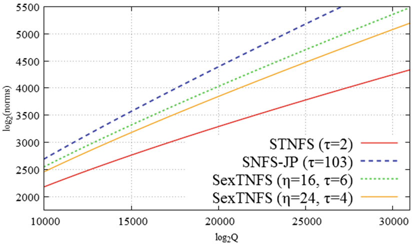


Fig. 2. Norms of SNFS algorithms in $\mathbb{F}_{p^{48}}$

Next, we plot the norms of three SNFS algorithms (i.e. STNFS, SNFS-JP, and SexTNFS) in Fig. 2. In the BLS-48, $\deg(p(x)) = 18$. The E_S is as follows:

$$E_S = 2^{-4.3} L_{p^k}\left(\frac{1}{3}, \left(\frac{4}{9}\right)^{\frac{1}{3}}\right)$$

- STNFS. The norm of the STNFS algorithm is expressed as $E_G^{\frac{2(d+1)}{\tau}} (p^k)^{\frac{\tau-1}{d}}$. The optimal value of τ is 2 when the norm is $E_G^{\frac{38}{\tau}} (p^k)^{\frac{\tau-1}{18}}$.
- SNFS-JP. The norm of the SNFS-JP algorithm is expressed as $E_G^{\frac{2k(d+1)}{\tau}} (p^k)^{\frac{\tau-1}{kd}}$. The optimal value of τ is 103 when the norm is $E_G^{\frac{1824}{\tau}} (p^k)^{\frac{\tau-1}{864}}$.
- SexTNFS. The norm of the SexTNFS algorithm is expressed as $E_G^{\frac{2\kappa(d+1)}{\tau}} (p^k)^{\frac{\tau-1}{\kappa d}}$. We also use the all case of (η, κ). When (η, κ) = (2, 24), (3, 16), (4, 12), (6, 8), (8, 6), (12, 4), (16, 3), (24, 2), the optimal value of τ is 51, 34, 26, 17, 13, 9, 6, 4, respectively.

In Figs. 1 and 2, the STFNS algorithm has the smallest norm. When the norm is the initial norm of 4006-bit, the bitlength of the STNFS algorithm is 27410-bit. Therefore, we can estimate that the bitlength at 256-bit security level is 27410-bit.

5.2 Revised Bitlength at 256-Bit Security Level

We revise to estimate the bitlength for the five candidate pairing-friendly curves (i.e. the BLS-k with $k = 24, 42, 48$ and KSS-k with $k = 32, 36$) at 256-bit security level based on the Kim and Barbulescu's method [28]. The results are listed in Table 4. According to ENISA [15, Table 3.6], the bitlength of the pairing requires more than 15360-bit to achieve the 256-bit security level [16]. However, our revised estimation shows that it is necessary to increase the bitlength by more than 10000-bit to achieve the 256-bit security level. In other word, it is necessary to approximately multiply the bitlength by 1.7 times to achieve the 256-bit security level.

Table 4. Revised bitlength at 256-bit security level

	BLS-24	KSS-32	KSS-36	BLS-42	BLS-48
len(p^k)	25,990	27,410	28,280	28,150	27,410

6 Comparison of Timing Among Candidate Pairing-Friendly Curves

In this section, we measure and compare the timing of the main operations of PBC among the five candidate pairing-friendly curves using the revised bitlength to show a suitable pairing-friendly curve at 256-bit security level. Our implementation uses the efficient algorithms for computing the main operations of PBC.

6.1 Specific Parameter for Implementation

The specific parameter x_0 for each candidate pairing-friendly curve is required to decide the parameters of each curve in Table 1 and implement the main operations of PBC. The x_0 for the BLS-24 showed in [13] satisfies the revised bitlength of p^k in Table 4, but there are no documents showed the parameter satisfying the revised bitlength of p^k in Table 4 for other curves. Therefore, we should search for x_0 for each KSS-32, KSS-36, BLS-42, and BLS-48.

To efficiently compute the pairing, we search for the specific parameter x_0 with a low Hamming weight and $\text{len}(r) \geq 512$ and $\text{len}(p^k)$ always more than the bitlength in Table 4. Table 6 shows x_0, the bitlength of parameters, and Hamming weight of x_0 for the five candidate pairing-friendly curves. Note that the $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_3$ are better to satisfy subgroup security [12] in order to resist against small-subgroup attacks as an optional security requirement. The information of implementation (i.e. the tower construction, elliptic curve E, twist E' of E, and $\ell_{T,T'}(P)$) are showed in Table 5.

6.2 Our Implemented Algorithms

In this subsection, we give an overview of the implemented efficient algorithms for computing the main operations of PBC. We implement the base field $\mathbb{F}_p$ arithmetic by using the GMP library [18]. Additionally, in the arithmetic of the tower field, we use the lazy reduction technique [42] which can reduce the number of the modulo operations of $\mathbb{F}_p$.

Pairing. The formulas of optimal ate pairing for each candidate pairing-friendly curve are given in Table 7. Note that these formulas can be produced from Eq. (1) and [47, Eq. (9)]. There are two steps involved in computing the optimal ate pairing; the miller loop (ML) $f' = f_{x,Q}(P) \cdot g$, where g is the part other than $f_{x,Q}(P)$ in Table 7, and final exponentiation (FE) $f'^{(p^k-1)/r}$.

Table 5. Information of implementation for the five candidate pairing-friendly curves

BLS-24	Fields	$\mathbb{F}_p \xrightarrow{u^2+1} \mathbb{F}_{p^2} \xrightarrow{v^2+u+1} \mathbb{F}_{p^4} \xrightarrow{w^3+v} \mathbb{F}_{p^{12}} \xrightarrow{z^2+w} \mathbb{F}_{p^{24}}$
	$E,\, E'$	$E/\mathbb{F}_p : y^2 = x^3+1,\ E'/\mathbb{F}_{p^4} : y^2 = x^3 - 1/v$
	$\ell_{T,T'}(P)$	$[\overbrace{\underbrace{y_P}_{1} : \underbrace{0}_{w} : \underbrace{0}_{w^2}}^{1} \mid \overbrace{\underbrace{(-\lambda \cdot x_P)u}_{1} : \underbrace{cu}_{w} : \underbrace{0}_{w^2}}^{z}]$
KSS-32	Fields	$\mathbb{F}_p \xrightarrow{u^2+2} \mathbb{F}_{p^2} \xrightarrow{v^2-u} \mathbb{F}_{p^4} \xrightarrow{w^2-v} \mathbb{F}_{p^8} \xrightarrow{z^2-w} \mathbb{F}_{p^{16}} \xrightarrow{s^2-z} \mathbb{F}_{p^{32}}$
	$E,\, E'$	$E/\mathbb{F}_p : y^2 = x^3+2x,\ E'/\mathbb{F}_{p^8} : y^2 = x^3 + 2x/w$
	$\ell_{T,T'}(P)$	$[\overbrace{\underbrace{y_P}_{1} : \underbrace{0}_{z}}^{1} \mid \overbrace{\underbrace{-\lambda \cdot x_P}_{1} : \underbrace{c}_{z}}^{s}]$
KSS-36	Fields	$\mathbb{F}_p \xrightarrow{u^2+1} \mathbb{F}_{p^2} \xrightarrow{v^3+u+1} \mathbb{F}_{p^6} \xrightarrow{w^3+v} \mathbb{F}_{p^{18}} \xrightarrow{z^2+w} \mathbb{F}_{p^{36}}$
	$E,\, E'$	$E/\mathbb{F}_p : y^2 = x^3+2,\ E'/\mathbb{F}_{p^6} : y^2 = x^3 - 2/v$
	$\ell_{T,T'}(P)$	$[\overbrace{\underbrace{y_P}_{1} : \underbrace{0}_{w} : \underbrace{0}_{w^2}}^{1} \mid \overbrace{\underbrace{(-\lambda \cdot x_P)u}_{1} : \underbrace{cu}_{w} : \underbrace{0}_{w^2}}^{z}]$
BLS-42	Fields	$\mathbb{F}_p \xrightarrow{u^7+2} \mathbb{F}_{p^7} \xrightarrow{v^3+u-1} \mathbb{F}_{p^{21}} \xrightarrow{w^2-v} \mathbb{F}_{p^{42}}$
	$E,\, E'$	$E/\mathbb{F}_p : y^2 = x^3+1,\ E'/\mathbb{F}_{p^6} : y^2 = x^3 + 1/(1-u)$
	$\ell_{T,T'}(P)$	$[\overbrace{\underbrace{y_P}_{1} : \underbrace{0}_{v} : \underbrace{0}_{v^2}}^{1} \mid \overbrace{\underbrace{-\lambda \cdot x_P}_{1} : \underbrace{c}_{v} : \underbrace{0}_{v^2}}^{w}]$
BLS-48	Fields	$\mathbb{F}_p \xrightarrow{u^2+1} \mathbb{F}_{p^2} \xrightarrow{v^2+u+1} \mathbb{F}_{p^4} \xrightarrow{w^2+v} \mathbb{F}_{p^8} \xrightarrow{z^3+w} \mathbb{F}_{p^{24}} \xrightarrow{s^2+z} \mathbb{F}_{p^{48}}$
	$E,\, E'$	$E/\mathbb{F}_p : y^2 = x^3+1,\ E'/\mathbb{F}_{p^8} : y^2 = x^3 - 1/w$
	$\ell_{T,T'}(P)$	$[\overbrace{\underbrace{y_P}_{1} : \underbrace{0}_{z} : \underbrace{0}_{z^2}}^{1} \mid \overbrace{\underbrace{(-\lambda \cdot x_P)u}_{1} : \underbrace{cu}_{z} : \underbrace{0}_{z^2}}^{s}]$

In the ML, the rational function $f_{x,Q}(P)$ can be computed using Miller's algorithm [34]. The computational cost of the ML is affected by bitlength x_0 and the Hamming weight of x_0. We can reduce the computational cost of the multiplication on $\mathbb{F}_{p^k}$ by using the sparse multiplication technique [35]. We use the affine pairing [2] since the computation of the inversion in $\mathbb{G}_2$ is fast.

In the FE, the equation $f'^{(p^k-1)/r}$ can be broken down into three components by using cyclotomic polynomial Φ_k as follows [43].

$$(p^k-1)/r = \underbrace{[(p^{k/2}-1)] \cdot [(p^{k/2}+1)/\Phi_k(p)]}_{\text{easy part}} \cdot \underbrace{[\Phi_k(p)/r]}_{\text{hard part}}.$$

The computation of the easy part $m = f'^{(p^{k/2}-1)\cdot((p^{k/2}+1)/\Phi_k(p))}$ requires one conjugation, one inversion, some Frobenius operations and some multiplications on $\mathbb{F}_{p^k}$, so the computational cost of the easy part hardly affects that of the whole FE. The hard part can be computed by using the base p representation of $\Phi_k\left(p\right)/r$ as

Table 6. x_0, bitlength of parameters and Hamming weight of x_0

	x_0	len(x_0)	HW(x_0)	len(p^k)	len($p^{k/d}$)	len(p)	len(r)
BLS-24	$-1+2^{65}-2^{75}+2^{109}$	109	4	26122	4354	1089	872
KSS-32	$-1-2^{2}-2^{12}+2^{14}+2^{18}-2^{30}+2^{49}$	49	7	27536	6884	861	738
KSS-36	$2^{5}+2^{34}+2^{40}+2^{45}-2^{58}$	58	5	28699	4784	798	669
BLS-42	$-1+2^{2}-2^{8}+2^{43}$	43	4	28830	4805	687	516
BLS-48	$-1+2^{7}-2^{10}-2^{30}-2^{32}$	33	5	27851	4642	581	518

* HW() : Hamming weight

Table 7. Formulas for computing optimal ate pairing $a_k(P,Q)$

BLS-k with $k = 24, 42, 48$	$(f_{x,Q}(P))^{(p^k-1)/r}$
KSS-32	$\left(f_{x,Q}(P)\cdot f_{2,Q}^{p^9}(P)\cdot\left(\overline{f_{3,Q}(P)}\right)^p\cdot\ell_{xQ,-3pQ}(P)\right)^{(p^{32}-1)/r}$
KSS-36	$\left(f_{x,Q}(P)\cdot f_{2,Q}^{p^7}(P)\cdot\left(\overline{f_{3,Q}(P)}\right)^p\cdot\ell_{xQ,-3pQ}(P)\right)^{(p^{36}-1)/r}$

* $\overline{t}$: conjugation of t in $\mathbb{F}_{p^k}$

$$m^{\Phi_k(p)/r} = (m^{\lambda_0})\cdot(m^{\lambda_1})^p\cdots(m^{\lambda_{s-1}})^{p^{s-1}}\cdot(m^{\lambda_s})^{p^s}, \tag{3}$$

where λ_i is the polynomial by x and $s = \varphi(k) - 1$. For computing the Eq. (3), in the BLS-k, we can compute it with essentially just exponentiation by x since the BLS-k has a very convenient way to compute the each m^{λ_i} [13]. In the KSS-k, we apply the addition chain technique with Ghammam *et al.*'s λ_i-representation to compute the each m^{λ_i} efficiently since the coefficients of λ_i are dozens bits [23]. Additionally, to efficiently compute the exponentiation by x, we use the Karabina squaring technique [27] in the case of the curve with $d = 6$, and Granger-Scott squaring technique [19] in the case of the curve with $d = 4$ respectively. Hence, the computation of the hard part requires exponentiations by x deg($p(x)$) $- 1$ times, Frobenius operations s times, and squarings/multiplications on $\mathbb{F}_{p^k}$.

Scalar Mult. in $\mathbb{G}_1$ *and* $\mathbb{G}_2$. To efficiently compute the scalar multiplication in $\mathbb{G}_1$ and $\mathbb{G}_2$, we use the Gallant-Lambert-Vanstone (GLV) [20] and Galbraith-Lin-Scott (GLS) [21] which are the scalar decomposition methods By using the GLV/GLS, for given a scalar u and $P \in \mathbb{G}_1$ (or $\mathbb{G}_2$), the scalar u is decomposed into t scalars $u_1, u_2, \ldots, u_t$ with roughly the size len$(u)/t$, then we convert the multi-scalar multiplication $uP = u_1P + u_2\psi(P) + \cdots + u_t\psi^{t-1}(P)$ by using an efficient endomorphism ψ [9], where $t = 2$ in $\mathbb{G}_1$ and $t = \varphi(k)$ in $\mathbb{G}_2$ respectively. The number of the doubling in $\mathbb{G}_1$ and $\mathbb{G}_2$ can reduce to roughly $1/t$. Moreover, by using the width-w non adjacent form (w-NAF) [45], the number of the addition in $\mathbb{G}_1$ and $\mathbb{G}_2$ can be reduced in the computing the each scalar multiplication $u_i\psi^{i-1}(P)$. Note that we chose the optimal window size w for the scalars u_i. Let $\mathtt{I}_i$ and $\mathtt{M}_i$ be the cost of inversion and multiplication in $\mathbb{F}_{p^i}$ respectively. We use the Jacobian coordinates in $\mathbb{G}_1$ since $\mathtt{I}_1 \approx 17.7\mathtt{M}_1$. We use the affine coordinates in $\mathbb{G}_2$ since $\mathtt{I}_{k/d} \approx 3.3\mathtt{M}_{k/d}$ for BLS-24, KSS-32, KSS-36, BLS-48, and the Jacobian coordinates in $\mathbb{G}_2$ since $\mathtt{I}_7 \approx 17.6\mathtt{M}_7$ for BLS-42.

Table 8. Timing of computing pairing (ML, FE), scalar multiplication in $\mathbb{G}_1$, $\mathbb{G}_2$ and exponentiation in $\mathbb{G}_3$ (M clk: million clocks)

		BLS-24	KSS-32	KSS-36	BLS-42	BLS-48
Pairing	ML	53.80	32.04	37.61	36.36	**20.48**
	FE	**89.84**	197.26	147.49	100.95	96.36
	Total	143.64	229.30	185.10	137.31	**116.84**
Scalar Mult. in $\mathbb{G}_1$		12.00	8.31	6.13	3.94	**3.56**
Scalar Mult. in $\mathbb{G}_2$		38.87	53.32	35.01	49.43	**25.18**
Exp. in $\mathbb{G}_3$		71.00	62.88	77.30	**56.00**	63.46

* Scalar Mult. in $\mathbb{G}_2$ of BLS-42 only used the Jacobian coordinates because of $\mathtt{I}_7 \approx 17.6\mathtt{M}_7$.

Exp. in $\mathbb{G}_3$*.* To efficiently compute the exponentiation in $\mathbb{G}_3$, we can use the GLS method and w-NAF since $\mathbb{G}_3 \subseteq \mathbb{G}_{\Phi_k(p)} = \{\alpha \in \mathbb{F}_{p^k} \mid \alpha^{\Phi_k(p)} = 1\}$ and the inversion in $\mathbb{G}_3$ can be efficiently computed by the conjugation [4].

6.3 Timing and Comparison

In this subsection, we show the timing of the main operations of PBC and compare those among the five candidate pairing-friendly curves.

Environment. We implement in C language, and its compiler is gcc 6.2.0 with -O3 option. We also measure on an Intel Core i7-6700 @ 3.4 GHz, RAM: 32 GB and OS: Ubuntu 16.04 (64-bit).

Results. Table 8 shows the timing of computing the pairing (ML, FE), scalar multiplication in $\mathbb{G}_1$, $\mathbb{G}_2$, and exponentiation in $\mathbb{G}_3$ for each candidate pairing-friendly curve. We discuss the timing of these operations among the five candidate pairing-friendly curves.

Pairing. The computational cost of the ML is affected by the bitlength and Hamming weight of x_0. The effect on the timing of the ML by the Hamming weight of x_0 is small since the Hamming weight of x_0 of each five candidate pairing-friendly curves is sufficiently small and much the same. As the embedding degree k increases, the bitlength of x_0 decrease, so the timing of the ML in the BLS-48 can be computed most efficiently.

The computational cost of the FE is affected by the coefficient of λ_i of Eq. (3), $\varphi(k)$, degree of $p(x)$, the bitlength of x_0 and p^k. The timings of the FE in the KSS-k are slower than that of the BLS-k since the computational cost of addition chain is required in addition to the exponentiation by x to compute the each m^{λ_i} in Eq. (3) of KSS-k. In the BLS-k, the coefficient of each λ_i of Eq. (3) is a few bits. Hence, the cost to compute all m^{λ_i} in the BLS-k is affected by the bitlength of p^k. The timing of the FE in the BLS-24 is faster than other BLS-k since the bitlength of x_0 of the BLS-24 is smaller.

Consequently, the timing of the pairing in the BLS-48 is faster than other pairing-friendly curves. The multi-pairing requires computing multiple MLs and one FE. Hence, the pairing-friendly curve with fast calculation of the ML has a significant effect on the efficiency of the computing the multi-pairing.

Scalar Mult. in $\mathbb{G}_1$. The input of scalar multiplication in $\mathbb{G}_1$ is a random element $P \in \mathbb{G}_1$ and a random scalar value of less than r. The computational cost is affected by the bitlength of r and the point addition/doubling in $\mathbb{G}_1$ affected by the bitlength of p. As k increases, the bitlength of p and r decrease. Hence, the timing of the scalar multiplication in $\mathbb{G}_1$ in the BLS-48 is faster than other pairing-friendly curves.

Scalar Mult. in $\mathbb{G}_2$. The input of scalar multiplication in $\mathbb{G}_2$ is a random element of $P \in \mathbb{G}_2$ and a random scalar value of less than r. Its computational cost is affected by the degree of twist d, and the bitlength r and $p^{k/d}$. The group $\mathbb{G}_2$ is a subgroup on $E'(\mathbb{F}_{p^{k/d}})$, and the bitlength of $p^{k/d}$ can be small when d is large. The bitlength of $p^{k/d}$ in the KSS-32 with $d = 4$ is about 2000-bit larger than that in the BLS-24, KSS-32, KSS-36, and BLS-48 with $d = 6$. Hence, the timing of the scalar multiplication in $\mathbb{G}_2$ in the KSS-32 is slower than other curves. Among the BLS-24, KSS-36 and BLS-48, as k increases, the bitlength of r decrease. Hence, the timing of the scalar multiplication in $\mathbb{G}_2$ in the BLS-48 is faster than other curves.

Exp. in $\mathbb{G}_3$. The input of exponentiation in $\mathbb{G}_3$ is a random element of $g \in \mathbb{G}_3$ and a random scalar value of less than r. The group $\mathbb{G}_3$ is a subgroup on finite field $\mathbb{F}_{p^k}$, and then the computational cost of the exponentiation in $\mathbb{G}_3$ is affected by the bitlength of r and the multiplication/squaring in $\mathbb{F}_{p^k}$. The BLS-42 and BLS-48 are theoretically better to compute the exponentiation in $\mathbb{G}_3$ efficiently since the bitlength of r is small rather than other curves. The number of the squaring in $\mathbb{F}_{p^k}$ in the BLS-48 is less than that in the BLS-42 because of the decomposition size. To compute the multi-exponentiation after the decomposition, the number of the multiplication in $\mathbb{F}_{p^k}$ in the BLS-42 is more reduced rather than in the BLS-48 because the bigger window size can be use in BLS-42 by w-NAF. As the result, the timing of the exponentiation in $\mathbb{G}_3$ in the BLS-42 is fastest in all candidate curves.

6.4 Impact on Timing by Revised Bitlength

In this subsection, we show the impact on the timing of the main operations of PBC by revised bitlength of p^k at 256-bit security level by comparing between our and previous implementations. Note that it is difficult to directly compare the timing of the main operations of PBC between our and previous implementations since the implemented algorithms and techniques are different.

In previous implementations, Scott [44] showed that the timing of the pairing is 88.8M clk, and Bos *et al.* [9] showed that the timing of the scalar multiplication in $\mathbb{G}_1$ and $\mathbb{G}_2$, and exponentiation in $\mathbb{G}_3$ are 5.2, 27.6, 47.1M clk respectively. These implemented in BLS-24 with about 15000-bit p^k and 500-bit r.

We compare the timing between our BLS-48 implementation and the previous BLS-24 implementations. Our BLS-48 implementation of the scalar multiplication in $\mathbb{G}_1$ is approximately 1.5 times faster than the previous BLS-24 implementations of that because the bitlength of r is the same between these implementations. Then, our BLS-48 implementation of other operations is approximately 1.0–1.3 times slower than the previous BLS-24 implementations of that due to the effect of the efficient NFS algorithms.

7 Conclusion

We give for the first time the revised bitlength which the DLP is computationally infeasible against the efficient NFS algorithms (e.g. SexTNFS, STNFS), and the timing of the main operations of PBC for the five candidate pairing-friendly curves (i.e. the BLS-k with $k = 24, 42, 48$, KSS-k with $k = 32, 36$) at 256-bit security level. On the security side, we show that it is necessary to increase bitlengths by more than 10000-bit from the previous estimation to achieve the 256-bit security level. On the implementation side, we show that the BLS-48 curve is the suitable curve at the 256-bit security level by comparing the timing of the main operations of PBC among the five candidate pairing-friendly curves with revised bitlengths. For more speeding up, we should implement $\mathbb{F}_p$-arithmetic in assembly, apply other efficient algorithms, etc.

A Norm Plots of BLS-24, KSS-32, KSS-36 and BLS-42

In this appendix, we show the norm plots of the GNFS and SNFS algorithms for the BLS-24, KSS-32, KSS-36, and BLS-42 in order to revise the bitlength of the these curves using the same method discussed in Sect. 5 (Figs. 3, 4, 5, 6, 7, 8, 9 and 10).

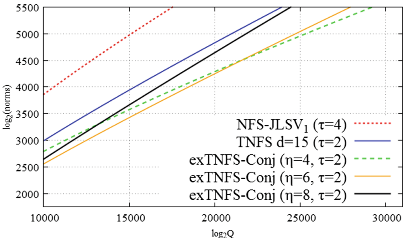


Fig. 3. Norms of GNFS algorithm in $\mathbb{F}_{p^{24}}$

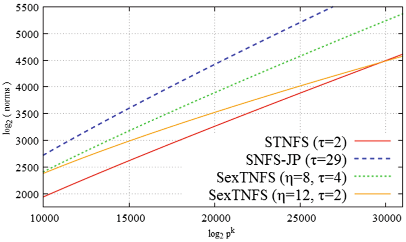


Fig. 4. Norms of SNFS algorithm in $\mathbb{F}_{p^{24}}$

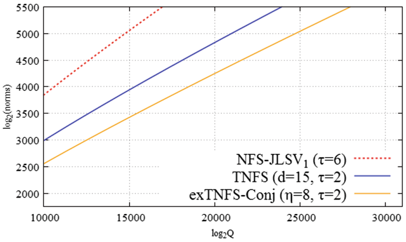


Fig. 5. Norms of GNFS algorithm in $\mathbb{F}_{p^{32}}$

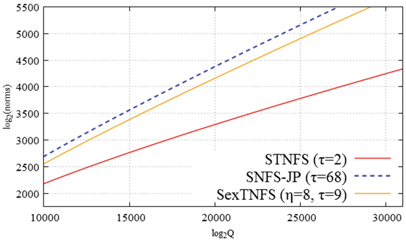


Fig. 6. Norms of SNFS algorithm in $\mathbb{F}_{p^{32}}$

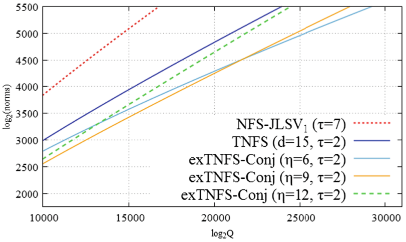


Fig. 7. Norms of GNFS algorithm in $\mathbb{F}_{p^{36}}$

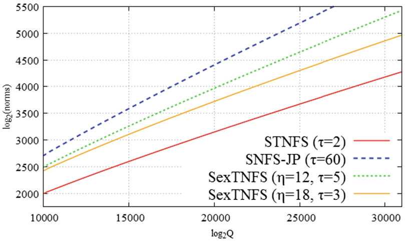


Fig. 8. Norms of the SNFS algorithm in $\mathbb{F}_{p^{36}}$

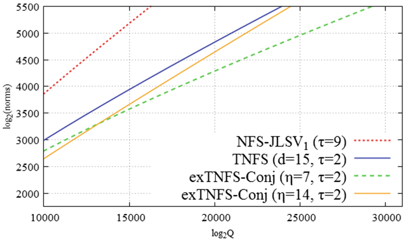


Fig. 9. Norms of GNFS algorithm in $\mathbb{F}_{p^{42}}$

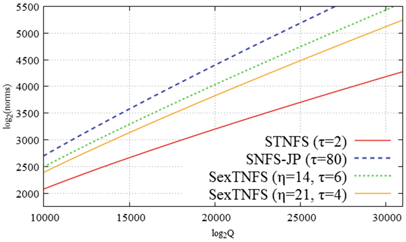


Fig. 10. Norms of SNFS algorithm in $\mathbb{F}_{p^{42}}$

References

1. Aoki, K., Franke, J., Kleinjung, T., Lenstra, A.K., Osvik, D.A.: A kilobit special number field sieve factorization. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 1–12. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76900-2_1
2. Acar, T., Lauter, K., Naehrig, M., Shumow, D.: Affine pairings on ARM. In: Abdalla, M., Lange, T. (eds.) Pairing 2012. LNCS, vol. 7708, pp. 203–209. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36334-4_13

3. Aranha, D.F., Fuentes-Castañeda, L., Knapp, E., Menezes, A., Rodríguez-Henríquez, F.: Implementing pairings at the 192-bit security level. In: Abdalla, M., Lange, T. (eds.) Pairing 2012. LNCS, vol. 7708, pp. 177–195. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36334-4_11
4. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López, J.: Faster explicit formulas for computing pairings over ordinary curves. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 48–68. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20465-4_5
5. Barker, E.B., Barker, W.C., Burr, W.E., Polk, W.T., Smid, M.E.: Recommendation for key management - part 1: General (Revision 4). NIST SP 800-57 (2016)
6. Barbulescu, R., Gaudry, P., Guillevic, A., Morain, F.: Improving NFS for the discrete logarithm problem in non-prime finite fields. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 129–155. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46800-5_6
7. Bouvier, C., Gaudry, P., Imbert, L., Jeljeli, H., Thom, E.: Discrete logarithms in GF(p) — 180 digits. Announcement available at the NMBRTHRY archives, item 004703 (2014)
8. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_13
9. Bos, J.W., Costello, C., Naehrig, M.: Exponentiating in pairing groups. In: Lange, T., Lauter, K., Lisoněk, P. (eds.) SAC 2013. LNCS, vol. 8282, pp. 438–455. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43414-7_22
10. Barbulescu, R., Gaudry, P., Kleinjung, T.: The tower number field sieve. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 31–55. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48800-3_2
11. Barreto, P.S.L.M., Lynn, B., Scott, M.: Constructing elliptic curves with prescribed embedding degrees. In: Cimato, S., Persiano, G., Galdi, C. (eds.) SCN 2002. LNCS, vol. 2576, pp. 257–267. Springer, Heidelberg (2003). doi:10.1007/3-540-36413-7_19
12. Barreto, P.S.L.M., Costello, C., Misoczki, R., Naehrig, M., Pereira, G.C.C.F., Zanon, G.: Subgroup security in pairing-based cryptography. In: Lauter, K., Rodríguez-Henríquez, F. (eds.) LATINCRYPT 2015. LNCS, vol. 9230, pp. 245–265. Springer, Cham (2015). doi:10.1007/978-3-319-22174-8_14
13. Costello, C., Lauter, K., Naehrig, M.: Attractive subfamilies of BLS curves for implementing high-security pairings. In: Bernstein, D.J., Chatterjee, S. (eds.) INDOCRYPT 2011. LNCS, vol. 7107, pp. 320–342. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25578-6_23
14. Danilov, S.A., Popovyan, I.A.: Factorization of RSA-180, Cryptology ePrint Archive, Report 2010/270 (2010)
15. European Union Agency of Network and Information Security (ENISA): Algorithms, key sizes and parameters report, 2013 recommandations, version 1.0, October 2013
16. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. J. Cryptol. **23**, 224–280 (2010)
17. Fried, J., Gaudry, P., Heninger, N., Thomé, E.: A kilobit hidden SNFS discrete logarithm computation. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 202–231. Springer, Cham (2017). doi:10.1007/978-3-319-56620-7_8
18. The GNU Multiple Precision Arithmetic Library. https://gmplib.org/

19. Granger, R., Scott, M.: Faster squaring in the cyclotomic subgroup of sixth degree extensions. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 209–223. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13013-7_13
20. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 190–200. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_11
21. Galbraith, S.D., Lin, X., Scott, M.: Endomorphisms for faster elliptic curve cryptography on a large class of curves. J. Crypto **24**, 446–469 (2011)
22. Guillevic, A., Morain, F., Thomé, E.: Solving discrete logarithms on a 170-bit MNT curve by pairing reduction, arXiv preprint arXiv:1605.07746 (2016)
23. Ghammam, L., Fouotsa, E.: Adequate elliptic curves for computing the product of n pairings. In: Duquesne, S., Petkova-Nikova, S. (eds.) WAIFI 2016. LNCS, vol. 10064, pp. 36–53. Springer, Cham (2016). doi:10.1007/978-3-319-55227-9_3
24. Hess, F., Smart, N., Vercauteren, F.: The eta pairing revisited. IEEE Trans. Inf. Theory **52**(10), 4595–4602 (2006)
25. Joux, A., Lercier, R., Smart, N., Vercauteren, F.: The number field sieve in the medium prime case. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 326–344. Springer, Heidelberg (2006). doi:10.1007/11818175_19
26. Joux, A., Pierrot, C.: The special number field sieve in $\mathbb{F}_{p^n}$. In: Cao, Z., Zhang, F. (eds.) Pairing 2013. LNCS, vol. 8365, pp. 45–61. Springer, Cham (2014). doi:10.1007/978-3-319-04873-4_3
27. Karabina, K.: Squaring in cyclotomic subgroups. Math. Comput. **82**, 555–579 (2013)
28. Kim, T., Barbulescu, R.: Extended tower number field sieve: a new complexity for the medium prime case. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 543–571. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53018-4_20
29. Kachisa, E.J., Schaefer, E.F., Scott, M.: Constructing brezing-weng pairing-friendly elliptic curves using elements in the cyclotomic field. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 126–135. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85538-5_9
30. Kim, T., Jeong, J.: Extended tower number field sieve with application to finite fields of arbitrary composite extension degree. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10174, pp. 388–408. Springer, Heidelberg (2017). doi:10.1007/978-3-662-54365-8_16
31. Kleinjung, T.: Discrete Logarithms in GF(p) – 768 bits. Announcement available at the NMBRTHRY archives, item **004917** (2016)
32. Lenstra, A.K.: Unbelievable security *matching AES security using public key systems*. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 67–86. Springer, Heidelberg (2001). doi:10.1007/3-540-45682-1_5
33. Lenstra, A.K., Lenstra, H.W. (eds.): The Development of the Number Field Sieve. LNM, vol. 1554. Springer, Heidelberg (1993). doi:10.1007/BFb0091534
34. Miller, V.S.: The weil pairing, and its efficient calculation. J. Cryptol. **17**, 235–261 (2004)
35. Mori, Y., Akagi, S., Nogami, Y., Shirase, M.: Pseudo 8–sparse multiplication for efficient ate–based pairing on barreto–naehrig curve. In: Cao, Z., Zhang, F. (eds.) Pairing 2013. LNCS, vol. 8365, pp. 186–198. Springer, Cham (2014). doi:10.1007/978-3-319-04873-4_11
36. Mitsunari, S.: A fast implementation of the optimal ate pairing over BN curve on intel haswell processor, Cryptology ePrint Archive, Report 2013/362 (2013)

37. Menezes, A., Sarker, P., Singh, S.: Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography, Cryptology ePrint Archive, Report 2016/1102 (2016)
38. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14623-7_11
39. Pollard, J.: Monte Carlo methods for index computation (mod p). Math. Comput. **32**(143), 918–924 (1978)
40. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: SCIS 2000, C-20, pp. 26–28 (2000)
41. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). doi:10.1007/11426639_27
42. Devegili, A.J., Scott, M., Dahab, R.: Implementing cryptographic pairings over barreto-naehrig curves. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 197–207. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73489-5_10
43. Scott, M., Benger, N., Charlemagne, M., Dominguez Perez, L.J., Kachisa, E.J.: On the final exponentiation for calculating pairings on ordinary elliptic curves. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 78–88. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03298-1_6
44. Scott, M.: On the efficient implementation of pairing-based protocols. In: Chen, L. (ed.) IMACC 2011. LNCS, vol. 7089, pp. 296–308. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25516-8_18
45. Schirokauer, O.: Using number fields to compute logarithms in finite fields. Math. Comp. **69**, 1267–1283 (2000)
46. Schirokauer, O.: Virtual logarithms. J. Algorithms **57**, 140–147 (2005)
47. Vercauteren, F.: Optimal pairings. IEEE Trans. Inf. Theory **56**(1), 455–461 (2010)

Data Protection and Mobile Security

No Free Charge Theorem: A Covert Channel via USB Charging Cable on Mobile Devices

Riccardo Spolaor[1](✉), Laila Abudahi[2], Veelasha Moonsamy[3], Mauro Conti[1], and Radha Poovendran[2]

[1] University of Padua, Padua, Italy
{rspolaor,conti}@math.unipd.it
[2] University of Washington, Seattle, USA
{abudahil,rp3}@uw.edu
[3] Radboud University, Nijmegen, The Netherlands
email@veelasha.org

Abstract. More and more people are regularly using mobile and battery-powered handsets, such as smartphones and tablets. At the same time, thanks to the technological innovation and to the high user demand, those devices are integrating extensive battery-draining functionalities, which results in a surge of energy consumption of these devices. This scenario leads many people to often look for opportunities to charge their devices at public charging stations: the presence of such stations is already prominent around public areas such as hotels, shopping malls, airports, gyms and museums, and is expected to significantly grow in the future. While most of the times the power comes for free, there is no guarantee that the charging station is not maliciously controlled by an adversary, with the intention to exfiltrate data from the devices that are connected to it.

In this paper, we illustrate for the first time how an adversary could leverage a maliciously controlled charging station to exfiltrate data from the smartphone via a USB charging cable (i.e., without using the data transfer functionality), controlling a simple app running on the device—and without requiring any permission to be granted by the user to send data out of the device. We show the feasibility of the proposed attack through a prototype implementation in Android, which is able to send out potentially sensitive information, such as IMEI and contacts' phone number.

1 Introduction

Market studies predicted that in 2011 smartphone sales would surpassed that of desktop PCs [31]. To this date, smartphones remain the most used handheld devices. This is partly due to the fact that these devices are more powerful and provide more functionalities than the traditional feature phones. As a result, users can perform a variety of tasks on an actual smartphone device, which in the past would have been possible only on a desktop PC. In order to carry out such tasks, the smartphone platform offers its users a plethora of applications (apps).

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 83–102, 2017.
DOI: 10.1007/978-3-319-61204-1_5

Moreover, as users are constantly using apps (e.g., the gaming app, Pokémon Go) and would eventually require to recharge their smartphones, the demand for public charging stations have increased significantly in the last decade. Such stations can be seen in public areas such as airports, shopping malls, gyms and museums, where users can recharge their devices for free. In fact, this trend is also giving rise to a special type of business[1], which allows shop owners to install charging stations in their stores so as to boost their sales by providing free phone recharge to shoppers.

As the phone recharging is usually for free, however, at the same time one cannot be sure that the public charging stations are not maliciously controlled by an adversary. The Snowden revelations gave us proof that civilians are constantly under surveillance and nations are competing against each other by deploying smart technologies for collecting sensitive information en mass. In our work, we consider an adversary (e.g., manufacturers of public charging stations, Government agencies) whose aim is to take control over the public charging station and whose motive is to exfiltrate data from the user's smartphone once the device is plugged into the station.

In this paper, we demonstrate the feasibility of using power consumption (in the form of power bursts) to send out data over a Universal Serial Bus (USB) charging cable, which acts as a covert channel, to the public charging station. We implemented a proof-of-concept app, *PowerSnitch*, that can send out bits of data in the form of power bursts by manipulating the power consumption of the device's CPU. Interestingly, PowerSnitch does not require any special permission from the user at install-time (nor at run-time) to exfiltrate data out of the smartphone over the USB cable. On the adversary's side, we designed and implemented a decoder to retrieve the bits that have been transmitted via power bursts. Our empirical results show that we can successfully decode a payload of 512 bits with a 0% Bit Error Ratio (BER). In addition, we stress that the goal of this paper is to assess for the first time the feasibility of data transmission on such a covert channel and not to optimize its performance, which we will tackle as future work.

We focus primarily on Android, as it is currently the leading platform and has a large user base. However, we believe that this attack can be deployed on any other smartphone operating systems, as long as the device is connected to a power source at the public charging station.

Our contributions are as follows:

1. To our knowledge, we are the first to demonstrate the practicality of using the power feature of a USB charging cable as a covert channel to exfiltrate data, in the form of power bursts, from a device while it is connected to a power supplier. The attack works in Airplane mode as well.
2. We implemented a prototype of the attack, i.e., we designed and implemented its two components: (i) We built a proof-of-concept app, *PowerSnitch*, which does not require any permission granted by the user to communicate bits

[1] chargeitspot.com, chargetech.com.

of information in the form of power bursts back to the adversary; (ii) The decoder is deployed on the adversary side, i.e., public charging station to retrieve the binary information embedded in the power bursts.
3. We are able with our prototype to actually send out data using power bursts. Our prototype demonstrate the practical feasibility of the attack.

The rest of the paper is organized as follows. In Sect. 2, we present a brief literature overview of covert channel and data exfiltration techniques on smartphones. In Sect. 3, we include some background knowledge on Android operating system, and signal transmission and processing. In Sect. 4, we provide a description of our covert channel and decoder design, followed by the experimental results in Sect. 5 and discussion in Sect. 6. We conclude the paper in Sect. 7.

2 Related Work

In this section, we survey the existing work in the area of covert channels on mobile devices. We also present other non-conventional attack vectors, such as side channel information leakage via embedded sensors which can be used for data exfiltration.

Covert Channels – A covert channel can be considered as a secret channel used to exfiltrate information from a secured environment in an undetected manner. Chandra et al. [8] investigated the existence of different covert channels that can be used to communicate between two malicious applications. They examined the common resources (such as battery) shared between two malicious applications and how they could be exploited for covert communication. Similar studies presented in [14,18,21,26] exploited unknown covert channels in malicious and clean applications to leak out private information.

As demonstrated by Aloraini et al. [1], the adversary is further empowered as smartphones continue to have more computational power and extensive functionalities. The authors empirically showed that speech-like data can be sent over a cellular voice channel. The attack was successfully carried out with the help of a custom-built rootkit installed on Android devices. In [10], Do et al. demonstrated the feasibility of covertly exfiltrating data via SMS and inaudible audio transmission, without the user's knowledge, to other mobile devices including laptops.

In our work, we present a novel covert channel which exploits the USB charging cable by leaking information from a smartphone via power bursts. Our proposed method is non-invasive and can be deployed on non-rooted Android devices. We explain the attack in more detail in Sect. 4.1.

Power Consumption by Smartphones – In order to prolong the longevity of the smartphone's battery, it is crucial to understand how apps consume energy during execution and how to optimize such consumption. To this end, several works [4,6,23,33] have been proposed. Furthermore, the authors from [13,17] studied apps' power consumption to detect anomalous behavior on smartphones, thus leading to detection of malware.

Since existing work focus on energy consumption on the device, our attack would therefore go undetected as the smartphone's CPU sends small chunks of encoded data, which are translated into power bursts, back to the public charging station. Additionally, state-of-the-art attacks that have been performed while the smartphone is charging [15,19] exploit vulnerabilities of USB interface rather than actual energy consumption.

Attack Vectors using Side Channel Leaks – Modern smartphones are embedded with a plethora of sensors that allow users to interact seamlessly with the apps on their smartphones. However, these sensors have access to an abundance of information stored on the device that can get exfiltrated. These data leaks can be used as a side channel to infer, otherwise undisclosed, sensitive information about the user or device [2,16,32].

The authors from [3,22] demonstrated how accelerometer readings can be used to infer tap-, gesture- and keyboard-based input from users to unlock their smartphones. Similarly, Spreitzer [27] showed that the ambient-light sensor can be exploited to infer users' PIN input. Moreover, considering network traffic as a side-channel, it is possible to identify the set of apps installed on a victim's mobile device [28,29], and even infer the actions the victim is performing with a specific app [9].

As pointed out in the aforementioned existing work, the adversarial model did not require any special privileges to exploit side channel leaks to recover data exfiltrated via sensors. In this paper, we show that our custom app, PowerSnitch, does not require any special permissions to be granted by the user in order to communicate information (in terms of power bursts) to the adversary. Furthermore, we stress that while the `INTERNET` permission is one approach of data exfiltration, our proposed work is different as we show the feasibility and practicability of using a USB cable to exfiltrate data. In particular, our attack still works even when the phone is switched to Airplane mode and defeats existing USB charging protection dongles, as in [7], since we only require the USB power pins to exfiltrate data.

3 Background Knowledge

In this section, we briefly recall several concepts that we use in our paper about Android operating system in Sect. 3.1, and signal transmission and processing in Sect. 3.2.

3.1 Android System and Permissions

In the Android Operating System (OS), apps are distributed as `APK` files. These files are simple archives which contain bytecode, resources and metadata. A user can install or uninstall an app (thus the APK file) by directly interacting with the smartphone. When an Android app is running, its code is executed in a sandbox. In practice, an app runs isolated from the rest of the system, and it cannot directly access other apps' memory. The only way an app could gain memory access is via the mediation of inter-process communication techniques made available by Android. These measures are in place to prevent the access of malicious apps to other apps' data, which could potentially be privacy-sensitive.

Since Android apps run in a sandbox, they not only have restriction in shared memory usage, but also to most system resources. Instead, the Android OS provides an extensive set of Accessible Programming Interfaces (APIs), which allows access to system resources and services. In particular, the APIs that give access to potentially privacy-violating services (e.g., camera, microphone) or sensitive data (e.g., contacts) are protected by the Android Permission System [11]. An app that wants access to protected data or service must declare in the form of permission (identified by a string) in its manifest file. The list of permissions needed by an app is shown to the user when installing the app, and cannot be changed while an app is installed on the device. With the introduction of Android M (i.e., 6.0), permissions can be dynamically granted (by users) during an app's execution.

The permission system has also the goal of reducing the damage in case of a successful attack that manages to take control of an app, by limiting the resources that app's process has access to. Unfortunately, permission over-provisioning is a common malpractice, so much so that research efforts have been spent in trying to detect this problem [5]. Moreover, an app asking for permissions not related to its purpose (or functionality) can hide malicious behaviors (i.e., spyware or malware apps) [20].

3.2 Signal Transmission and Processing

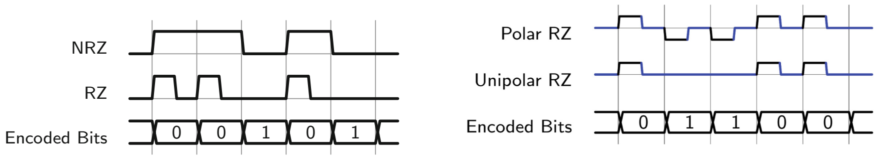
In this section, we provide some background information on bit transmission, and signal processing and decoding used in our proposed decoder (see Sect. 4.4).

Bit Transmission – To enable bit transmission over our channel, an understanding of basic digital communication systems is essential. For proof-of-concept purposes, the design of our bit transmission system was inspired by amplitude-based modulation in the digital communication literature.

Amplitude-Shift Keying (ASK) is a form of digital modulation where digital bits are represented by variations in the amplitude of a carrier signal. To send bits over our channel, we used On-Off Signaling (OOS), which is the simplest form of ASK where digital data is represented by the presence and absence of some pulse *p(t)* for a specific period of time. Figure 1a shows the difference between a Return-to-Zero (RZ) and a Non-Return-to-Zero (NRZ) on-off encoding. In NRZ encoding, bits are represented by a sufficient condition (a pulse) that occupies the entire bit period T_b while RZ encoding represents bits as pulses for a duration of $\mathrm{T}_b/2$ before it returns to zero for the following $\mathrm{T}_b/2$ period.

On the other hand, Fig. 1b shows the difference between a unipolar and a polar RZ on-off signaling. In a polar RZ encoding, two different conditions, different-sign pulses are used to encode different bits(zeros/ones) while the presence and absence of a single pulse, a positive one in our case, are used to encode different bits.

For the sake of our channel design, it is safe to assume that we can only increase the power consumption of a phone at certain times and hence, are able to generate only positive (high) bursts. Thus, a unipolar encoding seems more relative and applicable for our channel. Moreover, successive peaks, such as the first two zeros in Fig. 1a, are easier to identify, and thus decode, in the RZ-encoded signal than in the NRZ one. This advantage of RZ over NRZ becomes especially apparent in cases where the bit period is expected not to be restrictively fixed in the received signal whether it is due to expected high channel noises or lack of full control of the phone's CPU. Therefore, unipolar RZ on-off signaling was used to encode leaked bits over our covert channel.



(a) Return-to-Zero (RZ) and Non-Return-to-Zero (NRZ) On-Off Encoding.

(b) A Polar and a Unipolar encoding of an RZ On-Off Signal.

Fig. 1. A comparison between bit encoding methods

Signal Processing and Decoding – After choosing the appropriate encoding method to transmit bits, it is also essential to think about the optimal receiver design and how to process the received signal and decode bits with minimum error probability at the receiver side of the channel. As known in the digital communication literature, matched filters are the optimal receivers for Additive White Gaussian Noise (AWGN) channels. We refer the reader to Sect. 4.2 of [24] for a detailed proof.

Matched Filters are obtained by correlating the received signal $R(t)$ with the known pulse that was first used to encode a transmitted bit, in this case P(t) with period T_b. After correlation, the resulted signal is then sampled at time T_b, which means that the sampling rate equals to $1/\mathrm{T}_b$ samples/seconds. This way, each bit is guaranteed to be represented by only one sample. The decoding decision will then be made based on that one sample value; if the sample value is more than a given threshold, this indicates the presence of P(t); and hence a zero in our case, while a sample value below the threshold indicates the absence of P(t) and hence a one is decoded.

However and most importantly, for matched filters to work as expected, it is essential to have fixed bit period T_b throughout the entire received signal. If the periods of the received bits were varying, the matched filter samples taken with the $1/\mathrm{T}_b$ sampling rate will not be as optimal and representative of the bit data as expected and synchronization will be lost.

Since there exist infrequent phone-specific, OS-enforced conditions that can affect the power consumption of a phone, the noises on our channel are expected to be more complex to fit in an AWGN model. Hence, a matched filter receiver is most likely not the optimal receiver for our channel. More creative decoder design decisions are needed to maximize the throughput of our channel and minimize the error probability.

4 Covert Channel Using Mobile Device Energy Consumption

In this section, we elaborate on the components that make up our covert channel attack. We begin by giving an overview of the attack in Sect. 4.1. We then define the terms and parameters for transmission in Sect. 4.2, followed by a description of each component of the attack: PowerSnitch app in Sect. 4.3 and the energy traces decoder in Sect. 4.4.

4.1 Overview of Attack

As illustrated in Fig. 2, the attack scenario considers two components: the victim's Android mobile device (sender) and an accomplice's power supplier (receiver). Victim's mobile device is connected to a power supplier (controlled by the adversary) through a USB cable.

The left side of Fig. 2 depicts what happens after the victim has installed our proof-of-concept app, *PowerSnitch*. The app is able to exfiltrate victim's private information, which gets encoded as CPU bursts with a specific timing. Indeed, as the CPU is one of the most energy consuming resources in a device, a CPU burst can be directly measured as a "peak" based on the amount of energy absorbed by a mobile device. The right side of Fig. 2 illustrates how the energy supplier is able to measure (with a given sampling rate) the electric current provided to the mobile device connected to the public charging station. Then, such electric measurement, which is considered as a *signal*, is given as input to a decoder. It should be noted that the adversary, i.e., the public charging station, has control of the power supplier, and thus is able to control the amount of current provided to the device – even if it has the "fast charge" capability.

In our proposed covert channel attack, we consider situations in which users connect their mobile devices for more than 20 min. There are several scenarios that fulfill such time requirements. Examples are: (i) recharging a device overnight in a hotel room; (ii) making use of locked boxes in shopping malls for charging mobile phones; (iii) recharging devices on planes, in trains and cars.

In addition, we argue that those time requirements are more than reasonable since generally, 72% of users leave their phones on charging for more than 30 min, with an average time of 3 h and 54 min, as reported in [12]. This means that: (i) the mobile device is in stand-by mode; (ii) CPU and the use of other energy consuming resources (e.g., Wi-Fi or 3/4g data connection) usage is limited only to the OS and background apps. Moreover, since there is no user interaction, it is reasonable to assume that the phone screen, which has a relevant impact on energy consumption, will stay off for the aforementioned period of time.

Moreover, it is also worth noting that the attack is still feasible if there is no data connection between the victim's device and the power supplier, such as Media Transfer Protocol (MTP), Photo Transfer Protocol (PTP), Musical Instrument Digital Interface (MIDI). This is possible as our methodology only requires power consumption to send out the power bursts. Moreover, from Android version 6.0, when a device is connected via USB, it is set by default to "Charging" mode (i.e., just charge the device), thus no data connection is allowed unless the user switches on data connection manually. This improvement in security feature does not impact our proposed attack as we do not make use of data connection to transfer the power bursts.

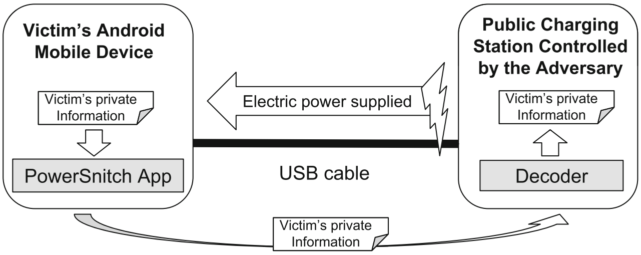

Victim's Android Mobile Device
Victim's private Information
PowerSnitch App
Electric power supplied
USB cable
Public Charging Station Controlled by the Adversary
Victim's private Information
Decoder
Victim's private Information

Fig. 2. The schema of the components involved in the attack.

4.2 Terminology and Transmission Parameters

In this section, we define the necessary terminology to identify concepts used in the rest of the paper:

- *Payload* is the information that has to be sent from the device to the receiver.
- *Transmission* is the whole sequence of bits transmitted in which the payload is encoded.

In order to obtain a successful communication, the sender and the receiver need to agree on the parameters of the transmission.

- *Period* is the time interval during which a bit is transmitted.
- *Duty cycle* is the ratio between burst and rest time in a period T_b. For example: if a burst lasts for $T_b/2$, the duty cycle will be 50%.
- *Preamble* is the sequence of bit used to synchronize the transmission. Usually a preamble is used at the beginning of a transmission, but it can also be used within a transmission in order to recover the synchronization in case of error. In our case, we used a preamble composed of 8 bits.

4.3 PowerSnitch App: Implementing the Attack on Android

The first component of our covert channel we discuss is the proof-of-concept which we called *PowerSnitch app*. This app, used for the covert channel exploit, has been designed as a service in order to be installed as a standalone app or a library in a repackaged app. Henceforth, we refer to both these variants simply with the term “app”.

PowerSnitch app requires only the `WAKE_LOCK` permission and does not need root access to work. Such permission allows PowerSnitch app to wake and force execute the CPU while the device is in sleep mode, so that it can start to transmit the payload. We stress that since it is running as a background service, PowerSnitch app still works even when user authentication mechanisms (e.g., PIN, password) are in place. Moreover, since it does not use any conventional communication technology (e.g., Wi-Fi, Bluetooth, NFC), PowerSnitch app can exfiltrate information even if the device is in airplane mode. It is worth mentioning that Android M (i.e., 6.0) introduced the Doze mode [30], a battery power-saving optimization which reduces the apps activity when the device is inactive and running on battery for extended periods of time. When it is in place, Doze mode stops background CPU and network activity (ignoring wakelocks, job scheduler, Wi-Fi scan, etc.). Then on periodic time intervals (i.e., maintenance windows), the system runs all pending jobs, synchronization and alarms. However, such optimization is not active when a device is connected to a power source or when the screen is on. This means that Doze does not affect our proposal since we need the wakelock function but also the device to be plugged to a power source. Moreover, since our proposed attack needs also the status of the battery, it does not need any permission in order to obtain such information: in fact, it is sufficient to only register at run-time (not even in the manifest) a specific broadcast receiver (i.e., `ACTION_BATTERY_CHANGED`).

In Fig. 3, we illustrate the modules of PowerSnitch app. It is composed of three modules: *Payload encoder*, *Transmission controller* and *Bursts generator*. *Payload encoder* takes the payload as input and outputs an array of bits. The payload can be any element that can be serialized into an array of bits. We use strings as payloads, they are first decomposed into an array of characters and then, using the ASCII code of each character, into an array of bits. *Payload Encoder* can also add to its output array synchronization bits (e.g., the preamble), and error checking codes (e.g., CRC).

Transmission controller is in charge of monitoring the status of the device with the purpose of understanding when it is feasible to transmit through the covert channel. Indeed, in order to not be detected by the user, it checks whether all the following conditions are satisfied: (i) the USB cable is connected; (ii) the screen is off; and (iii) the battery is sufficiently charged (see Sect. 6). If our app receives a broadcast intent from the Android OS that invalidates one of the aforementioned conditions, *Transmission controller* module will interrupt the transmission. It is worth noticing that to obtain all this information, PowerSnitch app does not need any additional permission. From the GUI app used in

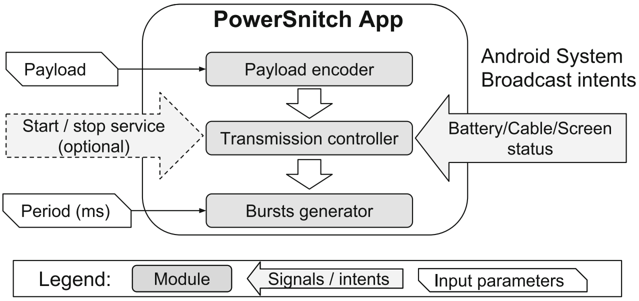


Fig. 3. The modules involved in the PowerSnitch app.

our experiments, we are also able to start or stop PowerSnitch app (represented in Fig. 3 with a dotted arrow).

The last component of PowerSnitch app is *Bursts generator*. The task of this component is to convert the encoded payload into bursts of energy consumption. These bursts will generate a signal that can be measured at the other end of the USB cable (i.e., the power supplier). In order to obtain these bursts of energy consumption, *Bursts generator* module can use a power consuming resource of the mobile device such as CPU, screen or flashlight. Our proof-of-concept, *Bursts generator* uses the CPU: a CPU burst is generated from a simple floating point operation repeated in a loop for a precise amount of time (given by transmission parameters).

4.4 Analysis of Energy Traces

To make better decoder design decisions, several channel traces were observed, collected and then used to calculate channel estimations and implement different simulations of the channel performance and behavior. A standard on-off signaling decoder needs to know the exact period of bits in the received signal in order to be able to decode them. However, a channel built based on a phone's power consumption is expected to have hard-to-model noises that, after examining the collected channel data traces, are actually affecting not only the peak periods but also the peak amplitudes. The amount of external power consumed by a phone can be largely affected by dominant OS-enforced, manufacturer-specific factors. For instance, different sudden drop patterns in power consumption especially when the phone is almost or completely charged, lack of control over the OS scheduler; when, how often and for how long do some heavy power-consuming OS background services run, as well as the precision and sampling rate of the power monitor on the receiver side of the channel.

Figure 4 shows a portion of the channel data captured after a transmission of ten successive bits (ten Zeros, therefore ten peaks) was initiated by our app on a Nexus 6 phone. It should be noted that the data was passed through a low-pass filter to get rid of harsh, high frequency noises in order to make the signal looks smoother. As a result, based on a threshold of 100 mA, ten peaks are successfully detected. Moreover, the width of each peak, and hence the period of each bit, is varying sufficiently. The first bit, for example, has a period of 300 ms while the eighth one has a period of only 195 ms. Although the intended bit period generated and transmitted by the app was 500 ms, the average period of the received bits was actually 311 ms, which the receiver has no way to predict in advance. Such variations in the received signal are expected to affect the performance of any decoder. An ideal matched filter receiver will have hard time decoding such inconsistent signal and synchronization will be lost very quickly. We elaborate further on this issue in the remaining sections.

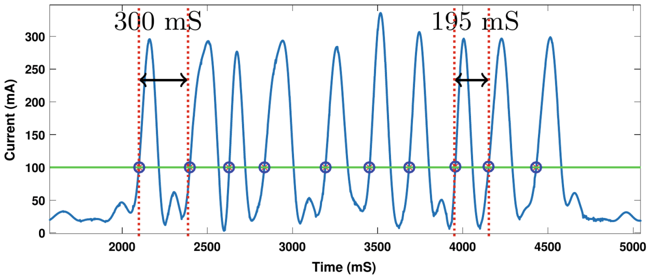


Fig. 4. A portion of a received signal showing the variations in peak widths and amplitudes.

Decoder Design. In this section, we provide additional explanation about the different processing stages that our decoder is taking the received signal through in order to overcome the channel inconsistencies and decode the sent bits with the minimum Bit Error Ratio (BER). In signal processing, the quality of a communication channel can be measured in terms of BER (represented as a percentage), which is the number of bit errors divided by the total number of transmitted bits over the channel. Channels affected by interference, distortion, noise, or synchronization errors have a high BER.

Figure 5 summarizes the different processing stages which will be discussed in the order they take place in, along with some background information and algorithm justifications, where applicable.

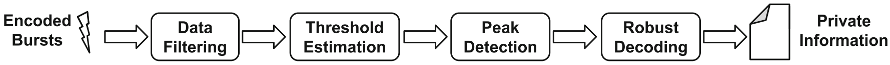
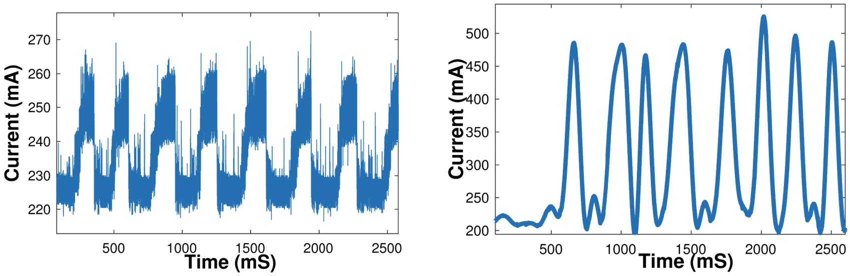


Fig. 5. Different phases of our decoder.

Data Filtering. First, the received signal is passed through a low-pass filter to get rid of the harsh high-frequency noises. For instance, Fig. 6 shows the same portion of a received signal before and after applying the low-pass filter. The low-pass filter helps not only to make the signal looks smoother, but also to make the threshold-based detection of real peaks easier by eliminating narrow-peak noises that can be falsely identified as real peaks or bits. Additionally, the low-pass filter used in our decoder adjusts its pass and stop frequencies based on the intended bit period generated by the phone in order to make sure that we do not over-filter or over-attenuate the signal.



(a) Raw received signal.

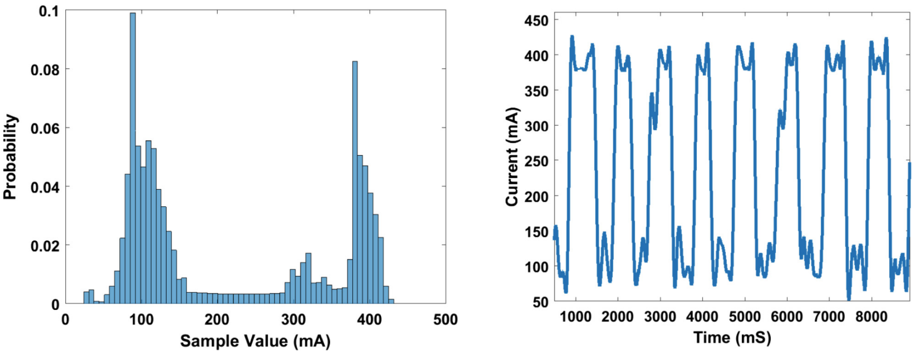
(b) Low-pass filtered received signal.

Fig. 6. A portion of a received signal before and after applying the low-pass filter.

Threshold Estimation. The decoder detects peaks by decoding unipolar RZ on-off encoded bits. The presence or absence of a peak (a 0 or a 1 in our case, respectively) at a certain time and for a specific period is then translated to the corresponding bit. Peak detection is usually done by setting an appropriate threshold; anything above the threshold is a peak and anything below is just noise. However, deciding which threshold to use is not a trivial process especially with the unpredictable noise in our channel and the variations in width and amplitude of the received peaks.

The threshold value used by the decoder is highly critical to peaks detection, the resulted width of detected peaks and the decoder performance. Hence, we primarily use a known preamble data sent prior to the actual packet to estimate the threshold. The preamble consists of eight known bits (eight zeros in our case) at the start of the transmission, which means that the decoder is expecting eight peaks at the start. Since a unipolar RZ on-off encoded zero has a pulse for half of the bit period, the preamble is expected to have roughly the same number of peak and no-peak samples. Therefore, a histogram of the preamble samples is expected to split into two portions; peak and no-peak portions. Figure 7a shows a histogram of the preamble samples shown in Fig. 7b. As observed, the histogram has two distinguishable densities; each of them look like the probability density function of a Gaussian distribution.

Estimating the parameters (mean and variance) of two Gaussians that are believed to exist in one overall distribution is a complicated statistical problem. However, the Gaussian Mixture Model (GMM), introduced and explained in [25], is a probabilistic model commonly used to address this type of problem and to statistically estimate the parameters of existing Gaussian populations. To estimate the threshold, as shown in Fig. 8, the decoder uses the GMM to fit two Gaussians to the two histogram portions, find the mean of each one of them and then compute the threshold as the middle point between the two means. As a result, our decoder is able to estimate the threshold independently and without any previous knowledge of the expected amplitudes of the received bits. After that, each sample is converted to either a peak sample or no-peak sample based on whether the sample value is above or below the estimated threshold.



(a) A histogram of the preamble samples.

(b) A received preamble signal.

Fig. 7. A histogram of the preamble samples shows a mixture of two Gaussian-like densities.

Robust Decoding. Generally, the way a decoder translates the peak and no-peak samples to zeros and ones is highly time-sensitive. For instance, if the bit period is fixed and equals to T$_b$, the decoder simply checks the presence or absence of the peak in each T$_b$ period. Since this decoding decision is made based on a very strict timing manner, the slightest error in the received bit periods will cause a quick loss of synchronization. As mentioned in the previous section, the received peak widths (and hence bit periods) over our channel are changing with a high variation around their mean. Therefore, our decoding decision cannot rely on an accurate notion of time. Instead, our decoder needs to assume a sufficient amount of error in the period of each received bit and to search for the peaks in a wider range instead of a strict period of time.

To address this level of time-insensitivity and achieve robustness to synchronization errors, our decoding decision was made based on the time difference between each two successive peaks. As an example, assume that two successive zeros were sent and hence two peaks were received. The difference between the start time of each peak should be rounded to the average bit period. It should be noted that the decoder computes the average bit period based on the received preamble data. However, if a zero-one-zero transmission was made, the time difference of the start of the two received peaks should be rounded to double of

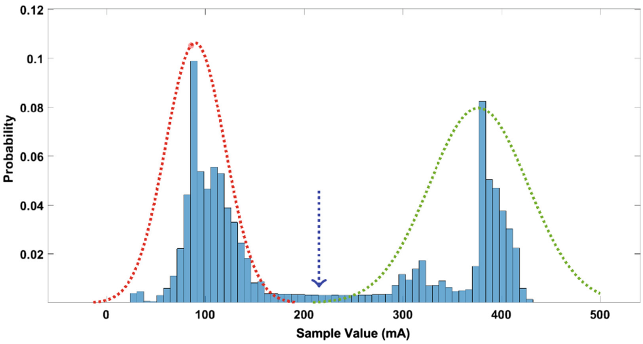


Fig. 8. Using the Gaussian Mixture Model to estimate the threshold.

the average bit period. If a zero-one-one-zero transmission was made, the difference should be rounded to triple the average period and so on. Eventually, synchronization is regained with every detected peak and based only on the time difference between peaks, the decoder makes a decision on how many no-peak bits (ones in our case) are transmitted between the zeros. The time difference does not have to be exactly equal to a multiple of the average bit period. Instead, a range of values can be rounded up to the same value and thus more flexible time-insensitive decoding decision is made.

5 Experimental Evaluation

In this section, we first describe the devices used in our experiments and the values for transmission parameters. We then report the results of the transmission evaluation.

5.1 Experiment Settings

In our experiments, we programmed the PowerSnitch app using Android Studio with API. The device used to measure the energy provided to the device via USB cable is Monsoon Power Monitor[2] in USB mode with 4.55 V in output. The decoder used to process signal was implemented in MATLAB. In order to evaluate the performance of the transmission, we send out a payload comprised of letters and numbers of ASCII code for a total of 512 bits. The values of period used range from 500 ms to 1000 ms with increments of 100 ms. It is worth mentioning that bits sent over our channel were not packeted and no error detection or correction techniques were used. For each phone and bit period, BER was computed after sending 512 bits at once and then number of bits that were incorrectly decoded was calculated.

[2] www.msoon.com/LabEquipment/PowerMonitor.

We evaluate the performance of our proposal on the following devices running Android OS: Nexus 4 with Android 5.1.1 (API 22), Nexus 5 with Android 6.0 (API 23), Nexus 6 with Android 6.0 (API 23) and Samsung S5 with Android 5.1.1 (API 22). We underline that the devices used in our experiments are actual personal devices, kindly lent by some users without any money reward. In order to replicate an actual real world scenario, we did not uninstall any app, nor stopped any app running in background. The only intervention we made on those devices is the installation of our PowerSnitch app.

5.2 Results

In Table 1, we report the performance of the decoder for processing the received power bursts on different mobile devices. The results presented in the table are in terms of BER in the transmission of the payload; the lower the BER, the better is the quality of the transmission. For Nexus devices (i.e., Nexus 4, 5 and 6), we achieve a zero or low BER of periods of 800 ms and 900 ms (i.e., 1.25 and 1.11 bits per seconds, respectively). While for Nexus 4 and 6, the BER remains under 20% and, for Nexus 5, it increases to 37% and 40% with periods 700 ms and 600 ms, respectively. For Samsung S5, the transmission BER is at 12.5% with a period of 1 s, and it slowly increases to around 21% with a period of half a second.

The higher BER for Nexus 5 (i.e., periods 700 ms and 600 ms in Table 1) are due to de-synchronization of the signal that the decoder was not able to recover. To cope with this problem, we can divide the payload into packets, where a packet header will be the preamble in order to recover the synchronization. A quick overview of the communication literature can show how a BER of 30% can be recovered using a simple Forward Error Correction (FEC) technique where the transmitter encodes the data using an Error Correction Code (ECC) prior to transmission; for example bits redundancy or parity checks.

Table 1. Results in terms of Bit Error Ratio (BER) as percentage.

Device		Period (milliseconds)					
Model	Operating system version	1000	900	800	700	600	500
Samsung S5	Android 5.1.1 (API 22)	12.5	13.5	13.31	16.33	17.9	21.42
Nexus 4	Android 5.1.1 (API 22)	13.5	0.78	0.0	0.0	13.33	16.21
Nexus 5	Android 6.0 (API 23)	21.0	0.0	0.95	36.82	40.35	13.4
Nexus 6	Android 6.0 (API 23)	1.07	0.0	0.21	0.0	4.05	7.42

6 Discussion and Optimizations

In this section, we elaborate further on the results obtained in the experimental evaluation of our proposed attack (Sect. 5). In particular, we discuss on interesting observation made during our experiments. We also present the optimizations that were implemented in the framework in order to make our proposed attack more robust.

An interesting phenomenon to notice is that, as observed in our experiments, the level of battery affects the quality of the transmission signal. In Fig. 9, we present the amount of electric current provided by the power supplier to a Nexus 6 during recharge (i.e., the first 35 min) and full battery states (i.e., after 35 min). Indeed, when the level for the battery is low (i.e., 0% to around 40%) the device consumes a high amount of energy, and almost all of it is used to recharge the battery.

When attempting to transmit data in the aforementioned conditions, we discover that the bursts were not easily distinguishable. In fact, the difference in terms of energy consumption between burst and rest was so small that it cannot be distinguished from noise; thus, they can be filtered out during the signal processing. Additionally, when the level of the battery is increased, the amount of energy consumed to recharge the battery gradually decreases. We observed that when the battery level is higher than 50%, the power bursts become more and more distinguishable. However the best condition under which the bursts are clear is when the battery is fully charged. Indeed, as we can notice from Fig. 9, the current drops down after the battery level reaches 100%, because there is no need to provide energy to the battery anymore - except to keep the device running.

The percentages mentioned above also depends from the power supplier used to provide energy to the device. In our experiments, we used Monsoon power monitor which provides as output at most 4.55 V. Due to the limitation of such power monitor, during the recharge of devices with fast charge technology (e.g., Samsung S5, Nexus 6 and 6P), which are able to work with 5.3 V and 2 mA, the energy consumed is almost constant until the battery is almost fully charged. Thus, we cannot decode any signal from the energy consumption.

In order to avoid to transmit when the receiver is not able to decode the signal, PowerSnitch checks whether the battery level is among a certain threshold ω. Such threshold ω can be obtained by PowerSnitch itself, simply knowing the model in which it is running. This information can be easily obtained without any permission (`android.os.Build.MODEL` and `MANUFACTURER`).

Optimizations. In what follows, we elaborate on the optimizations that were implemented in order to not be detected or make the victim suspicious. The first optimization is to keep a duty cycle (i.e., the time of burst in a period) under 50%. During an attack, if such optimization is not taken into account (i.e., a duty cycle greater than 75%), the victim may be alerted by two possible effects:

- the temperature of the device could increase significantly, in a way that could be perceived by touching it.
- if the attack takes place during the battery charge phase, the battery will take more time to recharge due to the high amount of energy used by CPU.

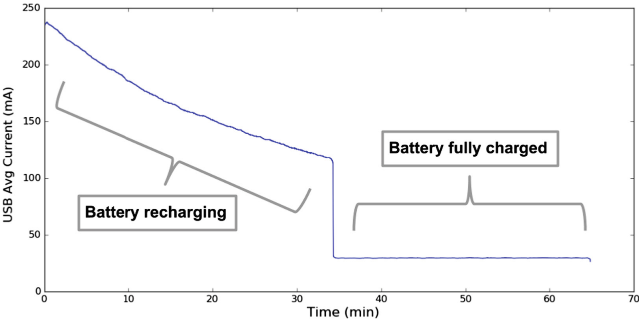


Fig. 9. Electric current provided to a Nexus 6 during recharge phase and battery fully charged.

However, as previously explained in Sect. 3.2, the duty cycle should be 50% of period (i.e., $T_b/2$) in order to achieve a RZ. Thus, the above effects are already taken care of in our proposed attack.

Another optimization involves the Android Debug Bridge (ADB) tool. It is possible to monitor CPU consumption of an Android device via ADB. Hence, one may use such debug tool to detect that something strange is happening on the device (i.e., a transmission on the covert channel using CPU bursts). Fortunately, PowerSnitch app could easily detect whether ADB setting is active through `Settings.Global.ADB_ENABLED`, once again provided by an Android API.

Another optimization to PowerSnitch app would be the ability to detect if the power supplier is an accomplice of the attack. The accomplice has to let PowerSnitch app know that it is listening to the covert channel by communicating something equivalent to a "hello message". In order to do so, we can rely on the information about the amount of electric current provided to recharge the battery. Such information is made available through `BatteryManager` object, provided by Android API. In particular, `BATTERY_PROPERTY_CURRENT_NOW` data field (available from API 21 and on devices with power gauge, such as Nexus series) of `BatteryManager` records an integer that represents the current entering the battery in terms of mA.

On one hand, the power supplier can then variate the current in output above and below a certain threshold θ with a precise timing. As a practical and non-limiting example, at a point in time during the recharging, the power supplier can output current with the following behavior: (i) below θ for t seconds, (ii) above θ for t seconds, (iii) again below θ for t seconds and finally (iv) above θ for good. On the other hand, since PowerSnitch app monitors `BATTERY_PROPERTY_CURRENT_NOW` and knows the aforementioned behavior (along with both θ and t), it will be able to understand that at the other end of the USB cable there is an accomplice power supplier ready to receive a transmission. This optimization is significant for reducing the chance to remain undetected, since PowerSnitch app will transmit data if and only if it is sure that an accomplice power supplier is listening. With such optimization, we will obtain a half-duplex communication channel, since the communication is bidirectional but only one participant (i.e., the device or the power source) is allowed to transmit at a time. This optimization is not currently implemented and will be considered as future work.

To summarize, the conditions under which the transmission of data is optimal and the chance of being detected is lowest are as follows: the mobile device has to be charged more than 50%, the screen has to be off, ADB tool should be switched off (which is true by default) and the phone must to be plugged with a USB charging cable to a public charging station which is controlled by the adversary.

7 Conclusion

In this paper, we demonstrate for the first time the practicality of using a (power-only) USB charging cable as a covert channel to exfiltrate data from a smartphone, which is connected to a charging station. Since there are no visible signs of the existence of a covert channel while the battery is recharging, the user is oblivious that data is being leaked from the device. Moreover, our proposed covert channel defeats existing USB charging protection dongles, as described in [7] because it requires only the USB power pins to exfiltrate data in the form of CPU power bursts.

To show the feasibility and practicality of our proposed covert channel, we implemented an app, *PowerSnitch*, which does not require the user to grant access to permissions at install-time (nor at run-time) on a non-rooted Android phone. Once the device is plugged in a compromised public charging station, the app encodes sensitive information and transmits it via power bursts back to the station. Our empirical results show that we are able to exfiltrate a payload encoded in power bursts at 1.25 bits per seconds with a BER under 1% on the Nexus 4-6 devices and a BER of around 13% for Samsung S5. As future work, we plan to investigate malicious power banks and how they can be exploited using our covert channel to exfiltrate data from smart devices. We will also work on the transmitter and decoder by extending the framework to include error correction algorithms and synchronization recover mechanisms to lower down the BER of data transmission—as this was not the main goal of this paper.

Acknowledgments. This work is supported by ONR grants N00014-14-1-0029 and N00014-16-1-2710, ARO grant W911NF-16-1-0485 and NSF grant CNS-1446866.

Veelasha Moonsamy is supported by the Technology Foundation STW (project 13499 - TYPHOON & ASPASIA) from the Dutch government.

Mauro Conti is supported by a Marie Curie Fellowship funded by the European Commission (agreement PCIG11-GA-2012-321980). This work is also partially supported by the EU TagItSmart! Project (agreement H2020-ICT30-2015-688061), the EU-India REACH Project (agreement ICI+/2014/342-896), "Physical-Layer Security for Wireless Communication", and "Content Centric Networking: Security and Privacy Issues" funded by the University of Padua. This work is partially supported by the grant n. 2017-166478 (3696) from Cisco University Research Program Fund and Silicon Valley Community Foundation.

We would like to thank Elia Dal Santo and Moreno Ambrosin for their insightful comments.

References

1. Aloraini, B., Johnson, D., Stackpole, B., Mishra, S.: A new covert channel over cellular voice channel in smartphones. Technical report (2015). arXiv preprint arXiv:1504.05647
2. Aviv, A.J., Gibson, K., Mossop, E., Blaze, M., Smith, J.M.: Smudge attacks on smartphone touch screens. In: Proceedings of USENIX WOOT (2010)
3. Aviv, A.J., Sapp, B., Blaze, M., Smith, J.M.: Practicality of accelerometer side channels on smartphones. In: Proceedings of USENIX ACSAC (2012)
4. Baghel, S., Keshav, K., Manepalli, V.: An investigation into traffic analysis for diverse data applications on smartphones. In: Proceedings of NCC (2012)
5. Bartel, A., Klein, J., Le Traon, Y., Monperrus, M.: Automatically securing permission-based software by reducing the attack surface: an application to android. In: Proceedings of ACM ASE (2012)
6. Carroll, A., Heiser, G.: An analysis of power consumption in a smartphone. In: Proceedings of USENIX ATC (2010)
7. Chacos, B.: USB condom promises to protect your dongle from infected ports. PC World, August 2014. http://tinyurl.com/hvlqkrt
8. Chandra, S., Lin, Z., Kundu, A., Khan, L.: Towards a systematic study of the covert channel attacks in smartphones. In: Tian, J., Jing, J., Srivatsa, M. (eds.) SecureComm 2014. LNICSSITE, vol. 152, pp. 427–435. Springer, Cham (2015). doi:10.1007/978-3-319-23829-6_29
9. Conti, M., Mancini, L.V., Spolaor, R., Verde, N.V.: Analyzing android encrypted network traffic to identify user actions. IEEE TIFS **11**(1), 114–125 (2016)
10. Do, Q., Martini, B., Choo, K.K.R.: Exfiltrating data from android devices. Comput. Secur. **48**, 74–91 (2015)
11. Felt, A.P., Chin, E., Hanna, S., Song, D., Wagner, D.: Android permissions demystified. In: Proceedings of ACM CCS (2011)
12. Ferreira, D., Dey, A.K., Kostakos, V.: Understanding human-smartphone concerns: a study of battery life. In: Proceedings of PerCom (2011)
13. Kim, H., Smith, J., Shin, K.G.: Detecting energy-greedy anomalies and mobile malware variants. In: Proceedings of ACM MobiSys (2008)
14. Lalande, J.-F., Wendzel, S.: Hiding privacy leaks in android applications using low-attention raising covert channels. In: Proceedings of ARES (2013)
15. Lau, B., Jang, Y., Song, C., Wang, T., Chung, P.H., Royal, P.: Mactans: injecting malware into IOS devices via malicious chargers. Black Hat, USA (2013)
16. Lin, L., Kasper, M., Güneysu, T., Paar, C., Burleson, W.: Trojan side-channels: lightweight hardware trojans through side-channel engineering. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 382–395. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04138-9_27
17. Liu, L., Yan, G., Zhang, X., Chen, S.: VirusMeter: preventing your cellphone from spies. In: Kirda, E., Jha, S., Balzarotti, D. (eds.) RAID 2009. LNCS, vol. 5758, pp. 244–264. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04342-0_13
18. Marforio, C., Ritzdorf, H., Francillon, A., Capkun, S.: Analysis of the communication between colluding applications on modern smartphones. In: Proceedings of USENIX ACSAC (2012)
19. Meng, W., Lee, W.H., Murali, S., Krishnan, S.: Charging me and i know your secrets!: towards juice filming attacks on smartphones. In: Proceedings of ACM CPS-SEC (2015)

20. Moonsamy, V., Rong, J., Liu, S.: Mining permission patterns for contrasting clean and malicious android applications. J. Future Gener. Comput. Syst. **36**, 122–132 (2013)
21. Novak, E., Tang, Y., Hao, Z., Li, Q., Zhang, Y.: Physical media covert channels on smart mobile devices. In: Proceedings of ACM UbiComp (2015)
22. Owusu, E., Han, J., Das, S., Perrig, A., Zhang, J.: ACCessory: password inference using accelerometers on smartphones. In: Proceedings of ACM HotMobile (2012)
23. Pathak, A., Charlie Hu, Y., Zhang, M.: Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with Eprof. In: Proceedings of ACM EuroSys (2012)
24. Proakis, J.G.: Intersymbol Interference in Digital Communication Systems. Wiley, Hoboken (2003)
25. Reynolds, D.: Gaussian mixture models. Encycl. Biom., 827–832 (2015)
26. Schlegel, R., Zhang, K., Zhou, X.Y., Intwala, M., Kapadia, A., Wang, X.: Soundcomber: a stealthy and context-aware sound trojan for smartphones. In: Proceedings of NDSS (2011)
27. Spreitzer, R.: Pin skimming: exploiting the ambient-light sensor in mobile devices. In: Proceedings of ACM CCS SPSM (2014)
28. Stöber, T., Frank, M., Schmitt, J., Martinovic, I.: Who do you sync you are?: Smartphone fingerprinting via application behaviour. In: Proceedings of ACM WiSec (2013)
29. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: Appscanner: automatic fingerprinting of smartphone apps from encrypted network traffic. In: Proceedings of IEEE EuroS&P (2016)
30. Android Developers. Optimizing for Doze and App Standby. http://tinyurl.com/zvphw46
31. Business Insider. The Smartphone Market Is Now Bigger Than The PC Market (2011). http://goo.gl/XkM8XM
32. Yan, L., Guo, Y., Chen, X., Mei, H.: A study on power side channels on mobile devices. In: Proceedings of ACM Internetware (2015)
33. Yoon, C., Kim, D., Jung, W., Kang, C., Cha, H.: AppScope: application Energy metering framework for android smartphone using kernel activity monitoring. In: Proceedings of ATC (2012)

Are You Lying: Validating the Time-Location of Outdoor Images

Xiaopeng Li, Wenyuan Xu(✉), Song Wang, and Xianshan Qu

Department of CSE, University of South Carolina, Columbia, SC, USA
{xl4,wyxu,songwang,xqu}@cec.sc.edu

Abstract. Photos have been commonly used in our society to convey information, and the associated contextual information (i.e., the capture time and location) is a key part of what a photo conveys. However, the contextual information can be easily tampered or falsely claimed by forgers to achieve malicious goals, e.g., creating fear among the general public or distorting public opinions. Thus, this paper aims at verifying the capture time and location using the content of the photos only. Motivated by how the ancients estimate the time of the day by shadows, we designed algorithms based on projective geometry to estimate the sun position by leveraging shadows in the image. Meanwhile, we compute the sun position by applying astronomical algorithms according to the claimed capture time and location. By comparing the two estimations of the sun position, we are able to validate the consistency of the capture time and location, and hence the time-location of the photos. Experimental results show that our algorithms can estimate sun position and detect the inconsistency caused by falsified time, date, and latitude of location. By choosing the thresholds to be 9.2° and 4.8° for the sun position distance and altitude angle distance respectively, our framework can correctly identify 91.1% of the positive samples, with 7.7% error in identifying the negative samples. Note that we assume that the photos contain at least one vertical object and its shadow. Nevertheless, we believe this work serves as the first and important attempt in verifying the consistency of the contextual information only using the content of the photos.

Keywords: Capture time and location · Sun position · Shadows · Consistency · Projective geometry

1 Introduction

Benefiting from the development of digital technologies and internet, photos become increasingly common in our society. A huge number of photos are shared through social media platforms. People use photos to convey information and express emotions, and even employ them to illustrate news stories [14]. Meanwhile, people are exposed to fake photos that had been used for malicious purposes: fooling the world and creating chaos as well as panic [3,7]. For example,

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 103–123, 2017.
DOI: 10.1007/978-3-319-61204-1_6

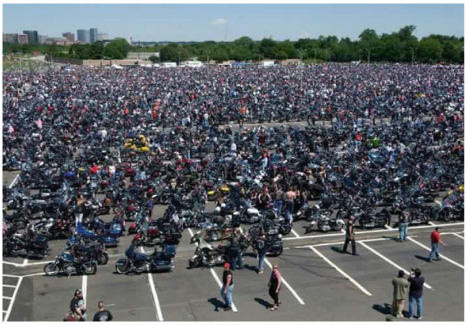


Fig. 1. A photo that was taken in Sept. 2013 was used for a news event happened in Jan. 2017.

the Hurricane Sandy hit the northeastern U. S. in 2012: numerious fake disaster photos and rumors were spread through social networks and caused panic and fear among the general public [10]. Therefore, the U.S. Federal Emergency Management Agency had to set up a "rumor control" section to defend against misinformation including fake photos on social networks [1]. In addition, fake photos have been used to distort public opinions. For instance, the fake refugee photos were shared online in the Europe's refugee crisis in 2015 and used to twist public opinion on asylum seekers [5].

Previous studies mainly focused on devising forensic techniques to detect photo tampering and manipulation. For example, researchers have proposed approaches to demonstrate copy-move manipulation [2,8,23] and leveraged shadows and lighting to determine photo tampering [4,20]. However, in addition to manipulating the content of a photo itself, the contextual information (i.e., the capture time and location) can also be falsified. For instance, the photo in Fig. 1 was claimed to be taken in January 2017, and was used on social medias to illustrate the news that a fleet of bikers were on the way to Washington D.C for President Trump's inauguration. However, the photo was actually published in 2013 for the anniversary of 9/11[1]. Thus, it is promising if we can validate the capture time and locations immediately purely using the photos themselves.

Determining whether the capture time or the location of an image is real is promising yet challenging. Although most images have timestamps and GPS information enclosed, these can be altered without traces once the format is known. Deciding whether a picture is taken at a place simply by experiences is infeasible since the image scenes may appear to be similar in various places, such as public lawns, parking lots, beaches and roadsides. Even if the capture location is true, the capture time can be falsified without any traces. Finding evidences from the content of an image to verify the time is difficult. Objects that reveal time directly (e.g., clocks and watches) are rarely seen in images.

[1] https://www.buzzfeed.com/tasneemnashrulla/bikers-for-trump-inauguration-fake-pictures.

Objects such as clothing, or colors of trees may indicate the capture time, but these indicators can only reveal a relatively long time span (e.g., a T-shirt is suitable from April through October in many places). So far, limited research has addressed this problem. Garg *et al.* [9] demonstrated the feasibility of using the Electric Network Frequency signal as a natural timestamp for video data in an indoor enviroment. Junejo and Foroosh [17] and Wu and Cao [27] used shadow trajectories to estimate the geo-location of stationary cameras from multiple outdoor images. Tsai *et al.* [26] and Kakar and Sudha [18] developed approaches that leverage the geolocation of images and the sun information to estimate the capture time for outdoor images. However, to the best of our knowledge, none has been done to validate both the capture time and location. In this paper, we study how to validate whether the image's capture time and location are true from a single outdoor image that has at least one shadow. Although we require a shadow in an image, we believe our work serves as the first attempt towards a full-fledged solution.

The basic idea is that the position of the sun is determined by time and location and can be utilized to check time-location consistency of outdoor images. Specifically, we estimate the sun position from two sets of information: (1) utilize vertical objects and their shadows in images to estimate sun position, and (2) use the claimed capture time and location in the metadata of images for estimation. Finally, we compare these two values and decide whether the claimed capture time and location are true.

In summary, we outline our three main contributions as below:

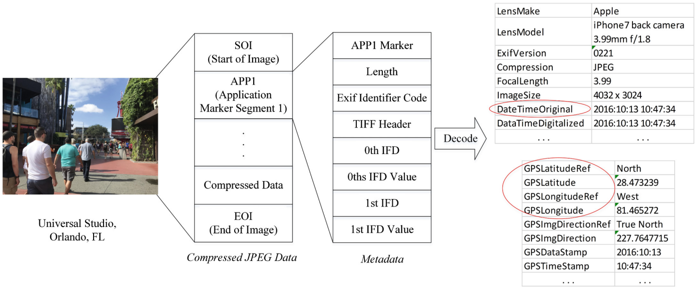
- We propose a framework that is called AYL for validating time-location consistency of outdoor images. We show that the variances of sun position correlate with the time and location, and the correlation can be used to determine whether the capture time and location of images are consistent.
- We demonstrate that the sun position can be acquired from shadows and design algorithms to estimate the sun position from one vertical object and its shadow in the image. The results show that the algorithms are effective.
- We implement the proposed framework and evaluate it using photos collected in 15 cities across the U.S. and China, which proves AYL to be effective.

2 Overview

We specify the threat model, overview the framework of AYL, and summarize the research challenges in this section.

2.1 Threat Model

We assume that an attacker modifies the capture time and location of an image for malicious purposes, but doesn't tamper or manipulate the image itself. Note that even if she modifies the image, we can detect it utilizing the prior works [2,4,8,20,23]. Below we describe how an attacker can modify the metadata.



LensMake	Apple
LensModel	iPhone7 back camera 3.99mm f/1.8
ExifVersion	0221
Compression	JPEG
FocalLength	3.99
ImageSize	4032 x 3024
DateTimeOriginal	2016:10:13 10:47:34
DataTimeDigitalized	2016:10:13 10:47:34
. . .	. . .

GPSLatitudeRef	North
GPSLatitude	28.473239
GPSLongitudeRef	West
GPSLongitude	81.465272
GPSImgDirectionRef	True North
GPSImgDirection	227.7647715
GPSDataStamp	2016:10:13
GPSTimeStamp	10:47:34
. . .	. . .

Fig. 2. The basic structure of JPEG compressed image files.

An image file contains not only the image itself but also the metadata that describes who, when, where, and how an image was taken [21,22]. Exchangeable image file format (Exif) is a popular standard that specifies the formats of images. The specification uses the existing file formats (e.g., JPEG) with the addition of specific metadata tags. Figure 2 shows the basic structure of JPEG compressed image files [16], and the application marker segment I (APPI) contains contextual information of images, e.g., the capture time, the image size, compression format, and details of cameras (focal length, camera maker) [16], etc. In particular, `DateTimeOriginal` records the capture time. `GPSLatitude` and `GPSLongitude` contain the GPS location (i.e., latitude and longitude) of where the image was taken. `GPSimgDirection` represents the direction measured by the magnetometer (i.e., the direction in which the camera faces). Modifying the capture time and the GPS information enclosed in metadata can be easily accomplished by using metadata editing tools such as ExifTool [12]. For example, the photo shown in Fig. 2 was taken in Orlando, FL, on 13th October 2016, at 10:47 a.m. An attacker can claim that the photo was taken in May 2016 in Los Angeles by changing `DateTimeOriginal` and other related GPS fields.

2.2 Overview of AYL

Our goal is to validate whether the claimed capture time and location are true. The capture time indicates the date and time when the photo was taken, and the capture location reveals where it was taken.

Basic Idea. Although an attacker can modify the metadata and claim that a photo was taken at time X and location Y, she won't be able to change the "time" and "location" information that is embedded in the photo. Thus, our framework works as follows. On one hand, we utilize the contents in outdoor images—vertical objects and resulting shadows—to extract the sun position that reflects when and where the image was taken. On the other hand, we utilize the metadata information—the claimed capture time and location—to obtain a second estimation of the sun position by applying astronomical algorithms. If these two estimations are close enough, we consider the capture time and location to be true with a high probability. Otherwise, they are considered to be falsified.

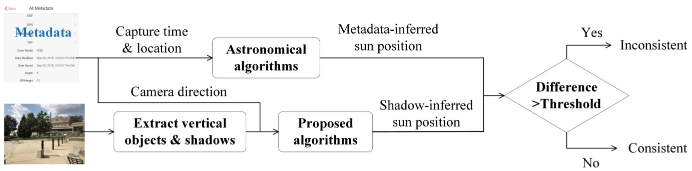


Fig. 3. The work flow of the proposed AYL framework.

Assumption. Without loss of generality, we assume that photos are taken using smartphone's rear cameras and the smartphone is held in such way that the camera looks at the front horizontally or vertically, and it is perpendicular to the ground. We further assume that the photographer stands on ground and the ground that interested shadows lie on is approximated to water level. Finally we assume that at least one vertical object and its shadow can be seen in the image. The objects can be human beings, road signs, lampposts, tree trunks and so on.

Workflow. Figure 3 shows the work flow of proposed approach. For convenience of description, we use the term *shadow-inferred sun position* to refer to the sun position estimated from shadows in the image, and use the term *metadata-inferred sun position* to refer to the sun position calculated from claimed capture time and location. In this paper, *capture time* denotes the date of year and the time of day unless otherwise indicated.

2.3 Research Challenges

Shadow-Inferred Sun Position. The first challenge is how to obtain sun position from a single image. Although shadows can be viewed in images, we still need to know the length ratio of objects and their shadows and the orientation of shadows to determine the sun position. However, the relative position of two objects in real world is no longer preserved when they are projected to a 2-d image. How to measure the actual length ratio and angles can be a challenging problem. Although single view reconstruction has been extensively studied, there is no generalized way to recover the relative positions of objects from one single image. To address above challenges, algorithms based on projective geometry are proposed in Sect. 4.

Validation. Once the *shadow-inferred sun position* is obtained, the next challenge would be how to validate that the capture time and location are true. To estimate the true capture time and location directly from the *shadow-inferred sun position* is difficult since a specific sun position can be viewed at various places and times. Conversely, the claimed capture time and location can determine a unique value of sun position that should be close enough to the *shadow-inferred sun position.* Then, we convert previous problem into a new problem: how to determine the two estimations—the *shadow-inferred sun position* and the *metadata-inferred sun position*—are close enough indicating the same sun position. Appropriate thresholds need to be selected to solve this problem. The sun moves across the sky at a varying speed. The changes of sun position with respect to time and location on the earth are not constant, which further complicates the selection of thresholds. We will discuss this problem in Sect. 5 and our experimental results are presented in Sect. 6.

3 Background

We discuss the basics on how the sun changes its position in the section, which serves as the foundation of our algorithm.

3.1 Sun Position Definition

The position of the sun in the sky is defined by an azimuth angle and an altitude angle. An azimuth angle describes the direction of the sun, whereas an altitude angle defines the height of the sun [24]. As shown in Fig. 4, the sun azimuth angle A is measured clockwise in the horizontal plane, from the north to the direction of the sun. Its value varies from 0° (north) through 90° (east), 180° (south), 270° (west), and up to 360° (north again). The altitude angle h is measured from the horizontal to the sun and it thus ranges from $-90°$ (at the nadir) through 0° (on the horizon), up to 90° (at the zenith). For instance, when the sun crosses the meridian, its azimuth is 180° and altitude is at its largest value in a day.

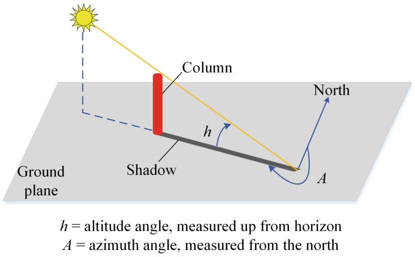

Column
North
h
Ground plane
Shadow
A
h = altitude angle, measured up from horizon
A = azimuth angle, measured from the north

Fig. 4. An illustration of the altitude and azimuth angles of the sun.

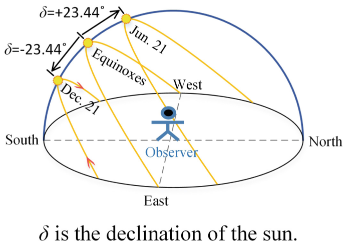


Fig. 5. The path of the sun across the sky as observed on various dates in the northern hemisphere.

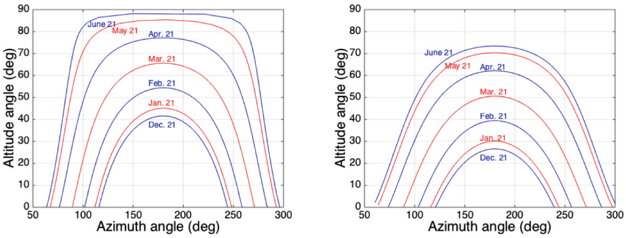
3.2 How Does the Sun Move

Observed from any location on the earth, the sun moves continuously across the sky throughout days and years. The relative position change is mainly caused by two types of motions of the earth: the rotation around its axis, and the revolution around the sun [24]. It takes about 24 h for the earth to finish one rotation around the earth's axis and about 365 days to complete one revolution around the sun. For an observer on the earth, the first motion contributes to the alternation of day and night, and the second motion leads to the alternation of seasons.

Daily Sun Path. Because of the earth's daily rotation, the sun appears to move along with the celestial sphere every day. It makes a 360° journey around the celestial sphere every 24 h. To an observer on the earth, the sun rises somewhere along the eastern horizon, and goes up to the highest point (zenith) around the noon, then goes down until it sets along the western horizon. Figure 5 shows three of the sun's daily paths viewed on the earth. Accordingly, the cast shadows of any objects move oppositely from somewhere along the west to somewhere along the east. The shadows' lengths vary with the sun's altitude angle. They become shorter and shorter since sunrise and reach the shortest when the sun is at its zenith. Then they become longer over time till sunset. Thus, the shadow that a camera takes at the same day and location but different times of the day will be totally different.

Yearly Sun Path. The sun's daily path across the sky also changes throughout the year. This is because the earth does not rotate on a stationary axis and the tilt in the axis varies each day with respect to the earth's orbit plane. To an observer on the earth, the sun looks higher in the summer than it looks in the winter at the same time in the day. As shown in Fig. 5, the sun follows different circles at different days in one year: most northerly on June 21st and most southerly on December 21st. The sun's motion along the north-south axis over a year is known as the declination of the sun, denoted by δ. Thus, the sun position inferred from photos taken at the same location and time but different days in a year will be different due to the sun's declination.

Sun Path at Different Latitudes. As the sun travels across the sky, the observed altitude angle varies based on the latitude of the observer. The further north or south we go from the equator, the lower the sun's altitude becomes. Figure 6 shows the sun's altitude angle versus the azimuth angle observed at 25° north latitude and 40° north latitude respectively. The sun's altitude angle observed at 25° north latitude is higher than the altitude angle observed at 40° north latitude at the same time. Thus, the sun position inferred from photos taken at the same time but different latitudes will be different.

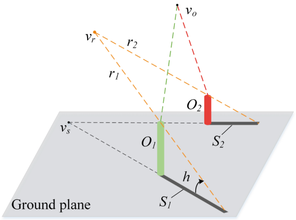


(a) at 25° north latitude in the U.S. (b) at 40° north latitude in the U.S.

Fig. 6. The same path of the sun observed at two latitudes.

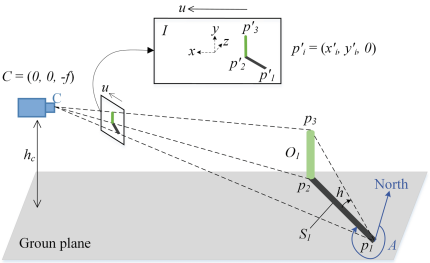
4 Shadow-Inferred Sun Position

The framework AYL uses both the azimuth angle and altitude angle to determine the position of the sun in the sky. As shown in Fig. 4, the sun's altitude angle equals the angle between the shadow and the sun ray, and the sun's azimuth angle equals the angle measured across the shadow point of the top of the column, clockwise from the north to the direction of the shadow. In this section, we provide algorithms to estimate the altitude and azimuth angles of the sun from shadows in a photo. We study two scenarios and two corresponding algorithms to estimate the altitude angle. We also design an algorithm to measure the azimuth angle. For both algorithms, their sensitivities are analyzed.



O_1, O_2 = two vertical objects (e.g. columns)
S_1, S_2 = the resulting shadows

Fig. 7. Estimate the sun's altitude angle with two shadows.



The origin of the *xyz* coordinate is at the center of image *I*.

Fig. 8. Estimate the sun's altitude and azimuth angle with one shadow.

4.1 Estimate Altitude

We consider two scenarios for estimating the sun's altitude angles: (a) photos that contain two vertical objects and their shadows, and (b) photos that contain only one vertical object and its shadow. Vertical objects refer to the ones that are perpendicular to the ground plane.

Two-Shadow Estimation. Figure 7 illustrates the first scenario, where two objects O_1 and O_2 cast shadows S_1 and S_2 on the ground plane, respectively. The sun's altitude angle h is the angle between the shadow and the sun ray. From the graphical perspective, a set of parallel lines in space intersect at one point when they are projected onto a 2-d image. This point is called vanishing point. In Fig. 7, shadows S_1 and S_2 of two vertical objects are parallel in space, and they intersect at vanishing point $\boldsymbol{v_s}$ on the ground plane. Since the sun is far away from the earth, the sun rays r_1 and r_2 can be considered to be parallel and intersect at $\boldsymbol{v_r}$. The sun's altitude angle h can be calculated according to the following formula [11]:

$$h = \arccos\left(\frac{\boldsymbol{v_r}^T \boldsymbol{\omega} \boldsymbol{v_s}}{\sqrt{\boldsymbol{v_r}^T \boldsymbol{\omega} \boldsymbol{v_r}}\sqrt{\boldsymbol{v_s}^T \boldsymbol{\omega} \boldsymbol{v_s}}}\right), \tag{1}$$

Algorithm 1. Estimating the altitude angle h

```
Input: $I$: an image, $f$: the camera's focal length
Output: $h$
$G \leftarrow$ find the equation of the ground plane;
find the equations of lines $\{p'_1p_1,\ p'_2p_2,\ p'_3p_3\}$;
$\{(x_1, y_1, z_1),\ (x_2, y_2, z_2)\} \leftarrow$ compute the coordinates of the points $\{p_1,\ p_2\}$ by solving a set of equations $G$ and $p'_ip_i$ accordingly;
find the equation of line $p_2p_3$
$(x_3, y_3, z_3) \leftarrow$ compute the coordinates of $p_3$ by solving a set of equations $p_2p_3$ and $p'_3p_3$;
$\{\overrightarrow{p_1p_2},\ \overrightarrow{p_1p_3}\} \leftarrow \{(x_2 - x_1, y_2 - y_1, z_2 - z_1),\ (x_3 - x_1, y_3 - y_1, z_3 - z_1)\}$
$h \leftarrow$ compute the angle between $\overrightarrow{p_1p_2}$ and $\overrightarrow{p_1p_3}$ using Eq. 9
return $h$
```

where $\boldsymbol{\omega}$ is called the image of the absolute conic and given by the expression [11,27]:

$$\boldsymbol{\omega} \sim \begin{bmatrix} 1 & 0 & -u_0 \\ 0 & 1 & -v_0 \\ -u_0 & -v_0 & f^2 + u_0^2 + v_0^2 \end{bmatrix}. \tag{2}$$

This expression assumes that the camera has zero skew, the intersection of the optical axis and the image plane is at the center of the image, and the pixels are square. Such assumptions are true for current camera technologies [11,27]. In Eq. 2, (u_0, v_0) denotes the coordinates of the center point of the image, and f denotes the camera's focal length. f is either included in the metadata of the image or can be calculated by the following constraint on $\boldsymbol{\omega}$ with respect to f:

$$\boldsymbol{v_s}^T \boldsymbol{\omega} \boldsymbol{v_o} = 0, \tag{3}$$

where $\boldsymbol{v_o}$ is the vanishing points of the two vertical objects O_1 and O_2. When the objects and their shadows are at perpendicular directions, $\boldsymbol{v_o}$ and $\boldsymbol{v_s}$ will satisfy Eq. 3 [11]. Once we have the coordinates of $\boldsymbol{v_o}$ and $\boldsymbol{v_s}$, we can obtain f by solving Eq. 3.

One-Shadow Estimation. In this scenario, only one vertical object and its shadow are visible in the image. Figure 8 illustrates this scenario where C denotes the camera and I is the image. We assume that the image plane is perpendicular to the ground plane and the direction $\overrightarrow{u}$ is parallel to the ground plane. So the angle between the image plane and the ground plane is 90°. Let's denote the image plane to be $z = 0$, and the coordinate frame is shown in Fig. 8. The center of the image is the origin point $(0, 0, 0)$.

Algorithm 1 describes the steps to measure the altitude angle h given a vertical object and its shadow. Firstly, to find the equation of the ground plane G, we define the distance between the camera and the ground plane to be h_c. As we know G is perpendicular to the XY plane of the coordinate system, the equation of G can be written as:

$$y = -h_c. \tag{4}$$

Next, we compute the equations of the lines p'_1p_1, p'_2p_2 and p'_3p_3. Since the line p'_ip_i passes through the point C $(0, 0, f)$ and the point p'_i whose coordinates can be obtained from the image, it can be described by the two points as:

$$\frac{x}{x'_i} = \frac{y}{y'_i} = \frac{z+f}{f}, \tag{5}$$

where $(x'_i, y'_i, 0)$ are the coordinates of p'_i for $i = 1, 2, 3$.

Lines p'_1p_1 and p'_2p_2 intersect with plane G at points p_1 and p_2 respectively. By solving the Eqs. 4 and 5, the coordinates of p_1 and p_2 can be computed as follows:

$$p_i = (x'_it_i, -h_c, f(t_i - 1)), \tag{6}$$

where $t_i = -\frac{h_c}{y'_i}$ for $i = 1, 2$. Then we have vector $\overrightarrow{p_1p_2} = (x'_2t_2 - x'_1t_1, 0, f(t_2 - t_1))$.

Now, we determine the coordinates of p_3 which is the intersection point of lines p'_3p_3 and p_2p_3. The equation of line p_2p_3 is given by:

$$x = x'_2t_2, \quad z = f(t_2 - 1). \tag{7}$$

By solving the equations of p'_3p_3 and p_2p_3, we can obtain the coordinates of the point p_3:

$$p_3 = (x'_2t_2, y'_3t_2, f(t_2 - 1)). \tag{8}$$

Using the coordinates of p_2 and p_3, we have vector $\overrightarrow{p_1p_3} = (x'_2t_2 - x'_1t_1, h_c + y'_3t_2, f(t_2 - t_1))$. The angle between $\overrightarrow{p_1p_3}$ and $\overrightarrow{p_1p_2}$ is the altitude angle and can be computed as follows:

$$\begin{aligned} h &= \arccos \frac{(\overrightarrow{p_1p_3})^T\overrightarrow{p_1p_2}}{\sqrt{(\overrightarrow{p_1p_3})^T\overrightarrow{p_1p_3}}\sqrt{(\overrightarrow{p_1p_2})^T\overrightarrow{p_1p_2}}}, \\ &= \arccos \frac{m}{\sqrt{m + (y'_3/y'_2 - 1)^2}\sqrt{m}}. \end{aligned} \tag{9}$$

where the intermediate variable $m = (\frac{x'_2}{y'_2} - \frac{x'_1}{y'_1})^2 + f^2(\frac{1}{y'_2} - \frac{1}{y'_1})^2$.

4.2 Estimate Azimuth

To estimate the sun's azimuth angle A from one shadow in an image, we design the following algorithm. The scenario is illustrated in Fig. 8. In particular, the point p_3 is not necessary to be visible for estimating the azimuth angle. Let C be the camera and the unit vector $\overrightarrow{u} = (1, 0, 0)$. The true north N is set to be the reference direction in our algorithm. The orientation of $\overrightarrow{u}$ with respect to N can be obtained by subtracting 90° from the image direction which is included in the metadata of the image.

The sun azimuth angle A equals the angle measured clockwise around point p_1 from due north to the shadow. We calculate A as follows:

$$A = \angle(N, \overrightarrow{u}) + \angle(\overrightarrow{u}, \overrightarrow{p_1p_2}), \tag{10}$$

where $\angle(\overrightarrow{u}, \overrightarrow{p_1p_2})$ denotes the angle measured clockwise from $\overrightarrow{u}$ to $\overrightarrow{p_1p_2}$, and $\angle(N, \overrightarrow{u})$ is the angle measured clockwise from N to $\overrightarrow{u}$, which is the orientation of $\overrightarrow{u}$. $\angle(\overrightarrow{u}, \overrightarrow{p_1p_2})$ is the only unknown variable in Eq. 10.

Next, we define the angle between $\overrightarrow{u}$ and $\overrightarrow{p_1p_2}$ to be α. $\angle(\overrightarrow{u}, \overrightarrow{p_1p_2})$ equals α if it is an acute angle. Otherwise, $\angle(\overrightarrow{u}, \overrightarrow{p_1p_2})$ is equal to $(360° - \alpha)$. The angle α can be calculated as:

$$\alpha = \arccos \frac{\overrightarrow{u}^T \overrightarrow{p_1p_2}}{\sqrt{\overrightarrow{u}^T \overrightarrow{u}} \sqrt{\overrightarrow{p_1p_2}^T \overrightarrow{p_1p_2}}}, \tag{11}$$

where $\overrightarrow{p_1p_2}$ has been calculated in Algorithm 1: $\overrightarrow{p_1p_2} = (x'_2t_2 - x'_1t_1, 0, f(t_2 - t_1))$ and $\overrightarrow{u} = (1, 0, 0)$. Then, we replace $\overrightarrow{u}$ and $\overrightarrow{p_1p_2}$ in Eq. 11 and compute it as:

$$\alpha = \arccos(\frac{(\frac{x'_1}{y'_1} - \frac{x'_2}{y'_2})}{\sqrt{(\frac{x'_1}{y'_1} - \frac{x'_2}{y'_2})^2 + f^2(\frac{1}{y'_1} - \frac{1}{y'_2})^2}}). \tag{12}$$

4.3 Sensitivity Analysis

In this section, we quantify the estimation errors in the computing of the altitude angles and azimuth angles.

Errors of the Altitude Angle Inferred from Two Shadows. The estimation errors of altitude angles stem from the following factors: camera distortion and the detection errors of the objects and shadows. For a well designed camera, the systematic errors (e.g. camera distortion) are constant and can be calibrated if necessary. The detection errors of the shadows and objects in the image can be modeled as random variables. Consequently, the detection errors will result in random errors in the calculation of v_r and v_s. Without loss of generality, we consider the errors of v_r and v_s to be linear to the detection errors and define them to be Δv_r and Δv_s, respectively.

From the graphical perspective, the sun altitude angle h derived from vanishing points v_r and v_s has the geometric meaning as described in Fig. 9. Let C be the camera. The lines Cv_r and Cv_s are parallel to the shadow and the sun ray respectively. h represents the sun altitude angle and equals the angle formed by v_r, C and v_s. The error range of each vanishing point is a circle centered at

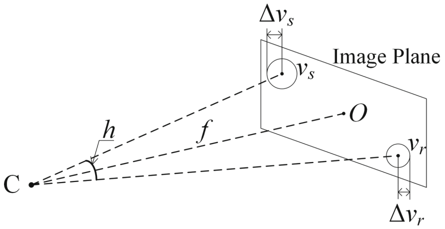


Fig. 9. Errors in the estimated altitude angle with two shadows.

the vanishing point with a radius of the maximum random error. Since Δv_r and Δv_s are small enough compared to the length of $|Cv_r|$ and $|Cv_s|$, they can be considered as two arcs with the center at C. In the worst case, the error Δh of the altitude angle can be calculated as below:

$$\Delta h = (\frac{\Delta v_r}{|Cv_r|} + \frac{\Delta v_s}{|Cv_s|})\frac{180^\circ}{\pi}, \qquad (13)$$

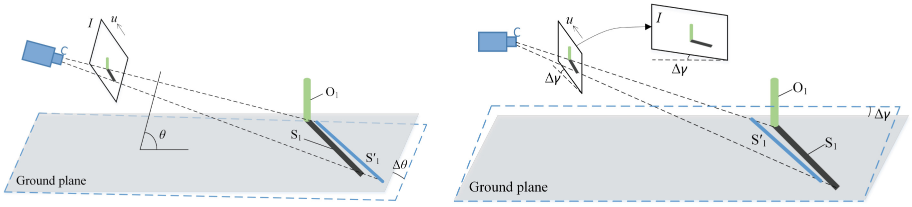
where $|Cv_r|$ and $|Cv_s|$ are the lengths between the camera and the two corresponding vanishing points: v_r and v_s. Thus, the error Δh depends on the random errors Δv_r and Δv_s.

Errors of the Altitude Angle Inferred from One Shadow. The sources of random errors in estimating the altitude angle from one shadow include the slope of the ground and the detection errors of interested object and its shadow. If the ground where the shadow located is not flat and has an error of ΔG with respect to the horizontal plane, ΔG will propagate as the altitude angle is estimated. In addition, the detection errors of the vertical object and its shadow can cause estimation errors. The detection errors can affect the angle estimation depending on the distances from the camera to the object and its shadow. The farther the distance, the larger the uncertainty of the estimated altitude angle. The errors in the altitude angle can be linear to the detection errors.

Errors of the Azimuth Angle. The sources of random errors in estimating the azimuth angle include the camera's orientation errors and the detection errors of shadows. To understand how a camera's orientation affects the estimation error of the angle between $\overrightarrow{u}$ and the shadow S_1, we define θ to be the angle between the image plane and the horizontal ground plane, and γ to be the angle between the camera and the horizontal plane. Assume $\theta = 90^\circ$ and $\gamma = 0^\circ$. And the estimated camera orientation is $\theta = 90^\circ + \Delta\theta$ and $\gamma = 0^\circ + \Delta\gamma$, where $\Delta\theta$ and $\Delta\gamma$ are random errors.

Figure 10 shows the impact of $\Delta\theta$ and $\Delta\gamma$ on the estimated direction of the shadow. First, the error $\Delta\theta$ will be propagated as we estimate the ground plane according to the camera's orientation. And due to this error, the estimated shadow direction will deviate from the true direction of the shadow. The deviation will be $\Delta\theta$ in the worst case. Second, the error $\Delta\gamma$ will also be propagated to the estimated ground plane. And this error will lead to a deviation in the estimated shadow direction as well, which is $\Delta\gamma$ in the worst case. In summary, the estimated shadow direction deviates from its true direction at most $\Delta\theta + \Delta\gamma$, which can produce $\Delta\theta + \Delta\gamma$ error in the estimated azimuth angle in the worst case.

In summary, we find three main sources of the errors: the detection errors of the objects and their shadows, the ground slope, and the camera's orientation errors. In general, the estimation errors of the sun position are linear to the three types of errors. The detection errors in our algorithms can be reduced by choosing the objects and shadows that are clear enough and using effective image detection algorithms. The errors caused by the slope of the ground will not be



(a) Effect caused by $\varDelta\theta$

(b) Effect caused by $\varDelta\gamma$

Fig. 10. Effects caused by random errors in camera's orientation

greater than the slope angle and can be reduced greatly by measuring this angle. In addition, the camera's orientation errors can be reduced using inertial sensors to obtain the camera orientation.

5 Metadata-Inferred Sun Position and Validation

In this section, we describe the process to validate the consistency of a photo's capture time and location. The key idea is the following: we calculate the sun position using the capture time and location in the metadata of images. If the capture time and location are true, the sun position will match the one we estimated from shadows.

5.1 Metadata-Inferred Sun Position

As mentioned in Sect. 3, the position of the sun depends on the time of day, the date and the location of the observer. Its movement across the sky obeys the rules that have been studied in astronomy. In this section, we discuss the astronomical algorithms that are used to calculate *metadata-inferred sun position*, given the time and location.

We refer the time of day as the local time based on the standard time offsets of Coordinated Universal Time (UTC). However, the local standard time doesn't provide an intuitive connection with the sun position. In astronomy, the solar time is often used to discuss the sun position. It works because the sun finishes a 360° rotation around the celestial sphere every 24 h. The completed journey is divided into 24 h, and one solar hour means that the sun travels a 15° arc [19]. The instant when the sun is due south in the sky or the shadow points to exactly north is called solar noon, which is 12:00 for solar time. Every 15° arc the sun travels, one hour is added to 12:00 under the 24-h clock system, and the angle distance that the sun passes on the celestial sphere is defined as the hour angle H [19]. It is measured from the sun's solar noon position, and ranges from 0° to +180° westwards and from 0° to −180° eastwards. The conversion between the local standard time t_l to the solar time t_s is as follows [13,24]:

$$t_s = t_l + ET + \frac{4\ min}{deg}(\lambda_{std} - \lambda_l), \tag{14}$$

where λ_l denotes the local longitude, and λ_{std} is the local longitude of standard time meridian, and ET stands for the equation of time, which describes the difference of the true solar time and the mean solar time [13]. The sun's hour angle is calculated as follows:

$$H = 15^\circ(t_s - 12). \tag{15}$$

Using the observer's local horizon as a reference plane, the azimuth and altitude angles of the sun can be calculated as follows [24]:

$$\tan(A) = \frac{\sin H}{\sin\varphi\cos H - \cos\varphi\tan\delta}, \tag{16}$$

$$\sin(h) = \sin\delta\sin\varphi + \cos\varphi\cos\delta\cos H, \tag{17}$$

where φ is the latitude of the observer's location, and δ is the sun's declination angle and it can be calculated as below [15,24]:

$$\delta = -23.44^\circ\cos(\frac{360^\circ(N+10)}{365^\circ}), \tag{18}$$

where N is the number of days since January 1st. Note that the azimuth angle A calculated in Eq. 16 uses south as a reference. We can derive the azimuth angle according to its definition in Sect. 3.

5.2 Consistency Validation

Once obtaining the *shadow-inferred sun position* and *metadata-inferred sun position*, we check the difference between these two estimations by comparing their altitude angles and azimuth angles respectively. However, since there exists random and systemic errors in the *shadow-inferred sun position*, the estimation may not equal the "true" sun position. Thus, we have to select a threshold that is large enough to tolerate the errors yet small enough to detect the inconsistency between the *shadow-inferred sun position* and *metadata-inferred sun position*. Intuitively, the closer these two sun positions are to each other, the more likely the capture time and location are true.

We define the altitude angles of *shadow-inferred sun position* and *metadata-inferred sun position* to be h_s and h_m respectively, and the corresponding azimuth angles to be A_s and A_m. Then the distance of the two altitude angles is $d_h = |h_s - h_m|$, and the distance of the two azimuth angles is computed as $d_A = |A_s - A_m|$. The likelihood of the consistency is inversely proportional to d_h and d_A. However, the effects on d_h and d_A caused by fake capture time and/or location are different. For example, modifying the capture time from 12:00 p.m. to 13:00 p.m. may lead to 10° in d_A but only 2° in d_h. So two different thresholds for d_h and d_A have to be selected. The capture time and location are considered to be true only when both d_h and d_A are within the thresholds. Besides, the sun position can be described by a pair of azimuth angle and altitude angle: (A, h).

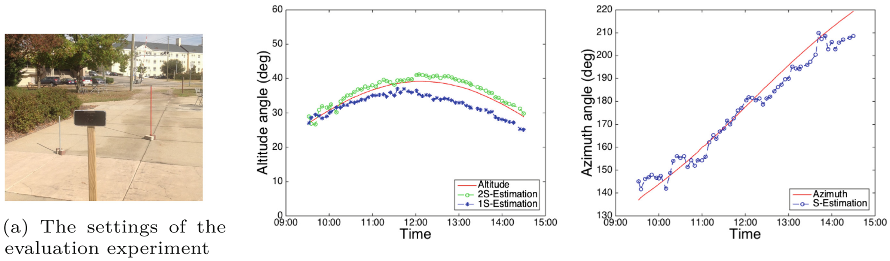
We can also use the sun position distance that is computed as $d_p = \sqrt{d_A^2 + d_h^2}$ to distinguish the two estimations of the sun position. Our goal is to choose appropriate variables and thresholds that can increase the probability of correct validation for inconsistent images and decrease the probability of false validation for consistent images. Section 6 details the selection of thresholds in the validation experiment.

6 Evaluation

This section presents the results of our experiments. To evaluate the performance of the sun position estimation algorithms, we conducted an experiment on November 8, 2016 in the U.S. and collected 60 photos. To validate the effectiveness of the framework AYL, we gathered 124 photos in China and the U.S in the span of four months, and examined whether we can detect the modifications of capture time, date and location.

6.1 Sun Position Estimation

To evaluate the accuracy of our sun position estimation algorithms, we collected 60 photos using the rear camera of an iPhone 7 from 9:30 a.m. to 14:30 p.m. at an interval of about 5 min on November 8, 2016 in Columbia, SC. As shown in Fig. 11(a) we set up the experiment in a relatively ideal situation: we place two columns (the red one and the grey one) on the ground vertically, and fixed the iPhone 7 on another vertical stick to take photos of these two columns and their shadows. Figure 11(b) shows the estimated altitude angles by applying the **two-shadow estimation** and **one-shadow estimation** algorithms to the photos. The ground truth sun positions are calculated using the astronomical algorithms in Sect. 5.1 according to the real time, latitude and longitude. The ground truth altitude angles are labeled with red and denoted as "Altitude". The other two



(a) The settings of the evaluation experiment

(b) Comparison of estimated altitude angle to ground truth.

(c) Comparison of estimated azimuth angle to ground truth.

Fig. 11. The experiment setting is shown in (a). And (b) presents the comparison of estimated altitude angle to ground truth altitude angle. (c) shows the comparison of estimated azimuth angle to ground truth azimuth angle.

curves in Fig. 11(b) represent the estimated altitude angles inferred from shadows. “2S-Estimation” is obtained by applying the **two-shadow estimation** algorithm, while “1S-Estimation” is plotted by applying the **one-shadow estimation** algorithm. We find that the average error in “2S-Estimation” is 1.43°, while it is 2.98° in “1S-Estimation”. Figure 11(c) presents the estimated azimuth angles versus the ground truth azimuth angles. The curve in red is plotted using the ground truth azimuth angles, while the curve in blue is plotted using the data of estimated azimuth angles. The average error is approximately 4.3°.

The estimation errors of sun positions are mainly contributed by three factors. First, due to the ground slope and the camera’s orientation, the image plane may not be precisely perpendicular to the ground plane, which causes errors. The second type of errors is random one that is introduced when extracting objects and shadows from the photos. Finally, errors can be created by the measurement drift of the compass over time. Due to the nature of the two algorithms, these types of errors will have different levels of impact on them. Figure 11(b) indicates that the **two-shadow estimation** algorithm outperforms the **one-shadow estimation** algorithm. It is partly because the **one-shadow estimation** algorithm is more sensitive to the ground slope. We believe that given the measurement of the slope, we shall be able to reduce the error. In summary, the algorithms in Sect. 4 are able to infer the sun position, either from two vertical objects and their shadows or from one object and its shadow.

6.2 Consistency Validation

To evaluate the performance of `AYL` and to understand threshold selection, we conducted a set of experiments.

Dataset. The data in this experiment was captured at 15 cities around the USA and China since September 2016. Our dataset consists of 124 photographs taken by 10 iPhones, including iPhone 5s, 6, 6 plus, 6s, 6s plus and 7. 61 out of the 124 photos were taken in China. Each photo encloses the metadata that includes the real capture time and location. 72 out of the 124 photos contain at least two vertical objects and their shadows, while 52 photos only contain one vertical object and its shadow. Our dataset mainly contains three types of vertical objects: standing people, poles (e.g. road signs, lampposts) and tree trunks. We chose these objects because they are common in reality and are mostly vertical to the ground. Our experimental results confirm that our algorithms work well on these objects. We refer to the true metadata of the 124 photos as the *positive samples.*

We generate the attack data by falsifying the metadata of the 124 photos. Note that multiple types of metadata may result in the same effect. For instance, modifying longitude one degree more to the west has the same effect on the sun position as changing the local time forward by four minutes. Thus, falsifying either longitude or the local time is equivalent. To simplify the analysis yet without loss of generality, we focus on three types of attacks that modify the following metadata:

- The falsified time of day, and true date and location.
- The falsified date, and true time and location.
- The falsified latitude of location, and true time and date.

We refer to the attack metadata as the *negative samples*. We have 124 negative samples for each type of attack metadata, The "fake" times of day are randomly generated in the range from 8:00 a.m. to 17:00 p.m. when the sun is likely to be seen. The "fake" dates are randomly generated from the range within one year. The "fake" latitudes of location are randomly generated in the range of 25° and 50° of the Northern Hemisphere where most of the U.S. and China locate. Here we didn't consider the attack data with falsified longitude. Because the result produced by only falsifying longitude can be equivalent to the result caused by falsifying the time of day accordingly.

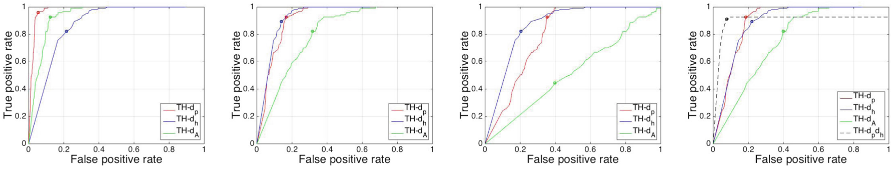
Metric. We use ROC curves to evaluate the performance of `AYL` by varying thresholds for our system. An ROC curve represents Receiver Operating Characteristic curve and is created by plotting true positive rate (TPR) against false positive rate (FPR), as the threshold varies [6]. The true positive rate and false positive rate are defined as below.

$$TPR = \frac{\#\text{ of true positives}}{\#\text{ of (true positives + false negatives)}} = \frac{\#\text{ of true positives}}{\#\text{ of positives}}$$
$$FPR = \frac{\#\text{ of false positives}}{\#\text{ of (true negatives + false positives)}} = \frac{\#\text{ of false positives}}{\#\text{ of negatives}}$$

where a true positive denotes the result that a positive sample is correctly identified as such, and a false positive is the one that a negative sample is identified as a positive sample by mistakes. The point $(0, 1)$ on the ROC curve denotes 0 FPR and 100% TPR, which indicates an ideal system that can correctly identify all genuine photos and reject all falsified photos [25]. In our experiment, we select the optimal threshold as the one that yields the minimum distance from the corresponding point on the ROC curve to the ideal point $(0, 1)$. Another indicator that we use to evaluate the average performance of the validation is the area under the ROC curve (AUC). The closer it is to 1, the better the average performance is [6].

Performance and Threshold Selection. Based on the framework `AYL`, we performed consistency validation using the three types of attack metadata. To understand how the altitude angle and azimuth angle influence the performance of the validation, we examine three distances separately: the distance of the altitude angles d_h, the distance of the azimuth angles d_A, and the distance of the sun positions d_p. Here, the sun position is defined to be (A, h), in which A refers to the azimuth angle and h refers to the altitude angle. To decide the best distance variable which can yield the maximum AUC and the optimal threshold of the variable, we analyze the ROC curves that are plotted by varying the threshold of each type of distance.

The results are presented in the set of ROC curves shown in Fig. 12. Each ROC curve with distinct color is plotted by varying the threshold of one type



(a) The attack metadata with falsified time of day.

(b) The attack metadata with falsified date.

(c) The attack metadata with falsified latitude.

(d) A collective of the three types of attack metadata.

Fig. 12. ROC curves based on different distance variables and different types of attack metadata.

of the three distances. "$\mathrm{TH}-\mathrm{d_p}$" denotes varying the threshold of the sun position distance d_p. "$\mathrm{TH}-\mathrm{d_h}$" and "$\mathrm{TH}-\mathrm{d_A}$" denote varying the threshold of the altitude angle distance d_h and the azimuth angle distance d_A respectively. For each type of attack metadata, the randomly generating of 124 negative samples is repeated 5 times. Each false positive rate on the ROC curve is averaged over these repeated attack metadata. Figure 12(a) indicates that the detection based on d_A slightly outperforms the one based on d_h, to detect the attacks that falsify the photo's time of day. However, the d_h based detection achieves better performance in detecting the other types of attacks as shown in Fig. 12(b–c), especially in detecting falsified latitude. The result implies that d_h is more important in distinguishing different positions of the sun compared to d_A in general. Such a conclusion confirms with the result reported in Sect. 6.1, i.e., the average estimation error of the altitude angles is smaller than that of the azimuth angles. If only d_h is used for consistency validation, Fig. 12(d) guides us to choose the optimal threshold of d_h to be 3° and it achieves combined $(TPR,\ FPR)$ values of (89.5%, 22%), which means that 89.5% of positive samples can be correctly validated but 22% of negative samples will be mistakenly identified. In addition, Fig. 12(a–c) shows that the d_p based detection achieves the best performance in detecting falsified time, and has almost the same performance as the d_h based detection in detecting the other types of attacks. Once we only use d_p for consistency validation as shown in Fig. 12(d), we choose the optimal threshold of d_p to be 9.2°, which achieves combined $(TPR,\ FPR)$ values of $(92.7\%, 18.6\%)$ for all attacks.

To improve the performance further, we examine both the d_p and d_h to validate the consistency of time and location. That is, a sample has to satisfy both the thresholds of d_h and d_p to be accepted by `AYL`. Plotting the ROC curves and finding the global optimal thresholds by varying two thresholds can be tricky. Thus, we chose the local optimal threshold for one variable and varied the other threshold to plot the ROC curve. This approach may not generate the global optimal thresholds for the two variables, but it strikes a balance between the optimum and the computational cost. We chose the threshold of d_p to be 9.2° and varied the threshold of d_h. The resulting curve illustrates an improved performance than the one of using a single threshold as shown in Fig. 12(d).

Note that we cannot plot an integral ROC curve when the threshold of d_p is fixed since the highest true positive rate will be decided by the fixed threshold, which is 92.7%. The curve "TH – $\mathrm{d_p d_h}$" in Fig. 12(d) indicates that choosing the optimal threshold of d_h to be 4.8° can correctly identify 91.1% positive samples, and cannot identify 7.7% of negative samples.

Attacks Against `AYL`. Based on the above results, we analyze the robustness of the framework `AYL` when falsifying one of the three parameters—time of day, date and latitude of location, and falsifying more than one parameters. `AYL` cannot detect the falsifications that do not cause violations of both the thresholds of the altitude angle distance and the sun position distance. If an attacker modifies both the time and location of a photo such that the altitude angle and the sun position are within the thresholds, then the modification can fool `AYL`. Luckily. the motivation of falsifying the metadata of a photo is to use it for a chose event and the attacker may not be guaranteed to find such a combination.

Our framework can detect that the image shown in Fig. 1 was not taken at the claimed date and location. Although we do not have the required metadata (e.g., the time of day, the image direction and the camera orientation) and cannot estimate the azimuth angle as well as the exact sun position, we can estimate the altitude angle from the image. Given that the photo was claimed to be taken on or before January 16th in Florida, we can calculate the possible maximum altitude angle between January 1st and 16th to be 41°. Based on the image, we estimate the focal length to be 1287 pixels and the altitude angle to be 47.6°. The distance between the two estimates will be 6.6° which is larger than the threshold 4.8° in our experiments. Thus, we conclude that the date and location of this image were spoofed.

6.3 Discussions and Limitations

When estimating the altitude angle using the **one-shadow estimation** algorithm, an integrated vertical object and its shadow are required. However, vertical objects in the real world may not be absolutely vertical. By examining the scenario, we find that the direction of the sun ray is determined by a point on the object and the resulting point on the shadow. Even if the object is not exactly vertical, these two points are still able to decide the path of the sun ray and the altitude angle can be obtained from the sun ray and the shadow. Thus, we believe that our algorithm can eliminate this requirement, and `AYL` may not require to have the entire object in the photo if there exists a distinct point on the object.

In this paper, we assume that the camera is perpendicular to the ground and looks front horizontally or vertically when taking photos. Such assumption is used to simplify the algorithms for estimating the sun position. In fact, most smartphotnes are equipped with inertial sensors that have been widely used to estimate the orientation of the smartphone. If the sensor data is enclosed in the metadata, the device orientation can be obtained and used to determine the relationship between the device and the ground as well as the shadows.

A direction of future work is to estimate the sun position regardless of how cameras are oriented when taking photos.

7 Conclusion

We presented a new framework AYL which uses two estimations of sun position—*shadow-inferred sun position* and *metadata-inferred sun position*—to check whether the capture time and location of an outdoor image are true. Our framework exploits the relationship between the sun position in the sky and the time and location of an observer. We designed algorithms to obtain *shadow-inferred sun position* using only one vertical object and its shadow in the image. Our experiments show that the algorithms can estimate the sun position from shadows in the image with satisfactory accuracy. AYL utilizes both the altitude angle and azimuth angle for the consistency validation. The evaluation results guide us to choose the thresholds of altitude angle distance and sun position distance to be 4.8° and 9.2° respectively, which achieves combined $(TPR,\ FPR)$ values of $(91.1\%, 7.7\%)$ for the consistency validation. We believe that our results illustrate the potential of using sun position to validate the consistency of the capture time and location. Our work raises an open question that whether other image contents can be leveraged for validating the consistency of image's contextual information.

References

1. Fema now has a rumor control section for misinformation. https://twitter.com/fema/status/264800761119113216
2. Bayram, S., Sencar, H.T., Memon, N.: An efficient and robust method for detecting copy-move forgery. In: IEEE ICASSP 2009, pp. 1053–1056. IEEE (2009)
3. Boididou, C., Papadopoulos, S., Kompatsiaris, Y., Schifferes, S., Newman, N.: Challenges of computational verification in social multimedia. In: ACM WWW 2014, pp. 743–748. ACM (2014)
4. Carvalho, T., Farid, H., Kee, E.: Exposing photo manipulation from user-guided 3D lighting analysis. In: SPIE/IS&T Electronic Imaging, p. 940902. International Society for Optics and Photonics (2015)
5. Dearden, L.: The fake refugee images that are being used to distort public opinion on asylum seekers, September 2015. http://www.independent.co.uk/news/world/europe/the-fake-refugee-images-that-are-being-used-to-distort-public-opinion-on-asylum-seekers-10503703.html
6. Fawcett, T.: An introduction to ROC analysis. Pattern Recogn. Lett. **27**(8), 861–874 (2006)
7. Ferrara, E.: Manipulation and abuse on social media by Emilio Ferrara with Ching-man Au Yeung as coordinator. ACM SIGWEB Newslett. (Spring), Article No. 4 (2015)
8. Fridrich, J., Soukal, D., Lukas, J.: Detection of copy move forgery in digital images. In: Digital Forensic Research Workshop, August 2003

9. Garg, R., Varna, A.L., Wu, M.: Seeing ENF: natural time stamp for digital video via optical sensing and signal processing. In: ACM MM 2011, pp. 23–32. ACM (2011)
10. Gupta, A., Lamba, H., Kumaraguru, P., Joshi, A.: Faking sandy: characterizing and identifying fake images on twitter during hurricane sandy. In: ACM WWW 2013, pp. 729–736. ACM (2013)
11. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2004)
12. Harvey, P.: Exiftool. http://www.sno.phy.queensu.ca/~phil/exiftool/
13. Holbert, K.E., Srinivasan, D.: Solar energy calculations. In: Handbook Of Renewable Energy Technology, pp. 189–204 (2011)
14. Imran, M., Elbassuoni, S.M., Castillo, C., Diaz, F., Meier, P.: Extracting information nuggets from disaster-related messages in social media. In: Proceedings of ISCRAM, Baden-Baden, Germany (2013)
15. Iqbal, M.: An Introduction to Solar Radiation. Elsevier, Amsterdam (2012)
16. Japan Electronics and Information Technology Industries Association: Exchangeable image file format for digital still cameras: Exif Version 2.2, April 2002
17. Junejo, I.N., Foroosh, H.: Estimating geo-temporal location of stationary cameras using shadow trajectories. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008. LNCS, vol. 5302, pp. 318–331. Springer, Heidelberg (2008). doi:10.1007/978-3-540-88682-2_25
18. Kakar, P., Sudha, N.: Verifying temporal data in geotagged images via sun azimuth estimation. IEEE Trans. Inf. Forensics and Secur. **7**(3), 1029–1039 (2012)
19. Karttunen, H., Kroger, P., Oja, H., Poutanen, M., Donner, K.J.: Fundamental Astronomy, 5th edn. Springer, Heidelberg (2007)
20. Kee, E., O'brien, J.F., Farid, H.: Exposing photo manipulation from shading and shadows. ACM Trans. Graph. **33**(5), 165:1–165:21 (2014)
21. Metadata Working Group: Guidelines For Handling Image Metadata v2.0, November 2010
22. National Information Standards Organization: Understanding Metadata (2004)
23. Pan, X., Lyu, S.: Detecting image region duplication using sift features. In: IEEE ICASSP 2010, pp. 1706–1709. IEEE (2010)
24. Savoie, D.: Sundials: Design, Construction, and Use. Praxis Publishing, Chichester (2009)
25. Tian, J., Qu, C., Xu, W., Wang, S.: Kinwrite: handwriting-based authentication using kinect. In: NDSS 2013 (2013)
26. Tsai, T.H., Jhou, W.C., Cheng, W.H., Hu, M.C., Shen, I.C., Lim, T., Hua, K.L., Ghoneim, A., Hossain, M.A., Hidayati, S.C.: Photo sundial: estimating the time of capture in consumer photos. Neurocomputing **177**, 529–542 (2016)
27. Wu, L., Cao, X.: Geo-location estimation from two shadow trajectories. In: IEEE CVPR 2010, pp. 585–590 (2010)

Lights, Camera, Action! Exploring Effects of Visual Distractions on Completion of Security Tasks

Bruce Berg, Tyler Kaczmarek(✉), Alfred Kobsa, and Gene Tsudik

University of California Irvine, Irvine, CA, USA
{bgberg,tkaczmar,kobsa}@uci.edu, gts@ics.uci.edu

Abstract. Human errors in performing security-critical tasks are typically blamed on the complexity of those tasks. However, such errors can also occur because of (possibly unexpected) sensory distractions. A sensory distraction that produces negative effects can be abused by the adversary that controls the environment. Meanwhile, a distraction with positive effects can be artificially introduced to improve user performance.

The goal of this work is to explore the effects of visual stimuli on the performance of security-critical tasks. To this end, we experimented with a large number of subjects who were exposed to a range of unexpected visual stimuli while attempting to perform Bluetooth Pairing. Our results clearly demonstrate substantially increased task completion times and markedly lower task success rates. These negative effects are noteworthy, especially, when contrasted with prior results on audio distractions which had positive effects on performance of similar tasks. Experiments were conducted in a novel (fully automated and completely unattended) experimental environment. This yielded more uniform experiments, better scalability and significantly lower financial and logistical burdens. We discuss this experience, including benefits and limitations of the unattended automated experiment paradigm.

1 Introduction

It is widely believed that the human user is the weakest link in the security chain. Nonetheless, human participation is unavoidable in many security protocols. Such protocols require extensive usability testing, since users are unlikely to perform well when faced with overly difficult or intricate tasks. Typically, security-related usability testing entails evaluating human performance in a "best-case" scenario. In other words, testing is usually conducted in sterile lab-like environments.

At the same time, security protocols involving human users have become more commonplace. Examples include activities, such as: (1) using a personal device for verification of transaction amounts, (2) entering a PIN or a password and (3) solving a CAPTCHA, (4) comparing PINs when pairing Bluetooth devices, and (5) answering personal security questions.

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 124–144, 2017.
DOI: 10.1007/978-3-319-61204-1_7

Since overall security of these tasks is determined by the human user (as the weakest link), extensive usability studies have been conducted. They aimed to assess users' ability to perform security tasks correctly and without undue delays, while providing an acceptable level of security [5,9,11,17].

However, the focus on maximizing successful protocol completion led developers to evaluate usability under contrived and unrealistic settings. In practice, security tasks can take place in noisy environments. In real-world settings, users are often exposed to various sensory stimuli. The impact of such stimuli on performance and completion of security tasks has not been well studied. A particular stimulus (e.g., a fire alarm or flickering lights) can be unintentional or hostile, i.e., introduced by the adversary that controls the physical environment. Furthermore, recent emergence of Internet of Things (IoT) devices (such as smart speakers and light fixtures) in home and office settings creates environments where compromised (malware-infected) devices can expose users to a variety of visual and audio stimuli.

There has been just one prior study that studied the effects of stimuli on the completion of security-critical tasks. It showed that introduction of unexpected audio stimuli during Bluetooth pairing actually improved subject performance [8]. This initial result, though interesting, motivates a more thorough study in order to fully understand the effects of a range of unexpected (and potentially malicious) stimuli.

Since modern user-aided security protocols focus on maximizing successful outcomes in an ideal environment, human errors are quite rare. For example, Uzun et al. [22] assume that:

> "...[A]ny non-zero fatal error rate in the sample size of 40 is unacceptable for security applications."

Consequently, numerous trials with many subjects are needed to gather data sufficient for making claims about human error rates. The scale is further exacerbated by the need to test multiple modalities, each with a distinct set of subjects. (This is because a given subject is less likely to make a similar mistake twice, even under different conditions.) Therefore, the number of required participants can quickly grow into hundreds, which presents a logistical challenge. To ease the burden of conducting a large-scale study, we designed and employed an entirely unattended and automated experimental setup, wherein subjects receive recorded instructions from a life-sized projection of a video-recorded experimenter ("avatar"), instead of a live experimenter.

We extensively experimented with subjects attempting to pair two Bluetooth devices (one of which was the subject's own device) in the presence of various unexpected visual stimuli. We tested a total of 169 subjects in the fully unattended experiment setting.[1] We initially hypothesized that visual stimuli would have beneficial or facilitatory effects on subject task completion, as was recently experienced with its audio counterpart [8]. Surprisingly, we discovered a marked

[1] All experiments described in this paper were fully authorized by the Institutional Review Board (IRB).

slowdown in task completion times across the board, and lower task success rates under certain stimuli.

The rest of the paper is organized as follows: The next section overviews related work and background material. Then, we present the design and setup of our experiments, followed by the presentation of our experimental results. Next, we derive conclusions and summarize lessons learned. The paper concludes with the discussion of limitations of our approach and directions for future work. Appendix 1 presents and analyzes performance of subjects arriving in groups. Appendix 2 contains the description of color spaces used to generate our stimuli. Details on the unattended experiment setup are in Appendix 3.

2 Background and Related Work

This section overviews related work in automated experiments, and human-assisted security methods. We also provide background information in psychology, particularly effects of sensory arousal on task performance, as well as effects of visual stimuli on arousal level and emotive state.

2.1 Automated Experiments

Other than recent results describing effects of audio distractions [8], we are unaware of any prior usability studies utilizing a fully automated and unattended physical environment.

However, some prior work reinforces validity of virtually-attended remote experiments and unattended online surveys, in contrast with same efforts in a traditional lab-based setting. Ollesch et al. [18] collected psychometric data in: (1) a physically attended experimental lab setting and (2) its virtually attended remote counterpart. No significant differences were found. This is further reinforced by Riva et al. [21] who compared data collected from (1) unattended online, and (2) attended offline, questionnaires. Finally, Lazem and Gracanin [14] replicated two classical social psychology experiments where both the participants and the experimenter were represented by avatars in Second Life[2], instead of being physically co-present. Here too, no significant differences were observed.

2.2 User Studies of Secure Device Pairing

Secure device pairing (mostly, but not only, via Bluetooth) has been extensively researched by experts in both security and usability. While initially pairing, the two devices have no prior knowledge of one another, i.e., there is no prior security context. Also, they can not rely on either a Trusted Third Party (TTP) or a Public Key Infrastructure (PKI) to facilitate the protocol. This makes device pairing especially vulnerable to man-in-the-middle (MiTM) attacks. This prompted the

[2] See secondlife.com.

design of numerous protocols requiring human involvement (integrity verification) over some out-of-band (OOB) channel, e.g., visual or audio comparison or copying/entering numbers.

For example, Short Authenticated String (SAS) protocols ask the user to compare two strings of about 20 bits each [13].

Uzun et al. [22] performed the first usability study of Bluetooth pairing techniques using SAS. It determined that the "compare-and-confirm" method – which involves the user comparing two 4-to-6-digit decimal numbers and indicating a match or lack thereof – was the most accurate and usable approach.

Kobsa et al. [11] compiled a comprehensive comparative usability study of eleven major secure device pairing methods. They measured task performance times, completion times, completion rates, perceived usability and perceived security. This led to the identification of most problematic as well as most effective pairing methods, for various device configurations.

Goodrich et al. [5] proposed an authentication protocol that used "Mad-Lib" style SAS. Each device in this protocol creates a nonsensical phrase based on the protocol outcome, and the user then determine if the two phrases match. This approach was found to be easier for non-specialist users.

Kainda et al. [9] examined usability of device pairing in a group setting. In this setting, up to 6 users tried to connect their devices to one another by participating in a SAS protocol. It was found that group effort decreased the expected rate of security and non-security failures. However, if a single individual was shown a SAS different from that of all others participants, the former often lied about the SAS in order to fit in with the group, demonstrating so-called "insecurity of conformity."

Gallego et al. [4] discovered that subject's performance in secure device pairing could be improved if it were to be scored. In other words, notifying subjects about their performance score resulted in fewer errors.

2.3 Effects of Sensory Stimulation

Sensory stimulation has variable impact on task performance. This is due to many contributing factors, including the subject's current level of arousal. The Yerkes-Dodson Law stipulates an inverse quadratic relationship between arousal and task performance [2]. It implies that, across all contributing stimulants, subjects who are either at a very low, or very high, level of arousal are not likely to perform well, and there exists an optimal level of arousal for correct task completion.

An extension to this law is the notion that completion of less complex tasks that produce lower levels of initial arousal in subjects benefits from inclusion of external stimuli. At the same time, completion of complex tasks that produce a high level of initial arousal suffers from the inclusion of external stimuli. Hockey [7] and Benignus et al. [1] classified this causal relationship by defining the complexity of a task as a function of the task's event rate (i.e., how many subtasks must be completed in a given time-frame) and the number of sources that originate these subtasks. External stimulation can serve to sharpen the focus of a subject at a low arousal level, improving task performance [19]. Conversely, it can overload subjects that are already at a high level of arousal, and induce errors in task completion [6].

O'Malley and Poplawsky [20] argued that sensory noise affects behavioral selectivity. Specifically, while a consistent positive or negative effect on task completion may not occur, a consistent negative effect was observed for tasks that require subjects to react to signals on their periphery. Meanwhile, a consistent positive effect on task completion was observed for tasks that require subjects to react to signals in the center of their field of attention. This leads to the claim that sensory stimulation has the effect of narrowing the subject's area of attention.

2.4 Unique Effects of Visual Stimuli

In addition to being general external stimuli that serve to raise arousal level, visual stimuli, particularly colors, have social and emotional implications. Naz and Epps [15] surveyed 98 college students about their emotional responses to five principal hues (red, blue, purple, green and yellow), five intermediate hues (yellow-red, green-yellow, blue-green, and red-purple) as well as three achromatic colors (white, gray, and black.) They found that principal hues are more likely to foster positive emotive responses. Furthermore, different colors within each group induce differing levels of arousal: some (red or green-yellow) increase arousal, while others (blue and green) are perceived as relaxing.

Moreover, visual stimuli were found to be dominating in multi-sensory contexts. Eimer [3] showed that in experiments with tactile, visual, and audio stimuli, subjects overwhelmingly utilized visual queues to localize tactile and auditory events.

3 Methodology

This section describes our experimental setup, procedures and subject parameters.

3.1 Apparatus

The experimental setting was designed to facilitate fully automated experiments with a wide range of sensory inputs. We located the experiment in a public, but low-traffic alcove at the top floor of the Computer Science Department building in a large public university. Figure 1 shows our setup from the subject's perspective (front view), and Fig. 2 depicts it from the side. More photos can be found in Appendix 2. The setup is comprised of readily available off-the-shelf components:

- A 60"-by-45" touch-sensitive interactive Smartboard (See footnote 2) whiteboard with a Hitachi CP-A300N short-throw projector (See footnote 2). The Smartboard acts as both an input and a display device. It reacts to tactile input, i.e., the user touches its surface, similar to a large touch-screen.

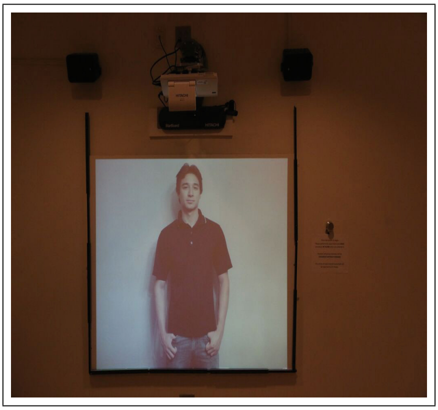


Fig. 1. Experimental environment: subject's perspective

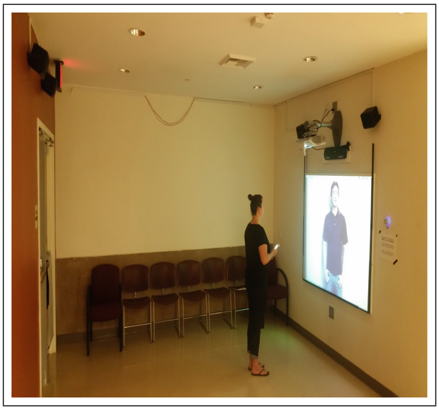


Fig. 2. Experimental environment: side view

- A Logitech C920 HD Webcam (See footnote 2).
- Two pairs of BIC America RtR V44-2 speakers (See footnote 2): one alongside the smartboard, and the other – on the opposite wall. Their arrangement is such that the subject is typically standing in the center of the four speakers.
- Four programmable wirelessly controllable Phillips Hue A19 LED lightbulbs[3] to deliver the visual stimuli.

device. All prospective subjects were explicitly informed, during recruitment, that they would need to use their own personal device that supports Bluetooth communication. We could have instead provided a device to the subjects, which might have fostered a more uniform subject experience. However, there would have been some drawbacks:

- We wanted to avoid accidental errors due to the use of an unfamiliar device that might have a different user interface from that of the subject's own device. Mitigating this unfamiliarity would have required some training, which is incompatible with the unattended experiment setting.
- Virtually all current Bluetooth pairing scenarios involve at least one of the devices being owned by the person performing the pairing. Forcing subjects to use our device would have resulted in a more contrived or synthetic experience.
- From a purely practical perspective, an unattended portable device provided by us would have been more prone to damage or theft than other components, which are bulky and attached to walls and/or ceilings.

Not surprisingly, the majority of subjects' devices (152 out of 169) were smartphones. Tablets (13) and laptops (4) accounted for the rest.

[3] See: meethue.com for Hue Bulbs, smarttech.com for the Smartboard, logitech.com for the Webcam, bicamerica.com for speakers, and hitachi.com for the projector.

Bluetooth pairing is not as common as other security-critical tasks, such as password entry or CAPTCHA solving. However, we believe that Bluetooth pairing is the ideal security-critical task for the unattended experiment setup. It is preferred to passwords and PINs since it does not require subjects to reveal existing, or to select new, secrets. The security task at the core of Bluetooth pairing involves the user comparing two 6-digit decimal numbers – one displayed by each device being paired – and pressing a single button. This is a much more discrete and uniform activity than solving CAPTCHA-s, which vary widely in terms of difficulty and require higher-resolution displays as well as more extensive user input. These factors, even without external stimuli, would yield large variations in error rates and completion times.

3.2 Procedures

As mentioned earlier, instead of a live experimenter, we used a life-size video/audio recording of a experimenter giving instructions. This avatar is the subjects' only source of information about the experiment. Actual experimenter involvement is limited to strictly off-line activities, such as infrequent recalibration of avatar video volume and visual effects, as well as occasional repair of some components that suffered minor wear-and-tear damage throughout the study. This unattended setup allows the experiment to run without interruption 24/7 over a 5 month period.

Recall that the central goal of the experiment is to measure performance of subjects who attempt to pair their personal Bluetooth device to our Bluetooth device – an iMAC that uses the SmartBoard as an external display. This iMAC is hidden from the subject's view; it is situated directly on the other side of the SmartBoard wall in a separate office. During the pairing process, each subject is exposed to one randomly selected (from a fixed set) visual stimulus. This is done by rapid change in the ambient lighting of the room's four overhead lightbulbs to the chosen stimulus condition.

The experiment runs in four phases:

1. Initial: the subject walks in, presses a button on the wall which activates the experiment. Duration: instant.
2. Instruction: the avatar delivers instructions via Smartboard display and speakers. Duration: 45 s.
3. Pairing: the subject attempts to pair personal device with SmartBoard which represents the hidden iMAC desktop. In this phase, the subject is exposed to one (randomly selected out of 7) visual distraction stimulus. Duration: up to 3 min.
4. Final: the subject is prompted, on the SmartBoard, to enter some basic demographic information, as well as an email address to deliver the reward – an Amazon discount coupon. The information is entered directly into the SmartBoard, acting as a touch-screen input device. Duration: up to 6 min.

The total duration of the experiment ranged between 5 and 10 min.

In order to mitigate any disparities in task completion times between subjects that already had Bluetooth Discovery enabled and those who did not, the avatar informs subjects in the first 15 s of the instruction dialog that they will need to perform Bluetooth pairing with their personal device. This gives subjects over 30 s to enable Bluetooth Discovery Mode on thier device, if it is not enabled already.

We selected 6 visual effects that differed across two dimensions: color and intensity. In terms of color, we picked 3 values in the CIE chromatic space: Red, Blue, and Yellow-Green. Each is either *Solid*, i.e., shown at constant maximum intensity for the duration of the effect, or *Flickering*, i.e., its intensity grows and shrinks from the minimum to the maximum and back, completing one full cycle every second. In all settings, the maximum saturation was used. Color and intensity parameters for the 4 Phillips Hue bulbs under each condition are as follows (CCV stands for CIE Chromatic Value) [23]:

1. Red, CCV: X = 0.674, Y = 0.322
2. Blue, CCV: X = 0.168, Y = 0.041
3. Yellow-Green, CCV: X = 0.408, Y = 0.517
4. Solid intensity lumen output: 600 lm
5. Flickering intensity lumen range: 6 lm–600 lm

These color conditions were picked based on capabilities of programmable bulbs as well as background knowledge about emotive effects of color. Phillips Hue is an LED system based on creating white light. It can not create a blacklight effect or any achromatic light, which limits color selection to the subspace of the CIE color space [23] that Hue supports. (See Appendix 2 for more information).

With that restriction, we looked to the state-of-the-art about emotive reception and sensory effects of various colors in the Munsell color space [16]. (See Appendix 2 for more information). It has been shown that *principal hues* – Red, Yellow, Purple, Blue, and Green – are typically positively received. In contrast, *intermediate hues*, i.e., mixtures of any two principal hues, are more often negatively associated. Also, various colors have been shown to have either an arousing or a relaxing effect on subjects exposed to them. Based on this information, we chose three colors that differ as much as possible [15]:

- Red: Principal hue with positive emotional connotations, high associated arousal levels
- Blue: Principal hue with positive emotional connotations, low associated arousal levels
- Yellow-Green: Intermediate hue with negative emotional connotation, high associated arousal levels

Furthermore, we chose to have multiple modalities of light intensity for each color, with the expectation that a more complex modality would be more arousing and have a greater effect than its simple counterpart [12]. Not having found any previous work on the impact of exposure to colored light on performance of security-critical tasks, we include *Solid* light – the simplest modality of exposure that corresponds to the base level of stimulation. As a more complex modality, we included *Flickering* light.

Clearly, these two modalities were not the only possible choices. For example, it might have been intuitive to include even a more complex and startling *Strobing* light modality, achievable through rapid modulation of light intensity. It would have probably engendered a more profound impact on the subjects. However, ethical considerations coupled with the unattended nature of the experiment preclude using any modality that could endanger subjects with certain sensitivity conditions, such as photosensitive epilepsy. This led us to select a safe flickering frequency of 1 Hz.

We also found that all three light colors (under both intensity modalities) do not interfere with readability of a backlit personal wireless device or the image projected on the Smartboard. All experimenters, including one who used corrective lenses, could *correctly* read the screens of their personal devices, under all color conditions and intensity modalities.

3.3 Prior Results with a Similar Setup

A very similar setup was used in a previous study that assessed effects of unexpected audio distractions on 147 subjects performing Bluetooth pairing. As reported in [8], introduction of audio stimuli *significantly increased subject success rates* for every stimulus used. There was no significant impact on task completion time for any stimulus condition. This phenomenon was likely due to increased sensory arousal, as discussed in [8]. Our expectations for the impact of unexpected visual stimuli are rooted in these prior results.

3.4 Initial Hypotheses

We started out by hypothesizing that introduction of unexpected visual distractions during the process of human-aided pairing of two Bluetooth devices would have similar effects to those observed in prior experiments with audio distractions. Specifically, we expected two outcomes, as compared to a distraction-free setting:

[H1]: Lower error rates, and
[H2]: No effect on task completion times

3.5 Recruitment

The main challenge we encountered in the recruitment process is the scale of the experiments. Prior studies of usability of human-aided pairing protocols [5,9,17], demonstrated that 20–25 subjects per tested condition represents acceptable size for obtaining statistically significant findings. Our experiment has one condition for each of the six visual distraction variations, plus the control condition with no distractions. Therefore, collecting a meaningful amount of data requires at least 140 iterations of the experiment.

We used a four-pronged strategy to recruit subjects:

1. Email announcements sent to both graduate and undergraduate Computer Science students.
2. Posters placed (as signboards) near the entrance, and in the lobby, of a large campus building which housed the experimental setup.
3. Several instructors promoted participation in the experiment in their lectures.
4. Printed fliers handed out at various campus locations during daily peak pedestrian traffic times.

Recruitment efforts yielded 169 subjects in total, of whom 125 were male and 44 – female, corresponding to a 74%–26% gender split. This is expected, given that the location of our experimental setup was in the Computer Science and Engineering part of campus. Most subjects (161) were of college age (18–24 years), while 8 were in the 30+ group. This distribution is not surprising given the university population and the fact that older subjects generally correspond to researchers, faculty and staff, all of whom are much less likely to be attracted to being a subject in an experiment.

As follows from the above, our subjects' demographic was dominated by young, tech-savvy male undergraduate students.

4 Results

This section discusses the results, starting with data cleaning and proceeding to subject task completion effects.

4.1 Data Cleaning

We had to discard subject data for three reasons.

First, although instructions (in fliers, announcements and signs near the setup) specifically stated that subjects were to arrive alone, and perform the experiment without anyone else present, 37 groups (2 or more) of subjects participated. We found that the initial participant from each group performed in a manner consistent with individual subjects. However, subsequent group members who tried the experiment were (not surprisingly) significantly faster and more accurate in their task completion. Consequently, we discarded data of every subject who arrived in a group and was not the initial participant. We discuss this issue in more detail in Appendix 1.

The second reason for discarding data would have been due to subject auditory and/or visual impairment. A subject with an auditory impairment would have difficulties understanding the avatar's spoken instructions. A visually impaired subject would have difficulties with using the Smartboard and with the pairing process which relies on reading and comparing numbers. After carefully reviewing all subject video records, we could not identify any obvious visual or auditory impairment in any subject.

Some subjects successfully completed the experiment several times, perhaps hoping to receive multiple participation rewards. This occurred despite explicit instructions to the contrary. The system automatically rejected any repeated pairing attempts from devices already paired with the system, and any repeated attempts with different devices were discovered by visual inspection of subject trials. Every such repeated instance was discarded.

4.2 Task Failure Rate

Table 1 shows the number of subjects who, respectively, succeeded and failed at Bluetooth device pairing under each stimulus condition. It also details the failure rate for each condition.

Table 2 shows results from Barnard's exact test applied pairwise to the subject failure rate of the control condition and each stimulus. It demonstrates that differences between failure rates are statistically significant at the $\alpha = 0.05$ level with respect to all *Flickering* conditions: *Flickering Red*, *Flickering Blue*, and *Flickering Yellow-Green*. This even holds if we apply a conservative Bonferroni correction to account for three pairwise comparisons. This leads us to the mixed rejection of the initial hypothesis **H1**, as the failure rate increases significantly with the introduction of certain kinds of visual distractions, and remains unaffected by others. The next section discusses this further.

Table 3 shows odds ratios and 95% confidence interval for the failure rates under each stimulus, as compared to the control condition's failure rate. Interestingly, under this analysis, only the confidence intervals of *Flickering Blue* and *Flickering Yellow-Green* do not include a possible odds ratio of 1.0. Therefore – under this method of analysis – they are the only statistically significant stimuli at the $\alpha = 0.05$ level. The confidence interval defined for the *Flickering Red* condition challenges the claim of statistical significance at the $\alpha = 0.05$ level, as established by Barnard's exact test.

Table 1. Subject failure statistics

Stimulus	#Successful subjects	# failed subjects	Failure rate
None (control)	32	15	0.32
Solid Red	11	9	0.45
Flickering Red	9	11	0.55
Solid Blue	14	6	0.30
Flickering Blue	8	12	0.60
Solid Yellow-Green	10	12	0.54
Flickering Yellow-Green	7	13	0.65
Total	91	78	0.46

Table 2. Barnard's exact test on failure rates

Stimulus	Total pairings	Failure rate	Wald statistic	Nuisance parameter	p
None (control)	47	0.32	–	–	–
Solid Red	20	0.45	1.02	0.88	0.17
Flickering Red	20	0.55	1.77	0.86	0.04
Solid Blue	20	0.30	0.15	0.05	0.49
Flickering Blue	20	0.60	2.14	0.96	0.03
Solid Yellow-Green	22	0.54	1.79	0.94	0.06
Flickering Yellow-Green	20	0.65	2.51	0.91	0.01

Table 3. Subject failure rate by gender

Stimulus	Odds ratio wrt control	95% Confidence interval wrt control
None (control)	-	–
Solid Red	1.70	0.60-5.11
Flickering Red	2.61	0.89-7.63
Solid Blue	0.91	0.29-2.85
Flickering Blue	3.20	1.08–9.47
Solid Yellow-Green	1.79	0.91–7.24
Flickering Yellow-Green	3.96	1.31–11.6

Table 4. Subject failure rate by gender

Gender	# Successful subjects	# Unsuccessful subjects	Failure rate
Male	65	59	0.48
Female	25	20	0.44

We also examined subject failure rates by gender. As shown by Table 4 there is no statistically significant difference in failure rates between male and female participants; Wald statistic = 0.36, nuisance parameter = 0.01, $p = 0.46$.

4.3 Task Completion Times

Table 5[4] [5] shows average completion times in successful trials under each stimulus. After applying a conservative Bonferroni correction to account for six pairwise comparisons between individual stimulus conditions and the control condition, every stimulus condition shows an overwhelmingly large, statistically significant departure from the control condition. This results in rejection of hypothesis **H2**. The following section examines possible causes of this slowdown, as well as its implications.

Table 5. Avg times (sec) for successful pairing.

Stimulus	Mean time	Std Dev	DF wrt control	t-value wrt control	p
None	34.50	11.93	–	–	–
Solid Red	87.81	24.56	41	9.56	<0.001
Flickering Red	90.44	15.62	39	11.59	<0.001
Solid Blue	106.36	17.39	44	16.32	<0.001
Flickering Blue	91.25	24.11	38	9.61	<0.001
Solid Yellow-Green	90.30	19.08	40	11.1	<0.001
Flickering Yellow-Green	90.29	19.06	37	10.01	<0.001

Table 6. Cohen's *d* on completion times wrt Control

Stimulus	Cohen's d wrt control
None (control)	-
Solid Red	−3.42
Flickering Red	−4.49
Solid Blue	−5.33
Flickering Blue	−3.90
Solid Yellow-Green	−4.12
Flickering Yellow-Green	−4.29

[4] Std Dev = Standard Deviation.

[5] DF = Degrees of Freedom.

Table 6 shows Cohen's d for completion times under each stimulus when compared to the control condition. $|d| > 1.0$ in all cases, which means that every stimulus condition shows an overwhelmingly large, statistically significant departure from the control condition for the evaluation of Cohen's d. This result is statistically significant: it indicates that, with convincing probability, the mean completion time observed under the control is representative of a different distribution than that observed under each stimulus condition. This supports rejection of hypothesis **H2**.

Next, we looked into subject completion times for successful completion attempts by gender. Results are displayed in Table 7. A pairwise t-test shows that observed differences are not statistically significant; $t(84) = 0.04$, $p = 0.96$.

Table 7. Avg times (sec) by gender

Gender	Mean time	Standard deviation
Male	75.27	22.31
Female	75.20	24.10

Table 8. One-Way ANOVA test

	Sum of squares	DF	Mean square	F	p
Between groups	2964.28	5	592.86	1.466	0.217
Within groups	21440.33	53	404.535		
Total	24404.61	58			

Finally, we preformed Bartlett's test for homogeneity of variances as well as a One-Way analysis of variance (ANOVA) test between average task completion times of all stimulus conditions, excluding the control. Bartlett's test failed to reject the null hypothesis that all stimulus conditions share the same variance ($\chi^2 = 2.80$, $p = 0.731$). Furthermore, the one-way ANOVA test indicated no significant difference between any sample distributions (F = 1.466, p = 0.217.) Table 8 shows the results; their implications are discussed in the following section.

5 Discussion of Observed Effects

Several types of visual stimuli appear to have a negative effect on the subjects' successful completion of the Bluetooth Pairing task. However, collected data shows that this is not consistent across all stimuli. Instead, the negative effect may be tied to certain features of the particular stimulus. Instances of significant degradation in subject success rates were linked to the *Flickering* modality, for all color stimuli. This result implies that emotional perception of the stimulus may not be as much of a contributing factor to the overall increase of subject arousal as the presence of a dynamic visual stimulus. Also, in contrast with a previous study of audio distractions that observed positive effects [8], we noted no benefit to subject success rates under **any** visual stimulus.

These negative and neutral responses to static and dynamic light stimuli, respectively, are reinforced by the psychological concept of attentional selectivity. This concept assumes that the capture of an individual's attention by an aversive stimulus is likely to be momentary, occurring primarily when the stimulus is first introduced. In cognitive science, attention is conceptualized as a limited resource. For good evolutionary reason, the greatest demand on attention is in response to any change in one's environment. Once an assessment of the stimulus is made, and determined not to require additional action, attentional devotion to that stimulus fades quickly. This means that – while a static, adverse lighting change may remain adverse throughout its duration – its capacity to interfere with subject performance will fade rapidly after its onset. Instead, dynamically changing stimuli can more effectively capture subject attention and impair their performance, since many assessments are needed for many environmental changes occurring throughout the stimulus's duration.

Negative impact on subject task completion rates prompts a new attack vector for the adversary who controls ambient lighting. By taking advantage of color effects with shifting intensity levels, the adversary could force a user into failing Bluetooth pairing as a denial-of-service (DoS) attack. Moreover, the adversary might induce failure by using positively perceived colors of varying intensity. These colors may not even register as malicious in the user's mind, as they are innately associated with beneficial or pleasant emotions.

However, a much greater effect was observed in terms of average completion time. During review of subject trials, we noted that, upon exposure to the stimulus, subjects often take their gaze off their personal device (or the avatar) and focus their attention to the colorful, and possibly flickering, lights. The resulting delay frequently caused the subject's device to exit the Bluetooth pairing menu due to a time-out, and re-initiate the pairing protocol, resulting in much longer completion times overall.

Furthermore, as shown by Table 8, the introduced delay in subject task completion time was not based on the particular stimulus. Instead, the mere presence of a visual stimulus was enough to slow down successful subjects. Similar to the result in inducing user failure, the adversary is not forced to rely on an overtly malicious stimulus in order to cause substantial slowdown in task completion. However, the adversary has even more choices in stimulus selection, since all stimuli (including those with static intensity levels) were shown to impact task completion times the same way.

This effect shows further power for the adversary in control of ambient lighting. One possibility is that the adversary's goal is a denial-of-service attack by frustrating user's pairing attempts. In a more sinister scenario the adversary could try to "buy time" by introducing its own malicious device(s) alongside changes to ambient lighting and then leverage the user's lapse in focus (when being exposed to new sensory stimuli) to trick the user into pairing with that device. In the worst case, the adversary might take advantage of the user's inattentiveness while their gaze shifts away from their device and trick them into accepting a non-matching authenticator.

6 Unattended Setup: Limitations

Based on our earlier discussion of Data Cleaning, some subjects' data had to be removed from the dataset because they did not conduct the experiment alone. This occurred even though all recruitment materials (and means) as well as the avatar's instructions stated that subjects were to perform the task alone. This illustrates a basic limitation of the unattended setup: no one is present to enforce the rules in real time.[6]

We did not manage to capture fine-grained data about the subjects' awareness of a distraction. We have some anecdotal evidence from video recordings showing that some subjects noticed the distraction in obvious ways, e.g., verbal remarks or turning their heads. However, we have no evidence of subjects who failed to notice the stimulus. Information about subjects noticing a change in the environment is very important to the development of a realistic adversary model for future studies.

7 Study Shortcomings

In this section we discuss some shortcomings of our study.

7.1 Homogeneous Subjects

Our subject group was dominated by young, tech-savvy male college students. This is a consequence of the experiment's location. Replication of our experiment in a non-academic setting would be useful. However, recruiting a really diverse group of subjects is hard. Ideal venues might be stadiums, concert halls, fairgrounds or shopping malls. Unfortunately, deployment of our unattended setup in such public locations is logistically infeasible. Since these public areas already have many sensory stimuli, reliable adjustment of our subjects' arousal level in a consistent manner would be very hard. Furthermore, it would be very difficult to secure specialized and expensive experimental equipment.

In addition to being tech-savvy, young subjects are in general more apt to quickly recover from changes in the lighting of their surroundings than older adults [10]. It is possible that unexpected visual stimuli would have a different effect on an older (less technologically adept) population.

7.2 Sufficiently Diverse Stimuli

We selected six conditions to obtain as many diverse stimuli types as we could rigorously test, in addition to control. We first varied them by changing the regularity of the stimulus, expecting that a varying signal would have greater

[6] However, it would have been possible (though quite difficult in practice) to instrument our recording of the experiment to abort upon detecting simultaneous presence of multiple subjects.

impact on subjects' arousal than a steady signal. We then varied the colors, with the expectation that using colors that evoked different emotive responses and general arousal levels would impact task performance differently.

An ideal experiment would have included a stimulus with negative emotional connotation and low arousal levels. However, between three colors, two intensity conditions, and the control, we had seven total conditions to test. Furthermore, due to the nature of our experiment, we could only reasonably expect each subject to be tested under a single condition, since prior knowledge about the experiment would clearly bias the results. Adding just one additional stimulus (for both intensity modalities) would have required at least 40 more subjects. This would have placed a heavy logistical burden for our already nearly-depleted subject pool.

We also note that variance in intensity of our flickering modality did not approach the technical limit of Philips Hue bulbs. Instead, we deliberately limited the frequency of intensity fluctuations to 1Hz in order to avoid any possible negative reaction from light-sensitive subjects. This ethical issue does not reflect real-world conditions where an adversary (with no ethical qualms) could create a very fast strobing effect, possibly causing physical harm.

7.3 Synthetic Environment

Our unattended setup, while a step closer to an everyday setting than a sterile and highly controlled lab, is still quite synthetic. First, our choice to place it in a low-traffic area makes it quieter than many common settings. Second, our choice to situate it indoors makes it free of temperature fluctuations, air flow, and exposure to sunlight. Finally, our equipment (such as the Smartboard projector system) is not commonly encountered by most subjects.

7.4 Ideal Setting

Drawing upon aforementioned shortcomings, the ideal setting for our experiment would be one where:

- Subject demographics are more varied
- Subjects are not aware of the nature of the experiment until they are debriefed after task completion
- The environment is more commonplace
- The task is more security-critical

All of these criteria could be trivially met if, for example, we conducted the experiment at a busy bank ATM. The task at hand would be the obviously security-critical entry of the subject's PIN. A modern ATM comes standard with all of the features needed for our experiment: it has a keypad, a screen, a speaker (for visually impaired users), a video camera, and are in areas that are artificially lit. Similarly, a busy gas station would fit our needs, as each fuel pump typically includes a keypad for PIN entry, speakers, a screen, artificial lighting, and a video camera recording the transaction. However, despite their attractive qualities, there would be serious ethical and logistical obstacles to setting up an unattended automated experiment in one these location examples.

8 Conclusions and Future Work

As human participation in security-critical tasks becomes more commonplace, so does the incidence of users performing these tasks while subject to accidental or malicious distractors. This strongly motivates exploring user error rates and their reactions to various external stimuli. Our efforts described in this paper shed some light on understanding human errors in security-critical tasks by studying the effects of visual stimuli on users attempting to pair two Bluetooth devices.

We feel that this unattended experiment paradigm is a valuable approach that deserves further study. The development of standardized unattended and automated experimental setups could greatly lower the logistical and financial burdens associated with conducting large-scale user studies.

Given the observed negative effect on subject completion times, one interesting next step would be to conduct a similar experiment, where, instead of measuring subjects' ability to pair Bluetooth devices, we would examine the rates of incorrect pairing when the subjects are shown mis-matched numbers during the pairing process. This could help us determine whether (and how) visual distractions make users more likely to pair their device to some other (perhaps adversary-controlled) device.

Another direction is investigating effects of hybrid (e.g., audio/visual) distractions. Finally, we plan to conduct a study of subjects performing security-critical tasks, while being exposed to *multiple visual stimuli* lasting longer than 3 min. This might allow us to learn whether subjects' sensory arousal is the result of the surprise (due to the sudden visual stimulus), or an unavoidable psychophysical reaction.

Acknowledgments. This research has been supported by NSF Grant CNS-1544373.

Appendix 1: Analysis of Group Initiators

We considered potential differences in failure rates between subjects who performed the task alone, and those who did it as part of a group. As mentioned in the discussion of Data Cleaning, for each group, we only consider the initial participating group member, referred to as the Group Initiator. As Table 9 shows, there is no significant difference between failure rates of individual subjects and Group Initiators; Wald Statistic = 0.34, Nuisance parameter = 0.01, p = 0.51. Furthermore, as Table 10 shows, a pairwise t-test of completion times for individuals – compared to group initiators – shows that observed differences are not statistically significant; $t(84) = 0.09$, $p = 0.93$.

Table 9. Failure rates: initiators vs. individuals

Participant type	#Successful subjects	#Unsuccessful subjects	Failure rate
Group initiator	19	18	0.49
Individual	72	60	0.45

Table 10. Avg times (sec): initiators vs. individuals

Participant type	Mean time	Standard deviation
Group initiator	76.63	23.00
Individual	76.20	17.93

Appendix 2: A Few Colorful Words

Munsell Color System

The Munsell Color System is used for creating and describing colors. In it, all colors are grouped into two categories: primary and intermediate hues. Primary hues include: Red, Yellow, Purple, Blue, and Green, arranged in a circular shape as in Fig. 3. Intermediate hues are mixtures of two adjacent primary hues, such as Yellow-Green or Purple-Blue. Colors are defined on three dimensions: hue, lightness, and color purity. The Munsell system is based on human perception which makes it useful for rigorously defining human reaction to specific color forms. However basing the system on human perception makes the Munsell system a poor tool for direct conversion of light described by its physical wavelength into human-perceptible color.

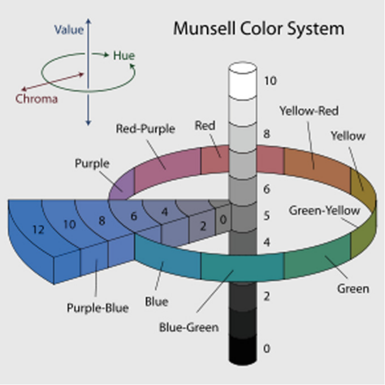


Fig. 3. Munsell color space (Image best viewed in color)

CIE Color Space

The Phillips Hue bulbs use the CIE color space. In CIE, colors are defined as a 2-dimensional space with X and Y values moving along a roughly triangular curve that corresponds to the translation of wavelengths of light to their human perception in the visible spectrum. The exact color range of the Philips Hue bulb is shown in Fig. 4.

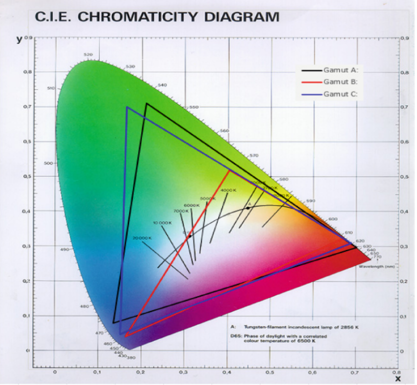


Fig. 4. Phillips Hue CIE color space (Image best viewed in color)

Appendix 3: Unattended Experiment Setup

Figures 5, 6, 7 and 8 provide additional details about our experimental setup.

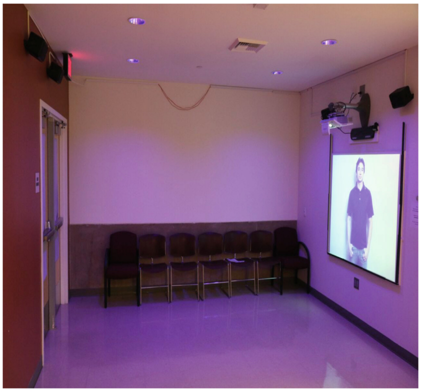


Fig. 5. The experiment environment during the Solid Blue condition (Image best viewed in color)

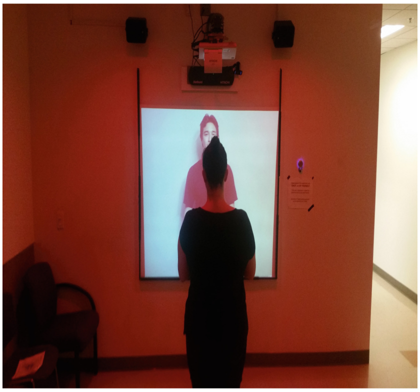


Fig. 6. The subject's perspective during the Solid Red condition (Image best viewed in color.)

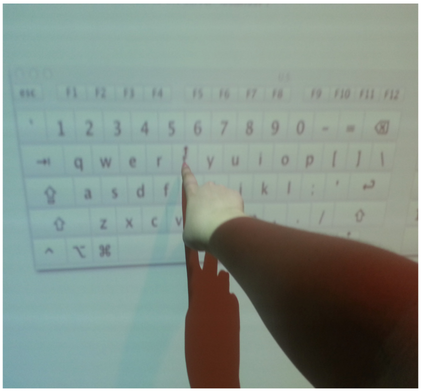


Fig. 7. Subject entering email address on Smartboard

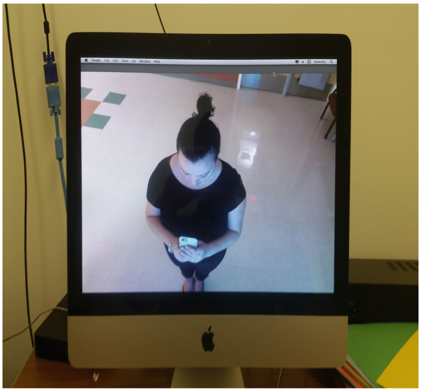


Fig. 8. Post-experimental review of video recordings (separate office)

References

1. Benignus, V.A., Otto, D.A., Knelson, J.H.: Effect of low-frequency random noises on performance of a numeric monitoring task. Percept. Mot. Skills **40**(1), 231–239 (1975)
2. Cohen, R.A.: Yerkes-Dodson law. In: Kreutzer, J.S., DeLuca, J., Caplan, B. (eds.) Encyclopedia of Clinical Neuropsychology, pp. 2737–2738. Springer, New York (2011)
3. Eimer, M.: Multisensory integration: how visual experience shapes spatial perception. Curr. Biol. **14**(3), R115–R117 (2004)
4. Gallego, A., Saxena, N., Voris, J.: Exploring extrinsic motivation for better security: a usability study of scoring-enhanced device pairing. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 60–68. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39884-1_6
5. Goodrich, M.T., Sirivianos, M., Solis, J., Soriente, C., Tsudik, G., Uzun, E.: Using audio in secure device pairing. Int. J. Secure. Netw. **4**(1), 57–68 (2009)
6. Harris, W., Stress, P.: The effects of intense noise stimulation and noxious stimulation upon perceptual performance. Ph.D. thesis, University of Southern California (1960)
7. Hockey, G.R.J.: Effect of loud noise on attentional selectivity. Q. J. Exp. Psychol. **22**(1), 28–36 (1970)
8. Kaczmarek, T., Kobsa, A., Sy, R., Tsudik, G.: An unattended study of users performing security critical tasks under adversarial noise. In: Proceedings of the NDSS Workshop on Useable Security, pp. 14:1–14:12 (2015)
9. Kainda, R., Flechais, I., Roscoe, A.W.: Usability and security of out-of-band channels in secure device pairing protocols. In: Proceedings of the 5th Symposium on Usable Privacy and Security, pp. 11:1–11:12 (2009). ACM ID: 1572547
10. Kline, D.W., Schieber F.: Vision and aging (1985)
11. Kobsa, A., Sonawalla, R., Tsudik, G., Uzun, E., Wang, Y.: Serial hook-ups: a comparative usability study of secure device pairing methods. In: Proceedings of the 5th Symposium on Usable Privacy and Security, pp. 10:1–10:12 (2009). ACM ID: 1572546

12. Koelega, H.S., Brinkman, J.-A., Zwep, B., Verbaten, M.N.: Dynamic vs static stimuli in their effect on visual vigilance performance. Percept. Mot. Skills **70**(3), 823–831 (1990)
13. Laur, S., Asokan, N., Nyberg, K.: Efficient mutual data authentication using manually authenticated strings. Cryptology ePrint Archive, report 2005/424 (2005). http://eprint.iacr.org/
14. Lazem, S., Gracanin, D.: Social traps in second life. In: 2010 Second International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES), pp. 133–140, March 2010
15. Naz, K., Epps, H.: Relationship between color and emotion: a study of college students. Coll. Stud. J. **38**(3), 396 (2004)
16. Nickerson, D.: History of the Munsell color system and its scientific application. J. Opt. Soc., 575–586 (1940)
17. Nithyanand, R., Saxena, N., Tsudik, G., Uzun, E.: Groupthink: usability of secure group association for wireless devices. In: Proceedings of the 12th ACM International Conference on Ubiquitous Computing, pp. 331–340 (2010). ACM ID: 1864399
18. Ollesch, H., Heineken, E., Schulte, F.P.: Physical or virtual presence of the experimenter: psychological online-experiments in different settings. Int. J. Internet Sci. **1**(1), 71–81 (2006)
19. Olmedo, E.L., Kirk, R.E.: Maintenance of vigilance by non-task-related stimulation in the monitoring environment. Percept. Mot. Skills **44**(3), 715–723 (1977)
20. O'Malley, J.J., Poplawsky, A.: Noise-induced arousal and breadth of attention. Percept. Mot. Skills **33**(3), 887–890 (1971)
21. Riva, G., Teruzzi, T., Anolli, L.: The use of the internet in psychological research: comparison of online and offline questionnaires. CyberPsychol. Behav. **6**(1), 73–80 (2003)
22. Uzun, E., Karvonen, K., Asokan, N.: Usability analysis of secure pairing methods. In: Dietrich, S., Dhamija, R. (eds.) FC 2007. LNCS, vol. 4886, pp. 307–324. Springer, Heidelberg (2007). doi:10.1007/978-3-540-77366-5_29
23. Wyszecki, G., Stiles, W.S.: Color Science, vol. 8. Wiley, New York (1982)

A Pilot Study of Multiple Password Interference Between Text and Map-Based Passwords

Weizhi Meng[1](✉), Wenjuan Li[2], Wang Hao Lee[3], Lijun Jiang[2], and Jianying Zhou[4]

[1] Department of Applied Mathematics and Computer Science,
Technical University of Denmark, Kongens Lyngby, Denmark
weme@dtu.dk

[2] Department of Computer Science, City University of Hong Kong,
Kowloon Tong, Hong Kong

[3] Infocomm Security Department, Institute for Infocomm Research,
Singapore, Singapore

[4] Singapore University of Technology and Design, Singapore, Singapore
jianying_zhou@sutd.edu.sg

Abstract. Today's computer users have to remember several passwords for each of their accounts. It is easily noticed that people may have difficulty in remembering multiple passwords, which result in a weak password selection. Previous studies have shown that recall success rates are not statistically dissimilar between textual passwords and graphical passwords. With the advent of map-based graphical passwords, this paper focuses on multiple password interference and presents a pilot study consisting of 60 participants to study the recall of multiple passwords between text passwords and map-based passwords under various account scenarios. Each participant has to create six distinct passwords for different account scenarios. It is found that participants in the map-based graphical password scheme could perform better than the textual password scheme in both short-term (one-hour session) and long term (after two weeks) password memorability tests (i.e., they made higher success rates). Our effort attempts to complement existing studies and stimulate more research on this issue.

Keywords: User authentication · Graphical passwords · Usable security · Multiple password interference · HCI

1 Introduction

Over the past few decades, text-based passwords are the most widely adopted method for user authentication. However, users may suffer from many issues when using text or pattern in the aspects of security and usability [6,35,38]. As an example, users find it difficult to remember their textual information for a long time due to the long-term memory (LTM) limitation [37]. As a result, they are likely to choose and use weak textual passwords. To mitigate these

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 145–162, 2017.
DOI: 10.1007/978-3-319-61204-1_8

drawbacks, graphical passwords (GPs) have been proposed as an alternative in the literature, where users are able to interact with images during the registration and authentication [29]. Generally, graphical passwords can be categorized as recognition-based, pure recall-based and cued recall passwords.

Many research studies have shown that graphical passwords can indeed help users in remembering passwords (see Sect. 2). However, nowadays users have to remember not just one password, but many, which would add a significant burden on users' memory. As a result, the concern of *multiple password interference* is raised, where for users, remembering a password for an account (e.g., email account) affects their memory of other accounts' passwords (e.g., Facebook account). In the literature, this issue has not been widely investigated, whilst one previous research had paid attention to this issue. They made a study between text-based passwords and click-based graphical passwords (*PassPoints*) [8]. In particular, the click-based GP system requires users to click on several points on an image rather than enter textual information [36]. They found that the click-based GPs were significantly less susceptible to multiple password interference in the short-term, while the results were not statistically different from textual passwords in the long-term (i.e., after two weeks).

Our current work was motivated by two observations. (1) In the literature, there have been few studies investigating this important issue of multiple password memory in the area of graphical passwords. (2) Map-based graphical passwords have been recently developed, where users can click on several locations on a world map as their secrets. A map is believed to provide better recall for users, but has not been investigated for multiple password memory. Based on this claim, our main goal of this paper is to investigate the multiple password interference between text-based passwords and map-based graphical passwords. More specifically, we develop a prototype of map-based GP in our study, which allows a sequence of three click-points on a digital world map. Furthermore, we follow the steps in the most relevant work [8], in order to facilitate the comparison and validation of the collected results.

More formally, *memory interference* can be described as "the impaired ability to remember an item when it is similar to other items stored in memory" [3]. In this work, our primary goal is to investigate the multiple password interference in text passwords and map-based GPs. The contributions of our paper are summarized below:

- A user study consisting of 60 participants was conducted to investigate the multiple password interference issue between textual passwords and map-based graphical passwords; where 50 of them successfully return after two weeks to test recall of those passwords they had created. They were assigned to use either text password system or map-based GP system. Each participant should create six distinct passwords for different account scenarios.
- It is found that participants can perform better in recalling multiple passwords using the map-based graphical passwords, not only in the short-term (e.g., a one-hour session) but also in the long-term (namely two weeks). In comparison with the results from former study (e.g., PassPoints), our results

indicate that map-based graphical passwords can offer a better and positive impact on recalling multiple passwords in both short and long term.

It is worth emphasizing that our study aims to complement existing research and reveals that recall of multiple passwords between text and map-based graphical password schemes has a statistically significant difference in both short and long term. The results also encourage research studies to develop proper map-based schemes to improve the issue of multiple password interference.

The remainder of the paper is organized as follows. Section 2 describes related work about graphical password classification, map-based graphical passwords and multiple password interference. Section 3 presents our developed map-based GP system and introduces our study methodology including procedure and steps. Section 4 analyzes the collected results. We provide a discussion in Sect. 5 and present some limitations in Sect. 6. Finally, we conclude our paper in Sect. 7.

2 Related Work

2.1 Graphical Passwords

Graphical password schemes can be classified into three broad categories [5,7]: namely, recognition-based, pure recall-based and cued recall-based.

- *Recognition-based GPs.* This kind of scheme requires users to select images from a large gallery. For example, PassFaces [10,25] lets users identify a set of human faces during the authentication phase. To create a password, the user chooses four images of human faces from a fixed portfolio of faces. The Story scheme [10] lets users identify a few images with a theme (e.g., people and food) from an image gallery.
- *Pure recall-based GPs.* Such kind of scheme lets users draw a secret on an image. An example is the DAS proposed by Jermyn et al. [17], where users draw the password on a grid. Tao and Adams [30] proposed Pass-Go that lets users select intersections within a grid as a way to enter their password. Based on Pass-Go, Android unlock patterns are developed on Android phones, which are a tuned application requiring users to unlock their phones by inputting correct patterns. Then, Dunphy and Yan [12] explored whether a background image can improve the performance of graphical passwords.
- *Cued recall-based GPs.* This kind of scheme requires users to click on a sequence of points to construct their passwords. *PassPoints* by Wiedenbeck et al. [36] belongs to this category. In *PassPoints*, users recall a sequence of five selected points by clicking on them. For authentication, users have to click close to the chosen click points within some (adjustable) tolerance distance. Later, Chiasson et al. [9] proposed *Persuasive Cued Click-Points* (*PCCP*), which lets users click a point on each of a sequence of background images.

The current graphical password schemes are designed around the actions of click, choice or draw. Combinations of these schemes have also been studied. For instance, Meng [20] developed a click-draw based graphical password scheme (CD-GPS) with the purpose of improving the image-based authentication in both security and usability by combining the actions of clicking, selecting and drawing. Several new GP systems can be referred to [14,18,19,23]. There are many studies focusing on security aspect of graphical passwords [4,16,17,26,32,33] and relevant user behaviors [1,11,21,27,37,39].

2.2 Map-Based Graphical Passwords

Map-based GPs can be categorized as a kind of cued recall-based graphical password, in which users can recall their selected places on a world map. Despite some early research work, map-based GPs start becoming popular in recent years. In 2012, Georgakakis et al. [15] proposed a map-based graphical password scheme called *NAVI*, where the credentials of the user are his username and password formulated by drawing a route on a predefined Google map. In particular, a user selects the starting and the ending point, so that the route is calculated by the provider of the route planning service.

During the same time, Sun et al. [28] extended the above idea and suggested to use an extremely large image as the password space. They proposed a world map based graphical password authentication system called *PassMap*, in which a password consists of a sequence of two click-points on a large world map. To construct secrets, users can shift the world map to selected areas, and zoom the map to the desired zoom-level. An example of *PassMap* is shown in Fig. 1. Their collected results showed that after a week, the accuracy of login is 92.59%, and claimed that the performance was better than *PassPoints*. Later, Thorpe et al. [31] designed *GeoPass*, an interface for digital map-based authentication, where users can choose a single place as his or her password. For this scheme, users only have to remember the final location, rather than their method of navigating there. They included 35 participants in their study and found that nearly 97% of users could correctly remember their location-based-passwords over 8–9 days with very few failed login attempts. It is worth noting that *PassMap* and *GeoPass* are very similar in that secrets are constructed by clicking one or two places on a world map (e.g., Google map).

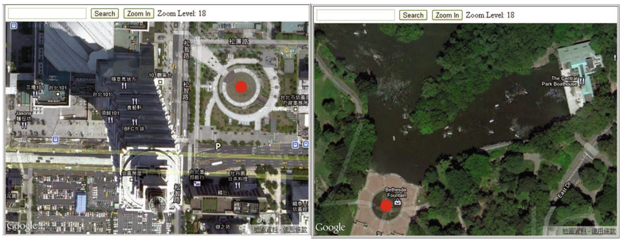


Fig. 1. An example of *PassMap*. The first click-point with red (Left) and the second click with red (Right). (Color figure online)

Recently, Meng [22] proposed *RouteMap* for better multiple password memory, which allows users to draw a route on a map as their password. They found that users could achieve better performance using *RouteMap* in terms of multiple password memory. From these studies, it is noticeable that users may have a better memorability regarding map-based graphical passwords.

2.3 Multiple Password Interference

Today, users have to remember more passwords rather than before in the course of their daily lives such as social networking accounts, personal email accounts, commercial email accounts and so on. In these cases, *multiple password interference* may become an important issue.

In the text password scheme, Vu et al. [34] had explored the memorability of multiple textual passwords for different accounts. The study discovered that users who were given five passwords have difficulty remembering them compared to those who only had to remember three. It is also revealed that users often selected passwords that have direct contextual relationship with the account(s) in question.

For the graphical password scheme, Everitt et al. [13] presented an early study on how frequency of access and interference affects recognition-based graphical passwords such as frequency of access to a graphical password. They employed *PassFaces* and discovered that many factors can noticeably impact authenticating with multiple facial graphical passwords. For example, participants who accessed four different graphical passwords per week were ten times more likely to completely fail to authenticate than participants who accessed a single password once per week.

Chiasson et al. [8] has conducted a study on comparing the recall of multiple text passwords against that of multiple click-based graphical passwords (namely *PassPoints*). Six account scenarios were simulated and they found that in the short-term, participants in the graphical password scheme performed significantly better than participants using the textual password scheme. They made fewer errors when recalling their graphical passwords. However, they found that recall success rates were not statistically different from each other after two weeks. More recently, Al-Ameen and Wright [2] showed that mental stories could be used to improve the interference issue in *GeoPass*. Meng et al. [24] presented a study on the memorability of multiple passwords between textual passwords and unlock patterns. The study also explored the difficulty of remembering those patterns for a prolonged period of time.

It is worth noting that recent research studies had showed that the performance of map-based GPs could be better than *PassPoints*; thus our current work is timely for the investigation of multiple password interference between textual passwords and map-based GPs. Our work is different from [2,24], as our used map-based scheme is not the same (i.e., *GeoPass* only allows one location to be selected). Thus, our study can complement existing research results.

3 User Study

Following the methodology in the literature [8], we conducted a lab-based user study involving two sessions: Session-1 and Session-2. All participants were students from a university who had no prior information security training, as well as no prior experience with graphical passwords.

The first session, Session-1 lasted for approximately an hour, recruiting a total of 60 participants (M = 25.6 years; SD = 3; 32 females). During the study, we randomly selected 30 participants to the text password scheme (MText), while the others were assigned to the graphical password scheme (MMP). For Session-2, after two weeks, 50 participants were successfully returned to the lab and recalled their created passwords. Also, we provide a feedback form for each session asking for users' feedback and attitude.

3.1 Implementation of Map-Based Graphical Passwords

As the source of existing map-based graphical passwords was not released, we developed a prototype of map-based graphical passwords in our lab computers with a 17-inch screen (Intel R, CPU 2.6 GHz). It is worth noting that our system provides a common platform that is able to realize several existing map-based GPs such as *PassMap* and *GeoPass*.

To obtain a world map, we utilized the JavaScript based Google Maps API, which provides an extensive move-by-dragging, zooming and search functions. When users zoom in/out on the map, our system reports the zoom-levels. The search function allows users to shift to a specific part of the map quickly and further zoom in to locate a specific area. Similar to [28], our system embedded a 640 × 420 pixel frame block for displaying the world map in a web page and road map instead of satellite-type map is used by default. The tolerance areas are 21 × 21 pixels.

We describe the registration and login phase of our developed map-based GP in Fig. 2 and Fig. 3, respectively.

- *Registration.* Figure 2 depicts that users are able to choose three locations on a world map in constructing their passwords. All the information like coordinates will be forwarded and stored in a database.
- *Login.* Figure 3 presents that users have to submit their names and three locations to the system for authentication. The system will compare the inputs with the stored information (i.e., checking whether the inputs are within tolerance). A user is successfully verified only if both credentials are correct.

It is worth noting that *PassPoints* requires users to select five points on an image as their secrets, *GeoPass* requires users to select one place while *PassMap* needs users to click two places. Intuitively, we consider the memory of one place in a map should be quite easy. To make a better comparison with *PassPoints*, our map-based GP system increases the memory burden a bit and thus demands users to choose three locations on a world map (as a study).

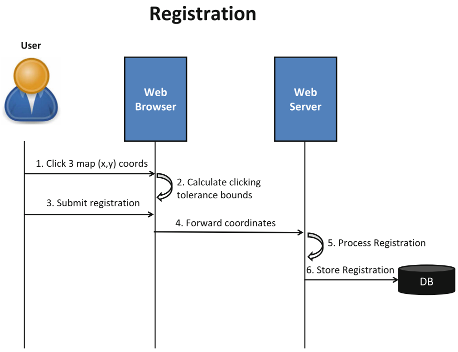


Fig. 2. Our developed system of map-based GP: registration phase.

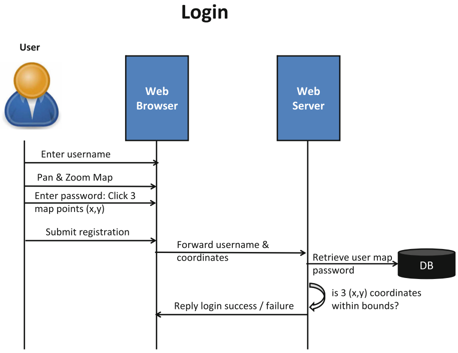

Login
User
Web Browser
Web Server
Enter username
Pan & Zoom Map
Enter password: Click 3 map points (x,y)
Submit registration
Forward username & coordinates
Retrieve user map password
DB
is 3 (x,y) coordinates within bounds?
Reply login success / failure

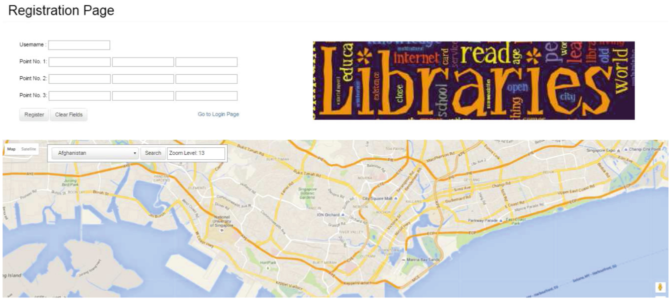
Fig. 3. Our developed system of map-based GP: login phase.

3.2 Session-1: Procedure and Steps

Session-1 consists of three phases: password practice, password creation and password retention. During the first phase, participants must complete two trials as practice to get them familiarized with the graphical password creation. In this phase, participants are not required to remember their practice passwords.

In the second phase (password generation), each participant has to complete six trials. Each password is associated with a different account scenario. A total of six account scenarios are constructed: email, bank, instant messenger, library, online dating and work (commercial). The accounts were identified by distinct banners at the top of the system interface. For map-based GP system, Fig. 4 depicts an example of registration interface with library account. It is worth noting that the banners are the same for text and map-based passwords.

In Fig. 5, we detail the process of registration, where Fig. 5(a) shows the interface that requires users to input their names and Fig. 5(b) allows users to select locations on a Google world map to create their secrets. After selecting locations, the click points including X and Y coordinates and zoom-level will be


Registration Page

Usemame :
Point No. 1:
Point No. 2:
Point No. 3:

Fig. 4. The interface of map-based password creation.

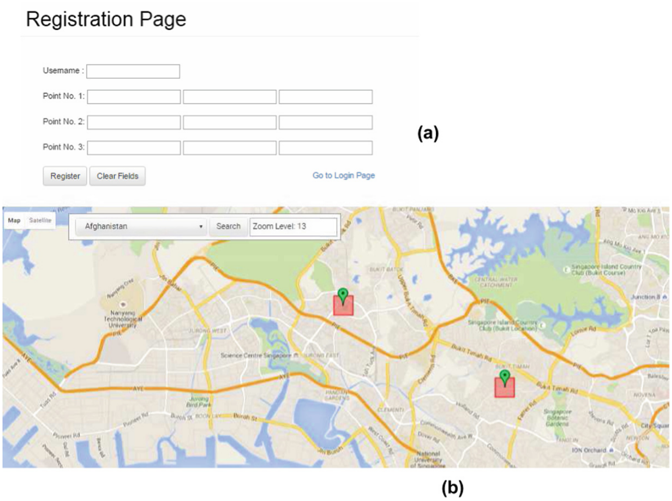


Fig. 5. Registration process: (a) inputting user name and (b) selecting locations.

shown in Fig. 5(a). To facilitate location search, our system provides a search bar in the map (see Fig. 5(b)) as well as zoom-level selection. Besides, users can switch the map between road and satellite map background.

In this stage, all accounts were presented to participants in the same order. In addition, participants were asked to construct their passwords which they could remember but would be difficult for attackers to guess. All participants were told that they would have to remember their created passwords for later use and each password was connected to a specific account scenario. Participants perform the steps below for each account:

- *Creation*. Participants have to create a password for each account. The textual password must consist of at least eight characters (this number is selected based on [8]) and entered twice. The password text was made visible during the creation process. For the graphical password, participants follow the steps in our system. This comprises choosing three different locations on a Google world map (see Fig. 4).
- *Confirmation*. In the text password scenario, participants have to enter their password (shown as asterisks), while they have to choose the same places in the graphical password scheme. If participants unexpectedly cannot remember their created password, they have the choice returning to the last step to create the password again.
- *Feedback*. Participants were invited to provide feedback to our questions (in the form of 10-point Likert-scale) about the creation and memory of the current passwords.
- *Distraction*. A 2-min interlude is given to each participant as a distraction task. First of all, participants counted backwards by 3 s from a random 4-digit number, and then participants were given a Mental Rotations Test puzzle

(MRT). Similar to [8], the purpose of these tasks was to clear participants' visual and textual working memory.

- *Login.* Participants were then invited to login with their created passwords. In the event that the password is mis-entered, they are allowed to re-enter their passwords multiple times with no pre-defined limit. If their created secrets are forgotten, they are allowed to return to the creation step.

Before the start of retention phase, participants were given a 10-min break where they were given a demographics questionnaire and could leave the experiment venue for the remaining break time. Upon returning from the break, account scenarios were presented to participants in a re-shuffled order with the help of a Latin square, where each row is connected with a participant and each column is connected with an account. All participants are required to recall each of their six textual or graphical passwords.

After this phase, participants were given another questionnaire to solicit feedback about their attitudes towards system usage and password memory.

3.3 Session-2: Procedure and Steps

Up to 50 participants returned to our lab after two weeks, among which 22 of them were previously assigned to the text password scheme. This session consists mainly of the 2-week retention phase, which follows the similar steps as per the initial retention phase during Session-1. The account scenarios were presented in the same order.

Similarly, participants are allowed to try multiple times with no limit to recall their created passwords until they gave up trying to recall. After the end of the session, a questionnaire was given to collect their feedback about password recall in these two schemes.

4 Results and Analysis

The Chi-square (χ^2) tests are mainly used to analyze data from the user studies. In all cases, we regard a value of $\rho < 0.05$ indicating that the results are statistically different between two schemes. In particular, we employ success rates as one of the major measures to evaluate participants' performance. The success rates for each step are presented in Table 1.

Table 1. The success rates of login, recall-1, recall-2 and password-confirmation. MText: multiple text passwords. MMP: multiple map-based passwords.

	Confirmation		Login		Recall-1			Recall-2		
	MText	MMP	MText	MMP	MText	MMP	χ^2	MText	MMP	χ^2
First attempt	97%	97%	94%	99%	62%	96%	$\chi^2 = 43.2,\ \rho < 0.05$	43%	72%	$\chi^2 = 30.1,\ \rho < 0.05$
Three attempts	99%	100%	98%	100%	86%	98%	$\chi^2 = 27.4,\ \rho < 0.05$	55%	79%	$\chi^2 = 23.3,\ \rho < 0.05$
More attempts	100%	100%	99%	100%	89%	99%	$\chi^2 = 17.5,\ \rho < 0.05$	64%	82%	$\chi^2 = 19.5,\ \rho < 0.05$

During password confirmation and login, it is found that there was no statistically significant difference in success rates between these two schemes. The observations are in line with the results from the previous research work [8]. In the next parts, we discuss success rates in retention and recall errors.

4.1 Recall Success Rates

Recall-1. In this phase (short term), it is found that the success-rate differences between the two password schemes are statistically significant. For correct-first-time attempts, participants in the text password scheme only achieved a success rate of 62%, but those in the map password scheme could reach a rate of 96%. With correct-on-multiple attempts, the textual password scheme yielded a success rate of up to 89%, indicating that 11% participants eventually gave up. The map password scheme in contrast yielded a rate of 99%. In other words, for 11% trials in the text password scheme, participants cannot remember their passwords, but there are only 1% unsuccessful trials for the map-based GP. These results indicated that participants in the graphical password scheme were more likely to successfully recall their created passwords than those who were under the text password scheme.

Recall-2. During this phase, we found that participants in the text password scheme had more difficulty in remembering their passwords, while those in the graphical password scheme could perform better. As an example, for the first attempt, the former could only achieve a success rate of 43%, but the latter could reach 72%. With multiple attempts, the text password scheme eventually yielded a rate of 64%, while a rate of 82% could be reached by the latter. Differing from the previous study [8], our existing results reveal that success rates have statistically significant differences between these two password schemes.

In Table 2, we present the success rates of male and female participants during Recall-2. For the graphical password scheme, 80% males can correctly enter map-based passwords within three attempts as compared to 78% female participants. Finally, male participants can reach 84% while females can reach 80%.

Table 2. Success rates of male and female for the Recall-2.

	MText: male	MText: female	χ^2
First attempt	42%	44%	$\chi^2 = 4.3,\ \rho > 0.05$
Within three attempts	53%	47%	$\chi^2 = 4.7,\ \rho > 0.05$
Multiple attempts	68%	60%	$\chi^2 = 5.4,\ \rho > 0.05$
	MMP: male	MMP: female	χ^2
First attempt	78%	76%	$\chi^2 = 4.1,\ \rho > 0.05$
Within three attempts	80%	78%	$\chi^2 = 4.5,\ \rho > 0.05$
Multiple attempts	84%	80%	$\chi^2 = 4.8,\ \rho > 0.05$

The results are similar under the text password scheme, where male participants could achieve 53% within three attempts and eventually 68% with multiple attempts. In contrast, female participants could achieve 47% and 60%, respectively. However, the results have no statistically significant differences ($\rho > 0.05$). This indicates that there is no apparent difference between male and female.

In [8], the results showed that males tend to perform better than females when using *PassPoints*, as males are good at visual-spatial tasks. By contrast, our system reduces such gap between males and females. Considering only statistics, participants in our study were performed generally better than the results reported by [8]. This implies that map-based GP can enable users to have a better memory of multiple passwords, as compared to *PassPoints*.

To understand the different observations, we further analyzed the collected data and informally interviewed the returned participants, where 90% of them gave their feedback. It is found that participants could connect their created map-based passwords with their past experience such as tours, visits, conference locations and so on. That is, when they have to input passwords in different account scenarios, they can link the account scenarios to their previous memory. For example, one participant said that "*When I have to input my map password to a work account, I can quickly remind of several locations where I have been*". During the interview, we found that such relationships can generally enhance the memory accuracy and strength in the long-term.

4.2 Recall Errors

Recall-1. As shown in Table 3, participants made 183 errors in the text password scheme, whilst they made 15 errors in the graphical password scheme. Since we allow participants to try many times for an account, the number of errors is bigger than the number of trials.

To look closer to the types of errors, participants are vulnerable to multiple password interference under the text password scheme. It is found that 21 out of 30 participants made recall errors.

- *Wrong account.* Many participants tried a wrong password for an account. For example, some of them tried a password from instant message on an online dating account. Some of them made mistakes between bank account and work account.
- *Wrong account variant.* This error means that participants entered some variant of the correct password for another account.
- *Misspelled variant.* Up to 20 errors were made because of wrong spells. For example, some participants entered "lxyy1987" instead of "Lxyy1987".
- *Unknown errors.* These errors were not belonging to any types above. Some participants reflected that they may try a password other than one they created during the study.

The errors made in the text password condition and the graphical password condition are summarized in Table 3 and Table 4, respectively. The main observations regarding the graphical password condition are discussed as below.

Table 3. Classification of recall errors for the text password scheme.

Type of error	Recall-1	Recall-2
Wrong account	90	88
Wrong account variant	42	24
Misspelled variant	20	73
Unknown	31	40
Total number of errors	183	225

Table 4. Classification of recall errors for the map-based password scheme.

Type of error	Recall-1	Recall-2
Outside tolerance	5	15
Incorrect click order	2	7
Forgotten locations	3	8
Incorrect zoom level	5	46
Total number of errors	15	76

- *Outside tolerance.* There are up to five errors made for this sake, where participants could remember the general location, but may choose a place a bit far from the correct location. For example, some participants selected a place of car park, but clicked a “house” near the car park during the recall.
- *Incorrect click order.* Not many, but still two errors were made since participants selected their locations in a wrong order.
- *Forgotten locations.* Some participants may forget their locations, or tried a location from another account. All these errors belong to this type.
- *Incorrect zoom level.* Zoom level selection is a feature of map-based graphical passwords, which can help enlarge the password space. However, we find that a few participants may make errors by selecting a wrong zoom level, or even forget the correct level.

Recall-2. Overall, participants made significantly more errors when inputting their passwords after two-weeks, under either the text password scheme, or the graphical password scheme. There are totally 225 errors made related to text passwords, while 76 errors made for map-based passwords. The recall results have statistically significant differences in these two schemes: MText: ($t = 3.73,\ \rho < 0.05$) and MMP ($t = 4.81,\ \rho < 0.05$).

For the text password scheme, most errors were made because of “wrong account” and “misspelled variant”, which cover up to 67% of the total errors. Some participants may cycle through their passwords or variations of their passwords when they forget the correct secret for a specific account. These observations are in line with [8].

For the graphical password scheme, most errors were caused by tolerance and zoom level, which claim a percentage of nearly 80% total errors. It is found that participants are more likely to click a place out of the tolerance or forget to check the zoom levels. For example, some participants could choose the correct location, but ignore the zoom level, resulting in a wrong trial.

5 Discussion

5.1 User Feedback

As stated earlier, after each session, we provided each participant some questions to collect their feedback and attitude towards system usage and multiple password interference. The major questions along with the scores are shown in Table 5. In particular, 10-point Likert scales were used in each feedback question ranging from 1 to 10, where 1-score indicates strong disagreement and 10-score indicates strong agreement.

- *System usage.* According to the first two questions, most participants considered that both system interfaces are easy to use (i.e., both scores are above 9 on average).
- *Recall-1.* Regarding the third and fourth question, it can be observed that most participants can remember multiple passwords within a short time. The score in the graphical password scheme is a bit higher than that in the text password scheme (8.9 versus 8.4).
- *Recall-2.* Most participants indicated that they had difficulty in remembering multiple textual passwords after two weeks, in which the score is only 4.5. In contrast, the score in the password scheme reached 7.6, which is much better. The feedback shows that participants can have a better capability of remembering multiple map-based passwords.
- *Preference.* It is found that participants gave a positive score (8.2 on average) to the map-based GP system, presenting their interests in using map-based passwords in practice. In comparison, the text password scheme only received a score of 5.3.

Table 5. Main questions and relevant scores from users' feedback.

Questions	Score (average)
1. The text password interface is easy to use	9.1
2. The map-based password interface is easy to use	9.0
3. I easily remembered the created map passwords after one hour	8.9
4. I easily remembered the created text passwords after one hour	8.4
5. I easily remembered the created map passwords after two weeks	7.6
6. I easily remembered the created text passwords after two weeks	4.5
7. I prefer using the map-based passwords instead of text passwords in practice	8.2
8. I prefer using the text passwords instead of map-based passwords in practice	5.3
9. I am able to manage multiple map-based passwords	8.3
10. I am able to manage multiple text passwords	6.3

- *Password management.* The last two questions have a score of 8.3 and 6.3, respectively. These indicate that most participants believed that they can handle multiple passwords in the graphical password scheme. Most of them stated that map locations are more easily to recall.

It is worth noting that these scores are subjective, but they reflect participants' confidence in their password generation and management. We also interviewed up to 80% (48 out of 60) participants to explore why they provide such scores. Our interests are mainly focusing on participants' feedback in relation to Recall-1, Recall-2 and password management.

- For Recall-1, most participants (40 out of 48) stated that they could remember both types of passwords in the short term. Some of them reflected that they had some techniques in remembering text passwords, or could employ some strategies based on their own experience.
- For Recall-2, most participants (43 out of 48) supported that they can remember map-based graphical passwords much better, as the map provides many cues for them to recall their selected locations. As an example, one participant revealed an example of the created map-based secret as: 'A school', 'B park' and 'C mall'. In real life, this participant will go to 'A school', through 'B park' and 'C mall' sequentially. Thus, it is easily to recall the password as long as he can remember this route for a specific account.

For password management, most participants (38 out of 48) advocated that they could have a better memory of multiple passwords in the graphical password scheme. For instance, one participant said that he could simply use different routes to create map-based passwords. Some participants said that their map-based passwords were generated based on their past experience in tourist and conference; thus, it is very convenient for them to remember the passwords for a long time.

5.2 Comparison with Map-Based GPs and PassPoints

Both map-based GPs and *PassPoints* belong to the kind of cued recall-based graphical password, where users are able to select a set of points on an image to construct the secrets. By considering the results from [8], both graphical passwords can be easier to recall than text passwords in the short term. These results indicate that most participants can manage multiple graphical passwords with different accounts.

By contrast, after two weeks, Chiasson et al. [8] showed there was no significant difference in recalling multiple passwords between *PassPoints* and text passwords. However, our study reveals that participants can still cope with multiple map-based passwords better than text passwords in the long term (after two weeks). There are two major reasons:

- *Background selection.* For map-based GPs, users can choose their own map background, i.e., they can zoom in or zoom out the map to a particular area and choose their preferred locations. In contrast, they have limited selections in *PassPoints*, with an image pool including several pre-loaded images.

- *Map locations.* Users can choose locations on a world map in the map-based GP. In *PassPoints*, they should click several points on an image. We find that locations are more useful for users to remember, as these places are usually familiar to them. At least, they have a general understanding of the selected location. Therefore, they can link the locations to facilitate their long-term memory. In comparison, five clicks in *PassPoints* may not provide much information for users to link these clicks to a particular account. For instance, users should click some objects to improve their memory; however, there may no strong relation between clicks, causing users unable to remember their clicks after a long period.

On the whole, our discussion revealed that users are empowered with better recall capability of multiple graphical passwords with the map-based graphical passwords rather than *PassPoints*.

6 Limitations

Lab Study. Our primary goal in this study is to examine the multiple password interference between textual and map-based graphical passwords. We followed the established methodology in the literature for study steps and working memory clearance. However, we acknowledge that the lab study may not reflect password usage in real lives. For instance, participants have to construct six new passwords one after another and recall them in a short time, which seldom occur in practice. In the lab environment, we found that participants may create passwords with medium strength. For example, participants in the text password scheme may generate some weak passwords, where easy patterns could be identified. By contrast, participants in the graphical password scheme can generate better secrets. Still, our study results provide useful information and complement existing research in this area.

Participants. Current participants in our study were mainly students, but this does not dimmish the results of our work. In future work, diverse participants can be considered to validate our results. Besides, all participants did not have any prior experience in utilizing any map-based graphical password system, but they have much experience in constructing text passwords. In this case, participants should have an advantage when recalling text passwords. Interestingly, our results showed that there was a significant difference in recalling text and map-based passwords. It is found that participants could handle multiple map-based passwords better than text passwords, since map can provide cues for participants when they recalled their secrets.

User Study. In the previous work with *PassPoints*, it is found that the connection between account and clicks is not clear. Participants were likely to select click-points in simple patterns such as a straight line or C-shape [8]. These indicate that participants tried to connect their passwords with their accounts for better recall. From our existing data, we reveal that there is a potential of enhancing multiple password memory through designing graphical passwords in an even proper way. That is, users can have a better recall capability of multiple graphical passwords when connecting passwords to their past experience. From this view, map provides many locations for users to select in terms of their own experience. Our implication could be verified in even larger studies.

Despite these limitations, our study provides interesting results to understand the effects of password interference, which can complement existing research.

7 Conclusion

Motivated by the recent development of map-based graphical password schemes, we conducted a two-phase user study with 60 participants to investigate the issue of multiple password interference between text passwords and map-based passwords. After two weeks, up to 50 participants successfully returned. In the study, each participant has to create and remember district passwords for six different account scenarios. Overall, we find that participants in the graphical password scheme can perform better than those in the text password scheme, not only in the short-term (one-hour session), but also in the long-term (after two weeks). Our current results reveal that participants can cope better with map when recalling multiple passwords. In particular, it is found that participants indeed made fewer recall errors in the map-based graphical password scheme than those in the text password scheme. Our work aims to complement existing work and stimulate more research in this area.

Acknowledgments. We would like to thank all participants for their hard work and collaboration in the user studies (e.g., data collection), and thank all anonymous reviewers for their helpful comments. This work was partially supported by SUTD start-up research grant SRG-ISTD-2017-124.

References

1. Alt, F., Schneegass, S., Shirazi, A.S., Hassib, M., Bulling, A.: Graphical passwords in the wild - understanding how users choose pictures and passwords in image-based authentication schemes. In: Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI), pp. 316–322 (2015)
2. Al-Ameen, M.N., Wright, M.: Multiple-password interference in the GeoPass user authentication scheme. In: Proceedings of NDSS Workshop on Usable Security (USEC), pp. 1–6 (2015)
3. Anderson, M.C., Neely, J.H.: Interference and inhibition in memory retrieval. In: Memory. Handbook of Perception and Cognition, chap. 8, 2nd edn, pp. 237–313. Academic Press (1996)
4. Bianchi, A., Oakley, I., Kim, H.: PassBYOP: bring your own picture for securing graphical passwords. IEEE Trans. Hum.-Mach. Syst. **46**(3), 380–389 (2015)
5. Biddle, R., Chiasson, S., Van Oorschot, P.C.: Graphical passwords: learning from the first twelve years. ACM Comput. Surv. **44**(4), 19 (2012)
6. Bonneau, J.: The science of guessing: analyzing an anonymized corpus of 70 million passwords. In: Proceedings of the 2012 IEEE Symposium on Security and Privacy, pp. 538–552 (2012)

7. Chiasson, S., Biddle, R., van Oorschot, P.C.: A second look at the usability of click-based graphical passwords. In: Proceedings of the 3rd Symposium on Usable Privacy and Security (SOUPS), pp. 1–12 (2007)
8. Chiasson, S., Forget, A., Stobert, E., van Oorschot, P.C., Biddle, R.: Multiple password interference in text passwords and click-based graphical passwords. In: Proceedings of the 2009 ACM Conference on Computer and Communications Security (CCS), pp. 500–511 (2009)
9. Chiasson, S., Stobert, E., Forget, A., Biddle, R.: Persuasive cued click-points: design, implementation, and evaluation of a knowledge-based authentication mechanism. IEEE Trans. Dependable Secure Comput. **9**(2), 222–235 (2012)
10. Davis, D., Monrose, F., Reiter, M.K.: On user choice in graphical password schemes. In: Proceedings of the 13th Conference on USENIX Security Symposium, pp. 1–11. USENIX Association (2004)
11. Dirik, A.E., Memon, N., Birget, J.C.: Modeling user choice in the PassPoints graphical password scheme. In: Proceedings of the 3rd Symposium on Usable Privacy and Security (SOUPS), pp. 20–28 (2007)
12. Dunphy, P., Yan, J.: Do background images improve "draw a secret" graphical passwords? In: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS), pp. 36–47 (2007)
13. Everitt, K.M., Bragin, T., Fogarty, J., Kohno, T.: A comprehensive study of frequency, interference, and training of multiple graphical passwords. In: Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI), pp. 889–898 (2009)
14. Gao, H., Liu, X.: A new graphical password scheme against spyware by using CAPTCHA. In: Proceedings of the 5th Symposium on Usable Privacy and Security (SOUPS), Article No. 21 (2009)
15. Georgakakis, E., Komninos, N., Douligeris, C.: NAVI: novel authentication with visual information. In: Proceedings of the 2012 IEEE Symposium on Computers and Communications (ISCC), pp. 588–595 (2012)
16. Gołofit, K.: Click passwords under investigation. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 343–358. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74835-9_23
17. Jermyn, I., Mayer, A., Monrose, F., Reiter, M.K., Rubin, A.D.: The design and analysis of graphical passwords. In: Proceedings of the 8th Conference on USENIX Security Symposium, pp. 1–14. USENIX Association, Berkeley (1999)
18. Liu, C.-L., Tsai, C.-J., Chang, T.-Y., Tsai, W.-J., Zhong, P.-K.: Implementing multiple biometric features for a recall-based graphical keystroke dynamics authentication system on a smart phone. J. Netw. Comput. Appl. **53**, 128–139 (2015)
19. Lopez, N., Rodriguez, M., Fellegi, C., Long, D., Schwarz, T.: Even or odd: a simple graphical authentication system. IEEE Lat. Am. Trans. **13**(3), 804–809 (2015)
20. Meng, Y.: Designing click-draw based graphical password scheme for better authentication. In: Proceedings of the 7th IEEE International Conference on Networking, Architecture, and Storage (NAS), pp. 39–48 (2012)
21. Meng, Y., Li, W.: Evaluating the effect of tolerance on click-draw based graphical password scheme. In: Chim, T.W., Yuen, T.H. (eds.) ICICS 2012. LNCS, vol. 7618, pp. 349–356. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34129-8_32
22. Meng, W.: RouteMap: a route and map based graphical password scheme for better multiple password memory. Network and System Security. LNCS, vol. 9408, pp. 147–161. Springer, Cham (2015). doi:10.1007/978-3-319-25645-0_10

23. Meng, W., Wong, D.S., Furnell, S., Zhou, J.: Surveying the development of biometric user authentication on mobile phones. IEEE Commun. Surv. Tutor. **17**(3), 1268–1293 (2015)
24. Meng, W., Li, W., Jiang, L., Meng, L.: On multiple password interference of touch screen patterns and text passwords. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI), pp. 4818–4822 (2016)
25. Passfaces. http://www.realuser.com/
26. Pinkas, B., Sander, T.: Securing passwords against dictionary attacks. In: Proceedings of the 9th ACM Conference on Computer and communications security (CCS), pp. 161–170 (2002)
27. Shin, J., Kancharlapalli, S., Farcasin, M., Chan-Tin, E.: SmartPass: a smarter geolocation-based authentication scheme. Secur. Commun. Netw. **8**(18), 3927–3938 (2015)
28. Sun, H.-M., Chen, Y.-H., Fang, C.-C., Chang, S.-Y.: PassMap: a map based graphical-password authentication system. In: Proceedings of the 7th ACM Symposium on Information, Compuer and Communications Security (ASIACCS), pp. 99–100 (2012)
29. Suo, X., Zhu, Y., Owen, G.S.: Graphical passwords: a survey. In: Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC), pp. 463–472 (2005)
30. Tao, H., Adams, C.: Pass-go: a proposal to improve the usability of graphical passwords. Int. J. Netw. Secur. **2**(7), 273–292 (2008)
31. Thorpe, J., MacRae, B., Salehi-Abari, A.: Usability and security evaluation of GeoPass: a geographic location-password scheme. In: Proceedings of the 2013 Symposium on Usable Privacy and Security (SOUPS), pp. 1–14 (2013)
32. van Oorschot, P.C., Stubblebine, S.: On countering online dictionary attacks with login histories and humans-in-the-loop. ACM Trans. Inf. Syst. Secur. **9**(3), 235–258 (2006)
33. van Oorschot, P.C., Salehi-Abari, A., Thorpe, J.: Purely automated attacks on passpoints-style graphical passwords. IEEE Trans. Inf. Forensics Secur. **5**(3), 393–405 (2010)
34. Vu, K.P.L., Proctor, R.W., Bhargav-Spantzel, A., Tai, B.-L., Cook, J., Schultz, E.E.: Improving password security and memorability to protect personal and organizational information. Int. J. Hum. Comput. Stud. **65**(8), 744–757 (2007)
35. Weir, M., Aggarwal, S., Collins, M., Stern, H.: Testing metrics for password creation policies by attacking large sets of revealed passwords. In: Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS), pp. 162–175 (2010)
36. Wiedenbeck, S., Waters, J., Birget, J.-C., Brodskiy, A., Memon, N.: PassPoints: design and longitudinal evaluation of a graphical password system. Int. J. Hum. Comput. Stud. **63**(1–2), 102–127 (2005)
37. Wiedenbeck, S., Waters, J., Birget, J.-C., Brodskiy, A., Memon, N.: Authentication using graphical passwords: effects of tolerance and image choice. In: Proceedings of the 1st Symposium on Usable Privacy and Security (SOUPS) (2005)
38. Yan, J., Blackwell, A., Anderson, R., Grant, A.: Password memorability and security: empirical results. IEEE Secur. Priv. **2**, 25–31 (2004)
39. Zhu, B.B., Yan, J., Bao, G., Yang, M., Xu, N.: Captcha as graphical passwords - a new security primitive based on hard AI problems. IEEE Trans. Inf. Forensics Secur. **9**(6), 891–904 (2014)

Security Analysis

Hierarchical Key Assignment with Dynamic Read-Write Privilege Enforcement and Extended KI-Security

Yi-Ruei Chen(✉) and Wen-Guey Tzeng

Department of Computer Science, National Chiao Tung University,
Hsinchu 30010, Taiwan
yrchen.cs98g@nctu.edu.tw, wgtzeng@cs.nctu.edu.tw

Abstract. This paper addresses the problem of key assignment for controlling access of encrypted data in access hierarchies. We propose a hierarchical key assignment (HKA) scheme RW-HKA that supports dynamic reading and writing privilege enforcement simultaneously. It not only provides typical confidentiality guarantee in data encryption, but also allows users to verify the integrity of encrypted data. It can be applied to cloud-based systems for providing flexible access control on encrypted data in the clouds. For security, we define the extended key indistinguishable (EKI) security for RW-HKA schemes. An EKI-secure RW-HKA scheme is resistant to collusion such that no subset of users can conspire to distinguish a data decryption key, that is not legally accessible, from random strings. In this paper, we provide a generic construction of EKI-secure RW-HKA schemes based on sID-CPA secure identity-based broadcast encryption (IBBE) and strong one-time signature schemes. Furthermore, we provide a new IBBE scheme that is suitable in constructing an efficient RW-HKA scheme with a constant number of user private keys, constant size of encrypted data, and constant computation cost of a user in deriving a key for decryption. It is the first HKA scheme that achieves the aforementioned performance while supporting dynamic reading and writing privilege enforcement simultaneously.

Keywords: Hierarchical key assignment · Access control · Data outsourcing

1 Introduction

We address the problem of key assignment in access hierarchies to ensure that only authorized users are allowed to access certain data. Users are organized as a hierarchy, which is a poset of n disjoint security classes, according to the roles and clearances of users. A user with access privilege of a class is granted to access to the data in the descendant classes. A hierarchical key assignment

The research was supported in part by project 104-2221-E-009-112-MY3, Taiwan.

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 165–183, 2017.
DOI: 10.1007/978-3-319-61204-1_9

(HKA) scheme is a method of assigning cryptographic keys to each class of the hierarchy. These keys are used to encrypt data and allow a user in a class to compute the data decryption keys of the descendant classes [1,18].

After the pioneer work of Akl and Taylor [1], the HKA problem has been extensively studied during the past 3 decades [1,12,18,38]. However, most of these schemes enforce dynamic reading privileges only. They cannot be directly applicable to enforce dynamic writing privileges simultaneously. An HKA scheme with dynamic reading and writing privileges enforcement can be used to provide flexible access control in cloud-based systems. For example, a private corporation can realize mobile office by outsourcing its encrypted data to the cloud and controlling the access right of its employees in a hierarchical structure [38]. This kind of HKA scheme is also useful in developing e-Health systems [33] and network-based computer systems in mobile ad hoc networks [28].

At the first glance, we can assign each class an asymmetric key pair (ek, dk) for data encryption and decryption. The writing and reading privileges can be separately assigned to the users by giving away ek and dk, respectively. Unfortunately, not all the asymmetric cryptosystems can be applicable to provide such a privilege separation successfully. For example, in the ElGamal encryption scheme, a user who knows dk can compute ek by itself. Given dk for the reading privilege would also give away ek for the writing privilege at the same time. Therefore, the separation of writing and reading privileges fails.

Contribution. In this paper, we propose a generic HKA scheme called *RW-HKA*, which supports dynamic reading and writing privilege enforcement simultaneously. The enforcement of privileges follows the known "no read-up" and "no write-down" principles. An authorized user in a class can only read the data in the descendant classes and write a datum into the ancestor classes in access hierarchies. Our RW-HKA scheme not only provides typical confidentiality guarantee in data encryption, but also allows users to verify the integrity of encrypted data. For security, we define the *extended key indistinguishable* (EKI) security for RW-HKA schemes. An EKI-secure RW-HKA scheme is resistant to collusion such that no subset of users can conspire to distinguish a data decryption key, that is not legally accessible, from random strings.

The construction of our generic RW-HKA scheme is based on sID-CPA secure identity-based broadcast encryption (IBBE) and strong one-time signature schemes. There are some suitable IBBE and signature schemes for implementing an RW-HKA scheme. In this paper, we provide a new IBBE scheme for constructing an efficient RW-HKA scheme in which the number of user private keys and the size of encrypted data are only small constants. In particular, the computation cost of a user in deriving the key of a descendant class for decryption is also a constant. This is the first HKA scheme that achieves the aforementioned performance while supporting dynamic reading and writing privilege enforcement.

Related Work. Akl and Taylor [1] first addressed the problem of assigning cryptographic keys in access hierarchies and proposed the first HKA scheme based on the hardness of factor problem. Later, Mackinnon et al. [31,32] improved the efficiency of Akl and Taylor's result by proposing an optimal algorithm for key generation and key derivation. Sandhu [34] addressed the rekeying issue for inserting or deleting classes and proposed a solution based on one-way functions in a specific situation that the security classes are organized as a rooted tree. Similar results were then proposed by Harn and Lin [26], Shen and Chen [37], Yang and Li [40], and Das et al. [21] for enhancing the cost for dynamic updates of access hierarchies. However, the above HKA schemes and some of their alternative schemes have two main common issues: inefficient rekeying and no formal security proofs. The cost of updating a cryptographic key is as the same as regenerating all the keys in the hierarchy. On the other hand, most of these schemes lacked rigorous security proofs. A relative large number of these schemes were shown to be vulnerable to collusion attacks [21,37,40].

To deal with the above issues, Atallah et al. [2,3] formalized two security notion of HKA schemes: key recovery (KR-security) and key indistinguishability (KI-security). KR-security ensures that a set of collusive users cannot obtain the keys for data decryption. In KI-security, the collusive users should not even be able to distinguish a data decryption key from random strings. Atallah et al. [2,3] also provided two dynamic and efficient approaches based on pseudorandom functions and symmetric key encryption schemes. The schemes are provably secure against KR-security and KI-security, respectively. After Atallah et al.'s work, many different constructions with the same security notions have been proposed [19,20,23,35,36]. De Santis et al. [35,36] provided two efficient constructions based on symmetric encryption schemes and public-key broadcast encryption schemes. D'Arco et al. [19,20] analyzed Akl and Taylor's scheme and considered different key assignment strategies to get the variations which are secure against KR-security. They also showed how to turn Akl and Taylor's scheme to be a KI-secure one. Freire and Paterson [23] provided a construction that is KI-secure based on the hardness of factoring Blum integers.

In 2013, Freire et al. [24] proposed a new security notion for HKA schemes called strong key indistinguishability (SKI-security). Comparing wit the KI-security, this notion provides an adversary with additional compromise ability to obtain data encryption keys. Freire et al. also proposed two SKI-secure HKA schemes based on pseudorandom functions and forward-secure pseudorandom generators. In 2014, Castiglione et al. [7] proved that the SKI-security notion is not stronger than the KI-security notion, i.e., these two security notions are equivalent. Cafaro et al. [6] showed a similar result in the unconditionally secure setting.

Recently, some of the approaches have been proposed to deal with the HKA problem in cloud computing [9,12,38]. Chen et al. [12] provided the first cloud-based HKA scheme called CloudHKA, which is proved to be KI-secure. CloudHKA first addressed the issue of revoking a user from accessing old data ciphers that were entitled to the user. In user revocation, the scheme outsources the cipher re-encryption to semi-trusted clouds by giving away some public re-encryption keys only. Tang et al. [38] proposed an KI-secure HKA scheme based on linear geometry. The scheme belongs to the directed derivation HKA schemes such as the schemes in [11,17,30], which only need to perform one (or constant) time of operation in key derivation. When comparing with other directed scheme, the scheme in Tang et al. [38] is the most efficient one in key derivation. Castiglione et al. [9] extended the conventional hierarchical access control model as a general one for certain additional sets of qualified users. They also proposed two KI-secure HKA schemes in the new model. The first construction is based on symmetric encryption schemes and secret sharing, and the second one is based on public-key threshold broadcast encryption schemes. Castiglione et al. [8] defined the concept of HKA schemes supporting dynamic updates and formalizing the relative security model. They also proposed a corresponding KI-secure HKA scheme with efficient key derivation and updating procedures, while requiring each user to store one private key only. Castiglione et al. [10] analyzed the Akl-Taylor scheme in the dynamic setting characterizing storage clouds. They considered different key assignment strategies and provided the corresponding schemes with KR-security.

2 Preliminaries

In this section, we introduce hierarchical access control (HAC) policy and two main building blocks of our generic RW-HKA scheme. The first one is identity-based broadcast encryption (IBBE) scheme and the second one is strong one-time signature scheme.

2.1 Hierarchical Access Control (HAC) Policy

A hierarchical access control (HAC) policy $\mathcal{P}$ is a 5-tuple $(\mathcal{SC}, \preceq, \mathcal{U}, \mathcal{D}, \lambda)$, where $\mathcal{SC} = \{SC_i : 1 \leq i \leq n\}$ is a set of security classes, $\preceq$ is a binary relation on $\mathcal{SC} \times \mathcal{SC}$, $\mathcal{U}$ is a set of users, $\mathcal{D}$ is a set of data, and $\lambda : \mathcal{U} \cup \mathcal{D} \rightarrow \mathcal{SC}$ is a function that associates each user and datum with a security class. $(\mathcal{SC}, \preceq)$ is a partial order set (poset) and $SC_j \preceq SC_i$ means that the security level of class SC_i is higher than or equal to that of SC_j. $\mathcal{P}$ requires the following two properties that indicate the "no read-up" and "no write-down" principles.

(1) *Simple security property*: A user $U \in \mathcal{U}$ can read a datum $D \in \mathcal{D}$ only if $\lambda(D) \preceq \lambda(U)$.
(2) $\star$*-property*: A user $U \in \mathcal{U}$ can write a datum $D \in \mathcal{D}$ only if $\lambda(U) \preceq \lambda(D)$.

The poset $(\mathcal{SC}, \preceq)$ is represented as a directed graph (access hierarchy) G. Each class SC_i is a node and there is a path of directed edges from SC_i to SC_j if and only if $SC_j \preceq SC_i$. G can be simplified by eliminating the edges that are implied by the transitive closure property. For example, Fig. 1 is an access hierarchy G with the nodes SC_1, SC_2, ..., SC_6 and directed edges (SC_1, SC_2), (SC_1, SC_3), (SC_2, SC_4), (SC_2, SC_5), (SC_3, SC_5), and (SC_3, SC_6).

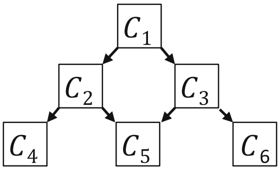


Fig. 1. An access hierarchy with 6 nodes and 6 directed edges.

2.2 Identity-Based Broadcast Encryption (IBBE) Scheme

An identity-based broadcast encryption (IBBE) scheme $\varPsi$ is a 4-tuple of poly-time algorithms $\varPsi = (\mathbf{Gen}, \mathbf{Ext}, \mathbf{Enc}, \mathbf{Dec})$, where

- $\mathbf{Gen}(\tau, n) \rightarrow (msk, mpk)$. On input a security pkameter τ and the maximal size n of the set of receivers for each encryption, this key generation algorithm outputs a master secret key msk and public key mpk.
- $\mathbf{Ext}(mpk, msk, id_i) \rightarrow dk_i$. On input mpk, msk, and a receiver's identity id_i, this key extraction algorithm outputs a private key dk_i.
- $\mathbf{Enc}(mpk, \mathcal{S}, k) \rightarrow \mathsf{Hdr}_{\mathcal{S}}$. On input mpk, a set of receivers' identities $\mathcal{S}$, and a symmetric encryption key k, this encryption algorithm outputs a header cipher $\mathsf{Hdr}_{\mathcal{S}}$. The header $\mathsf{Hdr}_{\mathcal{S}}$ is for the receivers in $\mathcal{S}$ to derive k and k is used to encrypt a message m as a cipher broadcasted to $\mathcal{S}$.
- $\mathbf{Dec}(mpk, \mathsf{Hdr}_{\mathcal{S}}, dk_i) \rightarrow k$ or $\perp$. On input mpk, $\mathsf{Hdr}_{\mathcal{S}}$, and dk_i, this decryption algorithm outputs k for decryption if $id_i \in \mathcal{S}$, and outputs the stop symbol $\perp$ otherwise.

For correctness, we require that for all key pairs (msk, mpk) generated by **Gen**, all secret keys dk_i output by **Ext**, and all encryption key k,

$$\mathbf{Dec}(mpk, \mathbf{Enc}(mpk, \mathcal{S}, k), dk_i) = k$$

A standard security notion for IBBE schemes is sID-CPA security [4]. This notion of security is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. In the beginning, $\mathcal{A}$ chooses a target identity set $\mathcal{S}^*$ that it wants to attack.

Setup phase: $\mathcal{C}$ runs $\mathbf{Gen}(\tau, n)$ to obtain (msk, mpk), and gives mpk to $\mathcal{A}$.

Query phase 1: In this phase, $\mathcal{A}$ is allowed to issue the key extraction oracle $\mathcal{O}_{ext}$. For a query of identity $id_i \notin \mathcal{S}^*$, $\mathcal{O}_{ext}(id_i)$ returns dk_i.

Challenge phase: After $\mathcal{A}$ decides that query phase 1 is over, $\mathcal{C}$ runs $\mathbf{Enc}(mpk, \mathcal{S}^*, k)$ to obtain $\mathsf{Hdr}_{\mathcal{S}^*}$. Then $\mathcal{C}$ flips a random coin $b \in \{0, 1\}$ and sets $k_b \leftarrow k$ and $k_{1-b} \leftarrow \$$, where \$ is a random string. $\mathcal{C}$ returns $(\mathsf{Hdr}_{\mathcal{S}^*}, k_0, k_1)$ to $\mathcal{A}$.

Query phase 2: In this phase, $\mathcal{A}$ is also allowed to issue the oracle queries as in query phase 1.

Guess phase: $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b.

$\mathcal{A}$ wins the sID-CPA security game if $b' = b$. The advantage of $\mathcal{A}$ in winning the sID-CPA security game is defined as:

$$\mathsf{Adv}_{\text{IBBE},\mathcal{A}}^{\text{sID-CPA}}(\tau) := |\Pr[\mathcal{A} \text{ outputs } 0|b = 0] - \Pr[\mathcal{A} \text{ outputs } 0|b = 1]|$$

We say that an IBBE scheme is sID-CPA secure if $\mathsf{Adv}_{\text{IBBE},\mathcal{A}}^{\text{sID-CPA}}(\tau)$ is negligible in τ for all probabilistic and poly-time adversary $\mathcal{A}$.

2.3 Strong One-Time Signature Scheme

A signature scheme Ω is a triple poly-time algorithms $\Omega = (\mathbf{KG}, \mathbf{Sig}, \mathbf{Ver})$, where

- $\mathbf{KG}(\kappa) \to (vk, sk)$. On input a security parameter κ, this key generation algorithm outputs a verification key vk and a signing key sk.
- $\mathbf{Sig}(sk, m) \to \sigma$. On input sk and a message m, this signing algorithm outputs a signature σ of m.
- $\mathbf{Ver}(vk, m, \sigma) \to b$. On input vk and a pair of (m, σ), this verification algorithm outputs a bit $b \in \{0, 1\}$, where $b = 1$ signifies "acceptance" and $b = 0$ signifies "rejection".

For correctness, we require that for all key pairs (vk, sk) generated by $\mathbf{KG}$ and all message m,

$$\mathbf{Ver}(vk, m, \mathbf{Sig}(sk, m)) = 1$$

In the standard security of one-time signature schemes, an adversary should be unable to forge a valid message and signature pair (m, σ) for all m chosen by the adversary. In the strong security case, it should be infeasible for an adversary to generate different signatures on the same message m. The security is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Setup phase. $\mathcal{C}$ runs $\mathbf{KG}(\kappa)$ to obtain (vk, sk) and gives vk to $\mathcal{A}$.

Attack phase. In this phase, $\mathcal{A}$ does one of the following two attacks. Firstly, $\mathcal{A}$ outputs (m^*, σ^*) for undefined m^* and σ^*. Secondly, $\mathcal{A}$ choses a message m and obtain its signature $\sigma^* \leftarrow \mathbf{Sig}(sk, m)$, then $\mathcal{A}$ outputs (m^*, σ^*) for $m^* \neq m$.

$\mathcal{A}$ wins the security game if $\mathbf{Ver}(vk, m^*, \sigma^*) = 1$. We say that an signature scheme is strong one-time secure if the probability of $\mathcal{A}$ wins the security game is negligible in κ for all probabilistic and poly-time adversary $\mathcal{A}$.

3 The Proposed RW-HKA Scheme

In this section, we first give an overview of our generic RW-HKA scheme and an intuition of its security. Then we provide a detailed description of our RW-HKA scheme, the corresponding EKI-security notion, and related analyzes in the rest sections.

3.1 An Overview

Our RW-HKA consists of the initialization algorithm **Initial** and key derivation algorithm **Derive**. In the beginning, the initializer runs **Initial** to generate all system parameters according to a given HAC policy $\mathcal{P}$, and assigns the keys for reading and writing access to authorized users accordingly. A user who has writing privilege of a class can encrypt-and-sign data into the corresponding classes such that an authorized user with appropriate reading privilege can run **Derive** for decryption.

More precisely, the initializer uses the signature scheme $\varOmega$ to generate a pair (vk_i, sk_i) of verification and signing keys for each class SC_i, and regards vk_i as the public "identity" of SC_i. The decryption key dk_j w.r.t. the identity vk_j is generated with the underlying IBBE scheme $\varPsi$. In key assignment, a user with reading privilege of SC_i is assigned dk_i for decrypting the data ciphers of SC_i and its descendant classes in the hierarchy. This is the simple security property (no read-up). A user with writing privilege of a class SC_i is assigned sk_z for generating a valid signature of Hdr_z in SC_z for $SC_i \preceq SC_z$. This is the $\star$-property (no write-down). The reading and writing privileges are separated in our RW-HKA since the generation of dk_i requires a master secret in $\varPsi$ and the generation of sk_i is independent of the generation of dk_i.

To encrypt a datum into SC_i, an authorized user takes the ancestor class set $\mathbb{A}_i = \{vk_z : SC_i \preceq SC_z\}$ as the broadcast set $\mathcal{S}$ and encrypts a data encryption key k with the underlying IBBE scheme $\varPsi$. The resulting Hdr_i is then signed by using sk_i to obtain σ_i. The final cipher consists of the verification key vk_i, the IBBE cipher Hdr_i, and the signature σ. To decrypt a cipher $c_i = (vk_i, \mathsf{Hdr}_i, \sigma_i)$ in class SC_i, an authorized user first verifies the signature of Hdr_i w.r.t. vk_i and uses dk_j to decrypt Hdr_i if the verification has passed. The final decryption is successful if $SC_i \preceq SC_j$ since the "identity" vk_j is in the target broadcast set $\mathbb{A}_i$ for encrypting k.

In defining the EKI-security for RW-HKA, we model the collusion of authorized users as an adversary $\mathcal{A}$ for distinguishing data encryption keys from random strings in non-corrupted classes. $\mathcal{A}$ is allowed to access the oracles for corrupting (dk_i, sk_i)'s and decrypting c_i's for certain chosen classes. Consider a cipher $c = (vk, \mathsf{Hdr}, \sigma)$ that is not supposed to be decryptable by $\mathcal{A}$. The security of $\varOmega$ ensures that any valid cipher $c' = (vk', \mathsf{Hdr}', \sigma')$ submitted by $\mathcal{A}$ to the decryption oracle have to satisfy $vk' \neq vk$. Simultaneously, the security of $\varPsi$ ensures that decryption of c' does not give $\mathcal{A}$ any further advantage in decrypting c.

3.2 Notion of RW-HKA Scheme with EKI-Security

In this section, we give a formal description of RW-HKA scheme and its corresponding EKI-security notion. The RW-HKA scheme $\varPi$ is a pair of poly-time algorithms $(\mathbf{Initial}, \mathbf{Derive})$, where

- $\mathbf{Initial}(\lambda, \tau, \mathcal{P}) \rightarrow (Pub, Sec)$. On input two security parameters λ, τ and an HAC policy $\mathcal{P}$, this initialization algorithm generates a pair (msk, mpk) of master secret key and public key. For each SC_i of $\mathcal{P}$, **Initial** generates a pair (ek_i, dk_i) of write-key and read-key and assigns them to the authorized users with writing and reading privileges accordingly. The public and secret information (Pub, Sec) are set as:

$$\begin{aligned} Pub &= \{\mathcal{P}, mpk\} \\ Sec &= \{msk\} \cup \{ek_i, dk_i : \forall SC_i\} \end{aligned}$$

An authorized user can use ek_i to encrypt data encryption key k_i as c_i and use dk_i to decrypt c_i.

- **Derive**$(\mathcal{P}, mpk, c_i, dk_j) \rightarrow k_i$ or $\perp$. On input $\mathcal{P}$, mpk, a cipher c_i, and dk_j, this derivation algorithm outputs the stop symbol $\perp$ if $SC_i \npreceq SC_j$. Otherwise, **Derive** outputs the data decryption key k_i.

Correctness. We require that given an HAC policy $\mathcal{P}$, for all system information (Pub, Sec) generated by **Initial**, for all c_i of SC_i, and for all SC_j with $SC_i \preceq SC_j$, we have

$$\textbf{Derive}(\mathcal{P}, mpk, c_i, dk_j) = k_i$$

To separate the assignment of read and write access privileges, we require that the write-key ek_i and read-key dk_i cannot be derived from each other.

The security notion for an RW-HKA scheme is the extended key indistinguishability (EKI). $\mathcal{A}$ is allowed to access all public information and all secret information associated with a number of classes of its choice, and chooses a non-corrupted class SC_{i^*} that it want to attack. Formally, this notion of security is defined by the following EKI-security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Setup phase. Given an HAC policy $\mathcal{P}$, $\mathcal{C}$ runs **Initial**$(\lambda, \tau, \mathcal{P})$ to generate (Pub, Sec) and gives Pub to $\mathcal{A}$.

Query phase 1. In this phase, $\mathcal{A}$ is allowed to issue the read-key corruption oracle $\mathcal{O}_{rc}$, the write-key corruption oracle $\mathcal{O}_{wc}$, and derivation oracle $\mathcal{O}_{der}$. For the query of a class index i and a valid access cipher c_i, $\mathcal{O}_{rc}(i)$, $\mathcal{O}_{wc}(i)$, and $\mathcal{O}_{der}(c_i)$ return the decryption key dk_i of SC_i, the encryption key ek_i of SC_i, and the data encryption key k_i of SC_i, respectively.

Challenge phase. After $\mathcal{A}$ decides that query phase 1 is over, $\mathcal{A}$ specifies a class index i^*, subject to $SC_{i^*} \npreceq SC_i$ for any corrupted dk_i in query phase 1. $\mathcal{C}$ flips a random coin $b \in \{0, 1\}$ and gives k_{i^*} of c_{i^*} to $\mathcal{A}$ if $b = 0$. Otherwise, it gives a random string \$ (with the same length of k_{i^*}) to $\mathcal{A}$.

Query phase 2. In this phase, $\mathcal{A}$ is also allowed to issue the oracle queries as in query phase 1, excepts that $\mathcal{O}_{rc}(i)$ for $SC_{i^*} \preceq SC_i$ and $\mathcal{O}_{der}(i^*)$.

Guess phase. $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b.

$\mathcal{A}$ wins the EKI-security game if $b' = b$. The advantage of $\mathcal{A}$ winning the EKI-security game is defined as

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{EKI}}(\lambda) := |\Pr[b' = b] - \frac{1}{2}|$$

We say that an RW-HKA scheme $\mathit{\Pi}$ is EKI-secure if $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{EKI}}(\lambda)$ is negligible in λ for all probabilistic and poly-time adversary $\mathcal{A}$.

3.3 A Generic Construction of RW-HKA Scheme

In this section, we provide our generic RW-HKA based on IBBE and signature schemes. Let $\mathbb{A}_i$ be the identity set of the ancestors of the class SC_i in $\mathcal{P}$, i.e., $\mathbb{A}_i = \{id_z : SC_i \preceq SC_z\}$. Let $\varPsi = (\varPsi.\mathbf{Gen}, \varPsi.\mathbf{Ext}, \varPsi.\mathbf{Enc}, \varPsi.\mathbf{Dec})$ be an sID-CPA secure IBBE scheme, and $\varOmega = (\varOmega.\mathbf{KG}, \varOmega.\mathbf{Sig}, \varOmega.\mathbf{Ver})$ be a strong one-time signature scheme. Our RW-HKA scheme $\varPi = (\mathbf{Initial}, \mathbf{Derive})$ is as follows.

- $\mathbf{Initial}(\lambda, \tau, \mathcal{P}) \rightarrow (Pub, Sec)$. On input two security parameters λ, τ and an HAC policy $\mathcal{P}$ with n security classes, this initialization algorithm first runs $\varPsi.\mathbf{Gen}(\lambda, n)$ to generate a pair (msk, mpk) of system master secret key and public key. For each class SC_i of $\mathcal{P}$, this algorithm generates verification and signing key pair $(vk_i, sk_i) \leftarrow \varOmega.\mathbf{KG}(\lambda)$, sets the identity id_i as the verification key vk_i, and assigns sk_i as the private part of encryption key ek_i. To generate the decryption key dk_i of SC_i, this algorithm runs $\varPsi.\mathbf{Ext}$ to obtain $dk_i \leftarrow \varPsi.\mathbf{Ext}(mpk, msk, id_i)$. The public and secret information (Pub, Sec) are set as:

$$Pub = \{\mathcal{P}, mpk\} \cup \{vk_i : \forall SC_i\}$$
$$Sec = \{msk\} \cup \{sk_i, dk_i : \forall SC_i\}$$

  The initializer keeps Sec as secret and publishes Pub. To assign access privileges, the initializer gives dk_i to the users who has reading privilege of SC_i, and gives sk_i to the users who has write privilege of SC_i though secure channels. To encrypt data, an authorized user randomly picks a data encryption key k_i and encrypts it as

$$c_i = (vk_i, \mathsf{Hdr}_i, \sigma_i),$$

  where $\mathsf{Hdr}_i \leftarrow \varPsi.\mathbf{Enc}(mpk,\ \mathbb{A}_i,\ k_i)$ and $\sigma_i \leftarrow \varOmega.\mathbf{Sig}(sk_i,\ \mathsf{Hdr}_i)$.
- $\mathbf{Derive}(\mathcal{P}, mpk, c_i, dk_j) \rightarrow k_i$ or $\perp$. On input $\mathcal{P}$, mpk, $c_i = (vk_i, \mathsf{Hdr}_i, \sigma_i)$, and dk_j, if $SC_j \prec SC_i$, this algorithm outputs $\perp$. Otherwise, this algorithm outputs k_i by computing

$$k_i \leftarrow \varPsi.\mathbf{Dec}(mpk, \mathsf{Hdr}_i, dk_j)$$

  if $\varOmega.\mathbf{Ver}(vk_i, \mathsf{Hdr}_{\mathbb{A}_i}, \sigma_i) = 1$ and outputs $\perp$ otherwise.

Correctness. Given an HAC policy $\mathcal{P}$, for all (Pub, Sec) generated by **Initial**, and for all SC_j with $SC_i \preceq SC_j$, we have $\mathbf{Derive}(\mathcal{P},\ mpk,\ c_i,\ dk_j) = k_i$ since c_i can pass the verification of the signature scheme $\varOmega$ and c_i is an encryption of k_i under the IBBE scheme $\varPsi$ with the broadcast set $\mathbb{A}_i$ that contains id_j. In addition, our RW-HKA scheme ensures that the write-key ek_i and read-key dk_i of a security class SC_i cannot be derived from each other since msk is kept secret and the generation of sk_i is independent of the generation of dk_i.

3.4 Supporting Dynamic Access Hierarchies and User Privileges

In this section, we describe the procedures of dealing with dynamic access hierarchies and user privileges. They are relation insertion, relation deletion, class insertion, and class deletion, as well as the procedures of revoking user privileges.

Relation Insertion. After inserting a new relation (SC_i, SC_j) into $\mathcal{P}$, the data owner updates $\mathcal{P}$ as $\mathcal{P}'$. The ciphers c_z of SC_z, $SC_z \preceq SC_i$, have to be updated since $\mathbb{A}_z$ is changed as $\mathbb{A}'_z \leftarrow \mathbb{A}_z \cup \mathbb{A}_i$. The initializer updates $c_z = (vk_z, \mathsf{Hdr}_z, \sigma_z)$ as $c'_z = (vk_z, \mathsf{Hdr}'_z, \sigma'_z)$, where $\mathsf{Hdr}'_i \leftarrow \varPsi.\mathbf{Enc}(mpk, \mathbb{A}'_z, k_z)$ and $\sigma'_i \leftarrow \varOmega.\mathbf{Sig}(sk_i, \mathsf{Hdr}'_z)$.

Relation Deletion. To delete a relation (SC_i, SC_j) from $\mathcal{P}$, the corresponding procedure is as the same as that in relation insertion with new $\mathbb{A}'_z \leftarrow \mathbb{A}_z \setminus \mathbb{A}_i$ for all $SC_z \prec SC_i$.

Class Insertion. To insert a new class SC_i into $\mathcal{P}$, the initializer associates SC_i with the class secrets as in our RW-HKA scheme. Then the initializer runs relation insertion to insert the incoming and outgoing relations of SC_i.

Class Deletion. To delete a class SC_i from $\mathcal{P}$, the initializer removes the associated information of SC_i, and deletes each of the incoming and outgoing relations of SC_i by using relation deletion.

Access Right Revocation. In our RW-HKA, an authorized user has writing or reading privilege of some classes. To revoke the writing privilege of SC_i from a user, the initializer updates (vk_i, sk_i) to $(vk'_i, sk'_i) \leftarrow \varOmega.\mathbf{KG}(\lambda)$ and updates id_i and ek_i accordingly. Then the initializer redistributes the new sk'_i to the other users who has writing privilege of SC_i through secure channels. Afterthen, the revoked user cannot generate a new c_i since it cannot compute a valid signature that can pass the verification with vk'_i.

To revoke the reading privilege of SC_i from a user, the initializer needs to remove ability of decrypting the ciphers associated with the descendant classes of SC_i. For each class $SC_z \preceq SC_i$, the initializer updates (vk_z, sk_z) to $(vk'_z, sk'_z) \leftarrow \varOmega.\mathbf{KG}(\lambda)$, and updates id_i and ek_i accordingly. The initializer also needs to update dk_z to $dk'_z \leftarrow \varPsi.\mathbf{Ext}(mpk, msk, id'_z)$ and c_i to c'_i with newly generated k'_z. Finally, the initializer redistributes the new ek'_z and dk'_z, $SC_z \preceq SC_i$, to the corresponding users through secure channels.

Local Rekeying. In the end of revoking the access privileges of a user from SC_i, the key redistribution is required in each of the descendant or ancestor classes of SC_i. It consumes a large bandwidth for transmission if there are many users in the affected classes. Similar to the results in [2,12,36], our RW-HKA can also support "local rekeying" with a slight modification. With the local rekeying property, the key redistribution is only required locally, i.e. only in SC_i, and do not "propagate" to the descendant or ancestor classes in the hierarchy. We modify our RW-HKA for supporting local rekeying property as follow.

In each class SC_z, the initializer prepares two extra symmetric keys rk_z and wk_z to encrypt dk_z and sk_z as the public access-tokens rt_z and wt_z, respectively. Rather than assigning dk_z and sk_z, the initializer assigns the private

rk_z and wk_z to the authorized users with writing and reading privileges, respectively. A user can obtain the keys dk_z and sk_z by using rk_z and wk_z to decrypt rt_z and wt_z, respectively.

Now, for the revocation of user privileges in SC_i, the updates of dk_z and sk_z in the other classes can be done by updating the public tokens rt_z and wt_z without changing the user private keys rk_z and wk_z. The key redistribution is only occurred in the local class SC_i since only rk_i and wk_i are required to be updated. Furthermore, we can use the group key management (GKM) schemes such as LKH [39] to maintain the keys rk_i and wk_i in each class SC_i. Here we recommend to use the semi-stateful GKM schemes [13–16] for practical application scenarios.

3.5 Security Analysis

In this section, we proof the security of our generic RW-HKA scheme. The following theorem shows that our RW-HKA scheme $\varPi$ is EKI-secure based on the sID-CPA security of $\varPsi$ and the strong one-time security of $\varOmega$.

Theorem 1 (EKI-secure RW-HKA). *If $\varPsi$ is a sID-CPA secure IBBE scheme and $\varOmega$ is a strong one-time signature scheme, our generic RW-HKA scheme $\varPi$ is secure against extended key indistinguishability.*

Proof. We show how to turn an adversary $\mathcal{A}$ against our RW-HKA scheme $\varPi$ into a forger against the signature scheme $\varOmega$ and an attacker against the IBBE scheme $\varPsi$. Assume that $\mathcal{A}$ specifies the class SC_{i^*} to attack.

Let SC_1, SC_2, …, $SC_m = SC_{i^*}$ be a topological ordering of the ancestor classes of SC_{i^*} in $\mathcal{P}$. We define a sequence of computational indistinguishable games $\mathbf{G}_0$, $\mathbf{G}_1$, …, $\mathbf{G}_m$. Game $\mathbf{G}_0$ is identical to the original EKI-security game of $\varPi$. From $\mathbf{G}_0$ to $\mathbf{G}_m$, the later one incrementally makes a slight modification to the previous one while maintaining the indistinguishability among these two games on $\mathcal{A}$'s view. In each game $\mathbf{G}_i$, $0 \leq i \leq m$, the goal of $\mathcal{A}$ is to output a correct guess b' of b. More precisely, game $\mathbf{G}_i$ is defined as follows.

Game $\mathbf{G}_i, 1 \leq i \leq m$. This game is identical to game $\mathbf{G}_{i-1}$, except that the **Initial** algorithm is modified in such a way that the data encryption key k_i is substituted with a random string \$.

Let E_i be the event that $b' = b$ in Game $\mathbf{G}_i$. Let F_i be the event that $\mathcal{A}$ makes a valid forger of the challenged access cipher c_{i^*}, i.e., $\mathcal{A}$ submits a valid cipher $c'_{i^*} = (vk_{i^*}, \mathsf{Hdr}'_{i^*}, \sigma'_{i^*})$ with $(\mathsf{Hdr}'_{i^*}, \sigma'_{i^*}) \neq (\mathsf{Hdr}_{i^*}, \sigma_{i^*})$ to the derivation oracle $\mathcal{O}_{der}$. We have the following two lemmas:

Lemma 1. $\Pr[F_i]$ is negligible.
Lemma 2. $|\Pr[E_{i-1} \wedge \overline{F}_{i-1}] - \Pr[E_i \wedge \overline{F}_i]|$ is negligible.

Lemma 1 ensures that $\mathcal{A}$ cannot forge a signature of the challenged cipher Hdr_{i^*}, i.e., it cannot obtain k_{i^*} by feeding Hdr_{i^*} to the derivation oracle $\mathcal{O}_{der}$. Besides the case of forgery, Lemma 2 ensures that $\mathcal{A}$ cannot distinguish the difference between game $\mathbf{G}_{i-1}$ and game $\mathbf{G}_i$.

To see that these two claims imply the theorem, we have:

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathrm{EKI}}(\lambda) &= |\Pr[b' = b] - \frac{1}{2}| \\ &\leq \sum_{i=1}^{m} |\Pr[E_{i-1}] - \Pr[E_i]| \\ &\leq \sum_{i=1}^{m} |\Pr[F_{i-1}] - \Pr[F_i]| + \sum_{i=1}^{m} |\Pr[E_{i-1} \wedge \overline{F}_{i-1}] - \Pr[E_i \wedge \overline{F}_i]|,\end{aligned}$$

since $\Pr[E_i]$ can be expressed as

$$\begin{aligned}\Pr[E_i] &\leq |\Pr[E_i \wedge F_i] - \frac{1}{2}\Pr[F_i]| + |\Pr[E_i \wedge \overline{F}_i] + \frac{1}{2}\Pr[F_i]| \\ &\leq \frac{1}{2}\Pr[F_i] + \Pr[E_i \wedge \overline{F}_i] + \frac{1}{2}\Pr[F_i] \\ &= \Pr[F_i] + \Pr[E_i \wedge \overline{F}_i]\end{aligned}$$

Note that in the last game $\mathbf{G}_m$, there is no information about the key $k_m = k_{i^*}$ is presented on $\mathcal{A}$'s view. It follows that the probability of a correct guess of b by $\mathcal{A}$ in game $\mathbf{G}_m$ is $\frac{1}{2}$, i.e., $\Pr[T_m] = \frac{1}{2}$.

Thus, $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{EKI}}(\lambda)$ is negligible given the two lemmas. This concludes the proof of the theorem. □

Lemma 1. $\Pr[F_i]$ *is negligible.*

Proof. We construct a probabilistic poly-time forger $\mathcal{F}$ that forges the signature σ_{i^*} w.r.t. the signature scheme Ω with probability $\Pr[F_i]$ as follows.

Given an HAC policy $\mathcal{P}$, $\mathcal{F}$ runs $\mathbf{Initial}(\lambda, \tau, \mathcal{P})$ to generate (Pub, Sec) and gives Pub to $\mathcal{A}$. $\mathcal{F}$ answers the oracle queries of $\mathcal{A}$ as in the EKI-security game. If $\mathcal{A}$ submits a valid access cipher $c_{i^*} = (vk_{i^*}, \mathsf{Hdr}'_{i^*}, \sigma'_{i^*})$ with $(\mathsf{Hdr}'_{i^*}, \sigma'_{i^*}) \neq (\mathsf{Hdr}_{i^*}, \sigma_{i^*})$ to the derivation oracle $\mathcal{O}_{der}$ in the query phase, $\mathcal{F}$ outputs $(\mathsf{Hdr}'_{i^*}, \sigma'_{i^*})$ as its forgery and stops. It is easy to see that the success probability of $\mathcal{F}$ to forge the signature at SC_{i^*} is exactly $\Pr[F_i]$. Thus, the security of the signature scheme Ω implies this lemma. □

Lemma 2. $|\Pr[E_{i-1} \wedge \overline{F}_{i-1}] - \Pr[E_i \wedge \overline{F}_i]|$ *is negligible.*

Proof. Assume that $\mathcal{A}$ is able to distinguish game $\mathbf{G}_i$ and game $\mathbf{G}_{i-1}$. We use $\mathcal{A}$ to construct a probabilistic poly-time adversary $\mathcal{B}$ which breaks the sID-CPA security of Ψ as follows.

1. For a given HAC policy $\mathcal{P}$, the challenger $\mathcal{C}_\Psi$ generates $(msk, mpk) \leftarrow \mathbf{Gen}(\tau, n)$ and gives pk to $\mathcal{B}$.

2. $\mathcal{B}$ runs the following modified $\textbf{Initial}(\lambda, \tau, \mathcal{P})$ to generates (Pub, Sec) and gives Pub to $\mathcal{A}$. In the beginning, $\mathcal{B}$ generates $(msk, mpk) \leftarrow \Psi.\textbf{Gen}(\lambda, n)$. For each SC_z of $\mathcal{P}$, $\mathcal{B}$ generates (vk_z, sk_z), and k_z, and sets id_z and ek_z as in our RW-HKA scheme. To generate dk_z of each SC_z, $SC_{i^*} \npreceq SC_z$, $\mathcal{B}$ makes the key extraction oracle query to obtain $dk_z \leftarrow \mathcal{O}_{ext}(id_z)$. For each $SC_z \neq SC_i$, $\mathcal{B}$ generates the cipher $c_z \leftarrow (vk_z, \mathsf{Hdr}_z, \sigma_z)$ as in our RW-HKA scheme. To generate c_i, $\mathcal{B}$ asks $\mathcal{C}_\Psi$ for the challenged message (Hdr_i, k_0, k_1), where $(k_b, k_{1-b}) \leftarrow (k, \$)$ with a random bit b chosen by $\mathcal{C}_\Psi$, and $\mathsf{Hdr}_i \leftarrow \textbf{Enc}(ek_i, \mathbb{A}_i, k)$. $\mathcal{B}$ sets $c_i \leftarrow (vk_i, \mathsf{Hdr}_i, \sigma_i)$ and replaces k_i as k_0.
3. $\mathcal{B}$ answers the oracle queries of $\mathcal{A}$ as follows. For a read-key corruption oracle query $\mathcal{O}_{rc}(z)$, $\mathcal{B}$ returns dk_z if $SC_{i^*} \npreceq SC_z$ and $\perp$ otherwise. For a write-key corruption oracle query $\mathcal{O}_{wc}(z)$, $\mathcal{B}$ returns sk_z. For a derivation oracle query $\mathcal{O}_{der}(z)$, $\mathcal{B}$ returns k_z if $SC_z \neq SC_{i^*}$ and $\perp$ otherwise.
4. At some point, $\mathcal{B}$ starts the challenge phase with $\mathcal{A}$ as in the EKI-security game. Afterthen, $\mathcal{A}$ can make more oracle queries as the above and returns its best answer in the end. Finally, $\mathcal{B}$ outputs 0 if $\mathcal{A}$ wins the game and outputs 1 otherwise.

In the above construction, $\mathcal{B}$ simulates the environment of $\mathcal{A}$ through interpolating between game $\mathbf{G}_{i-1}$ and game $\mathbf{G}_i$. In the generation c_i, $\mathcal{B}$ embeds the given Hdr_i into c_i and sets k_i as k_0. This is equivalent to game $\mathbf{G}_0$ if $b = 0$ since c_i is the cipher of k_0, and equivalent to game $\mathbf{G}_1$ otherwise. Eventually, $\mathcal{A}$ outputs its best answer. If $\mathcal{A}$ wins the game, $\mathcal{B}$ outputs 0, guessing for that $\mathcal{C}_\Psi$ encrypts k_0 as $\mathsf{Hdr}_{\mathbb{A}_i}$, and outputs 1 otherwise. Now we have

$$\begin{aligned}\mathsf{Adv}^{\text{sID-CPA}}_{\Psi,\mathcal{B}}(\lambda) &= |\Pr[\mathcal{B} \text{ outputs } 0|b=0] - \Pr[\mathcal{B} \text{ outputs } 0|b=1]| \\ &= |\Pr[\mathcal{A} \text{ wins } |\mathbf{G}_0] - \Pr[\mathcal{A} \text{ wins } |\mathbf{G}_1]| \\ &= |\Pr[E_{i-1} \wedge \overline{F}_{i-1}] - \Pr[E_i \wedge \overline{F}_i]|\end{aligned}$$

Thus, the security of the IBBE scheme Ψ implies this lemma. □

3.6 Efficiency Analysis

In this section, we illustrate the efficiency of our generic RW-HKA scheme. We also demonstrate the efficiency of the result scheme when applying the existing and our newly constructed IBBE schemes.

Efficiency of Generic RW-HKA Scheme. For storage cost, the size of private information Sec is $O(|msk| + n)$ and the the size of public information Pub is $O(|\mathcal{P}| + |mpk| + n)$. For a user who has reading privilege of a class, it only needs to store one private key. A user who has writing privilege of a class SC_i is assigned $O(|\mathbb{A}_i|)$ keys for encrypting data into the ancestor classes of SC_i in the hierarchy. Each encrypted datum requires $O(|\mathsf{Hdr}_i|)$ of spaces for its header cipher.

For computation cost, the initializer runs one time of $\Psi.\textbf{Gen}$, n times of $\Omega.\textbf{KG}$, and n times of $\Psi.\textbf{Ext}$ in generating system parameters. A user runs one time of Ψ.**Enc** and one time of Ω.**Sig** for encrypting a datum. To derive a data encryption key, a user only needs to run one time of Ω.**Ver** and one time of Ψ.**Dec**.

Efficiency of Concrete RW-HKA Schemes. The requirement of the IBBE scheme for our RW-HKA scheme is the sID-CPA security. Most of the existing IBBE schemes satisfy this basic security notion. Table 1 shows the efficiency of our RW-HKA scheme when applying the concrete IBBE schemes proposed by Baek et al. [4], Delerablée [22], Gentry and Waters [25], Boneh et al. [5], Kim et al. [29], He et al. [27], and our newly constructed IBBE scheme proposed in Sect. 4, respectively.

Table 1. Efficiency of our RW-HKA scheme when applying concrete IBBE schemes

Applied IBBE	$\lvert mpk \rvert$	$\lvert dk_i \rvert$	$\lvert \mathsf{Hdr}_i \rvert$	der. cost	Security
Baek et al. [4]	$O(1)$	$O(1)$	$O(\lvert \mathbb{A}_i \rvert)$	$O(1)$	sID-CCA
Delerablée [22]	$O(n)$	$O(1)$	$O(1)$	$O(\lvert \mathbb{A}_i \rvert)$	sID-CPA
Gentry and Waters [25]	$O(n)$	$O(1)$	$O(\sqrt{n})$	$O(\sqrt{n})$	sID-CPA
Boneh et al. [5]	$O(\log n)$	$O(1)$	$O(1)$	$O(\lvert \mathbb{A}_i \rvert)$	sID-CPA
Kim et al. [29]	$O(n)$	$O(n)$	$O(1)$	$O(\lvert \mathbb{A}_i \rvert)$	sID-CCA
He et al. [27]	$O(1)$	$O(1)$	$O(\lvert \mathbb{A}_i \rvert)$	$O(1)$	sID-CCA
Ours	$O(n^2)$	$O(1)$	$O(1)$	$O(1)$	sID-CPA

† n: the number of security classes in hierarchy.

† $\mathbb{A}_i$: the set of ancestor classes of SC_i in hierarchy.

We consider the size of public information $|mpk|$, the number of private key of a user $|dk_i|$, the size of header cipher $|\mathsf{Hdr}_i|$ of each encrypted datum, and the computation cost of a user in deriving the decryption key of a datum. Here n is the number of security classes in the hierarchy and $\mathbb{A}_i$ is the set of the ancestor classes of SC_i in the hierarchy.

In Baek et al. [4] scheme, the size of public keys, user private keys, and decryption cost are constants. Thus, when applying this scheme for our RW-HKA, $|mpk|$, $|dk_i|$ and decryption cost are only constants. But the size of each cipher in Baek et al. [4] scheme is linear to the number of receivers so that $|\mathsf{Hdr}_i|$ of each cipher is linear to $|\mathbb{A}_i|$ in the result RW-HKA. It is not suitable in storing a large number of encrypted data since the total size of encrypted data is $O(\#(\text{data}) \cdot |\mathbb{A}_i|)$. In Delerablée [22], Boneh et al. [5], and Kim et al. [29] schemes, the size of each cipher remains a constant. When applying these schemes to our RW-HKA, it is efficient in storing many encrypted data since $|\mathsf{Hdr}_i|$ remains a constant. However, the decryption cost in these three schemes are linear to the number of receivers. It causes that each user needs $O(|\mathbb{A}_i|)$ of computation cost for key derivation in our RW-HKA. From Table 1, we see that only our IBBE scheme can provide constant size of $|dk_i|$ and $|\mathsf{Hdr}_i|$, and constant key derivation cost for RW-HKA simultaneously. Although the size $|mpk|$ of public information in our IBBE scheme is $O(n^2)$, a user of class SC_i only needs to take $|\mathbb{A}_i|$ of them in each time of encryption and $O(1)$ of them in each time of decryption.

4 Our Concrete Construction of IBBE Scheme

In this section, we propose a new IBBE scheme Ψ, which is proved to be sID-CPA secure. The scheme is modified from the IBBE scheme in Appendix A proposed by Delerablée [22]. The construction of our IBBE scheme $\Psi = (\mathbf{Gen}, \mathbf{Ext}, \mathbf{Enc}, \mathbf{Dec})$ is demonstrated as follows.

- $\mathbf{Gen}(\tau, n) \rightarrow (msk, mpk)$. Given a security parameter τ and an integer n, this algorithm constructs a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative groups with prime order p and $|p| = \tau$. This algorithm randomly picks two generators $g \in \mathbb{G}$ and $h \in \mathbb{G}_T$, a value $\gamma \in \mathbb{Z}_p^*$, and a cryptographic hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. This algorithm then computes $w = g^\gamma$, $v = \hat{e}(g, h)$, an n-vector $\mathbf{t} = (t_i)_{\forall SC_i}$ with $t_i = \prod_{z \in \mathbb{A}_i} \mathcal{H}(id_z)$, an $(n+1)$-vector $\mathbf{h} = (h, h^\gamma, \ldots, h^{\gamma^n})$ and an $n \times n$ matrix $\mathbf{D} = [d_{i,j}]_{n \times n}$ with entries
$$d_{i,j} = \begin{cases} h^{\gamma^{-1} \cdot \Delta_{i,j}} & \text{if } SC_i \preceq SC_j \\ 0 & \text{otherwise} \end{cases}, \text{ where}$$
$$\Delta_{i,j} = \prod_{z \in \mathbb{A}_i \setminus \{j\}} (\gamma + \mathcal{H}(id_z)) - t_i$$
The master secret key msk and public key mpk are defined as
$$msk = (g, \gamma)$$
$$mpk = (p, \mathbb{G}, \mathbb{G}_T, \hat{e}, w, v, \mathbf{t}, \mathbf{h}, \mathbf{D}, \mathcal{H})$$
- $\mathbf{Ext}(mpk, msk, id_i) \rightarrow dk_i$. Given mpk, msk, and an identity id_i, this algorithm outputs the private decryption key dk_i as
$$dk_i = g^{\frac{1}{\gamma + \mathcal{H}(id_i)}}$$
- $\mathbf{Enc}(mpk, \mathbb{A}_i, k_i) \rightarrow \mathsf{Hdr}_i$. Given mpk, $\mathbb{A}_i$, and a symmetric encryption key k_i, this algorithm randomly picks a value $r \in \mathbb{Z}_p^*$, and computes $\mathsf{Hdr}_i = (c_0, c_1, c_2)$, where
$$c_0 = k \cdot v^r$$
$$c_1 = w^{-r}$$
$$c_2 = h^{r \prod_{z \in \mathbb{A}_i} (\gamma + \mathcal{H}(id_z))}$$
- $\mathbf{Dec}(mpk, \mathsf{Hdr}_i, dk_j) \rightarrow k$ or $\perp$. Given mpk, $\mathsf{Hdr}_i = (c_0, c_1, c_2)$, and dk_j, if $SC_i \npreceq SC_j$, this algorithm outputs $\perp$. Otherwise, this algorithm computes
$$x = (\hat{e}(c_1, d_{i,j}) \cdot \hat{e}(dk_j, c_2))^{1/t_i}$$
$$k_i = c_0 / x$$

We verify the correctness of the decryption process as follows.

$$\begin{aligned}
x &= (\hat{e}(c_1, d_{i,j}) \cdot \hat{e}(dk_j, c_2))^{1/t_i} \\
&= (\hat{e}(w^{-r}, h^{\gamma^{-1}\cdot\Delta_{i,j}}) \cdot \hat{e}(g^{\frac{1}{\gamma+\mathcal{H}(id_j)}}, h^{r\cdot(\Delta_{i,j}+t_i)\cdot(\gamma+\mathcal{H}(id_j))}))^{1/t_i} \\
&= (\hat{e}(g, h)^{-r\cdot\Delta_{i,j}} \cdot \hat{e}(g, h)^{r\cdot(\Delta_{i,j}+t_i)})^{1/t_i} \\
&= (\hat{e}(g, h)^{r\cdot t_i})^{1/t_i} = v^r
\end{aligned}$$

Thus, we have

$$c_0/x = (k_i \cdot v^r)/v^r = k_i$$

The security of our IBBE scheme Ψ is stated with the following theorem. The key extraction and encryption algorithms are as the same with the scheme proposed by Delerablée [22]. In decryption, we move most of the computation to be pre-computed the key generation algorithm so that the decryptor only needs a constant time of operations in the result scheme. Notably, the pre-computational tasks are accomplished by using public information only. Thus, our IBBE scheme can be shown to be sID-CPA security with a standard reduction argument from the IBBE scheme proposed by Delerablée [22].

Theorem 2 (sID-CPA Security of Our IBBE Scheme). *Our IBBE scheme Ψ is secure against sID-CPA security.*

5 Conclusions

In this paper, we propose a generic HKA scheme called RW-HKA, which supports dynamic reading and writing privilege enforcement simultaneously. It not only provides typical confidentiality guarantee in data encryption, but also allows users to verify the integrity of encrypted data. The construction is based on sID-CPA secure IBBE and strong one-time signature schemes. It is proved to be secure against EKI-security, which is resistant to collusion of users for illegal accesses. We also provide a new IBBE scheme for constructing an efficient RW-HKA scheme with a constant number of user private keys, constant size of encrypted data, and constant computation cost of a user in deriving a key for decryption. It is the first HKA scheme that achieves such performance while supporting reading and writing privilege enforcement simultaneously.

A D07's IBBE Scheme [22]

- **Gen**$(\tau, n) \rightarrow (msk, pk)$. Given a security pkameter τ and an integer n, this algorithm constructs a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative groups with prime order p and $|p| = \tau$. This algorithm randomly picks two generators $g \in \mathbb{G}$ and $h \in \mathbb{G}_T$, a value $\gamma \in \mathbb{Z}_p^*$, and a cryptographic hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$. The master secret key msk and the public key pk are defined as

$$msk = (g, \gamma), pk = (p, \mathbb{G}, \mathbb{G}_T, \hat{e}, w, v, h, h^{\gamma}, \ldots, h^{\gamma^n}, \mathcal{H}),$$

where $w = g^{\gamma}$ and $v = \hat{e}(g, h)$.

- **Ext**$(pk, msk, id_i) \rightarrow dk_i$. Given pk, msk, and an identity id_i, this algorithm outputs the private key dk_i as

$$dk_i = g^{\frac{1}{\gamma + \mathcal{H}(id_i)}}$$

- **Enc**$(pk, \mathcal{S}, k) \rightarrow \mathsf{Hdr}_{\mathcal{S}}$. Given pk, a set $\mathcal{S}$ of some identities, and a symmetric encryption key k, this algorithm randomly picks a value $r \xleftarrow{\$} \mathbb{Z}_p^*$, and computes $\mathsf{Hdr}_{\mathcal{S}} = (c_0, c_1, c_2)$, where

$$c_0 = k \cdot v^r, c_1 = w^{-r}, c_2 = h^{r \prod_{id_j \in \mathcal{S}} (\gamma + \mathcal{H}(id_j))}$$

- **Dec**$(pk, \mathsf{Hdr}_{\mathcal{S}}, dk_i) \rightarrow k$ or $\perp$. Given pk, $\mathsf{Hdr}_{\mathcal{S}} = (c_0, c_1, c_2)$, and dk_i, if $id_i \notin \mathcal{S}$, this algorithm outputs $\perp$. Otherwise, this algorithm computes

$$k = (\hat{e}(c_1, h^{\Delta_{\gamma}(id_i, \mathcal{S})}) \cdot \hat{e}(dk_i, c_2))^{\frac{1}{\prod_{id_j \mathcal{S} \wedge id_j \neq id_i} \mathcal{H}(id_j)}},$$

where

$$\Delta_{\gamma}(id_i, \mathcal{S}) = \gamma^{-1}(\prod_{id_j \mathcal{S} \wedge id_j \neq id_i} (\gamma + \mathcal{H}(id_j) - \prod_{id_j \mathcal{S} \wedge id_j \neq id_i} \mathcal{H}(id_j))$$

References

1. Akl, S.G., Taylor, P.D.: Cryptographic solution to a problem of access control in a hierarchy. ACM Trans. Comput. Syst. (TOCS) **1**(3), 239–248 (1983)
2. Atallah, M.J., Blanton, M., Fazio, N., Frikken, K.B.: Dynamic and efficient key management for access hierarchies. ACM Trans. Inf. Syst. Secur. **12**(3), 18:1–18:43 (2009)
3. Mikhail, J.A., Keith, B.F., Marina, B.: Dynamic and efficient key management for access hierarchies. In: Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, 7–11 November 2005, pp. 190–202 (2005)
4. Baek, J., Safavi-Naini, R., Susilo, W.: Efficient multi-receiver identity-based encryption and its application to broadcast encryption. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 380–397. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30580-4_26
5. Boneh, D., Waters, B., Zhandry, M.: Low overhead broadcast encryption from multilinear maps. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 206–223. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44371-2_12
6. Cafaro, M., Civino, R., Masucci, B.: On the equivalence of two security notions for hierarchical key assignment schemes in the unconditional setting. IEEE Trans. Dependable Secure Comput. **12**(4), 485–490 (2015)

7. Castiglione, A., De Santis, A., Masucci, B.: Key indistinguishability versus strong key indistinguishability for hierarchical key assignment schemes. IEEE Trans. Dependable Secure Comput. **13**(4), 451–460 (2016)
8. Castiglione, A., De Santis, A., Masucci, B., Palmieri, F., Castiglione, A., Huang, X.: Cryptographic hierarchical access control for dynamic structures. IEEE Trans. Inf. Forensics Secur. **11**(10), 2349–2364 (2016)
9. Castiglione, A., De Santis, A., Masucci, B., Palmieri, F., Castiglione, A., Li, J., Huang, X.: Hierarchical and shared access control. IEEE Trans. Inf. Forensics Secur. **11**(4), 850–865 (2016)
10. Castiglione, A., De Santis, A., Masucci, B., Palmieri, F., Huang, X., Castiglione, A.: Supporting dynamic updates in storage clouds with the Akl-Taylor scheme. Inf. Sci. **387**, 56–74 (2017)
11. Chen, T.-S., Chung, Y.-F.: Hierarchical access control based on chinese remainder theorem and symmetric algorithm. Comput. Secur. **21**(6), 565–570 (2002)
12. Chen, Y.-R., Chu, C.-K., Tzeng, W.-G., Zhou, J.: CloudHKA: a cryptographic approach for hierarchical access control in cloud computing. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 37–52. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38980-1_3
13. Chen, Y.-R., Tygar, J.D., Tzeng, W.-G.: Secure group key management using unidirectional proxy re-encryption schemes. In: 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2011, 10–15 April 2011, Shanghai, China, pp. 1952–1960 (2011)
14. Chen, Y.-R., Tzeng, W.-G.: Efficient and provably-secure group key management scheme using key derivation. In: 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2012, Liverpool, United Kingdom, 25–27 June 2012, pp. 295–302 (2012)
15. Chen, Y.-R., Tzeng, W.-G.: Group key management with efficient rekey mechanism: a semi-stateful approach for out-of-synchronized members. Comput. Commun. **98**, 31–42 (2017)
16. Chou, K.-Y., Chen, Y.-R., Tzeng, W.-G.: An efficient and secure group key management scheme supporting frequent key updates on pay-tv systems. In: 13th Asia-Pacific Network Operations and Management Symposium, APNOMS 2011, Taipei, Taiwan, 21–23 September 2011, pp. 1–8 (2011)
17. Chung, Y.-F., Lee, H.-H., Lai, F., Chen, T.-S.: Access control in user hierarchy based on elliptic curve cryptosystem. Inf. Sci. **178**(1), 230–243 (2008)
18. Crampton, J., Martin, K.M., Wild, P.R.: On key assignment for hierarchical access control. In: 19th IEEE Computer Security Foundations Workshop, (CSFW-19 2006), Venice, Italy, 5–7 July 2006, pp. 98–111 (2006)
19. D'Arco, P., Santis, A., Ferrara, A.L., Masucci, B.: Security and tradeoffs of the Akl-Taylor scheme and its variants. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 247–257. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03816-7_22
20. D'Arco, P., De Santis, A., Ferrara, A.L., Masucci, B.: Variations on a theme by Akl and Taylor: security and tradeoffs. Theor. Comput. Sci. **411**(1), 213–227 (2010)
21. Das, M.L., Saxena, A., Gulati, V.P., Phatak, D.B.: Hierarchical key management scheme using polynomial interpolation. Oper. Syst. Rev. **39**(1), 40–47 (2005)
22. Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 200–215. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76900-2_12

23. Freire, E.S.V., Paterson, K.G.: Provably secure key assignment schemes from factoring. In: Parampalli, U., Hawkes, P. (eds.) ACISP 2011. LNCS, vol. 6812, pp. 292–309. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22497-3_19
24. Freire, E.S.V., Paterson, K.G., Poettering, B.: Simple, efficient and strongly KI-secure hierarchical key assignment schemes. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 101–114. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36095-4_7
25. Gentry, C., Waters, B.: Adaptive security in broadcast encryption systems (with short ciphertexts). In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 171–188. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01001-9_10
26. Harn, L., Lin, H.-Y.: A cryptographic key generation scheme for multilevel data security. Comput. Secur. **9**(6), 539–546 (1990)
27. He, K., Weng, J., Liu, J., Liu, J.K., Liu, W., Deng, R.H.: Anonymous identity-based broadcast encryption with chosen-ciphertext security. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, 30 May–3 June 2016, pp. 247–255, 2016
28. Huang, D., Medhi, D.: A secure group key management scheme for hierarchical mobile ad hoc networks. Ad Hoc Netw. **6**(4), 560–577 (2008)
29. Kim, J., Susilo, W., Au, M.H., Seberry, J.: Adaptively secure identity-based broadcast encryption with a constant-sized ciphertext. IEEE Trans. Inf. Forensics Secur. **10**(3), 679–693 (2015)
30. Lin, Y.-L., Hsu, C.-L.: Secure key management scheme for dynamic hierarchical access control based on ECC. J. Syst. Softw. **84**(4), 679–685 (2011)
31. MacKinnon, S.J., Akl, S.G.: New key generation algorithms for multilevel security. In: Proceedings of the 1983 IEEE Symposium on Security and Privacy, Oakland, California, USA, 25–27 April 983, pp. 72–78 (1983)
32. MacKinnon, S.J., Taylor, P.D., Meijer, H., Akl, S.G.: An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. IEEE Trans. Comput. **34**(9), 797–802 (1985)
33. Odelu, V., Das, A.K., Goswami, A.: An effective and secure key-management scheme for hierarchical access control in e-medicine system. J. Med. Syst. **37**(2), 9920 (2013)
34. Sandhu, R.S.: Cryptographic implementation of a tree hierarchy for access control. Inf. Process. Lett. **27**(2), 95–98 (1988)
35. De Santis, A., Ferrara, A.L., Masucci, B.: Efficient provably-secure hierarchical key assignment schemes. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 371–382. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74456-6_34
36. De Santis, A., Ferrara, A.L., Masucci, B.: Efficient provably-secure hierarchical key assignment schemes. Theoret. Comput. Sci. **412**(41), 5684–5699 (2011)
37. Shen, V.R.L., Chen, T.-S.: A novel key management scheme based on discrete logarithms and polynomial interpolations. Comput. Secur. **21**(2), 164–171 (2002)
38. Tang, S., Li, X., Huang, X., Xiang, Y., Lingling, X.: Achieving simple, secure and efficient hierarchical access control in cloud computing. IEEE Trans. Comput. **65**(7), 2325–2331 (2016)
39. Wong, C.K., Gouda, M.G., Lam, S.S.: Secure group communications using key graphs. IEEE/ACM Trans. Netw. **8**(1), 16–30 (2000)
40. Yang, C., Li, C.: Access control in a hierarchy using one-way hash functions. Comput. Secur. **23**(8), 659–664 (2004)

A Novel GPU-Based Implementation of the Cube Attack

Preliminary Results Against Trivium

Marco Cianfriglia[1,2(✉)], Stefano Guarino[2,3(✉)], Massimo Bernaschi[2], Flavio Lombardi[2], and Marco Pedicini[1,2]

[1] Roma Tre University, Rome, Italy
{cianfriglia,pedicini}@mat.uniroma3.it
[2] Istituto per le Applicazioni del Calcolo (IAC - CNR), Rome, Italy
{s.guarino,m.bernaschi,f.lombardi}@iac.cnr.it
[3] Sapienza University of Rome, Rome, Italy

Abstract. With black-box access to the cipher being its unique requirement, Dinur and Shamir's cube attack is a flexible cryptanalysis technique which can be applied to virtually any cipher. However, gaining a precise understanding of the characteristics that make a cipher vulnerable to the attack is still an open problem, and no implementation of the cube attack so far succeeded in breaking a real-world strong cipher. In this paper, we present a complete implementation of the cube attack on a GPU/CPU cluster able to improve state-of-the-art results against the Trivium cipher. In particular, our attack allows full key recovery up to 781 initialization rounds without brute-force, and yields the first ever maxterm after 800 initialization rounds. The proposed attack leverages a careful tuning of the available resources, based on an accurate analysis of the offline phase, that has been tailored to the characteristics of GPU computing. We discuss all design choices, detailing their respective advantages and drawbacks. Other than providing remarkable results, this paper shows how the cube attack can significantly benefit from accelerators like GPUs, paving the way for future work in the area.

Keywords: Cube attack · Trivium · GPU

1 Introduction

The security of a stream cipher relies on its ability to mimic the properties of the perfectly secure One Time Pad (OTP): predicting future keystream bits (*e.g.*, by recovering its inner state) must be computationally infeasible. In fact, as highlighted by *algebraic* and *correlation* attacks, any statistical correlation between output bits and linear combinations of input bits is a potential security breach for the cipher. Cryptographers are therefore caught in between implementation requirements, which suggest the use of efficient primitives such as *Feedback Shift Registers* (FSRs) or *Finite State Machines* (FSMs), and security requirements,

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 184–207, 2017.
DOI: 10.1007/978-3-319-61204-1_10

which demand for solutions able to disguise the dependence of keystream-bits on the inner state of the registers. Many recent stream ciphers therefore rely upon irregular clocks, mutual clock control, non-linear and/or mutual feedback among different registers, or combinations of these solutions.

The cube attack, proposed by Dinur and Shamir [10], can be classified as an algebraic known-plaintext attack. Assuming that a chunk of keystream can be recovered from a known plaintext-ciphertext pair, the attack allows determining a set of linear equations binding key-bits. However, cube attacks significantly deviate from traditional algebraic attacks in that the equations are not recovered symbolically, but rather extracted through exhaustive searches over selected public/IV bits – the edges of the *cubes* the attack is named after. The possibility that a cube yields a linear equation depends on both its size and on the algebraic properties of the cipher. Since the Algebraic Normal Form (ANF) of the cipher (that is, its representation as a binary polynomial) is generally unknown beforehand, in practice the attack usually runs without clear prior insights into a convenient strategy for selecting the cubes – an approach made possible by the fact that the attack only requires black-box access to the attacked cipher. Exploring cubes of different (possibly large) size, trying many different sets of indices, and varying the binary assignment of the public bits not belonging to the tested cube are all promising solutions, but they all come at an exponential cost. In a sense, cube attacks can be therefore assimilated to *Time-Memory-Data Trade-Off* (TMDTO) attacks, as their success rate strongly depends on the extensiveness of the pre-computation stage, on the memory available to store the results of that stage, and on the amount of data usable to implement it. Consequently, identifying the most favourable design choices is the main pillar of a possibly successful cube attack.

Contributions. The present paper motivates and discusses in depth an implementation for Graphics Processing Unit (GPU) of the cube attack. The target cipher is Trivium [8,22], already considered in the literature to test the viability of the cube attack [10,14]. Our contributions can be summarized as follows: (i) We tailor the design and implementation of the cube attack to the characteristics of GPUs, in order to fully exploit parallelization while coping with limited memory. Our framework is extremely flexible and can be adapted to any other cipher at no more cost than some fine (performance) tuning, mostly related to memory allocation. (ii) We show the performance gain with respect to a CPU implementation, including results obtained on latest generation GPU cards. (iii) Our implementation allows for exhaustively assigning values to (subsets of) public variables with negligible additional costs. This means extending the quest for superpolys to a dimension never explored in previous works, and, by not being tied to a very small set of IV combinations, potentially weakening one of the basic requirements of the cube attack, that is, the assumption of a completely tweakable IV. (iv) Even though we run the attack with only a few preliminary sets of cubes – specifically selected to both validate our code and compare our results with the literature – our findings improve on the state-of-the-art for attacks against reduced-round versions of Trivium.

Roadmap. This paper is organized as follows: Sect. 2 introduces the cube attack and the targeted cipher Trivium; our implementation of the attack is described in Sect. 3, whereas experimental results are reported and discussed in Sect. 4; Sect. 5 gives an overview of related works; finally, Sect. 6, draws conclusions and suggests possible directions for future work.

2 Preliminaries

In this section, we first describe the theoretical implant of the cube attack, and we then briefly introduce Trivium. More details about Trivium are reported in Appendix A.

The Cube Attack. Let z denote a generic keystream bit produced by a stream cipher $\mathcal{E}$. z is the result of a function $E : \mathbb{F}_2^{n+k} \rightarrow \mathbb{F}_2$, computed over the $n+k$ input bits obtained from an Initial Vector IV of length n and a secret key K of length k. It is well known that z can be expressed as $z = p(\mathbf{x}, \mathbf{y})$, where p is the polynomial representation of E, $\mathbf{x} = (x_1, \ldots, x_n)$ is the vector of public variables (IV), $\mathbf{y} = (y_1, \ldots, y_k)$ is the vector of secret variables (K), and all variables in p appear with degree 1, at most. The cube attack relies on extracting from p a set of linear equations binding private variables in $\mathbf{y}$, through a suitable offline pre-computation phase involving public variables in $\mathbf{x}$.

Let $I = \{i_1, \ldots, i_m\} \subset \{1, \ldots, n\}$ and let us introduce the complement $\overline{I} = \{1, \ldots, n\} \backslash I$ of the set I. With a slight abuse of notation, let us consider variables in $\mathbf{x}$ as partitioned by I: $\mathbf{x} = (\mathbf{x}_I, \mathbf{x}_{\overline{I}})$, *i.e.*, we tell apart the variables $\mathbf{x}_I$ indexed by I from those $\mathbf{x}_{\overline{I}}$ indexed by its complement $\overline{I}$. Let $t_I = x_{i_1} \cdots x_{i_m}$ be the monomial induced by I, that is, the product of all variables in $\mathbf{x}_I$. By writing $t_I(\mathbf{x}_I)$ we want to stress that t_I contains only variables in $\mathbf{x}_I$. If we factor $t_I(\mathbf{x}_I)$ out of $p(\mathbf{x}, \mathbf{y})$ we obtain

$$p(\mathbf{x}, \mathbf{y}) = t_I(\mathbf{x}_I) \cdot p_{S(I)}(\mathbf{x}, \mathbf{y}) + q(\mathbf{x}, \mathbf{y})$$

where the quotient $p_{S(I)}(\mathbf{x}, \mathbf{y})$ of the division is called the *superpoly* of I in p, whereas $q(\mathbf{x}, \mathbf{y})$ is the remainder of the division.

Now, for any binary vector $\mathbf{v}_{\overline{I}}$, we consider a fixed assignment for variables $\mathbf{x}_{\overline{I}}$[1], and let $C_I(\mathbf{v}_{\overline{I}})$ denote the *cube* induced by I and $\mathbf{v}_{\overline{I}}$, that is, the set of all 2^m possible binary assignments to $\mathbf{x}$ in which variables $\mathbf{x}_{\overline{I}}$ assume values specified by the binary vector $\mathbf{v}_{\overline{I}}$ and the remaining variables in $\mathbf{x}_I$ take all the possible combinations. It is easy to verify that all monomials in $p_{S(I)}$ do not contain any of the variables $\mathbf{x}_I$ (*i.e.*, $p_{S(I)}(\mathbf{x}, \mathbf{y}) = p_{S(I)}(\mathbf{x}_{\overline{I}}, \mathbf{y})$), whereas all monomials in q do not contain *at least* one of the variables in $\mathbf{x}_I$. For this reason, regardless of $\mathbf{y}$, the sum of $p(\mathbf{x}, \mathbf{y})$ over all elements $\mathbf{v}$ of $C_I(\mathbf{v}_{\overline{I}})$ yields [10]

$$\sum_{\mathbf{v} \in C_I(\mathbf{v}_{\overline{I}})} p(\mathbf{v}, \mathbf{y}) = p_{S(I)}(\mathbf{v}_{\overline{I}}, \mathbf{y}) \tag{1}$$

which obviously does not depend on variables $\mathbf{x}_I$ anymore.

[1] The standard assumption is $\mathbf{v}_{\overline{I}} = \mathbf{0}$, but this is not actually required.

If $p_{S(I)}(\mathbf{v}_{\overline{I}}, \mathbf{y})$ is *linear*, the monomial $t_I(\mathbf{x}_I)$ is called a *maxterm for p with the assignment* $\mathbf{v}_{\overline{I}}$. If we can identify maxterms and find the symbolic expression of their superpolys, we obtain a system of linear equations that can be used to recover the secret key.

As $\mathbf{v}_{\overline{I}}$ is always clear from the context, to improve readability in the following we simply denote $C_I(\mathbf{v}_{\overline{I}})$ and $p_{S(I)}(\mathbf{v}_{\overline{I}}, \mathbf{y})$ as C_I and $p_{S(I)}(\mathbf{y})$, respectively.

Trivium. Trivium [8] is a stream cipher conceived by Christophe De Cannière and Bart Preneel, part of the eSTREAM portfolio. It generates up to 2^{64} bits of output from an 80-bit key K and an 80-bit Initial Vector IV, and it shows remarkable resistance to cryptanalysis despite its simplicity and its excellent performance. Trivium is composed by a 288-bit internal state consisting of three shift registers of length 93, 84 and 111, respectively. The feedback to each of these registers and the output bit of the cipher are obtained through non-linear combinations involving in total 15 out of the 288 internal state bits. To initialize the cipher, K and IV are written into two of the shift registers, with a fixed pattern filling the remaining bits. 1152 initialization rounds guarantee that the output begins to be produced only after all key-bits and IV-bits have been sufficiently mixed together to define the internal state of the registers.

3 The Proposed GPU Implementation of the Attack

In this section, we present, detail, and discuss our attack, designed to run on a cluster equipped with Graphics Processing Units (GPU). As previously mentioned, the success of a cube attack is highly dependent on suitable implementation choices. In order to better explain our own approach, we start with an analysis of the cube attack from a more implementative perspective.

3.1 Practical Cube Attack

At a high level, any practical implementation of the cube attack requires performing the following steps:

S1 Find as many maxterms as possible;
S2 For each maxterm, find the corresponding linear equation(s);
S3 Solve the obtained linear system.

Step S1. This is the core of the attack, where cubes that yield linear equations are identified. Choosing candidate maxterms (*i.e.*, cubes) is non-trivial. Intuitively, the degree of most maxterms lies in a specific range that depends on the (unknown) degree distribution of the monomials of the polynomial p. If the degree of t_I is too small, then $p_{S(I)}$ is most likely non-linear, but if the degree of t_I is too large, then $p_{S(I)}$ will probably be constant (*e.g.*, null). Moreover, since the complexity of the offline phase scales exponentially with $|I|$, the degree of tested potential maxterms is strongly influenced by practical limitations.

In [10], the authors propose a *random walk* to explore a maximal cube $C_{I_{\max}}$, *i.e.*, starting from a random subset $I \subset I_{\max}$ and iteratively testing the superpoly $p_{S(I)}$ to decide whether the degree of t_I should be increased or decreased. The underlying idea is to use a probabilistic approach to identify the optimal size $|I|$. In [14], the authors evaluate the cipher upon all vertices of a maximal cube $C_{I_{\max}}$, store the results in a table T of size $|T| = 2^{|I_{\max}|}$, and then apply the *Moebius transform* to the entire table T, thus computing at once the sums over $\binom{|I_{\max}|}{d}$ sub-cubes of $C_{I_{\max}}$ of degree d, for $d = 0, \ldots, |I_{\max}|$. These cubes are all possible sub-cubes of $C_{I_{\max}}$ in which the variables outside the cube have been set equal to 0. In this case the rationale is minimizing processing cost by reusing partial computations as much as possible. Interestingly, the authors of [14] show that specific cubes perform better than others, at least for reduced-round variants of Trivium, and use their findings to select the most promising maximal set $I_{\max}$.

None of these two strategies is suitable for GPUs. The stochastic nature of the random walk prevents the sequence of steps from being determined *a priori*, since the computation is performed only when (and if) needed. On the other hand, the Moebius transform requires a rigid schema of calculations and a large number of alternating read and write operations in memory that must be synchronized. Both approaches are conceived for implementations in which computational power is a constraint (while memory is not), and all advantages of using the Moebius Transform are lost in case of parallel processing. We rather perform an exhaustive search over a portion of a maximal cube, a solution that is highly parallelizable and feasible with our computational resources.

For each candidate maxterm t_I, we need to verify whether the superpoly $p_{S(I)}$ is linear. The goal being recovering key bits, any fixed assignment of variables $\mathbf{x}_{\overline{I}}$ with the bit vector $\mathbf{v}_{\overline{I}}$ can be used to get rid of variables. In order to guarantee that the degree of each superpoly is reduced to the bare minimum, the assignment $\mathbf{v}_{\overline{I}}$ to $\mathbf{0}$ is usually preferred, but we argue that this is not necessarily the best choice, as motivated later in Sect. 4.2. In any case, at this stage the superpoly $p_{S(I)}$ only depends on $\mathbf{y}$. In principle, assessing the linearity of $p_{S(I)}(\mathbf{y})$ requires finding all of its coefficients, but efficient *probabilistic* linearity tests [7,21] can safely replace deterministic ones in most practical settings. Probabilistic tests involve verifying if

$$p_{S(I)}(\mathbf{u}_1 + \mathbf{u}_2) = p_{S(I)}(\mathbf{u}_1) + p_{S(I)}(\mathbf{u}_2) + p_{S(I)}(\mathbf{0}) \tag{2}$$

holds for random pairs of vectors $\mathbf{u}_1, \mathbf{u}_2$. Practically, this means evaluating *numerically* four sums: $\sum_{\mathbf{v}\in C_I} E(\mathbf{v}, \mathbf{0})$, $\sum_{\mathbf{v}\in C_I} E(\mathbf{v}, \mathbf{u}_1)$, $\sum_{\mathbf{v}\in C_I} E(\mathbf{v}, \mathbf{u}_2)$, and $\sum_{\mathbf{v}\in C_I} E(\mathbf{v}, \mathbf{u}_1 + \mathbf{u}_2)$.

Probabilistic tests rely on the fact that (2) must be true for all $\mathbf{u}_1, \mathbf{u}_2$ if $p_{S(I)}$ is linear, whereas, in general, it holds with probability $\frac{1}{2}$. In particular, as done for previous cube attacks [10,14], we will resort to a *complete-graph test* [21], which guarantees a slightly lesser accuracy than the (truly-random) BLR test [7] with far fewer evaluations of $p_{S(I)}$. Let us remark that what ultimately matters in the envisaged scenario is identifying "far-from-linear" superpolys [20]. To clarify, let us consider the superpoly $p_{S(I)}$

$$p_{S(I)}(\mathbf{y}) = l(\mathbf{y}) + \prod_{i=1}^{k} y_i$$

formed by a sum $l(\mathbf{y})$ of linear terms, plus one nonlinear term given by the product of all variables in $\mathbf{y}$. Despite the equality $p_{S(I)}(\mathbf{y}) = l(\mathbf{y})$ is *formally* wrong (the degree of $p_{S(I)}$ is as large as k), $p_{S(I)}(\mathbf{u}) = l(\mathbf{u})$ is *numerically* correct for all $\mathbf{u} \in \mathbb{F}_2^k$, except $\mathbf{u} = (1, 1, \ldots, 1)$. In other words, mistaking $p_{S(I)}$ for linear has practical consequences only if $\mathbf{u} = (1, 1, \ldots, 1)$.

Steps S2 and S3. Step **S2** consists in finding the *symbolic* expression of the superpoly of all identified maxterms, and the free term of the corresponding equation. Again, this turns into a set of *numerical* evaluations: the free term of $p_{S(I)}(\mathbf{y})$ is

$$p_{S(I)}(\mathbf{0}) = \sum_{\mathbf{v} \in C_I} E(\mathbf{v}, \mathbf{0})$$

whereas the coefficient of each variable y_i is

$$p_{S(I)}(\mathbf{e}_i) + p_{S(I)}(\mathbf{0}) = \sum_{\mathbf{v} \in C_I} E(\mathbf{v}, \mathbf{e}_i) + \sum_{\mathbf{v} \in C_I} E(\mathbf{v}, \mathbf{0})$$

where $\mathbf{e}_i$ is the unit vector with all null coordinates except $y_i = 1$. Once the polynomial $p_{S(I)}(\mathbf{y})$ is found, the attack assumes the availability of the 2^m keystream bits produced in correspondence to a fixed (unknown) assignment to the variables $\mathbf{y}$, as the variables $\mathbf{x}$ take all possible assignments in C_I. This produces the linear equation

$$p_{S(I)}(\mathbf{y}) = \sum_{\mathbf{v} \in C_I} E(\mathbf{v}, \mathbf{y})$$

whose left side is a linear combination of the key variables $\mathbf{y}$ with coefficients found *offline*, whereas the right side is a number found *online*, and whose solution is the sought unknown assignment to $\mathbf{y}$.

Finally, Step **S3** just requires solving the obtained linear system with any suitable technique described in the literature.

3.2 The Setting

Generally speaking, GPUs are processing units characterized by the following advantages and limitations:

Computing: Each unit features a large number (*i.e., thousands*) of simple cores, that make possible running a much higher number of parallel threads compared to a standard CPU. More precisely, the GPU's basic processing unit is the *warp* consisting of 32 threads each. Threads are designed to work on 32-bit words, and the performance is maximized if all threads belonging to the same warp execute exactly the same operations at the same time on different but contiguous data.

Memory: The so-called *global* memory available on a GPU is limited, typically between 4 and 12 GB. Each thread can independently access data (random access is fully supported, but costly performance-wise). However, when threads in a warp access consecutive 32-bit words, the cost is equivalent to a single memory operation. Concurrent readings and writings by different threads to the same resources, which require some level of synchronization, should be avoided to prevent serialization that defeats parallelism.

The basic step of the attack is the sum of $E(\mathbf{v}, \mathbf{y})$ over all elements $\mathbf{v}$ of a cube C_I. Each time we sum over a cube, the key variables $\mathbf{y}$ are fixed, either to a random $\mathbf{u}_j$ for the linearity tests, or to $\mathbf{0}$ and to versors $\mathbf{e}_i$ for determining the superpoly. In both cases exactly the same sum $\sum_{\mathbf{v}\in C_I} E(\mathbf{v}, \mathbf{u}_j)$ must be performed for all elements of a set of keys $\{\mathbf{u}_1, \ldots, \mathbf{u}_M\}$.

We define the following strategy for carrying out the sums over a cube with the goal of maximizing the parallelization and fully exploiting at its best the computational power offered by GPUs:

- Assigning to all the threads within a warp the computation of the same cube C_I but with a different key $\mathbf{u}_j$. This choice guarantees that all threads perform the same operation at the same time for the entire computation.
- Leveraging the GPU computational power to calculate all the elements of a cube C_I, providing to the threads just a bit-mask representing the set I. With this approach we can exploit all available GPU memory to store the cubes evaluations and minimize, at the same time, the number of memory access operations.
- Defining a keystream generator function $E(\mathbf{x}, \mathbf{y})$ which outputs a 32-bit word, and letting each thread work on the whole word, fully leveraging the GPU computing model. This approach offers two remarkable benefits: (i) considering 32 keystream bits altogether is equivalent to concurrently attacking 32 different polynomials, and (ii) working on 32-bit integers fits much better with the GPUs features, whereas forcing the threads to work on single bits would critically affect the performance of the attack. Therefore, attacking 32 keystream bits altogether reduces (of a factor 32) the memory needed for storing the cubes' evaluation, thus imposing some limitations on the size of the cubes to be tested, as we will clarify later.
- Choosing the number M of keys to be a multiple of the warp size in order to perform the probabilistic linearity test on 32 keystream bits at the same time and for all M keys.

3.3 The Attack

A severe constraint in any GPU implementation is represented by the amount of memory $|T|$ currently available on GPUs. Moreover, for each cube, we need to consider M different keys in order to run the linearity test, thus reducing the amount of available memory even further to $|T|/M$. Storing single evaluations of the cipher in T means testing only sub-cubes of a maximal cube of size $|I_{\max}| = \log_2(|T|/M)$. With the memory available in current GPUs, $\log_2(|T|/M)$ is not large enough for any reasonably strong cipher. The new approach we propose is highly parallelizable, it can fully exploit the computational resources offered by GPU, and it is able to exploit GPU memory to test high order maximal cubes.

The proposed design of the attack relies on the following rationale: exploring only a portion of the maximal cube $C_{I_{\max}}$, considering only subsets $I \subseteq I_{\max}$ characterized by a non-empty *minimal* intersection $I_{\min}$. Quite naturally, a similar design leads to two distinct CUDA[2] kernels, respectively responsible for: (1) computing many variants of the cube $C_{I_{\min}}$, one for each of the possible combinations of the indices in $I_{\max} \setminus I_{\min}$, and writing the results in memory; (2) combining the stored results to test all cubes C_I such that $I_{\min} \subseteq I \subseteq I_{\max}$. Following this approach, the size of the explored $I_{\max}$ can be raised to $|I_{\max}| = |I_{\min}| + \log_2(|T|/M)$, with read and write memory operations carried out by different kernels.

According to the notation introduced in Sect. 2, the public variables are $\mathbf{x} = (x_1, \ldots, x_n)$. Now, let us distinguish these n public variables into three sets $\mathbf{x}_{\text{fix}} = (x_{i_1}, \ldots, x_{i_{d_{\text{fix}}}})$, $\mathbf{x}_{\text{free}} = (x_{j_1}, \ldots, x_{j_{d_{\text{free}}}})$, and $\mathbf{x}^*$, of size d_{fix}, d_{free}, and $n - d$, respectively, where $d = d_{\text{fix}} - d_{\text{free}}$. The variables $\mathbf{x}_{\text{fix}}$ correspond to the *fixed* components of $C_{I_{\max}}$ identified by $I_{\min}$, *i.e.*, $I_{\min} = \{i_1, \ldots, i_{d_{\text{fix}}}\}$, whereas the variables $\mathbf{x}_{\text{free}}$ correspond to the remaining *free* components of $C_{I_{\max}}$, *i.e.*, $I_{\max} \setminus I_{\min} = \{j_1, \ldots, j_{d_{\text{free}}}\}$ and $|I_{\max}| = d$. The variables $\mathbf{u}^*$ are the remaining public variables that fall outside $I_{\max}$.

The two kernels of our attack can be described as follows:

Kernel 1: It uses $2^{d_{\text{free}}}$ warps. Since, as described before, the 32 threads belonging to the same warp perform exactly the same operations but for different keys, in the following we simply consider a representative thread per warp and ignore the private variables $\mathbf{y}$.[3] For $t = 0, \ldots, 2^{d_{\text{free}}} - 1$, thread (*i.e.*, warp) s sums $E(\mathbf{u}, \mathbf{y})$ over each vertex of the cube $C^s_{I_{\min}}$ of size d_{fix} determined by the assignment of the d_{free}-bit representation $\mathbf{u}_{\text{free}}$ of integer s to the variables $\mathbf{x}_{\text{free}}$ and of $\mathbf{0}$ to the variable $\mathbf{u}^*$. Finally, thread s writes the sum in the s^{th} entry of table T, so that, at the end of the execution of the kernel, each entry of T contains the sum over a cube of size d_{fix}. These evaluations allow for testing the monomial $t_{I_{\min}}$ with all the aforementioned assignments to the other $n - d_{\text{fix}}$ variables.

[2] CUDA is the software framework used for programming Nvidia GPUs.

[3] The work that here is assigned to a single thread can be actually split among any number of threads, reassembling the results at the end. We will not consider this possibility here for the sake of clarity.

Kernel 2: By simply combining the values stored in T at the end of Kernel 1, it is now possible to explore cubes of potentially any size $d_{\text{fix}} + \delta$, with $0 \leq \delta \leq d_{\text{free}}$. Although the exploration can potentially follow many other approaches (*e.g.*, a random walk as in [10]), the large computing power of our platform suggests to test cubes exhaustively. Moreover, we extend the exhaustive search to an area never reached, to the best of our knowledge, in the literature. For all I such that $I_{\min} \subseteq I \subseteq I_{\max}$, this kernel considers all variants of cube C_I obtained assigning all possible combinations of values to the variables in $I_{\max} \backslash I$. More precisely, for each possible choice of $\delta \in [0, d_{\text{free}}]$, there are exactly $\binom{d_{\text{free}}}{\delta} 2^{d_{\text{free}} - \delta}$ distinct cubes of size $d_{\text{fix}} + \delta$ available. In fact, we can choose δ free variables (the additional dimensions of the cube) in $\binom{d_{\text{free}}}{\delta}$ different ways, and we can choose the fixed assignment to the remaining $d_{\text{free}} - \delta$ variables in any of the $2^{d_{\text{free}} - \delta}$ possible combinations.

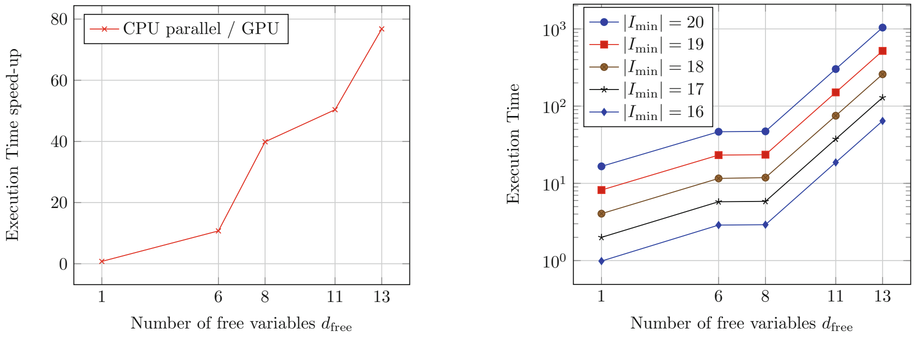
As a matter of fact, the number of cubes considered in [14] is $\sum_{\delta=0}^{d_{\text{free}}} \binom{d_{\text{free}}}{\delta} = 2^{d_{\text{free}}}$, whereas the number of cubes tested by our approach is significantly larger, namely, $\sum_{\delta=0}^{d_{\text{free}}} 2^{d_{\text{free}} - \delta} \binom{d_{\text{free}}}{\delta} = 3^{d_{\text{free}}}$. We would like to highlight that Kernel 2 is computationally dominated by Kernel 1, so the cost of our exhaustive search is negligible. Therefore, our design entails considering any possible assignment to variables outside the cube, to finally address the common conjecture (never proved in the literature), that assigning $\mathbf{0}$ is the best possible solution.

Let us underline that, in order to validate our implementation of the cube attack, we symbolically evaluated the polynomial p of Trivium up to 400 initialization rounds, and used p to identify all possible maxterms and their superpoly. We then ran the attack to find all maxterms whose variables belonged to selected sets I. Our experimental findings matched the symbolical findings. Further experimental validation of our code is reported in Sect. 4.

3.4 Performance Analysis

To evaluate the performance of our GPU based solution, we developed both a CPU and a GPU version of the cube attack. The cluster we used for the experiments is composed by 3 nodes, each equipped with 4 Tesla K80 with 12 GB of *global* memory and 4 Intel Xeon CPU E5-2640 with 128 GB of RAM. The CPU experiments were conducted on a parallel version based on OpenMP that exploits 32 cores of the four Intel(R) Xeon(R) CPU E5-2640. Each performance test was executed 5 times and the average time is reported. It is worth noticing that all versions rely on the same base functions to implement Trivium.

In Fig. 1a, we report the speed-up gained by the GPU version with respect to the parallel CPU version. We evaluated the two solutions over growing size maximal cubes $C_{I_{\max}}$, in which we anchor the size of $I_{\min}$, consequently causing the size of the set $I_{\max} \setminus I_{\min}$ to exponentially increase. Overall, the experiments show that the benefit of using the GPU version grows with the number of free variables d_{free} considered, reaching a speed-up up to 70× when $d_{\text{free}} = 13$. The rationale is that the execution time of the CPU version increases almost linearly with d_{free} from the very beginning, whereas a similar trend can be observed for



(a) Speedup of parallel CPU vs. GPU

(b) GPU performance analysis

Fig. 1. Performance experiments

the GPU version only when the number of blocks in use gets larger than the number of Streaming Multiprocessors (SMs) of the GPU, which happens when $d_{\text{free}} \geq 9$ in our case. Of course, slight fluctuations are possible, mostly due to the complex interactions among the multiple cache levels of a modern CPU. Moreover, we evaluated how the GPU solution scales when d_{free} increases. As reported in Fig. 1b, our solution scales linearly with the size of the problem, *i.e.*, exponentially with the size of the sub-cubes $C_{I_{\min}}$, thus paving the way for future works in the area.

Finally, we ran the attack under the control of the Nvidia profiler in order to measure the ALU occupancy achieved by our kernels. Kernel 1 is invoked just once per run to fill the whole table T, with an occupancy consistently over 95% when $d_{\text{free}} \geq 10$. Kernel 2 is instead invoked once per each $\delta \in [0, d_{\text{free}}]$, to compute all available cubes of size $d_{\text{fix}} + \delta$. The maximum occupancy exceeds 95% as soon as $d_{\text{free}} \geq 12$, with an average of approximately 50%. In either case the impact of d_{fix}, which determines the load of each thread, is negligible. Considering that d_{free} should be maximized to improve the attack success rate, our kernels guarantee an excellent use of resources in any realistic application. For instance, in our experiments discussed in Sect. 4 we set $d_{\text{free}} = 16$, which guarantees an occupancy above 99% for Kernel 1, and a maximum occupancy above 98% for Kernel 2.

4 Results

Finally, this section reports the results obtained by our GPU implementation of the cube attack against reduced-round Trivium. We recall that the attack ran on a cluster composed by 3 nodes, each equipped with 4 Tesla K80 with 12 GB of *global* memory and 4 Intel Xeon CPU E5-2640 with 128 GB of RAM.

As mentioned in Sect. 3.3, we performed a formal evaluation of our implementation, by checking our experimental results against Trivium's polynomials, explicitly computed up to 400 initialization rounds. In the following, the number of initialization rounds instead matches (and slightly overtakes) the best results from the literature, thus reaching a point where a symbolic evaluation would be prohibitive. Still, the results we exhibit are obtained from experiments specifically designed to reproduce tests carried out in the recent past [14], so as to provide, at the same time: (i) a direct comparison of our results with the state-of-the-art; (ii) an immediate means to assess the advantages of our approach, and (iii) a further validation of the correctness of our code.

Experimental Setting. In our attack, we consider two different reduced-round variants of Trivium, corresponding to 768 and 800 initialization rounds, respectively. As explained and motivated in Sect. 3.2, in our scheme, each call to Trivium produces 32 key-stream bits, which we use in our concurrent search for superpolys. The most significant practical consequence of a similar construction is the ability to devise attacks to Trivium reduced to any number of initialization rounds ranging from 768 to 831, at the cost of just two attacks, although the number of available superpolys decreases with the number of rounds. As a matter of fact, the j^{th} output bit after 768 rounds can also be interpreted as the $(j-i)^{th}$ bit of output after $768+i$ initialization rounds, for any $j \geq i$. In other words, an attack to Trivium reduced to $768+i$ initialization rounds can count upon all superpolys found in correspondence of the j^{th} output bit after 768 rounds, for all $j \geq i$.

For each of the two attacks (768 and 800 initialization rounds), we ran a set of independent runs, each using a different choice for the pair of sets of variables $I_{\min}, I_{\max}$ (with $I_{\min} \subset I_{\max}$) that define the minimal and maximal tested cubes $C_{I_{\min}}$ and $C_{I_{\max}}$. The size of $I_{\min}$ and $I_{\max} \setminus I_{\min}$ is $d_{\text{fix}} = 25$ and $d_{\text{free}} = 16$, respectively, for all runs, so that all maximal cubes have size $d = d_{\text{fix}} + d_{\text{free}} = 41$. Peculiarly to our implementation, when we test the monomial composed of all variables in some set $I_{\min} \subseteq I \subseteq I_{\max}$, we exhaustively assign values to all public variables in $I_{\max} \setminus I$, thus concurrently testing the linearity of $2^{41-|I|}$ possibly different superpolys. This feature of our attack – a possibility overlooked in the literature, but almost free-of-charge in our framework – provides primary benefits, as described in Sect. 4.2.

In all the reported experiments, we use a complete-graph linearity test based on combining 10 randomly sampled keys.

4.1 Summary of Results

As mentioned before, we implemented two attacks, against Trivium reduced to 768 (Trivium-768 in the following) and 800 (Trivium-800) initialization rounds, respectively. In both cases, our setting allows obtaining superpolys corresponding to 32 output bits altogether, at the cost of a single attack.

Results Against Trivium-768. For the attack against Trivium-768, we took inspiration from [14]: we launched 12 runs based on 12 different pairs $I_{\min}, I_{\max}$, chosen so as to guarantee that each of the 12 linearly independent superpolys found in [14] after 799 initialization rounds was to be found by one of our runs. The rationale of reproducing results from [14] was to both test the correctness of our implementation, and provide a better understanding of the advantages of our implementation with respect to the state-of-the-art. In this sense, let us highlight that a single run of ours cannot be directly compared with all results presented in [14], because each of our runs only explores the limited portion of the maximal cube $C_{I_{\max}}$ composed by all super-cubes of $C_{I_{\min}}$.

To better describe our results, let us introduce the binary matrix A whose element $A(i, j)$ is the coefficient of variable y_j in the i^{th} available superpoly. The rank of A, denoted $\mathrm{rk}(A)$, clearly determines the number of key bits that can be recovered in the online phase of the attack based on the available superpolys, before recurring to brute-force.

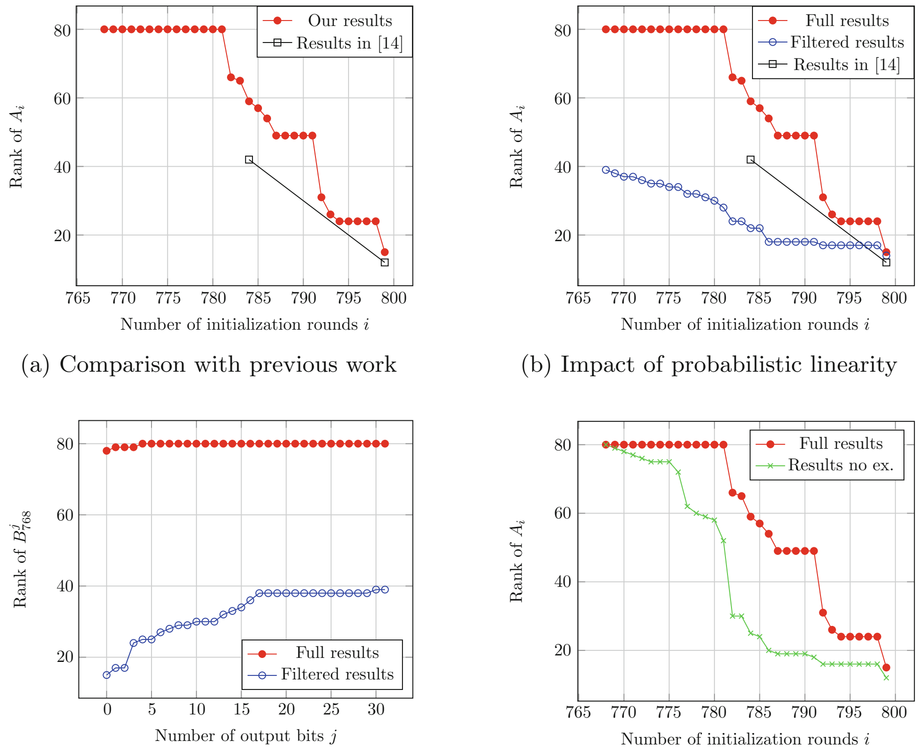
As described before, the superpolys yielded by the i^{th} output bit after round 768 are usable to attack Trivium for any number of initialization rounds between 768 and $768+i$. It is possible to define 32 different matrices $A_{768}, \ldots, A_{799}$: A_{768} includes all superpolys found, while each matrix A_{768+i} is obtained by incrementally removing the superpolys yielded by output bits $0, \ldots, i-1$. Figure 2a shows $\mathrm{rk}(A_i)$ as a function of i, comparing our findings with those of [14].

Overall, our results extend the state-of-the-art in a remarkable way, especially if we consider that our quest for maxterms was circumscribed to multiples of 12 *base* monomials of degree 25. In particular, let us highlight a few aspects that emerge from Fig. 2a:

- Since our runs were designed to include all 12 maxterms found in [14] after 799 initialization rounds, it is not surprising that $\mathrm{rk}(A_{799})$ is at least 12. Yet, it is indeed larger: we found 3 more linearly independent superpolys, reaching $\mathrm{rk}(A_{799}) = 15$.
- Although we did not force our tested cube to include the maxterms found in [14] after 784 rounds, we have $\mathrm{rk}(A_{784}) = 59$, compared with rank 42 found in [14].
- Finally, and probably most importantly, our attack allows a full key recovery up to 781 initialization rounds.

Selected superpolys that guarantee the above ranks are reported in Appendix B, together with the corresponding maxterms. Very interesting is also *how* novel superpolys were found, a point that is better described in the following.

Results Against Trivium-800. To provide a further test of the quality of our attack, we launched a preliminary attack against Trivium-800. We kept unvaried all the parameters of the attack ($d_{\text{fix}} = 25$, $d_{\text{free}} = 16$, 32 output bits attacked altogether), but this time we only launched 4 runs, and we chose the sets $I_{\min}, I_{\max}$ at random. In total, we were able to find a single maxterm corresponding to 800 rounds, and no maxterms afterwards. This maxterm and the corresponding superpoly are also reported in Appendix B. Although our findings only allow to cut in half the complexity of a brute force attack, this is the first ever superpoly found considering more than 799 initialization rounds. We



(a) Comparison with previous work

(b) Impact of probabilistic linearity

(c) Impact of using 32 output bits

(d) Impact of exhaustive search

Fig. 2. Our results

recall that our limited results should not appear as surprising: as previous work suggests [10,14], when the number of initialization rounds grows, a cube attack should increase the average degree of candidate maxterms and/or implement specific strategies for the selection of the index sets [14].

4.2 Further Discussion

Hereafter, we provide a more detailed analysis and a further discussion of our findings, considering two aspects in particular: the reliability of commonly used linearity tests, and the peculiar advantages of our attack design. Unless otherwise specified, in the following we always focus on Trivium-768.

On Probabilistic Linearity. A common practice in the cube attack related literature consists in using a probabilistic linearity test, meaning that a (small) chance exists that the superpolys found by an attack are not actually linear. In particular, the best results obtained with the cube attack against Trivium use a complete-graph test, which, with respect to the standard BLR test, trades-off accuracy for efficiency. The viability of a similar choice is supported by previous work [12,21], showing that the complete-graph test behaves essentially as a BLR test in testing a randomly chosen function f, with the quality of the former being especially high if the nonlinearity (minimum distance from any affine function) of f is large, that is, when the result of the test is particularly relevant.

Following the trend, we chose to implement a complete-graph test based on a set of 10 randomly chosen keys, exactly as done in [14]. However, while increasing the number of tests done *during* the attack was costly for us (it impacts on memory usage), implementing further test on the superpolys found *at the end* of the attack was not. We therefore decided to put our superpolys through additional tests involving other 15 keys chosen uniformly at random. Figure 2b compares $\mathrm{rk}(A_i)$ as a function of i, for our full results and our filtered results, in which all superpolys that failed at least one of the additional tests have been removed. Let us stress once more that these two sets of results cannot be defined as wrong and correct, but they rather correspond to two different levels of trust in the found superpolys. In a sense, choosing between the two sets is equivalent to selecting the desired trade-off between efficiency and reliability of the attack: our full results permit a faster attack, which however may fail for a subset of all possible keys. Of course, many middle ways/intermediate approaches are possible. Investigating whether the reason of these failing tests is related to any of our design choices is left to future work.

On Using 32 Output Bits. A significant novelty of our implementation consists in the ability to concurrently attack 32 different polynomials, which describe 32 consecutive output bits of the target cipher. This choice is induced by GPUs features – as discussed in Sect. 3.2 – yet it is natural to assess what benefits it introduces. In Sect. 4.1 we showed that looking at 32 output bits altogether can be considered a way to concurrently attack 32 different reduced-round variants of Trivium. However, aiming to extend the attack to the full version of the cipher, our implementation can be used to check whether the same set of monomials yield different superpolys, hopefully involving different key variables, when we focus on different output bits. To this end, let us introduce a new set of matrices $B^0_{768}, \ldots, B^{31}_{768}$, where each B^j_{768} is obtained considering only the superpolys yielded by output bits $0, \ldots, j$ after 768 initialization rounds (*i.e.*, $A_{768} = B^{31}_{768}$). Figure 2c shows $\mathrm{rk}(B^j_{768})$ as a function of j, for both our full results and our filtered results. What the figure highlights is that considering several output bits altogether for the same version of the cipher, albeit possibly causing issues related to memory usage, does introduce the expected benefit, indeed a remarkable benefit if the matrix rank is initially (*i.e.*, when $j = 0$) low. This is the first ever result showing that considering a larger set of output bits is a viable alternative to exploring a larger cube.

On the Advantages of the Exhaustive Search. As described before, our implementation allows to find significantly more linearly independent superpolys than previous attempts from the literature. One of the reasons of our findings is the parallelization that makes possible to carry out, at a negligible cost, an exhaustive search over all public variables in $I_{\max} \setminus I$ when the cube C_I is under test. Figure 2d, again focusing on $\mathrm{rk}(A_i)$, compares our full results with results obtained without exhaustive search (shortened "no ex."), *i.e.*, setting all variables in $I_{\max} \setminus I$ to 0, as usually done in related work. What emerges is that through an exhaustive search it is indeed possible to remarkably increase $\mathrm{rk}(A_i)$. Significantly, the exhaustive search is what allows us to improve on the state-of-the-art for $i = 799$, which, among other things, suggests that the benefits of the exhaustive search are particularly relevant when increasing the number of tested cubes would be difficult otherwise (*e.g.*, by considering other monomials).

Another consequence of implementing an exhaustive search is that we found many redundant superpolys, *i.e.*, superpolys that are identical or just linearly dependent with the ones composing the maximal rank matrix $\tilde{A}$. A similar finding is extremely interesting, because we expect it to provide a wide choice of different IV combinations yielding superpolys that compose a maximal rank sub-matrix $\tilde{A}$, thus weakening the standard assumption that cube attacks require a completely tweakable IV.

5 Related Work

The cube attack is a widely applicable method of cryptanalysis introduced by Dinur and Shamir [10]. The underlying idea, similar to Vielhaber's AIDA [24], can be extended, *e.g.*, by assigning a *dynamic* value to IV bits not belonging to the tested cube [3,11], or by replacing cubes with generic subspaces of the IV space [25], and it is used in so-called *cube testers* to detect nonrandom behaviour rather than performing key extraction [4,5]. Despite the cube attack and its variants have shown promising results against several ciphers (*e.g.*, Trivium [10], Grain [11], Hummingbird-2 [13], Katan and Simon [3], Quavium [26]), Bernstein [6] expressed harsh criticism to the feasibility and convenience of cube attacks. Indeed, a general trend for cube attacks is to focus on reduced-round variants of a cipher, without any evidence that the full version can be equally attacked. However, while Bernstein suggests that the cube attack only works if the ANF of the cipher has low degree, Fouque and Vannet [14] argue (and, to some extent, experimentally show) that effective cube attacks can be run not aiming at the maximum degree of the ANF, but rather exploiting a nonrandom ANF by searching for maxterms of significantly lower degree. Along this line, O'Neil [18] suggests that even the full version of Trivium exhibits limited randomness, thus indicating the potential vulnerability of this cipher to cube attacks.

In recent years, several implementations of the cube attack attempted at breaking Trivium, our target cipher described in Sect. A. Quedenfeld and Wolf [19] found cubes for Trivium up to round 446. Srinivasan et al. [23] introduces a sufficient condition for testing a superpoly for linearity in $\mathbb{F}_2$ with a time complexity $\mathrm{O}(2^{c+1}(k^2 + k))$, yielding 69 extremely sparse linearly independent superpolys for Trivium reduced to 576 rounds. In their seminal paper [10], Dinur and Shamir found 63, 53, and 35 linearly independent superpolys after, respectively, 672, 735, and 767 rounds. Fouque and Vannet [14] even improve over Dinur and Shamir, by obtaining 42 linearly independent superpolys after 784 rounds, and 12 linearly independent superpolys (plus 6 quadratic superpolys) after 799 rounds. To the best of our knowledge, these are the best results against Trivium to date, making our attack comparable to (or better than) the state-of-the-art.

Distributed computing and/or parallel processing have been explored in the literature to render attacks to crypto systems computationally or storage-wise feasible/practical. Smart et al. [15] develop a new methodology to assess cryptographic key strength using cloud computing. Marks et al. [16] provide numerical evidence of the potential of mixed GPU(AMD, Nvidia) and CPU technology to data encryption and decryption algorithms. Focusing on GPU, Milo et al. [17] leverage GPUs to quickly test passphrases used to protect private keyrings of OpenPGP cryptosystems, showing that the time complexity of the attack can be reduced up to three-orders of magnitude with respect to a standard procedure, and up to ten times with respect to a highly tuned CPU implementation. A relevant result is obtained by Agostini [2] leveraging GPUs to speed up Dictionary Attacks to the BitLocker technology commonly used in Windows OSes to encrypt disks. Finally, and most closely related to the present work, Fan and Gong [13] make use of GPUs to perform side channel cube attacks on Hummingbird-2. They describe an efficient term-by-term quadraticity test for extracting simple quadratic equations, leveraging the cube attack. Just like us, Fan and Gong speed-up the implementation of the proposed term-by-term quadraticity test by leveraging GPUs and finally recovering 48 out of 128 key bits of the Hummingbird-2 with a data complexity of about 2^{18} chosen plaintexts. However, we present a complete implementation of the cube attack thoroughly designed and optimized for GPUs. Our flexible construction allows an exhaustive exploration of subsets of IV bits, thus overcoming the limitations of dynamic cube attacks, which try to find the most suitable assignment to those bits by analyzing the target cipher.

6 Conclusions and Future Work

This work has discussed in depth an advanced GPU implementation of the cube attack aimed at breaking a reduced-round version of Trivium. The implemented attack allows extending the quest for superpolys to a dimension never explored in previous works, and weakens the previous cube attack assumption of a completely tweakable IV. An extensive experimental campaign is discussed and results validate the approach and improve over the state-of-the-art attacks against reduced-round versions of Trivium.

The tool, that we expect to release into the public domain, opens new perspectives by allowing a more comprehensive and hopefully exhaustive analysis of stream-ciphers security. For instance, along the line proposed in [1], we envisage developing our implementation to test the effectiveness of the generalized cube attack over $\mathbb{F}_n$.

A Trivium Specifications

Trivium [8] is a synchronous stream cipher conceived by Christophe De Cannière and Bart Preneel, not patented, and specified as an International Standard under ISO/IEC 29192-3. Trivium combines a flexible trade-off between speed and gate count in hardware, and a reasonably efficient software implementation. Citing [9]: "Trivium is a hardware oriented design focussed on flexibility. It aims to be compact in environments with restrictions on the gate count, power-efficient on platforms with limited power resources, and fast in applications that require high-speed encryption". Particularly interesting is the fact that any state bit stays unused for at least 64 iterations after it has been modified. This means that up to 64 iterations can be parallelized and computed at once, allowing for a factor 64 reduction in the clock frequency without affecting the throughput.

Trivium generates up to 2^{64} bits of output from an 80-bit key K and an 80-bit Initial Vector IV, and it shows remarkable resistance to cryptanalysis despite its simplicity and its excellent performance. The 80-bit key K and the 80-bit IV, are used in Trivium to initialize three FSRs of length 93, 84 and 111, respectively. The internal states of the three registers are denoted $(s_1, \dots, s_{93})$, $(s_{94}, \dots, s_{177})$ and $(s_{178}, \dots, s_{288})$ respectively. Fifteen out the 288 internal state bits are used at each round to compute the feedbacks to the three registers and the output bit of the cipher. However, to obtain a better mixing of the seed and to guarantee that each output bit is a complex non-linear function of all key-bits and IV-bits, the cipher undergoes 1152 initialization rounds without producing any output. In detail, the initial seed of the three registers is defined as follows:

$$\begin{aligned}
(s_1, \dots, s_{93}) &\leftarrow (K_1, \dots, K_{80}, 0, \dots, 0)\\
(s_{94}, \dots, s_{177}) &\leftarrow (IV_1, \dots, IV_{80}, 0, \dots, 0)\\
(s_{178}, \dots, s_{288}) &\leftarrow (0, \dots, 0, 1, 1, 1)
\end{aligned}$$

For each $t \geq 1$, the internal state of the cipher is updated as follows[4]:

$$\begin{aligned}
(s_1, s_2, \dots, s_{93}) &\leftarrow (s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}, s_1, \dots, s_{92})\\
(s_{94}, s_{95}, \dots, s_{177}) &\leftarrow (s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}, s_{94}, \dots, s_{176})\\
(s_{178}, s_{179}, \dots, s_{288}) &\leftarrow (s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}, s_{178}, \dots, s_{287})
\end{aligned}$$

Finally, for each $t > 1152$, the output bit z_t is computed as:

$$z_t \leftarrow s_{66} + s_{93} + s_{162} + s_{177} + s_{243} + s_{288}$$

[4] Symbols $+$ and $\cdot$ denote sum and product over $\mathbb{F}_2$, *i.e.*, bitwise XOR and AND.

B Tables of Maxterms and Superpolys

Trivium-781

maxterm bits	superpoly	round
3, 6, 8, 10, 12, 14, 18, 19, 20, 23, 25, 27, 31, 33, 38, 40, 43, 45, 48, 53, 54, 56, 58, 60, 62, 63, 69, 75, 77, 79, 80	x_{55}	781
1, 5, 7, 8, 10, 15, 16, 18, 20, 23, 25, 27, 32, 33, 36, 38, 40, 41, 43, 47, 49, 52, 53, 54, 56, 58, 63, 69, 71, 75, 77, 80	x_{69}	781
1, 6, 7, 8, 10, 12, 16, 19, 21, 24, 25, 27, 31, 33, 36, 38, 40, 41, 43, 47, 49, 52, 53, 56, 58, 63, 67, 69, 71, 73, 77, 80	x_{60}	781
1, 2, 3, 5, 6, 7, 8, 12, 14, 15, 16, 19, 21, 23, 25, 27, 36, 38, 40, 43, 45, 47, 49, 54, 56, 58, 60, 62, 69, 71, 73, 74, 80	$x_{51} + 1$	781
1, 2, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 23, 25, 27, 36, 38, 40, 43, 45, 47, 52, 54, 56, 58, 60, 62, 69, 71, 73, 76, 79, 80	x_{45}	781
1, 2, 5, 6, 7, 8, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 43, 45, 47, 49, 52, 54, 56, 58, 62, 65, 69, 71, 73, 76, 80	$x_{43} + x_{58}$	781
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 23, 27, 33, 36, 38, 40, 43, 45, 47, 52, 54, 56, 58, 60, 62, 69, 71, 73, 74, 79, 80	x_{23}	781
1, 3, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 33, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 65, 69, 71, 75, 77, 79, 80	$x_8 + x_{35} + x_{64}$	781
1, 5, 7, 8, 10, 12, 14, 15, 16, 18, 20, 23, 24, 25, 27, 32, 33, 36, 40, 41, 43, 47, 49, 52, 53, 56, 58, 63, 69, 71, 75, 77, 80	$x_{67} + 1$	781
3, 5, 6, 8, 10, 12, 14, 16, 18, 19, 20, 23, 24, 25, 27, 31, 33, 38, 43, 45, 48, 53, 54, 56, 58, 60, 62, 63, 69, 75, 77, 79, 80	x_2	781
6, 8, 10, 12, 14, 15, 16, 18, 19, 20, 21, 23, 25, 27, 29, 31, 33, 36, 38, 40, 41, 42, 45, 49, 54, 56, 60, 62, 63, 69, 73, 75, 80	x_{58}	781
1, 2, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 23, 27, 36, 38, 40, 43, 45, 47, 49, 52, 54, 56, 58, 60, 62, 69, 71, 73, 74, 76, 79, 80	$x_{62} + 1$	781
1, 3, 5, 6, 8, 10, 12, 14, 15, 16, 19, 21, 25, 30, 31, 32, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 54, 56, 63, 69, 71, 73, 75, 80	$x_3 + x_{25} + x_{39} + x_{40} + x_{51} + x_{66} + x_{67} + x_{78} + 1$	781
1, 3, 5, 6, 8, 12, 14, 15, 16, 19, 21, 25, 27, 31, 33, 38, 40, 41, 43, 45, 47, 48, 49, 50, 53, 54, 56, 60, 63, 69, 71, 73, 75, 80	$x_{10} + x_{13} + x_{14} + x_{19} + x_{25} + x_{28} + x_{29} + x_{31} + x_{37} + x_{40} + x_{46} + x_{52} + x_{53} + x_{55} + x_{56} + x_{57} + x_{60} + x_{61} + x_{62} + x_{64} + x_{66} + x_{68} + x_{69} + 1$	781
1, 3, 5, 7, 12, 14, 15, 16, 18, 19, 20, 21, 24, 25, 27, 31, 33, 36, 40, 41, 45, 49, 54, 56, 58, 60, 62, 63, 66, 71, 73, 75, 77, 80	x_{57}	781
1, 3, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 33, 36, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 61, 63, 65, 69, 71, 75, 77, 79, 80	$x_{43} + x_{58} + x_{64} + x_{66} + x_{70}$	781
1, 3, 5, 6, 7, 8, 12, 13, 14, 15, 16, 19, 21, 23, 25, 27, 36, 38, 40, 43, 45, 47, 49, 52, 54, 56, 58, 62, 65, 69, 71, 73, 76, 79, 80	x_{65}	781
1, 3, 5, 6, 8, 10, 12, 14, 15, 16, 18, 19, 24, 25, 31, 32, 33, 36, 38, 40, 41, 43, 45, 47, 49, 50, 53, 54, 56, 63, 69, 71, 73, 75, 80	$x_{23} + x_{39} + x_{50} + x_{66} + x_{67} + x_{79} + 1$	781
1, 3, 5, 6, 8, 10, 12, 14, 15, 16, 18, 19, 25, 30, 31, 32, 33, 38, 40, 41, 43, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 71, 73, 75, 80	$x_9 + x_{18} + x_{24} + x_{26} + x_{32} + x_{33} + x_{34} + x_{42} + x_{51} + x_{53} + x_{54} + x_{58} + x_{59} + x_{64} + x_{66} + x_{68} + x_{69} + x_{80} + 1$	781
1, 3, 5, 6, 8, 10, 12, 14, 15, 16, 19, 21, 24, 25, 31, 32, 33, 36, 38, 40, 41, 43, 45, 47, 49, 50, 53, 54, 56, 63, 69, 71, 73, 75, 80	$x_{52} + x_{66} + x_{67} + x_{79}$	781
1, 3, 5, 6, 8, 12, 14, 15, 16, 18, 19, 21, 25, 27, 31, 33, 38, 40, 41, 43, 45, 47, 48, 49, 50, 53, 54, 56, 60, 63, 69, 71, 73, 75, 80	$x_{13} + x_{14} + x_{19} + x_{25} + x_{27} + x_{28} + x_{29} + x_{31} + x_{39} + x_{41} + x_{42} + x_{46} + x_{51} + x_{52} + x_{54} + x_{55} + x_{56} + x_{57} + x_{61} + x_{62} + x_{64} + x_{65} + x_{66} + x_{69} + x_{78}$	781
1, 3, 5, 6, 8, 12, 14, 15, 16, 18, 19, 24, 25, 27, 31, 32, 33, 36, 38, 40, 41, 45, 47, 48, 49, 53, 54, 56, 60, 63, 69, 71, 73, 75, 80	$x_{16} + x_{26} + x_{27} + x_{38} + x_{43} + x_{53} + x_{54} + x_{56} + x_{65} + x_{67} + x_{80}$	781
1, 3, 5, 6, 8, 12, 14, 15, 16, 19, 21, 24, 25, 30, 31, 32, 33, 36, 38, 40, 41, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 71, 73, 75, 80	$x_{25} + x_{27} + x_{30} + x_{54} + x_{57}$	781
1, 2, 3, 5, 6, 7, 8, 12, 14, 15, 16, 19, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 52, 54, 56, 58, 60, 62, 65, 69, 70, 71, 73, 74, 76, 80	x_{42}	781
1, 3, 5, 6, 8, 10, 12, 14, 15, 16, 18, 19, 21, 25, 27, 30, 31, 32, 33, 38, 40, 41, 43, 45, 47, 48, 49, 53, 54, 56, 63, 69, 71, 73, 75, 80	$x_{14} + x_{29} + x_{41} + x_{55} + x_{61} + x_{62}$	781
1, 3, 5, 6, 8, 10, 12, 14, 15, 16, 18, 19, 24, 25, 31, 32, 33, 36, 38, 40, 41, 43, 45, 47, 48, 49, 50, 53, 54, 56, 63, 69, 71, 73, 75, 80	$x_{39} + x_{66}$	781
1, 3, 5, 6, 8, 10, 12, 14, 15, 16, 18, 19, 25, 30, 31, 32, 33, 38, 40, 41, 43, 45, 47, 48, 49, 50, 53, 54, 56, 60, 63, 69, 71, 73, 75, 80	$x_{24} + x_{55} + x_{61} + x_{66} + x_{67} + 1$	781
1, 3, 5, 6, 8, 10, 12, 14, 15, 16, 19, 21, 24, 25, 30, 31, 33, 38, 40, 41, 43, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 70, 71, 73, 75, 80	$x_{12} + x_{27} + x_{32} + x_{33} + x_{40} + x_{42} + x_{51} + x_{53} + x_{57} + x_{58} + x_{60} + x_{64} + x_{80} + 1$	781
1, 3, 5, 6, 8, 10, 12, 14, 15, 16, 19, 21, 24, 25, 30, 31, 33, 38, 40, 41, 43, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 70, 71, 73, 75, 80	$x_{27} + x_{32} + x_{42} + x_{53} + x_{58} + x_{60} + x_{64} + x_{78} + x_{80} + 1$	781
1, 3, 5, 6, 8, 12, 14, 15, 16, 18, 19, 24, 25, 27, 30, 31, 32, 33, 38, 40, 41, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 70, 71, 73, 75, 80	$x_{11} + x_{24} + x_{25} + x_{29} + x_{30} + x_{31} + x_{40} + x_{41} + x_{45} + x_{50} + x_{52} + x_{53} + x_{54} + x_{56} + x_{58} + x_{61} + x_{65} + x_{66} + x_{67} + x_{68} + x_{77} + x_{79} + x_{80} + 1$	781

1, 3, 5, 6, 8, 12, 14, 15, 16, 19, 21, 24, 25, 30, 31, 32, 33, 36, 38, 40, 41, 43, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 71, 73, 75, 80	$x_{14} + x_{16} + x_{27} + x_{29} + x_{30} + x_{31} + x_{40} + x_{41} + x_{42} + x_{43} + x_{54} + x_{55} + x_{56} + x_{57} + x_{58} + x_{64} + x_{79} + x_{80} + 1$	781
1, 3, 5, 6, 8, 12, 14, 15, 16, 19, 21, 24, 25, 30, 31, 32, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 54, 56, 60, 63, 69, 70, 71, 73, 75, 80	$x_{28} + x_{29} + x_{32} + x_{33} + x_{40} + x_{41} + x_{42} + x_{44} + x_{50} + x_{51} + x_{55} + x_{56} + x_{57} + x_{59} + x_{61} + x_{62} + x_{64} + x_{66} + x_{67} + x_{68} + x_{70} + x_{78} + 1$	781
1, 3, 5, 6, 8, 12, 14, 15, 16, 19, 21, 24, 25, 30, 31, 33, 36, 38, 40, 41, 43, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 70, 71, 73, 75, 80	$x_{38} + x_{39} + x_{41} + x_{44} + x_{45} + x_{50} + x_{51} + x_{52} + x_{53} + x_{55} + x_{57} + x_{58} + x_{60} + x_{66} + x_{68} + x_{72} + x_{78} + x_{79} + 1$	781
1, 3, 5, 6, 8, 12, 14, 15, 16, 19, 21, 24, 25, 31, 33, 36, 38, 40, 41, 43, 45, 47, 48, 49, 50, 53, 54, 56, 60, 63, 69, 70, 71, 73, 75, 80	$x_{43} + x_{50} + x_{52} + x_{55} + x_{58} + x_{66} + x_{70} + x_{77} + 1$	781
1, 5, 6, 8, 12, 14, 15, 16, 18, 19, 24, 25, 27, 30, 31, 32, 33, 38, 40, 41, 43, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 70, 71, 73, 75, 80	$x_{41} + x_{53} + x_{55} + x_{58} + x_{61} + x_{68}$	781
1, 5, 6, 8, 12, 14, 15, 16, 18, 19, 24, 25, 27, 30, 31, 32, 33, 38, 40, 41, 43, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 70, 71, 73, 75, 80	$x_{29} + x_{41} + x_{42} + x_{53} + x_{55} + x_{56} + x_{58} + x_{61} + x_{64} + x_{66} + x_{67} + x_{68} + x_{69}$	781
1, 5, 6, 8, 12, 14, 15, 16, 18, 19, 24, 25, 27, 30, 31, 32, 33, 38, 40, 41, 43, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 70, 71, 73, 75, 80	$x_{14} + x_{55} + x_{58} + x_{61} + x_{64} + x_{66} + x_{68} + x_{80}$	781
1, 5, 6, 10, 12, 14, 15, 18, 19, 20, 21, 22, 23, 25, 27, 28, 29, 31, 33, 36, 38, 40, 41, 42, 45, 48, 49, 54, 60, 62, 63, 69, 73, 75, 77, 80	x_{64}	781
5, 6, 8, 10, 12, 14, 15, 16, 18, 19, 20, 21, 22, 23, 25, 27, 29, 31, 33, 36, 38, 40, 41, 42, 45, 48, 49, 54, 60, 62, 63, 69, 73, 75, 77, 80	$x_{66} + 1$	781
1, 2, 3, 5, 6, 7, 8, 12, 13, 14, 15, 16, 19, 21, 23, 27, 33, 36, 38, 40, 43, 45, 47, 52, 54, 56, 58, 60, 62, 69, 70, 71, 73, 74, 76, 79, 80	x_{56}	781
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 33, 36, 40, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 65, 69, 71, 74, 75, 77, 78, 79, 80	$x_{21} + x_{36} + x_{48} + x_{58} + x_{63} + 1$	781
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 33, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 65, 69, 71, 74, 75, 77, 78, 79, 80	$x_{19}+x_{27}+x_{45}+x_{54}+x_{64}+x_{66}+x_{72}+1$	781
1, 3, 5, 6, 8, 12, 14, 15, 16, 18, 19, 24, 25, 30, 31, 32, 33, 36, 38, 40, 41, 43, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 70, 71, 73, 75, 80	$x_5 + x_8 + x_{24} + x_{26} + x_{32} + x_{33} + x_{39} + x_{40} + x_{41} + x_{42} + x_{44} + x_{47} + x_{51} + x_{54} + x_{57} + x_{59} + x_{60} + x_{65} + x_{66} + x_{68} + x_{69} + x_{78} + x_{79}$	781
1, 3, 5, 6, 8, 12, 14, 15, 16, 19, 21, 25, 27, 31, 32, 33, 36, 38, 40, 41, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 71, 73, 75, 80	$x_{25} + x_{52} + 1$	782
1, 5, 6, 7, 8, 10, 12, 14, 15, 19, 21, 24, 25, 27, 31, 36, 38, 39, 40, 41, 45, 47, 49, 53, 56, 58, 62, 63, 66, 69, 71, 73, 77, 80	x_{40}	782
1, 3, 5, 6, 7, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 56, 58, 61, 62, 63, 65, 69, 71, 74, 75, 77, 78	$x_{25} + 1$	782
1, 3, 5, 6, 10, 12, 14, 15, 16, 18, 19, 21, 25, 27, 31, 32, 33, 38, 40, 41, 43, 45, 47, 48, 49, 50, 53, 54, 56, 60, 63, 69, 71, 73, 75, 80	$x_{13} + x_{16} + x_{19} + x_{25} + x_{29} + x_{33} + x_{35} + x_{36} + x_{37} + x_{38} + x_{39} + x_{40} + x_{42} + x_{45} + x_{51} + x_{52} + x_{53} + x_{54} + x_{55} + x_{62} + x_{63} + x_{64} + x_{65} + x_{67} + x_{69} + x_{70} + x_{71} + x_{73} + x_{79} + x_{80} + 1$	782
5, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 34, 36, 38, 40, 45, 47, 48, 49, 53, 54, 56, 58, 60, 62, 63, 69, 71, 80	$x_{38} + 1$	783
3, 5, 6, 7, 8, 10, 14, 15, 16, 21, 23, 25, 27, 33, 34, 36, 38, 40, 45, 47, 48, 49, 53, 54, 55, 56, 58, 61, 62, 63, 69, 71, 74, 80	$x_{27} + 1$	783
1, 5, 6, 8, 10, 12, 14, 15, 16, 18, 19, 21, 24, 25, 27, 30, 31, 32, 33, 38, 40, 41, 45, 47, 48, 49, 50, 53, 54, 56, 63, 69, 71, 73, 75, 80	$x_{32} + x_{49} + x_{52} + x_{56} + x_{59} + x_{61} + x_{62} + x_{79} + 1$	783
1, 5, 6, 8, 10, 12, 14, 15, 16, 18, 19, 21, 24, 25, 31, 33, 36, 38, 40, 41, 43, 45, 47, 48, 49, 50, 53, 54, 56, 60, 63, 69, 71, 73, 75, 80	$x_7 + x_{16} + x_{40} + x_{43} + x_{49} + x_{52} + x_{58} + x_{62} + x_{70} + x_{79} + 1$	783
1, 3, 5, 6, 8, 10, 12, 14, 15, 16, 18, 19, 24, 25, 30, 31, 32, 33, 36, 38, 40, 41, 43, 45, 47, 49, 50, 53, 54, 56, 60, 63, 69, 71, 73, 75, 80	$x_{26} + x_{66} + x_{68} + 1$	783
1, 3, 6, 7, 10, 12, 15, 16, 19, 21, 23, 25, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 63, 65, 69, 71, 75, 77, 80	x_4	784
1, 5, 6, 7, 8, 10, 12, 14, 15, 16, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 55, 56, 58, 60, 62, 63, 67, 69, 71, 80	$x_{53} + 1$	784
1, 6, 8, 10, 12, 14, 15, 16, 18, 19, 20, 21, 22, 23, 25, 27, 29, 33, 36, 38, 40, 41, 42, 45, 48, 49, 54, 60, 62, 63, 69, 73, 75, 80	x_{37}	784
3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 60, 62, 63, 69, 71, 74, 80	x_{36}	784
1, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 29, 31, 33, 36, 38, 40, 41, 45, 47, 49, 53, 58, 60, 63, 71, 75, 76, 80	$x_{12} + 1$	785
1, 3, 6, 7, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 63, 65, 69, 71, 74, 75, 77, 78, 80	x_{34}	785
1, 5, 6, 7, 8, 12, 13, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 52, 54, 56, 58, 60, 62, 69, 71, 73, 79, 80	x_{54}	785
1, 3, 5, 6, 7, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 69, 71, 74, 75, 77, 78, 80	$x_{13} + x_{55} + x_{60} + x_{64}$	785
1, 3, 5, 6, 7, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 65, 69, 71, 74, 75, 77, 78, 80	$x_{22} + x_{49} + x_{64}$	786
1, 3, 6, 7, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 65, 69, 71, 74, 75, 77, 78, 80	$x_{14} + x_{23} + x_{41} + x_{47} + x_{49} + x_{50} + x_{58} + x_{64}$	786
1, 3, 5, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 65, 69, 71, 74, 75, 77, 78, 80	$x_3 + x_4 + x_{20} + x_{22} + x_{30} + x_{34} + x_{38} + x_{40} + x_{42} + x_{45} + x_{49} + x_{51} + x_{58} + x_{61} + x_{65} + x_{67} + x_{69} + x_{72} + x_{78}$	786
1, 3, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 38, 40, 43, 45, 47, 49, 53, 56, 58, 62, 63, 69, 71, 75, 77, 79	$x_9 + x_{29} + x_{30} + x_{32} + x_{42} + x_{43} + x_{49} + x_{51} + x_{57} + x_{58} + x_{59} + x_{60} + x_{62} + x_{64} + x_{66} + x_{67} + x_{68} + x_{69} + x_{70} + x_{72} + x_{76}$	791
1, 3, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 36, 40, 41, 43, 45, 47, 49, 53, 56, 58, 62, 63, 69, 71, 75, 77, 80	$x_{17} + x_{26} + x_{30} + x_{32} + x_{41} + x_{43} + x_{47} + x_{57} + x_{62} + x_{65} + x_{66} + x_{70} + x_{72} + x_{74} + 1$	791

1, 3, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 36, 40, 41, 43, 45, 47, 49, 53, 56, 58, 62, 63, 69, 71, 75, 77, 80	$x_{14} + x_{17} + x_{26} + x_{30} + x_{43} + x_{47} + x_{50} + x_{57} + x_{58} + x_{59} + x_{65} + x_{70} + x_{72} + x_{74} + x_{77} + 1$	791
1, 3, 5, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 40, 41, 43, 45, 47, 49, 53, 56, 58, 63, 65, 69, 71, 75, 77, 79	$x_{12}+x_{26}+x_{30}+x_{39}+x_{41}+x_{45}+x_{47}+x_{57}+x_{58}+x_{59}+x_{62}+x_{64}+x_{74}+x_{76}+1$	791
1, 3, 5, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 56, 58, 62, 63, 65, 69, 71, 75, 77	$x_5+x_{18}+x_{20}+x_{26}+x_{28}+x_{29}+x_{30}+x_{31}+x_{32}+x_{41}+x_{42}+x_{44}+x_{50}+x_{51}+x_{56}+x_{57}+x_{62}+x_{64}+x_{67}+x_{69}+x_{70}+x_{71}+x_{74}+x_{77}+x_{78}+1$	791
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 38, 40, 41, 43, 45, 47, 49, 53, 56, 58, 61, 62, 63, 65, 69, 71, 75, 77	$x_1+x_{28}+x_{32}+x_{47}+x_{58}+x_{59}+x_{62}+x_{64}+x_{74}$	791
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 40, 41, 43, 45, 47, 49, 53, 56, 58, 61, 62, 63, 65, 69, 71, 75, 77, 80	$x_{10}+x_{11}+x_{12}+x_{13}+x_{15}+x_{17}+x_{19}+x_{20}+x_{29}+x_{31}+x_{32}+x_{33}+x_{37}+x_{39}+x_{40}+x_{41}+x_{42}+x_{44}+x_{46}+x_{48}+x_{49}+x_{50}+x_{53}+x_{57}+x_{60}+x_{67}+x_{70}+x_{71}+x_{76}+x_{78}+x_{79}$	791
1, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 29, 31, 33, 36, 38, 40, 41, 42, 45, 47, 49, 53, 58, 63, 69, 71, 72, 76, 79, 80	x_{61}	791
1, 3, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 69, 71, 74, 75, 77, 78, 80	$x_{43}+x_{47}+x_{58}+x_{70}+x_{74}+1$	791
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 38, 40, 41, 43, 45, 47, 49, 53, 56, 58, 62, 63, 69, 71, 75, 77, 80	$x_{12}+x_{17}+x_{26}+x_{27}+x_{29}+x_{30}+x_{32}+x_{40}+x_{43}+x_{45}+x_{46}+x_{49}+x_{53}+x_{54}+x_{56}+x_{59}+x_{62}+x_{64}+x_{65}+x_{67}+x_{69}+x_{72}+x_{74}+x_{75}$	792
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 56, 58, 61, 63, 69, 71, 75, 77, 78, 79, 80	$x_{12}+x_{14}+x_{26}+x_{30}+x_{40}+x_{41}+x_{47}+x_{48}+x_{56}+x_{66}+x_{67}+x_{68}+x_{74}+x_{75}+1$	792
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 63, 69, 71, 74, 75, 77, 78, 79, 80	$x_{16}+x_{43}+x_{56}$	792
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 63, 65, 69, 71, 75, 77, 78, 79, 80	$x_{14}+x_{16}+x_{26}+x_{29}+x_{30}+x_{41}+x_{45}+x_{55}+x_{56}+x_{59}+x_{62}+x_{64}+x_{66}+x_{68}+x_{70}+x_{71}+x_{72}+1$	792
1, 3, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 61, 63, 69, 71, 75, 77, 80	$x_{45}+x_{72}$	793
1, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 27, 30, 31, 33, 38, 40, 43, 45, 47, 49, 51, 52, 56, 58, 63, 67, 69, 71, 73, 77, 80	$x_{10}+x_{55}$	798
1, 3, 5, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 65, 69, 71, 74, 75, 77, 80	$x_{36}+x_{52}+x_{60}+x_{63}$	798
1, 3, 5, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 65, 69, 71, 74, 75, 77, 80	$x_6+x_{11}+x_{25}+x_{33}+x_{36}+x_{53}+x_{60}+x_{62}+x_{63}+x_{64}+x_{79}$	798

Trivium-784

maxterm bits	superpoly	round
1, 3, 6, 7, 10, 12, 15, 16, 19, 21, 23, 25, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 63, 65, 69, 71, 75, 77, 80	x_4	784
1, 5, 6, 8, 10, 12, 14, 15, 16, 18, 19, 20, 21, 23, 25, 27, 29, 31, 33, 36, 38, 40, 41, 42, 45, 49, 54, 60, 62, 69, 73, 75, 80	x_{60}	784
5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 67, 69, 71, 80	$x_{56}+1$	784
1, 3, 5, 6, 7, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 63, 65, 69, 71, 75, 77, 78	$x_2+x_9+x_{13}+x_{14}+x_{22}+x_{23}+x_{30}+x_{36}+x_{38}+x_{39}+x_{40}+x_{42}+x_{47}+x_{48}+x_{51}+x_{56}+x_{65}+x_{67}+x_{68}+x_{69}+x_{74}+x_{75}$	784
1, 3, 5, 6, 8, 10, 12, 14, 15, 16, 19, 21, 25, 30, 31, 32, 33, 38, 40, 41, 43, 45, 47, 48, 49, 50, 53, 54, 56, 63, 69, 71, 75, 80	x_{38}	784
1, 3, 6, 7, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 61, 63, 69, 71, 75, 77, 78, 79	$x_{38}+x_{47}+x_{74}$	784
1, 5, 6, 7, 8, 10, 12, 14, 15, 16, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 55, 56, 58, 60, 62, 63, 67, 69, 71, 80	$x_{53}+1$	784
1, 5, 6, 7, 8, 10, 12, 14, 15, 16, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 67, 69, 71, 74, 80	x_{58}	784
1, 6, 8, 10, 12, 14, 15, 16, 18, 19, 20, 21, 22, 23, 25, 27, 29, 33, 36, 38, 40, 41, 42, 45, 48, 49, 54, 60, 62, 63, 69, 73, 75, 80	x_{37}	784
1, 6, 8, 10, 12, 14, 15, 16, 18, 19, 20, 21, 22, 23, 25, 27, 29, 33, 36, 38, 40, 41, 45, 48, 49, 54, 56, 60, 62, 69, 73, 75, 77, 80	x_{64}	784
3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 60, 62, 63, 69, 71, 74, 80	x_{36}	784
6, 8, 10, 12, 14, 15, 16, 18, 19, 20, 21, 22, 23, 25, 27, 29, 33, 36, 38, 40, 41, 45, 48, 49, 54, 56, 60, 62, 63, 69, 73, 75, 77, 80	x_{66}	784
5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 34, 36, 38, 40, 43, 45, 47, 49, 53, 54, 55, 56, 58, 60, 62, 63, 69, 71, 74, 80	$x_{67}+1$	784
5, 6, 8, 10, 12, 14, 15, 18, 19, 20, 21, 22, 23, 25, 27, 28, 29, 31, 33, 36, 38, 40, 41, 42, 45, 49, 54, 60, 62, 63, 69, 73, 75, 77, 80	x_{62}	784
1, 5, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 34, 36, 38, 40, 43, 45, 47, 48, 49, 53, 54, 56, 58, 60, 61, 62, 63, 69, 71, 74, 80	$x_{69}+1$	784
3, 5, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 48, 49, 53, 54, 56, 58, 60, 61, 62, 63, 67, 69, 71, 74, 80	x_{40}	784
1, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 29, 31, 33, 36, 38, 40, 41, 45, 47, 49, 53, 58, 60, 63, 71, 75, 76, 80	$x_{12}+1$	785
1, 5, 6, 7, 10, 12, 16, 19, 21, 23, 24, 25, 27, 31, 33, 36, 38, 40, 41, 43, 47, 49, 52, 56, 58, 60, 63, 67, 69, 71, 73, 77, 80	x_{42}	785

1, 5, 6, 10, 12, 14, 15, 16, 19, 21, 25, 27, 30, 31, 33, 36, 38, 40, 41, 43, 45, 47, 48, 49, 53, 54, 56, 63, 69, 71, 73, 75, 80	x_{55}	785
1, 3, 6, 7, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 63, 65, 69, 71, 74, 75, 77, 78, 80	x_{34}	785
1, 5, 6, 7, 8, 12, 13, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 52, 54, 56, 58, 60, 62, 69, 71, 73, 79, 80	x_{54}	785
1, 3, 5, 6, 7, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 69, 71, 74, 75, 77, 78, 80	$x_{13} + x_{55} + x_{60} + x_{64}$	785
1, 3, 5, 6, 7, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 65, 69, 71, 74, 75, 77, 78, 80	$x_{22} + x_{49} + x_{64}$	786
1, 3, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 65, 69, 71, 74, 75, 77, 78, 80	$x_3 + x_4 + x_7 + x_{12} + x_{20} + x_{22} + x_{30} + x_{36} + x_{39} + x_{42} + x_{43} + x_{45} + x_{47} + x_{51} + x_{58} + x_{63} + x_{69} + x_{70} + x_{72} + x_{78} + 1$	786
1, 3, 6, 7, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 65, 69, 71, 74, 75, 77, 78, 80	$x_{14} + x_{23} + x_{41} + x_{47} + x_{49} + x_{50} + x_{58} + x_{64}$	786
1, 3, 5, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 65, 69, 71, 74, 75, 77, 78, 80	$x_3 + x_4 + x_{14} + x_{22} + x_{30} + x_{34} + x_{38} + x_{41} + x_{42} + x_{45} + x_{47} + x_{49} + x_{51} + x_{58} + x_{61} + x_{65} + x_{69} + x_{72} + x_{78}$	786
1, 3, 5, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 65, 69, 71, 74, 75, 77, 78, 80	$x_3 + x_4 + x_{20} + x_{22} + x_{30} + x_{34} + x_{38} + x_{40} + x_{42} + x_{45} + x_{49} + x_{51} + x_{58} + x_{61} + x_{65} + x_{67} + x_{69} + x_{72} + x_{78}$	786
1, 3, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 38, 40, 43, 45, 47, 49, 53, 56, 58, 62, 63, 69, 71, 75, 77, 79	$x_9 + x_{29} + x_{30} + x_{32} + x_{42} + x_{43} + x_{49} + x_{51} + x_{57} + x_{58} + x_{59} + x_{60} + x_{62} + x_{64} + x_{66} + x_{67} + x_{68} + x_{69} + x_{70} + x_{72} + x_{76}$	791
1, 3, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 36, 40, 41, 43, 45, 47, 49, 53, 56, 58, 62, 63, 69, 71, 75, 77, 80	$x_{17} + x_{26} + x_{30} + x_{32} + x_{41} + x_{43} + x_{47} + x_{57} + x_{62} + x_{65} + x_{66} + x_{70} + x_{72} + x_{74} + 1$	791
1, 3, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 36, 40, 41, 43, 45, 47, 49, 53, 56, 58, 62, 63, 69, 71, 75, 77, 80	$x_1 + x_{17} + x_{26} + x_{28} + x_{41} + x_{43} + x_{47} + x_{49} + x_{59} + x_{62} + x_{64} + x_{65} + x_{66} + x_{70} + x_{72} + x_{74} + x_{76} + 1$	791
1, 3, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 36, 40, 41, 43, 45, 47, 49, 53, 56, 58, 62, 63, 69, 71, 75, 77, 80	$x_{14} + x_{17} + x_{26} + x_{30} + x_{43} + x_{47} + x_{50} + x_{57} + x_{58} + x_{59} + x_{65} + x_{70} + x_{72} + x_{74} + x_{77} + 1$	791
1, 3, 5, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 40, 41, 43, 45, 47, 49, 53, 56, 58, 63, 65, 69, 71, 75, 77, 79	$x_{12} + x_{26} + x_{30} + x_{39} + x_{41} + x_{45} + x_{47} + x_{57} + x_{58} + x_{59} + x_{62} + x_{64} + x_{74} + x_{76} + 1$	791
1, 3, 5, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 56, 58, 62, 63, 65, 69, 71, 75, 77	$x_5 + x_{18} + x_{20} + x_{26} + x_{28} + x_{29} + x_{30} + x_{31} + x_{32} + x_{41} + x_{42} + x_{44} + x_{50} + x_{51} + x_{56} + x_{57} + x_{62} + x_{64} + x_{67} + x_{69} + x_{70} + x_{71} + x_{74} + x_{77} + x_{78} + 1$	791
1, 3, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 40, 41, 43, 45, 47, 49, 53, 56, 58, 61, 62, 63, 65, 69, 71, 75, 77, 80	$x_7 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{15} + x_{17} + x_{19} + x_{20} + x_{26} + x_{30} + x_{31} + x_{33} + x_{37} + x_{39} + x_{40} + x_{41} + x_{42} + x_{44} + x_{45} + x_{46} + x_{47} + x_{48} + x_{50} + x_{51} + x_{53} + x_{56} + x_{59} + x_{60} + x_{64} + x_{67} + x_{68} + x_{70} + x_{71} + x_{72} + x_{74} + x_{78} + x_{79} + 1$	791
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 38, 40, 41, 43, 45, 47, 49, 53, 56, 58, 61, 62, 63, 65, 69, 71, 75, 77	$x_1 + x_{28} + x_{32} + x_{47} + x_{58} + x_{59} + x_{62} + x_{64} + x_{74}$	791
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 40, 41, 43, 45, 47, 49, 53, 56, 58, 61, 62, 63, 65, 69, 71, 75, 77, 80	$x_3 + x_4 + x_6 + x_7 + x_9 + x_{10} + x_{13} + x_{15} + x_{19} + x_{22} + x_{28} + x_{30} + x_{33} + x_{34} + x_{35} + x_{38} + x_{39} + x_{40} + x_{41} + x_{43} + x_{44} + x_{47} + x_{48} + x_{49} + x_{50} + x_{53} + x_{54} + x_{55} + x_{56} + x_{58} + x_{61} + x_{62} + x_{65} + x_{66} + x_{67} + x_{68} + x_{69} + x_{71} + x_{72} + x_{76} + x_{77} + x_{78}$	791
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 40, 41, 43, 45, 47, 49, 53, 56, 58, 61, 62, 63, 65, 69, 71, 75, 77, 80	$x_{10} + x_{11} + x_{12} + x_{13} + x_{15} + x_{17} + x_{19} + x_{20} + x_{29} + x_{31} + x_{32} + x_{33} + x_{37} + x_{39} + x_{40} + x_{41} + x_{42} + x_{44} + x_{46} + x_{48} + x_{49} + x_{50} + x_{53} + x_{57} + x_{60} + x_{67} + x_{70} + x_{71} + x_{76} + x_{78} + x_{79}$	791
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 40, 41, 43, 45, 47, 49, 53, 56, 58, 61, 62, 63, 65, 69, 71, 75, 77, 80	$x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{17} + x_{19} + x_{20} + x_{26} + x_{29} + x_{31} + x_{33} + x_{37} + x_{39} + x_{40} + x_{42} + x_{44} + x_{46} + x_{48} + x_{53} + x_{57} + x_{58} + x_{59} + x_{60} + x_{66} + x_{67} + x_{68} + x_{70} + x_{71} + x_{72} + x_{77} + x_{78} + x_{79} + 1$	791
1, 3, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 65, 69, 71, 75, 77, 79	$x_3 + x_4 + x_6 + x_{11} + x_{15} + x_{17} + x_{19} + x_{20} + x_{22} + x_{30} + x_{34} + x_{35} + x_{37} + x_{38} + x_{43} + x_{47} + x_{51} + x_{54} + x_{57} + x_{58} + x_{60} + x_{61} + x_{64} + x_{65} + x_{67} + x_{68} + x_{70} + x_{72} + x_{74} + x_{77} + x_{79} + 1$	791
1, 3, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 65, 69, 71, 75, 77, 79	$x_3 + x_4 + x_6 + x_{11} + x_{12} + x_{15} + x_{17} + x_{18} + x_{19} + x_{20} + x_{22} + x_{30} + x_{34} + x_{35} + x_{37} + x_{38} + x_{39} + x_{42} + x_{43} + x_{45} + x_{47} + x_{50} + x_{54} + x_{56} + x_{57} + x_{58} + x_{61} + x_{65} + x_{69} + x_{70} + x_{74} + x_{78} + x_{79} + 1$	791
1, 3, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 69, 71, 74, 75, 77, 78, 80	$x_1 + x_5 + x_9 + x_{14} + x_{18} + x_{20} + x_{26} + x_{28} + x_{32} + x_{41} + x_{42} + x_{43} + x_{45} + x_{47} + x_{49} + x_{66} + x_{67} + x_{69} + x_{70} + x_{76} + x_{78}$	791
1, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 29, 31, 33, 36, 38, 40, 41, 42, 45, 47, 49, 53, 58, 63, 69, 71, 72, 76, 79, 80	x_{61}	791
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 65, 69, 71, 75, 77, 79	$x_3 + x_4 + x_6 + x_7 + x_9 + x_{11} + x_{17} + x_{18} + x_{20} + x_{22} + x_{26} + x_{28} + x_{29} + x_{31} + x_{34} + x_{35} + x_{37} + x_{38} + x_{39} + x_{41} + x_{46} + x_{50} + x_{51} + x_{54} + x_{55} + x_{56} + x_{58} + x_{61} + x_{65} + x_{66} + x_{67} + x_{74} + x_{76} + x_{78} + x_{79} + 1$	791
1, 3, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 69, 71, 74, 75, 77, 78, 80	$x_{43} + x_{47} + x_{58} + x_{70} + x_{74} + 1$	791

1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 38, 40, 41, 43, 45, 47, 49, 53, 56, 58, 62, 63, 69, 71, 75, 77, 80	$x_{12} + x_{17} + x_{26} + x_{27} + x_{29} + x_{30} + x_{32} + x_{40} + x_{43} + x_{45} + x_{46} + x_{49} + x_{53} + x_{54} + x_{56} + x_{59} + x_{62} + x_{64} + x_{65} + x_{67} + x_{69} + x_{72} + x_{74} + x_{75}$	792
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 63, 69, 71, 75, 77, 78, 79, 80	$x_{12} + x_{26} + x_{39} + x_{56} + x_{68} + 1$	792
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 56, 58, 61, 63, 69, 71, 75, 77, 78, 79, 80	$x_{12}+x_{14}+x_{26}+x_{30}+x_{40}+x_{41}+x_{47}+x_{48}+x_{56}+x_{66}+x_{67}+x_{68}+x_{74}+x_{75}+1$	792
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 63, 69, 71, 74, 75, 77, 78, 79, 80	$x_{16} + x_{43} + x_{56}$	792
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 63, 65, 69, 71, 75, 77, 78, 79, 80	$x_{14} + x_{16} + x_{26} + x_{29} + x_{30} + x_{41} + x_{45} + x_{55} + x_{56} + x_{59} + x_{62} + x_{64} + x_{66} + x_{68} + x_{70} + x_{71} + x_{72} + 1$	792
1, 3, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 41, 43, 45, 47, 49, 53, 54, 56, 58, 61, 63, 69, 71, 75, 77, 80	$x_{45} + x_{72}$	793
1, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 27, 30, 31, 33, 38, 40, 43, 45, 47, 49, 51, 52, 56, 58, 63, 67, 69, 71, 73, 77, 80	$x_{10} + x_{55}$	798
1, 3, 5, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 65, 69, 71, 74, 75, 77, 80	$x_{36} + x_{52} + x_{60} + x_{63}$	798
1, 3, 5, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 65, 69, 71, 74, 75, 77, 80	$x_{10} + x_{17} + x_{27} + x_{36} + x_{37} + x_{40} + x_{52} + x_{59} + x_{60} + x_{63} + x_{66} + x_{67}$	798
1, 3, 5, 6, 7, 8, 10, 12, 14, 16, 19, 21, 23, 25, 33, 36, 38, 40, 43, 45, 47, 49, 53, 56, 58, 61, 62, 63, 65, 69, 71, 74, 75, 77, 78, 79	$x_{27} + x_{54} + x_{60}$	798
1, 3, 5, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 65, 69, 71, 74, 75, 77, 80	$x_6+x_{11}+x_{25}+x_{33}+x_{36}+x_{53}+x_{60}+x_{62} + x_{63} + x_{64} + x_{79}$	798
1, 3, 5, 6, 7, 8, 10, 12, 15, 16, 19, 21, 23, 25, 27, 33, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 61, 62, 63, 65, 69, 71, 74, 75, 77, 80	$x_6+x_{11}+x_{25}+x_{33}+x_{36}+x_{52}+x_{53}+x_{60} + x_{62} + x_{63} + x_{64} + x_{79}$	798
5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 34, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 60, 61, 62, 63, 67, 69, 71, 74, 80	$x_{65} + x_{66} + x_{67} + 1$	798
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 54, 56, 58, 62, 69, 71, 73, 80	$x_{25} + 1$	799
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 40, 43, 45, 47, 49, 53, 56, 58, 63, 69, 71, 75, 77, 80	$x_{12} + x_{38} + x_{39} + x_{40}$	799

Trivium-799

maxterm bits	superpoly	round
1, 6, 8, 10, 12, 14, 18, 20, 23, 25, 27, 31, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 69, 73, 75, 77, 80	x_{60}	799
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 54, 56, 58, 62, 69, 71, 73, 80	$x_{25} + 1$	799
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 40, 43, 45, 47, 49, 53, 56, 58, 63, 69, 71, 75, 77, 80	$x_{25} + x_{40}$	799
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 40, 43, 45, 47, 49, 53, 56, 58, 63, 69, 71, 75, 77, 80	$x_{12} + x_{38} + x_{39} + x_{40}$	799
1, 5, 6, 7, 8, 10, 12, 14, 15, 16, 18, 20, 23, 25, 27, 33, 36, 38, 40, 41, 43, 47, 49, 53, 56, 58, 63, 69, 71, 75, 77, 80	$x_{67} + 1$	799
1, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 31, 33, 36, 38, 40, 41, 45, 47, 49, 53, 56, 58, 63, 69, 71, 75, 80	x_{42}	799
1, 5, 6, 7, 8, 10, 12, 14, 15, 19, 21, 23, 25, 27, 31, 36, 38, 40, 41, 45, 47, 49, 53, 56, 58, 63, 69, 71, 73, 75, 77, 80	x_{53}	799
1, 5, 6, 8, 10, 12, 14, 15, 16, 18, 19, 20, 21, 23, 25, 27, 31, 33, 36, 38, 40, 41, 45, 49, 54, 56, 62, 69, 73, 75, 77, 80	x_{64}	799
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 69, 71, 80	$x_{36} + 1$	799
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 40, 41, 43, 45, 47, 49, 53, 56, 58, 63, 65, 69, 71, 75, 77, 80	x_{38}	799
1, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 54, 56, 58, 60, 62, 65, 69, 70, 71, 73, 80	x_{56}	799
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 33, 34, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 69, 71, 74, 80	$x_{69} + 1$	799
1, 6, 7, 8, 10, 12, 14, 16, 19, 21, 25, 27, 30, 31, 33, 36, 38, 40, 41, 43, 45, 47, 49, 51, 52, 56, 58, 63, 67, 69, 71, 73, 77, 80	$x_{66} + 1$	799
1, 3, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 25, 27, 33, 34, 36, 38, 40, 43, 45, 47, 49, 53, 54, 56, 58, 62, 63, 67, 69, 71, 74, 80	x_{58}	799
1, 5, 6, 7, 8, 10, 12, 14, 15, 16, 19, 21, 23, 25, 27, 33, 36, 38, 40, 43, 45, 47, 49, 53, 54, 55, 56, 58, 62, 63, 67, 69, 71, 74, 80	x_{37}	799

Trivium-800

maxterm bits	superpoly	round
0, 5, 6, 7, 9, 11, 13, 17, 19, 20, 22, 24, 26, 30, 32, 33, 35, 37, 39, 42, 44, 46, 48, 52, 55, 57, 61, 62, 66, 68, 72, 74, 76, 79	x_{63}	800

References

1. Agnesse, A., Pedicini, M.: Cube attack in finite fields of higher order. In: Proceedings of 9th Australasian Information Security Conference, AISC 2011, pp. 9–14. ACS, Inc. (2011)
2. Agostini, E.: Bitlocker dictionary attack using GPUs. In: University of Cambridge Passwords 2015 Conference (2015). https://www.cl.cam.ac.uk/events/passwords2015/preproceedings.pdf
3. Ahmadian, Z., Rasoolzadeh, S., Salmasizadeh, M., Aref, M.R.: Automated dynamic cube attack on block ciphers: cryptanalysis of SIMON and KATAN. IACR Cryptology ePrint Archive **2015**, 40 (2015)
4. Aumasson, J.-P., Dinur, I., Meier, W., Shamir, A.: Cube testers and key recovery attacks on reduced-round MD6 and trivium. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 1–22. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03317-9_1
5. Baksi, A., Maitra, S., Sarkar, S.: New distinguishers for reduced round trivium and trivia-SC using cube testers. In: WCC2015-9th International Workshop on Coding and Cryptography 2015 (2015)
6. Bernstein, D.J.: Why haven't cube attacks broken anything? https://cr.yp.to/cubeattacks.html. Accessed 11 Nov 2016
7. Blum, M., Luby, M., Rubinfeld, R.: Self-testing/correcting with applications to numerical problems. In: ACM Symposium on Theory of Computing, pp. 73–83. ACM (1990)
8. De Cannière, C.: Trivium: a stream cipher construction inspired by block cipher design principles. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 171–186. Springer, Heidelberg (2006). doi:10.1007/11836810_13
9. De Canniere, C., Preneel, B.: Trivium-specifications. eSTREAM, ECRYPT stream cipher project, report 2005/030 (2005)
10. Dinur, I., Shamir, A.: Cube attacks on tweakable black box polynomials. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01001-9_16
11. Dinur, I., Shamir, A.: Breaking grain-128 with dynamic cube attacks. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 167–187. Springer, Heidelberg (2011). doi:10.1007/978-3-642-21702-9_10
12. Dinur, I., Shamir, A.: Applying cube attacks to stream ciphers in realistic scenarios. Cryptogr. Commun. **4**(3–4), 217–232 (2012)
13. Fan, X., Gong, G.: On the security of Hummingbird-2 against side channel cube attacks. In: Armknecht, F., Lucks, S. (eds.) WEWoRC 2011. LNCS, vol. 7242, pp. 18–29. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34159-5_2
14. Fouque, P.-A., Vannet, T.: Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 502–517. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43933-3_26
15. Kleinjung, T., Lenstra, A.K., Page, D., Smart, N.P.: Using the cloud to determine key strengths. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 17–39. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34931-7_3
16. Marks, M., Jantura, J., Niewiadomska-Szynkiewicz, E., Strzelczyk, P., Góźdź, K.: Heterogeneous GPU&CPU cluster for high performance computing in cryptography. Comput. Sci. **13**(2), 63–79 (2012)

17. Milo, F., Bernaschi, M., Bisson, M.: A fast, GPU based, dictionary attack to OpenPGP secret keyrings. J. Syst. Softw. **84**(12), 2088–2096 (2011)
18. O'Neil, S.: Algebraic structure defectoscopy (2007). Tools for Cryptanalysis 2007 Workshop. http://eprint.iacr.org/2007/378
19. Quedenfeld, F.M., Wolf, C.: Algebraic properties of the cube attack. IACR Cryptology ePrint Archive **2013**, 800 (2013)
20. Samorodnitsky, A.: Low-degree tests at large distances. In: Proceedings of 39th ACM symposium on Theory of Computing, pp. 506–515. ACM (2007)
21. Samorodnitsky, A., Trevisan, L.: A PCP characterization of NP with optimal amortized query complexity. In: Proceedings ACM Symposium on ToC, pp. 191–199. ACM (2000)
22. Shanmugam, D., Annadurai, S.: Secure implementation of stream cipher: trivium. In: Bica, I., Naccache, D., Simion, E. (eds.) SECITC 2015. LNCS, vol. 9522, pp. 253–266. Springer, Cham (2015). doi:10.1007/978-3-319-27179-8_18
23. Srinivasan, C., Pillai, U.U., Lakshmy, K., Sethumadhavan, M.: Cube attack on stream ciphers using a modified linearity test. J. Discret. Math. Sci. Cryptogr. **18**(3), 301–311 (2015)
24. Vielhaber, M.: Breaking ONE.FIVIUM by AIDA an algebraic IV differential attack (2007). http://eprint.iacr.org/2007/413
25. Winter, R., Salagean, A., Phan, R.C.-W.: Comparison of cube attacks over different vector spaces. In: Groth, J. (ed.) IMACC 2015. LNCS, vol. 9496, pp. 225–238. Springer, Cham (2015). doi:10.1007/978-3-319-27239-9_14
26. Zhang, S., Chen, G., Li, J.: Cube attack on reduced-round Quavium. ICMII-15 Adv. Comput. Sci. Res. (2015). doi:10.2991/icmii-15.2015.25

Related-Key Impossible-Differential Attack on Reduced-Round SKINNY

Ralph Ankele[1](✉), Subhadeep Banik[2], Avik Chakraborti[3], Eik List[4],
Florian Mendel[5], Siang Meng Sim[2], and Gaoli Wang[6]

[1] Royal Holloway University of London, Egham, UK
ralph.ankele.2015@rhul.ac.uk
[2] Nanyang Technological University, Singapore, Singapore
bsubhadeep@ntu.edu.sg, ssim011@e.ntu.edu.sg
[3] NTT Secure Platform Laboratories, Tokyo, Japan
chakraborti.avik@lab.ntt.co.jp
[4] Bauhaus-Universität Weimar, Weimar, Germany
eik.list@uni-weimar.de
[5] Graz University of Technology, Graz, Austria
florian.mendel@iaik.tugraz.at
[6] East China Normal University, Shanghai, China
glwang@sei.ecnu.edu.cn

Abstract. At CRYPTO'16, Beierle et al. presented SKINNY, a family of lightweight tweakable block ciphers intended to compete with the NSA designs SIMON and SPECK. SKINNY can be implemented efficiently in both soft- and hardware and supports block sizes of 64 and 128 bits as well as tweakey sizes of 64, 128, 192 and 128, 256, 384 bits respectively. This paper presents a related-tweakey impossible-differential attack on up to 23 (out of 36) rounds of SKINNY-64/128 for different tweak sizes. All our attacks can be trivially extended to SKINNY-128/128.

Keywords: Symmetric Cryptography · Cryptanalysis · Tweakable block cipher · Impossible differential · Lightweight cryptography

1 Introduction

SKINNY is a family of lightweight tweakable block ciphers recently proposed at CRYPTO 2016 by Beierle et al. [3]. Its goal was to design a cipher that could be implemented highly efficiently on both soft- and hardware platforms, with performance comparable or better than the SIMON and SPECK families of block ciphers [1]. Like the NSA designs SIMON and SPECK, SKINNY supports a wide range of block sizes and tweak/key sizes – however, in contrast to the And-RX and Add-RX based NSA proposals, SKINNY is based on the better understood Substitution-Permutation-Network approach.

SKINNY offers a large security margin within the number of rounds for each member of the SKINNY family. The designers show that the currently best known attacks approach close to half of the number of rounds of the

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 208–228, 2017.
DOI: 10.1007/978-3-319-61204-1_11

cipher. To motivate third-party cryptanalysis, the designers of SKINNY recently announced a cryptanalysis competition [2] for SKINNY-64/128 and SKINNY-128/128 with the obvious challenge of attacking more rounds than the preliminary analysis, concerning both the single- and related-key models.

Table 1. Summary of our attacks and comparison to existing cryptanalysis of SKINNY-64/128.

Instance	Rounds	Attack type	Time	Data	Memory	Ref
SKINNY-64/128	20	Impossible	$2^{121.1}$	$2^{47.7}$	$2^{74.7}$	[9]
SKINNY-64/128	21	Rectangle	$2^{87.9}$	$2^{54.0}$	$2^{54.0}$	[7]
SKINNY-64/128	21	Impossible	$2^{71.4}$	$2^{71.4}$ [b]	$2^{68.0}$	Sect. 3.1
SKINNY-64/128	22	Rectangle	$2^{109.9}$	$2^{63.0}$	$2^{63.0}$	[7]
SKINNY-64/128	22	Impossible[a]	$2^{71.6}$	$2^{71.4}$ [b]	$2^{64.0}$	Sect. 3.2
SKINNY-64/128	23	Impossible	$2^{124.2}$	$2^{62.5}$	2^{124}	[7]
SKINNY-64/128	23	Impossible[a]	2^{79}	$2^{71.4}$ [b]	$2^{64.0}$	Sect. 3.3

[a] The data complexity of our 21-round attack is beyond codebook. Our attack is more efficient than a full codebook attack in the case where SKINNY is used in a tweak-updating mode (i.e. where the tweak changes every time, but the key stays the same). This does not effect the 22/23 round attack as 48 bits of the tweakey are public (i.e. data complexity for full codebook would be 2^{64} from the state + 2^{48} from the tweak).

[b] Our attack on 22/23 rounds uses the tweak against the recommendation of the SKINNY designers but still conform to the specification in [3].

Related Work. Recently and independent of our analysis Liu *et al.* [7] analyzed SKINNY in the related-tweakey model, showing impossible-differential and rectangle attacks on 19, 23, and 27 rounds of SKINNY-n/n, SKINNY-n/2n and SKINNY-n/3n, respectively. In [9], Tolba *et al.* showed impossible-differential attacks for 18, 20, 22 rounds of SKINNY-n/n, SKINNY-n/2n and SKINNY-n/3n, respectively. Additionally, Sadeghi *et al.* [8] studied related-tweakey impossible- differential and zero-correlation linear characteristics. In comparison to the other attacks, our 23-round related-tweakey impossible-differential attack on SKINNY-64/128 has the lowest time complexity so far. Table 1 summarizes our attacks and compares them to existing attacks on SKINNY-64/128.

Contributions and Outline. In this paper, we propose an impossible-differential attack on SKINNY-64/128 reduced to 23 rounds in the related-key model. The attack uses an 11-round impossible differential trail, to which six and four rounds can be added for obtaining a 21-round attack. Later, we show that another round can be appended leading to a 22-round attack, and even a 23-round attack.

The paper is organized as follows. In Sect. 2, we give a brief introduction to the SKINNY family of block ciphers. In Sect. 3, we detail the attack on SKINNY and provide time and memory complexities. Finally, Sect. 4 concludes the paper.

2 Description of SKINNY

Each round of SKINNY consists of the operations SubCells, AddRoundConstants, AddRoundTweakey, ShiftRows, and MixColumns. The round operations are schematically illustrated in Fig. 1. A cell represents a 4-bit value in SKINNY-64/* and an 8-bit value in SKINNY-128/*.

We concentrate on SKINNY-64/128, which has a 64-bit block size and a 128-bit tweakey size. The data is arranged nibble-by-nibble in a row-wise fashion in a 4 × 4-matrix. SKINNY-64/128 recommends 36 rounds.

SubCells (SC) substitutes each nibble x by $S(x)$, which is given below.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	c	6	9	0	1	a	2	b	3	8	5	d	4	e	7	f

AddRoundConstants (AC) adds LFSR-based round constants to Cells 0, 4, and 8 of the state.

AddRoundTweakey (ART) adds the round tweakey to the first two state rows.

ShiftRows (SR) rotates the i^{th} row, for $0 \leq i \leq 3$, by i positions to the right.

MixColumns (MC) multiplies each column of the state by a matrix M:

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Tweakey Schedule. The tweakey schedule of SKINNY, as illustrated in Fig. 2, follows the TWEAKEY framework [5]. In contrast to the previous TWEAKEY designs Deoxys-BC and Joltik-BC, SKINNY employs a significantly more lightweight strategy. In each round, only the two topmost rows of each tweakey word are extracted and XORed to the state. An additional round-dependent constant is also XORed to the state to prevent attacks from symmetry.

The 128-bit tweakey is arranged in two 64-bit tweakey words, represented by TK_1 and TK_2. In each round, the tweakey words are updated by a cell permutation P_T that ensures that the two bottom rows of a tweakey word in a

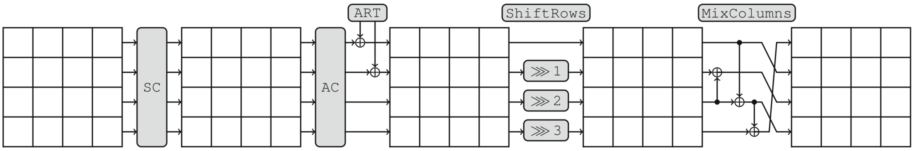


Fig. 1. Round function of SKINNY.

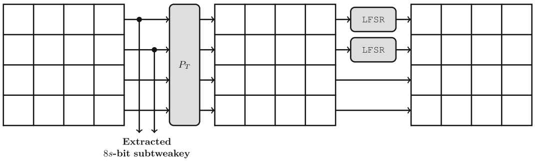


Fig. 2. Tweakey schedule of SKINNY.

certain round are exchanged with the two top rows in the tweakey word in the subsequent round. The permutation is given as:

$$P_T = \{9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7\}$$

The permutation P_T has a period of 16, as visualized in Fig. 7 in the appendix. Moreover, each individual cell in the two topmost rows of TK_2 is transformed by a 4-bit LFSR to minimise the cancellation of differences from TK_1 and TK_2; TK_1 employs no LFSR transformation. The LFSR transformation L is given by

$$L(x_3, x_2, x_1, x_0) := (x_2, x_1, x_0, x_3 \oplus x_2),$$

where x_3, x_2, x_1, x_0 represent the individual bits of every tweakey nibble.

3 Related-Key Impossible-Differential Attack

Impossible-differential attacks were introduced independently by Biham *et al.* [4] and Knudsen [6]. They are widely used as an important cryptanalytic technique. The attack starts with finding an input difference that can never result in an output difference. By adding rounds before and/or after the impossible differential, one can collect pairs with certain plaintext and ciphertext differences. If there exists a pair that meets the input and output values of the impossible differential under some subkey, these subkeys must be wrong. In this way, we filter as many wrong keys as possible and exhaustively search the rest of the keys.

Notations. Let us state a few notations that are used in the attack description:

K^r represents the r^{th} round key. This is equal to $TK_1^r \oplus TK_2^r$. Similarly, $k^r[i] = tk_1^r[i] \oplus tk_2^r[i]$ represents the individual i^{th} tweakey nibble in round r.
A^r represents the internal state before SC in round r.
B^r represents the internal state after SC in round r.
C^r represents the internal state after AT in round r.
D^r represents the internal state after SR in round r.
E^r represents the internal state after MC in round r. Furthermore, $E^r = A^{r+1}$.
L^t represents the t-times composition of LFSR function L.
$\overline{X}$ represents the corresponding variable X in the related-key setting.
$X[i]$ represents the i^{th} nibble of the corresponding variable X.

Impossible-Differential Trail. Figure 3 presents the 11-round related-key differential trail that we use. We introduce a nibble difference in Cell 8 of the combined tweakey. Since the initial difference is in Cell 8, *i.e.* in one of the bottom two rows in the tweakey, it does not affect the state in the first round, and will be added to the state from the second round onwards. Similarly in the backward trail, the difference in the 11^{th} round-tweakey appears in Cell 11 (in a bottom row), due to which we get an extra round in the backward direction.

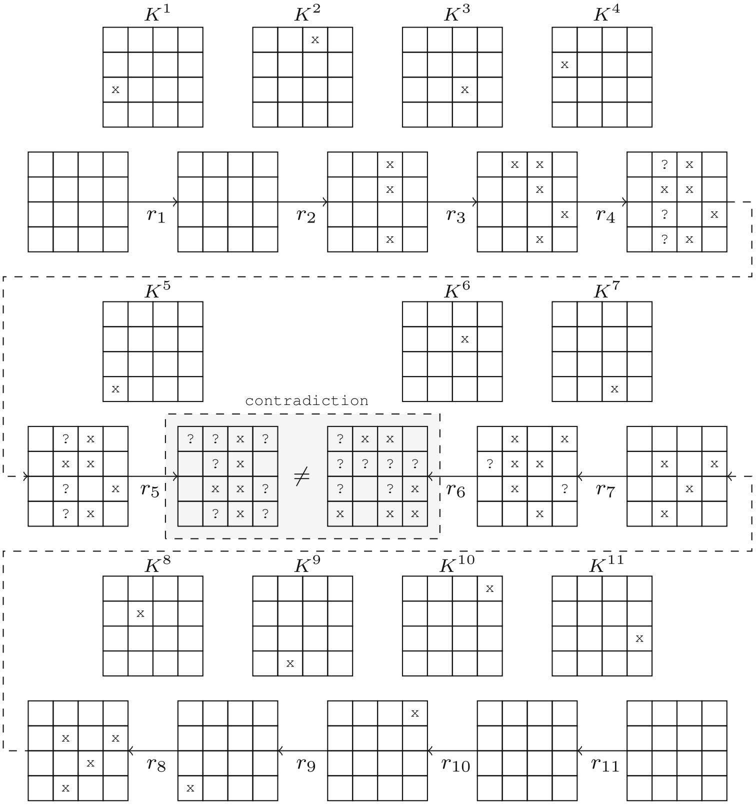


Fig. 3. Related-key impossible-differential trail over 11 rounds of SKINNY-64/128.

Lemma 1. *The equation $S(x \oplus \Delta_i) \oplus S(x) = \Delta_o$ has one solution x on average for $\Delta_i, \Delta_o \neq 0$. Similar result holds for the inverse S-Box S^{-1}.*

Proof. The above fact can be deduced by analyzing the Differential-Distribution Table (DDT) of the S-box S as illustrated in Table 2 in the appendix. The average can be calculated as $\frac{1}{225} \cdot \sum_{\Delta_i, \Delta_o \neq 0} DDT(\Delta_i, \Delta_o) \approx 1$. A similar exercise can be done for the inverse S-box yielding the same result.

Lemma 2. *For random values of x and $\varDelta_i, \varDelta_o \neq 0$, the equation $S(x \oplus \varDelta_i) \oplus S(x) = \varDelta_o$ holds with probability around 2^{-4}.*

Proof. The above fact can also be deduced by analyzing the Differential-Distribution Table (DDT) of the S-box S as illustrated in Table 2 in the appendix. The probability can be calculated as (let $\mathsf{Pr}[(x, \delta_i, \delta_o)$ denote the probability that the equation is satisfied for the triplet x, δ_i, δ_o)

$$\begin{aligned}\mathsf{Pr}[(x, \varDelta_i, \varDelta_o)] &= \sum_{\delta_i, \delta_o \neq 0} \mathsf{Pr}[(x, \delta_i, \delta_o) | \varDelta_i = \delta_i, \varDelta_o = \delta_o] \mathsf{Pr}[\varDelta_i = \delta_i, \varDelta_o = \delta_o] \\ &= \frac{1}{225} \cdot \sum_{\varDelta_i, \varDelta_o \neq 0} DDT(\varDelta_i, \varDelta_o) \cdot 2^{-4} \approx 2^{-4}\end{aligned}$$

Attack on 21 Rounds. The impossible differential trail described in Fig. 3 can be extended by six and four rounds in backward and forward direction as will be explained in the following two lemmas.

Lemma 3. *It is possible to find plaintext pairs $P, \overline{P}$ and related-tweakey pairs $K, \overline{K}$ such that if the tweakey pairs differ only in nibble position* 11*, then there is no difference in the internal state after executing six rounds of* SKINNY*-64/128 with the plaintext-tweakey pairs (P, K) and $(\overline{P}, \overline{K})$.*

Proof. We will show how the required plaintext and tweakey pairs are generated. We choose the nibble at Position 11 to introduce the initial difference because after completing six rounds, the difference is shuffled to Cell 8 of the round key, which coincides with the beginning of the impossible- differential trail, shown in Fig. 3. It can be seen that the AddRoundTweakey in the first round can be pushed behind the MixColumns operation by changing the first round key to $\mathsf{Lin}(K_1)$ where $\mathsf{Lin} = \mathrm{MC} \circ \mathrm{SR}$ represents the linear layer (refer to Fig. 4).

$$\mathsf{Lin}(K^1) = \begin{bmatrix} k^1[0] & k^1[1] & k^1[2] & k^1[3] \\ k^1[0] & k^1[1] & k^1[2] & k^1[3] \\ k^1[7] & k^1[4] & k^1[5] & k^1[6] \\ k^1[0] & k^1[1] & k^1[2] & k^1[3] \end{bmatrix}$$

Furthermore, the initial difference between $K = TK_1^1 \oplus TK_2^1$ and $\overline{K} = \overline{TK_1^1} \oplus \overline{TK_2^1}$ can be selected in a specific form, so that in Round 6, the tweakey difference is zero. Let us denote $\delta_1 = tk_1^1[11] \oplus \overline{tk_1^1}[11]$ and $\delta_2 = tk_2^1[11] \oplus \overline{tk_2^1}[11]$. In Round 6, the difference will appear in Cell 0 of the round key and so we want:

$$\begin{aligned}k^6[0] \oplus \overline{k^6}[0] &= tk_1^6[0] \oplus \overline{tk_1^6}[0] + tk_2^6[0] \oplus \overline{tk_2^6}[0] \\ &= tk_1^1[11] \oplus \overline{tk_1^1}[11] \oplus L^3 \left(tk_2^1[11]\right) \oplus L^3 \left(\overline{tk_2^1}[11]\right) \\ &= \delta_1 \oplus L^3 \left(\delta_2\right) = 0\end{aligned}$$

So, if the attacker chooses δ_1, δ_2 satisfying the equation $\delta_1 \oplus L^3(\delta_2) = 0$, then there is no difference introduced via the round-key addition in Round 6. The attacker should therefore follow the steps:

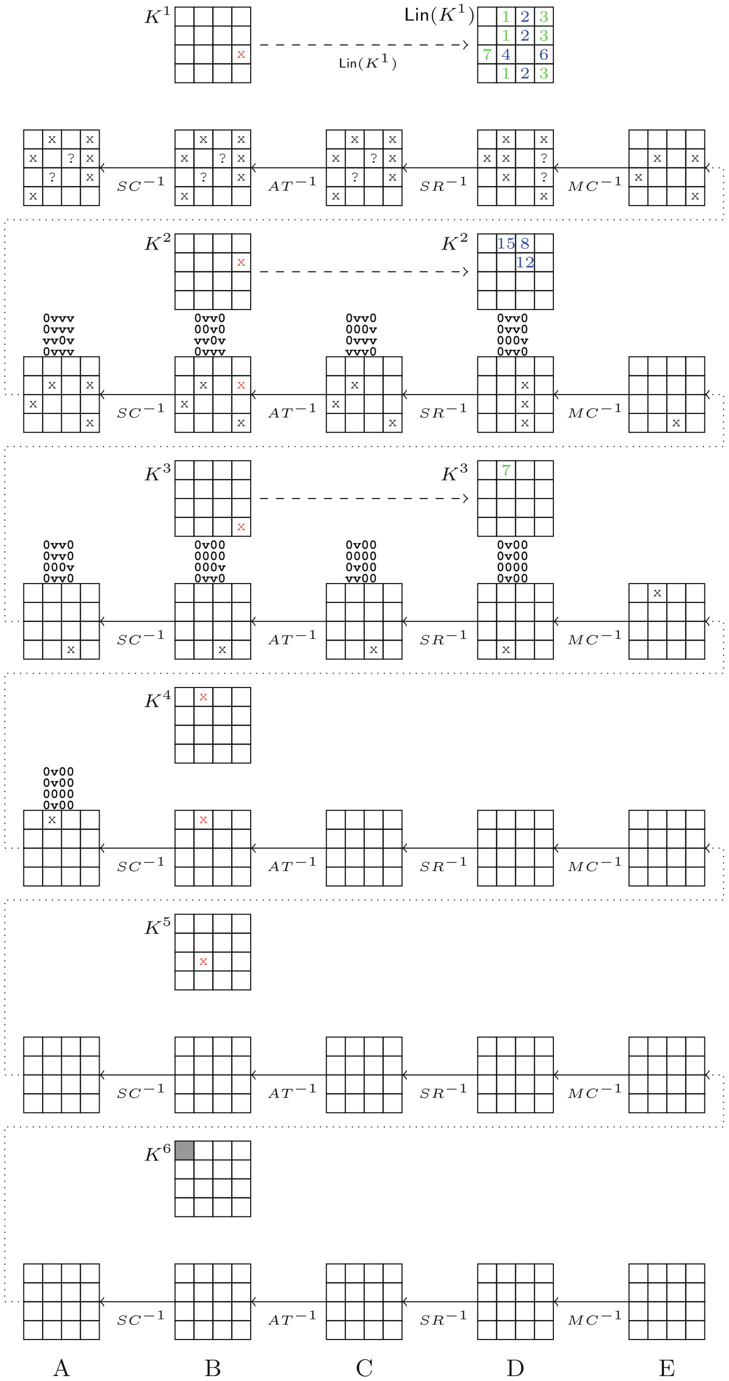


Fig. 4. Trail for the six forward rounds (the values of active nibbles in red are functions of δ_1, δ_2, the dark gray cell visualises the tweakey cancelation). (Color figure online)

1. Take any Plaintext P and compute the state after the first round MixColumns, *i.e.* E^1.
2. Take any three-nibble difference $\Delta_1, \Delta_3, \Delta_4$ to construct $\overline{E^1}$ such that

$$E^1 \oplus \overline{E^1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \Delta_1 & 0 & \Delta_2 \\ \Delta_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Delta_4 \end{bmatrix}$$

   The value of Δ_2 will be determined shortly. The attacker can recover $\overline{P}$ by inverting the MC, SR, AC and SC layers on $\overline{E^1}$.
3. The attacker chooses the difference α in Cell 14 of E^2. She calculates then $k^1[1]$, $k^1[3]$, $k^1[7]$ so that

$$B^2 \oplus \overline{B^2} = \mathsf{Lin}^{-1}(E^2) \oplus \mathsf{Lin}^{-1}(\overline{E^2}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & \beta \\ \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha \end{bmatrix}.$$

   For example, $k^1[1]$ is a solution of the equation:

$$S\left(E^1[5] \oplus k^1[1]\right) \oplus S\left(E^1[5] \oplus \Delta_1 \oplus k^1[1]\right) = \alpha.$$

   Lemma 1 ensures that the equation above has one solution on average.
4. β needs to be equal to $k^2[7] \oplus \overline{k^2}[7] = tk_1^2[7] \oplus tk_2^2[7] \oplus \overline{tk_1^2}[7] \oplus \overline{tk_2^2}[7]$. This is equal to $tk_1^1[11] \oplus L(tk_2^1[11]) \oplus \overline{tk_1^1}[11] \oplus L(\overline{tk_2^1}[11]) = \delta_1 \oplus L(\delta_2)$. So, the attacker chooses δ_1 and δ_2 satisfying $\delta_1 \oplus L^3(\delta_2) = 0$ and calculates $\beta = \delta_1 \oplus L(\delta_2)$. Δ_2 can then be determined as a solution of the equation:

$$S\left(E^1[7] \oplus k^1[3]\right) \oplus S\left(E^1[7] \oplus \Delta_2 \oplus k^1[3]\right) = \beta \tag{1}$$

   The attacker now has the values of Δ_1, Δ_2, Δ_3, Δ_4 and so, he can compute $E^1, \overline{E^1}$ and hence $P, \overline{P}$.
5. However, the attacker still needs that in Round 4, the active nibble in $B^4[1]$ is equal to $\delta_1 \oplus L^2(\delta_2)$ to make all the state cells inactive in C^4, D^4, and E^4.
6. The attacker needs to guess three roundkey values in Round 1 (*i.e.* $k^1[2]$, $k^1[4]$, $k^1[6]$) and three roundkey values in Round 2 (*i.e.* $k^2[1] = tk_1^1[15] \oplus L(tk_2^1[15])$, $k^2[2] = tk_1^1[8] \oplus L(tk_2^1[8])$, $k^2[6] = tk_1^1[12] \oplus L(tk_2^1[12])$).
   If the attacker can guess these values, then he knows the actual values (marked with v) of the state cells for the plaintext pair $P, \overline{P}$ as opposed to only differences (marked by 0) in both Figs. 4 and 5.
7. Guessing the tweakey nibbles mentioned above enables the attacker to calculate the value of $B^3[1]$. Then, she calculates $k^3[1] = tk_1^1[7] \oplus L(tk_2^1[7])$ as follows. Since $D^3[1] = B^3[1] \oplus k^3[1]$ holds, we have:

$$S\left(D^3[1] \oplus D^3[9] \oplus D^3[13]\right) \oplus S\left(D^3[1] \oplus D^3[9] \oplus \overline{D^3}[13]\right) = \delta_1 \oplus L^2(\delta_2).$$

Since the knowledge of the guessed key nibbles already allows the attacker to calculate $D^3[9]$, $D^3[13]$, and $\overline{D^3}[13]$, $k^3[1] = tk_1^1[7] \oplus L(tk_2^1[7])$ is the solution to the equation above. Again, Lemma 1 guarantees one solution on average. Since the attacker has already determined $k^1[7] = tk_1^1[7] \oplus tk_2^1[7]$, this also determines the values of $tk_1^1[7]$ and $tk_2^1[7]$.

8. This guarantees that there are no more active nibbles after Round 4. The key difference does not add to the state in Round 5, and due to the fact that $\delta_1 \oplus L^3(\delta_2) = 0$, the tweak difference becomes 0 in Round 6.

Thus, by guessing six and calculating three key nibbles, we can construct $P, \overline{P}$ and $K, \overline{K}$ so that the internal state after six rounds has no active nibbles.

Lemma 4. *Given $C, \overline{C}$ as the two output ciphertexts after querying plaintext-tweakey pairs (P, K) and $(\overline{P}, \overline{K})$ to a 21-round* SKINNY*-64/128 encryption oracle. Then for a fraction 2^{-40} of the ciphertext pairs, it is possible to construct a backward trail for round 21 to round 18 by guessing intermediate tweakey nibbles so that there are no active nibbles in the internal state at the end of round 17.*

Proof. The attacker starts working backward from the ciphertext pairs $C, \overline{C}$ and proceeds as follows (illustrated in Fig. 5):

1. The attacker rejects ciphertext pairs which do not have seven inactive cells in Cells 3, 4, 5, 8, 9, 11, and 14) after peeling off the final MixColumns layer (*i.e.* D^{21}). Thus, a fraction of 2^{-28} pairs are filtered after this stage.
2. Furthermore, the attacker rejects ciphertext pairs which do not have the difference $\delta_1 \oplus L^{10}(\delta_2)$ in Cell 13 of A^{21}, *i.e.* reject if $A^{21}[13] \oplus \overline{A^{21}}[13] \neq \delta_1 \oplus L^{10}(\delta_2)$. Since calculating this cell does not require any key guess, she can do this filtering instantly leaving a fraction of 2^{-4} pairs after this stage.
3. Since the two bottommost rows of the state are not affected by the tweakey addition, and since $tk_1^1[7], tk_2^1[7]$ are already known, the attacker can calculate the actual values in Cells 0, 8, and 12 in A^{21} for the ciphertext pairs. These have to be equal since they are the output of the 20^{th}-round MixColumns operation on the leftmost column which had only one active nibble in its input. If the active Cells 8 and 12 are different, the attacker can reject the pair. This adds another filter with probability 2^{-4}.
4. Since the actual values in Cell 0 in A^{21} for the ciphertext pairs were already calculated in the previous step, the attacker checks if the value of the active Cell 0 is equal to that of Cells 8 and 12, and rejects the pair otherwise. This adds another filter of probability 2^{-4}.
5. The attacker determines $k^{21}[5] = tk_1^1[4] \oplus L^{10}(tk_2^1[4])$ so that the active nibble in Cell 5 of A^{21} is $\delta_1 \oplus L^{10}(\delta_2)$. Since $A^{21}[5] = S^{-1}\left(k^{21}[5] \oplus C^{21}[5]\right)$, $k^{21}[5]$ is a solution to the equation below:

$$S^{-1}\left(k^{21}[5] \oplus C^{21}[5]\right) \oplus S^{-1}\left(k^{21}[5] \oplus \overline{C^{21}}[5]\right) = \delta_1 \oplus L^{10}(\delta_2).$$

6. The attacker determines $k^{21}[2] = tk_1^1[1] \oplus L^{10}(tk_2^1[1])$ and $k^{21}[6] = tk_1^1[2] \oplus L^{10}(tk_2^1[2])$ so that the active nibble in Cell 2 and 6 of A^{21} are equal to the

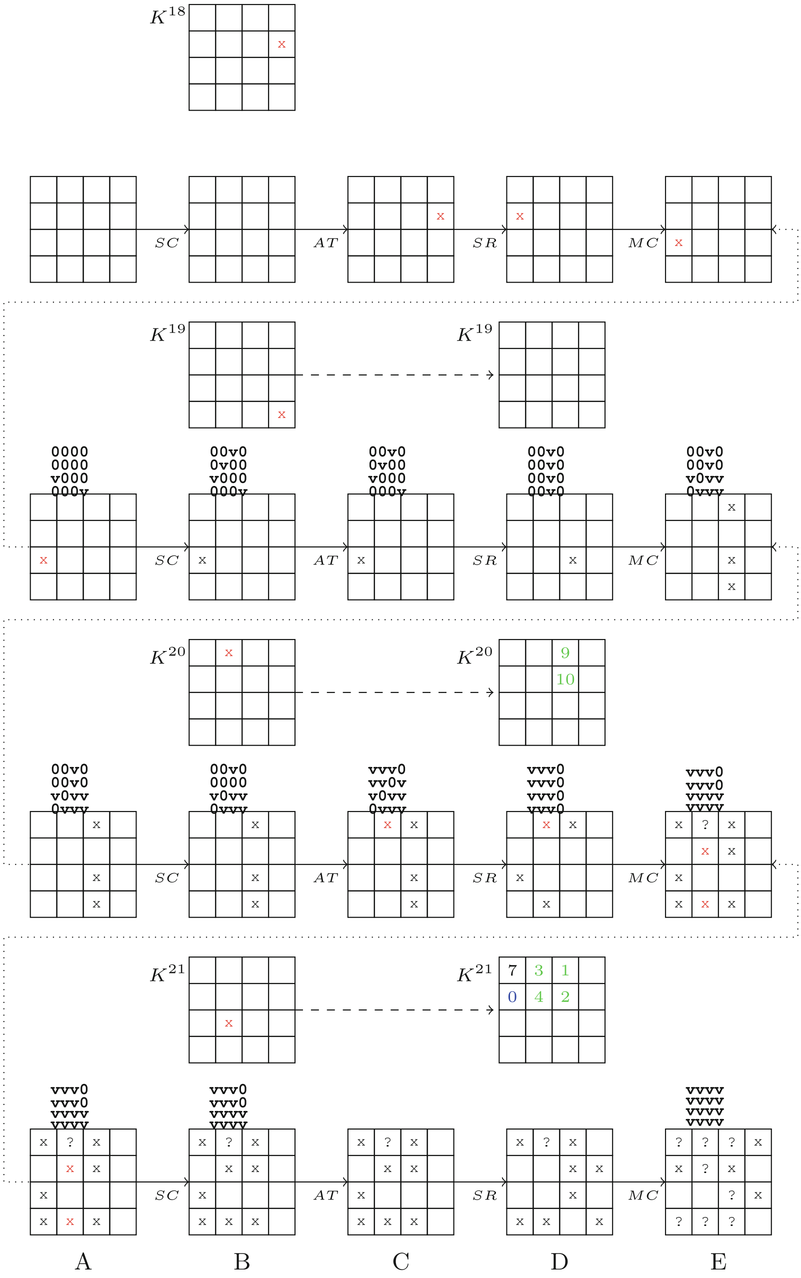


Fig. 5. Trail for the four backward rounds (the values of active nibbles in red are functions of δ_1 and δ_2). (Color figure online)

active nibble in Cell 14. Again, this works since those cells are output of the 20^{th}-round MixColumns operation on Column 2 which had only one active nibble in its input.

7. Additionally, the attacker guesses $k^{21}[4] = tk_1^1[0] \oplus L^{10}(tk_2^1[0])$. This enables the attacker to compute the actual values for the entire leftmost column of A^{21} and hence to compute the leftmost column of D^{20}.
8. The value of the active nibble in cell 10 of A^{20} is given as:

$$\begin{aligned} A^{20}[10] \oplus \overline{A^{20}}[10] &= S^{-1}\left(B^{20}[10]\right) \oplus S^{-1}\left(\overline{B^{20}}[10]\right) \\ &= S^{-1}\left(D^{20}[8]\right) \oplus S^{-1}\left(\overline{D^{20}}[8]\right) = \eta. \end{aligned} \tag{2}$$

Since the leftmost column of D^{20} is known, the attacker can calculate η, which must be equal to Cell 14 of A^{20} since they are output of the 19^{th}-round MixColumns operation with one active input nibble.

$$\begin{aligned} A^{20}[14] \oplus \overline{A^{20}}[14] &= S^{-1}\left(D^{20}[13]\right) \oplus S^{-1}\left(\overline{D^{20}}[13]\right) \\ &= S^{-1}\left(A^{21}[1] \oplus A^{21}[13]\right) \oplus S^{-1}\left(\overline{A^{21}}[1] \oplus \overline{A^{21}}[13]\right). \end{aligned} \tag{3}$$

It holds that $A^{21}[1] = S^{-1}\left(C^{21}[1] \oplus k^{21}[1]\right)$ and $\overline{A^{21}}[1] = S^{-1}(\overline{C^{21}}[1] \oplus k^{21}[1])$. By calculating Eqs. (2) and (3), the attacker can solve for $k^{21}[1] = tk_1^1[3] \oplus L^{10}(tk_2^1[3])$. One solution on average is guaranteed by Lemma 1.
9. The values $tk_1^1[i] \oplus tk_2^1[i]$, for $i = 1, 2, 3, 4$, were already determined during the calculation of the forward trail. So, using their values, the attacker can determine the actual values $tk_1^1[i]$, $tk_2^1[i]$ for $i = 1, 2, 3, 4$.
10. The attacker calculates $k^{20}[2] = tk_1^1[9] \oplus L^{10}(tk_2^1[9])$ so that the active nibble in Cell 2 in A^{20} is equal to the active value η in Cells 10 and 14 since they are output of the 19^{th}-round MixColumns operation with one active input nibble. This is done by solving

$$\eta = A^{20}[2] \oplus \overline{A^{20}}[2] = S^{-1}\left(C^{20}[2] \oplus k^{20}[2]\right) \oplus S^{-1}\left(\overline{C^{20}}[2] \oplus k^{20}[2]\right). \tag{4}$$

11. The final condition to be satisfied is that the active nibble in Cell 8 of A^{19} has to be equal to $\delta_1 \oplus L^9(\delta_2) = \gamma$.

$$\begin{aligned} \gamma &= S^{-1}\left(D^{19}[10]\right) \oplus S^{-1}\left(\overline{D^{19}}[10]\right) \\ &= S^{-1}\left(A^{20}[6] \oplus A^{20}[14]\right) \oplus S^{-1}\left(\overline{A^{20}}[6] \oplus \overline{A^{20}}[14]\right). \end{aligned} \tag{5}$$

Note that $A^{20}[6] = S^{-1}(C^{20}[6] \oplus k^{20}[6])$. And since $\overline{A^{20}}[6] = A^{20}[6]$, solving Eq. (5) helps to determine $k^{20}[6] = tk_1^1[10] \oplus L^{10}(tk_2^1[10])$.

The result follows since in the Steps 1–4, a total of $2^{-28-4-4-4} = 2^{-40}$ ciphertext pairs are filtered.

3.1 First Attack

Now, we put together the findings of Lemmas 3 and 4 into an attack procedure (see Fig. 8 in the appendix for details):

1. The attacker chooses the nibble values of the random base variable E^1 in all locations except Cells 5, 7, 8, and 15.
2. She chooses fixed differences δ_1, δ_2 satisfying $\delta_1 = L^3(\delta_2)$.
3. For each choice of $(E^1[5], E^1[7], E^1[8], E^1[15])$ (2^{16} choices):
   - Calculate P by inverting the first round.
   - Query the 21-round encryption oracle for P, K and $P, \overline{K}$.

So, for every choice of the base variable E^1, we have 2^{17} encryption calls. We can pair related plaintext and tweakey pairs in the following way: For every plaintext P_i, choose a plaintext P_j so that E^1 for P_i and P_j have a non-zero difference in all Cells 5, 7, 8, and 15. For every P_i, there exist $(2^4 - 1)^4 \approx 2^{15.6}$ such values of P_j, and so $2^{16+15.6} = 2^{31.6}$ pairs to work with. The attack now proceeds as follows. For each choice of P_i, P_j ($2^{31.6}$ choices):

- Denote $P = P_i$ and $\overline{P} = P_j$.
- The attacker can choose α and proceed with the steps of the above attack with one exception: She can no longer choose Δ_2 as in Step 4 of Lemma 3 since she has already chosen $P, \overline{P}, K, \overline{K}$.
- With probability 2^{-4} (as per Lemma 2), the plaintext pair satisfies Eq. (1) in Step 4 of Lemma 3 and proceeds; otherwise, she aborts.
- Request the ciphertext $\overline{C}$ for $(\overline{P}, \overline{K})$ and the ciphertext C for (P, K).
- If $C \oplus \overline{C}$ does not pass the 2^{-36} filter (Steps 1, 2, and 3 in Lemma 4), then abort and start again.
- If they pass the filter, the attacker can guess seven tweakey cells (2^{28} guesses) and calculate 17 key/tweak cells as follows:

#	Guessed	Rnd	Calculated	Rnd
1	$tk_1^1[i] \oplus tk_2^1[i]$ for $i = 2, 4, 6$	1		
2	$tk_1^1[i] \oplus L(tk_2^1[i])$ for $i = 8, 12, 15$	2		
3	$tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 0$	21		
4			$tk_1^1[i],\ tk_2^1[i]$ for $i = 7$	3
5			$tk_1^1[i],\ tk_2^1[i]$ for $i = 1, 2, 3, 4$	21
6			$tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 9, 10$	20

  The 17 tweakey nibbles used for elimination are therefore:

  (a) $tk_1^1[i],\ tk_2^1[i]$ for $i = 1, 2, 3, 4, 7$

  (b) $tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 9, 10$

  (c) $tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 0$

  (d) $tk_1^1[i] \oplus L(tk_2^1[i])$ for $i = 8, 12, 15$

  (e) $tk_1^1[i] \oplus tk_2^1[i]$ for $i = 6$

- A fraction of 2^{-4} tweakeys fulfills the condition required in Step 4 of Lemma 4.
- Therefore, the attacker has a set of $2^{28-4} = 2^{24}$ wrong key candidates.

The above procedure is repeated with 2^x chosen plaintexts until a single key solution remains for the 17 nibbles of the tweakey.

Complexity. For every base value of E^1, the attacker makes 2^{17} encryption calls. Out of these, she has $2^{31.6}$ pairs to work with. For each pair, the attacker can then choose α in $2^4 - 1$ ways, which gives her around $2^{35.6}$ initial guesses for the forward key nibbles $k^1[1]$, $k^1[3]$, and $k^1[7]$, of which a fraction of 2^{-4} passes the filter in Eq. (1). So, she has $2^{31.6}$ pairs to work with. In fact, for every pair (P_i, P_j) there is only one choice of α going forward on average.

$$\text{Time complexity} = \max \left\{2^{x+17} \text{ encryptions}, 2^{x-4.4+24} \text{ guesses}\right\} = 2^{x+19.6}.$$

The attacker gets wrong solutions for $2^{x-4.4+24} = 2^{x+19.6}$ incorrect solutions for 17 nibbles. To reduce the keyspace to 1 we need:

$$2^{17\times4} \cdot \left(1 - 2^{-17\times4}\right)^{2^{x+19.6}} \approx 2^{17\times4} e^{-2^{x-48.4}} = 1.$$

For this, we need $x = 55$. So, the total number of encryption calls to 21-round SKINNY-64/128 is $2^{55+17} = 2^{72}$ and the total number of guesses is $2^{74.6}$. We also need 2^{68} memory accesses, which are negligible in the total complexity. The memory complexity is upper bounded by storing one bit per key candidate which is therefore 2^{68} bits. The memory for storing the approximately $2 \cdot 2^{17}$ plaintexts and corresponding ciphertexts of a structure at each time is negligible.

3.2 22-Round Attack Under Partially Known Tweak

The attack above can be extended to 22-round SKINNY-64/128 under the assumption that 48 of the 128 bits in the tweakey are publicly known tweak (see Fig. 9 in the appendix for details). In particular, we assume that $tk_1^1[i], tk_2^1[i]$ for $i = 8, 11, 12, 13, 14, 15$ are reserved for the tweak. The remaining 80 bit constitute the secret key.

In this case, the attacker can add a round at the end (see Fig. 6 for details). Knowing six out of eight cells in the lower half of the tweakey blocks helps in the following way. From the ciphertext (i.e. E^{22}), one can revert the final round to compute E^{21} if we guess $k^{22}[4, 5]$, i.e. $tk_1^1[9, 10] \oplus L^{11}(tk_2^1[9, 10])$. The attack is almost the same as the previous attack, except that the tweakey indices $i = 8, 11, 12, 13, 14, 15$ and their functions are known and need not be guessed.

1. Generate $2^{31.6}$ plaintext/ciphertext pairs from every base choice of E^1 and 2^{17} encryption calls.
2. For each choice of P_i, P_j ($2^{31.6}$ choices):
   - Denote $P = P_i$ and $\overline{P} = P_j$.
   - The attacker can choose α and calculate $k^1[1]$, $k^1[3]$, and $k^1[7]$ as per Step 3 of Lemma 3.
   - She can no longer choose Δ_2 as in Step 4 of Lemma 3 since she has already chosen P, $\overline{P}$, K, $\overline{K}$.
   - With probability 2^{-4}, the plaintext pair satisfies Eq. (1) in Step 4 of Lemma 3 and proceeds; otherwise, she aborts.

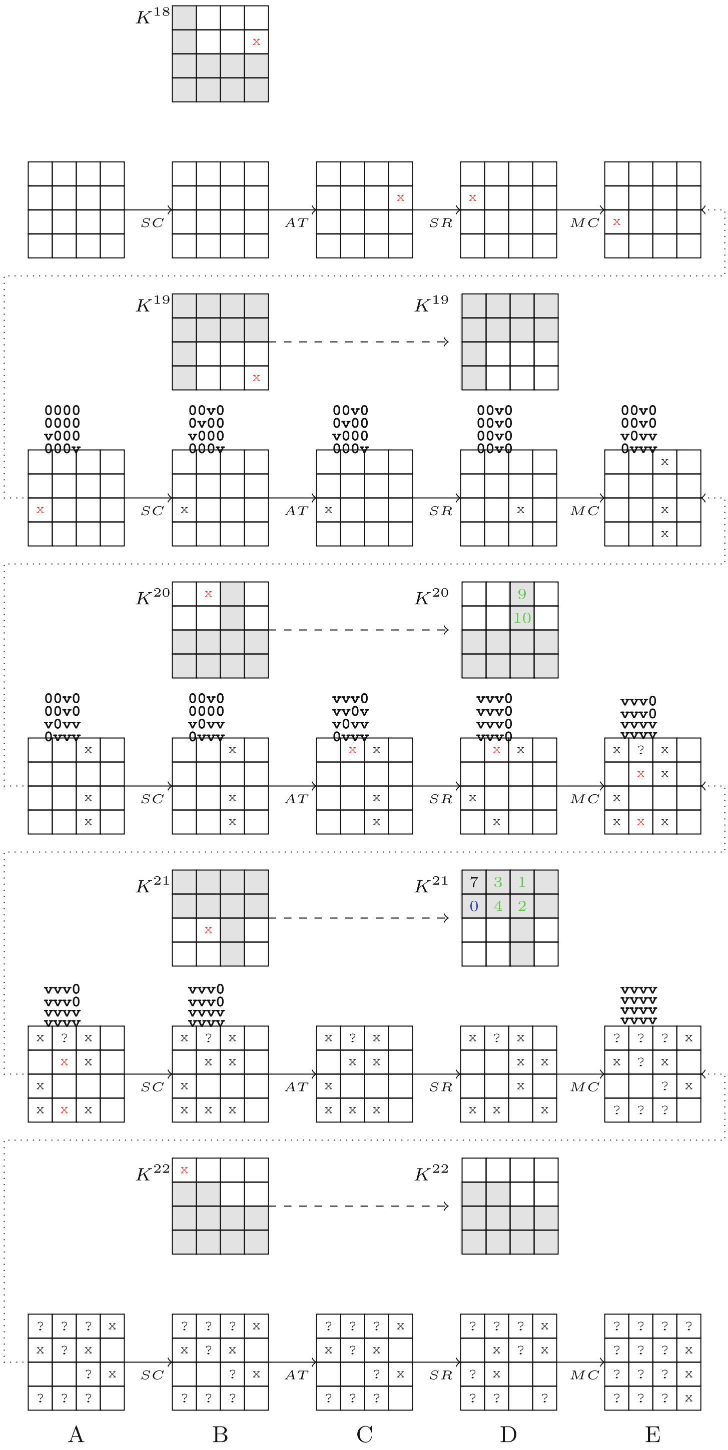


Fig. 6. Trail for the five backward rounds (the values of active nibbles in red are functions of δ_1, δ_2, grey cells are the key, white cells are the tweak). (Color figure online)

- The attacker doesn't need to guess the Round 2 tweakey nibbles since these are in the lower half of the tweakey blocks and therefore known.
- Retrieve the ciphertext $\overline{C}$ for $(\overline{P}, \overline{K})$ and the ciphertext C for (P, K).
- Guess $k^{22}[4, 5] = tk_1^1[9, 10] \oplus L^{11}(tk_2^1[9, 10])$ to get E_{21}.
- If $E_{21} \oplus \overline{E_{21}}$ does not pass the 2^{-36} filter, then abort and restart.
- After determining $k^{20}[2] = tk_1^1[9] \oplus L^{10}(tk_2^1[9])$ and $k^{20}[6] = tk_1^1[10] \oplus L^{10}(tk_2^1[10])$ in Steps 10 and 11 of Lemma 4, the attacker can uniquely determine $tk_1^1[9, 10]$ since $tk_1^1[9, 10] \oplus L^{11}(tk_2^1[9, 10])$ is already guessed.
- If they pass the filter, the attacker can guess six tweakey cells (2^{24} guesses) and calculate 16 key cells as follows:

#	Guessed	Rnd	Calculated	Rnd
1	$tk_1^1[i] \oplus tk_2^1[i]$ for $i = 2, 4, 6$	1		
2	$tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 0$	21		
3	$tk_1^1[i] \oplus L^{11}(tk_2^1[i])$ for $i = 9, 10$	22		
4			$tk_1^1[i], tk_2^1[i]$ for $i = 7$	3
5			$tk_1^1[i], tk_2^1[i]$ for $i = 1, 2, 3, 4$	21
6			$tk_1^1[i], tk_2^1[i]$ for $i = 9, 10$	20

  The 16 tweakey nibbles used for elimination are therefore:

  (a) $tk_1^1[i], tk_2^1[i]$ for $i = 1, 2, 3, 4, 7, 9, 10$. (c) $tk_1^1[i] \oplus tk_2^1[i]$ for $i = 6$.

  (b) $tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 0$.
- A fraction of 2^{-4} tweakeys fulfills the condition in Step 4 of Lemma 4.
- Therefore, the attacker has a set of $2^{24-4} = 2^{20}$ wrong key candidates.

The procedure above is repeated with 2^x chosen plaintexts until a single key solution remains for the 16 nibbles of the tweakey.

Complexity. For every base value of E^1, the attacker makes 2^{17} encryption calls. Out of these, she has $2^{31.6}$ pairs to work with. For each pair, the attacker can choose then α in $2^4 - 1$ ways, which gives her around $2^{35.6}$ initial guesses for the forward key nibbles $k^1[1], k^1[3], k^1[7]$, of which only a fraction of 2^{-4} passes the filter in Eq. (1). So, the attacker has $2^{31.6}$ pairs to work with. In effect, for every pair (P_i, P_j) there is only once choice of α going forward on average.

$$\text{Time complexity} = \max\left\{2^{x+17} \text{ encryptions}, 2^{x-4.4+20} \text{ guesses}\right\} = 2^{x+17}.$$

The attacker gets wrong solutions for $2^{x-4.4+20} = 2^{x+15.6}$ incorrect solutions for 16 nibbles. To reduce the keyspace to 1 we need:

$$2^{16\times4} \cdot \left(1 - 2^{-16\times4}\right)^{2^{x+15.6}} \approx 2^{16\times4} e^{-2^{x-48.4}} = 1.$$

For this, we need $x = 54$. So, the total number of encryption calls to 22-round SKINNY-64/128 is $2^{54+17} = 2^{71}$. We also need 2^{64} memory accesses, which are negligible in the total complexity. The memory complexity is upper bounded by storing one bit per key candidate which is therefore 2^{64} bits. The memory for storing the approximately $2 \cdot 2^{17}$ plaintexts and corresponding ciphertexts of a structure at each time is negligible.

3.3 23-Round Attack Under Partially Known Tweak

We can extend the 22 round attack to a 23 round attack by prepending one round at the beginning. In order to not disturb the notation, we denote the additonal round prepended at the beginning as the 0-th round. That is, the 23 rounds are labelled as rounds 0 to 22, and the variables A^0, B^0 etc. are defined as above. The plaintext is denoted by A^0 and the ciphertext by E^{22}. Note that, from the base value of E^1, the plaintext can be calculated if we guess $k^0[9, 10]$.

There are two principal differences to the 22-round attack.

1. When the attacker guesses $k^{22}[4, 5]$ which is $tk_1^1[9, 10] \oplus L^{11}(tk_2^1[9, 10])$ to invert the final round to get E_{21}, he uniquely determines $tk_1^1[9, 10]$ and $tk_2^1[9, 10]$. This is because at the beginning of the outer loop $k^0[9, 10]$ has already been guessed by the attacker to invert the initial round.
2. As the attacker can no longer determine $k^{20}[2] = tk_1^1[9] \oplus L^{10}(tk_2^1[9])$ and $k^{20}[6] = tk_1^1[10] \oplus L^{10}(tk_2^1[10])$ using Steps 10 and 11 of Lemma 4. The probability that with the given values of $tk_1^1[9, 10]$ and $tk_2^1[9, 10]$, Eqs. (4) and (5) are satisfied is 2^{-8}. This decreases the probability of ciphertext filter from 2^{-36} to 2^{-44}.

For each initial guess of $k^0[9, 10]$, the guessed and calculated key bytes are:

#	Guessed	Rnd	Calculated	Rnd
1	$tk_1^1[i] \oplus tk_2^1[i]$ for $i = 2, 4, 6$	1		
2	$tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 0$	21		
3	$tk_1^1[i] \oplus L^{11}(tk_2^1[i])$ for $i = 9, 10$	22		
4			$tk_1^1[i], tk_2^1[i]$ for $i = 7$	3
5			$tk_1^1[i], tk_2^1[i]$ for $i = 1, 2, 3, 4$	21

The 14 tweakey nibbles used for elimination are therefore:

(a) $tk_1^1[i], tk_2^1[i]$ for $i = 1, 2, 3, 4, 7$.

(b) $tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 0$.

(c) $tk_1^1[i] \oplus tk_2^1[i]$ for $i = 6$.

(d) $tk_1^1[i] \oplus L^{11}(tk_2^1[i])$ for $i = 9, 10$

As before, a fraction of 2^{-4} tweakeys fulfills the condition in Step 4 of Lemma 4. Therefore, the attacker has a set of $2^{24-4} = 2^{20}$ wrong key candidates.

Complexity. For each iteration of the outer loop, the complexity is calculated as follows: For every base value of E^1, the attacker makes 2^{17} encryption calls. Out of those, she has $2^{31.6}$ pairs to work with. For each pair, the attacker can choose then α in $2^4 - 1$ ways, which gives her around $2^{35.6}$ initial guesses for the forward key nibbles $k^1[1], k^1[3], k^1[7]$, of which only a fraction of 2^{-4} passes the filter in Eq. (1). In effect, for every pair (P_i, P_j) there is only one choice of α going forward on average.

$$\text{Time complexity} = \max\left\{2^{x+17} \text{ encryptions}, 2^{x+31.6-44+20} \text{ guesses}\right\} = 2^{x+17}.$$

The attacker gets $2^{x+31.6-44+20} = 2^{x+7.6}$ incorrect solutions for 14 nibbles. To reduce the keyspace to 1 we need:

$$2^{14\times 4} \cdot \left(1 - 2^{-14\times 4}\right)^{2^{x+7.6}} \approx 2^{14\times 4} e^{-2^{x-48.4}} = 1.$$

We need $x = 54$ leaving the total number of encryption calls to 22-round SKINNY-64/128 with $2^{54+17} = 2^{71}$. Multiplying this by 2^8 for the outer loop gives a total complexity of $2^{71+8} = 2^{79}$ which is just short of exhaustive search for the 80-bit key. We also need $2^{56+8} = 2^{64}$ memory accesses, which are negligible in the total complexity. The memory complexity is upper bounded by storing one bit per key candidate which is therefore 2^{64} bits. The memory for storing the approximately $2 \cdot 2^{17}$ plaintexts and ciphertexts of a structure is negligible.

4 Conclusion

In this paper, we outline related-key impossible-differential attacks against 21-round SKINNY-64/128 as well as attacks on 22 and 23 rounds under the assumption of having 48 of the 128-bit tweakey as public tweak. Our attacks are based on an 11-round impossible differential trail, to which we prepend six and append five rounds before and after the trail, respectively, to obtain an attack on 22 rounds. Finally, we can prepend a 23-rd round under similar assumptions.

Acknowledgements. This work was initiated during the group sessions of the 6th Asian Workshop on Symmetric Cryptography (ASK 2016) held in Nagoya, Japan. Ralph Ankele is supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. H2020-MSCA-ITN-2014-643161 ECRYPT-NET. Gaoli Wang is supported by National Natural Science Foundation of China (Grant Nos. 61572125, 61373142), Shanghai High-Tech Field Project (Grant No. 16511101400). Siang Meng Sim is supported by the Singapore National Research Foundation Fellowship 2012 (NRF-NRFF2012-06). This work has been supported in part by the Austrian Science Fund (project P26494-N15).

Appendix

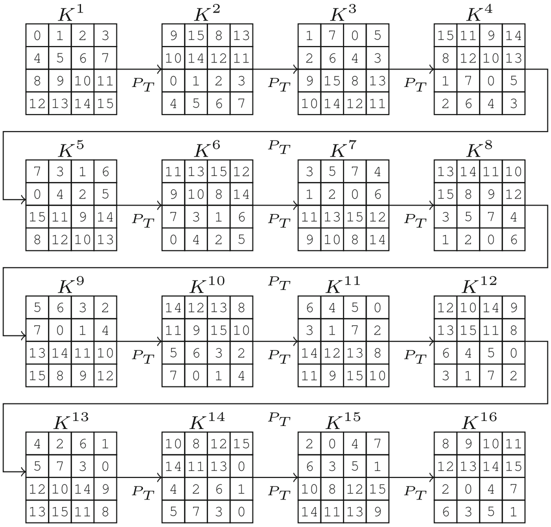


Fig. 7. The permutation P_T in the tweakey schedule has a period of 16.

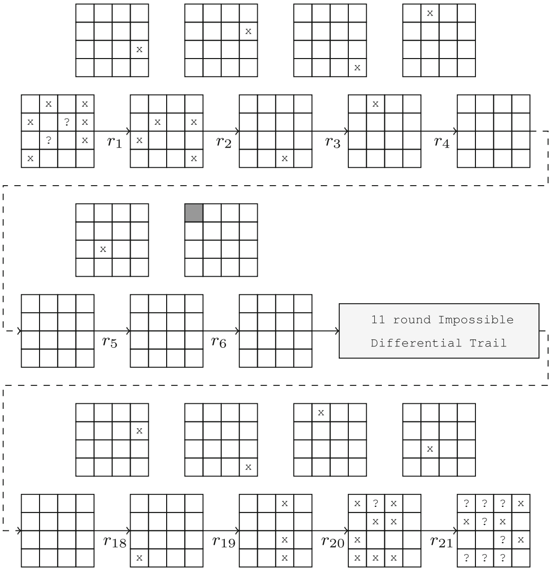


Fig. 8. Related-key impossible differential attack on 21-round SKINNY 64/128 (the dark gray cell visualises the cancelation of the tweakeys).

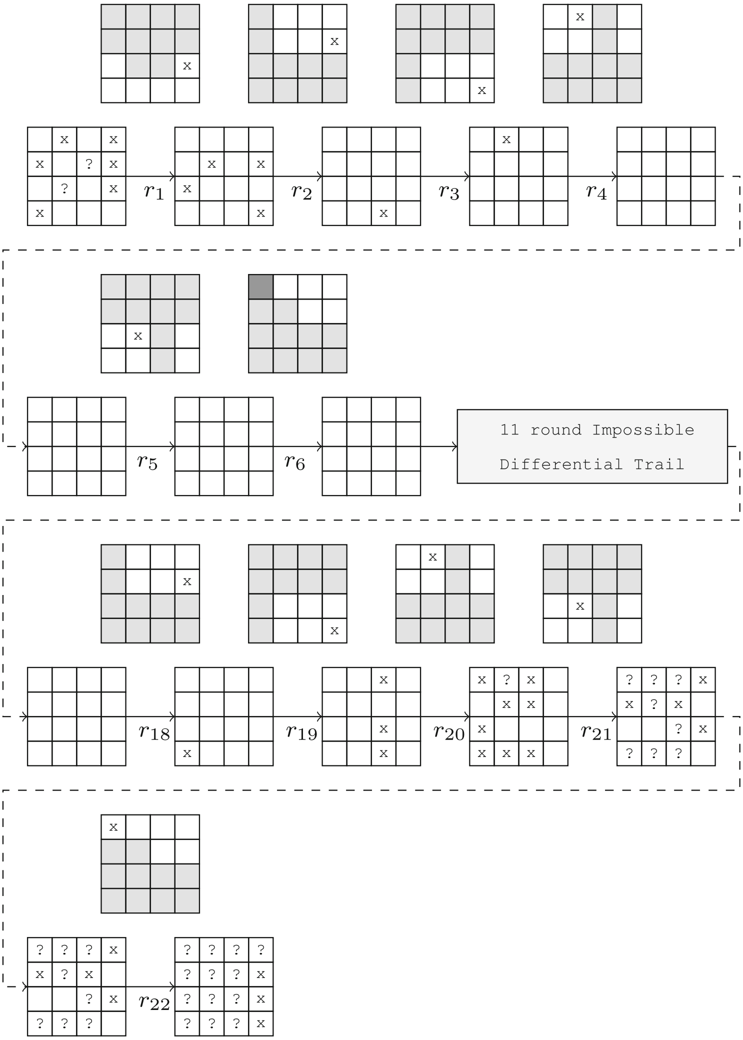


Fig. 9. Related-key impossible differential attack on 22 round SKINNY 64/128 (grey cells are the key, white cells are the tweak, the dark gray cell visualises the cancelation of the tweakeys)

Table 2. Difference-distribution table

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	.	.	.	.	.	.	.	.	4	4	4	4	.	.	.	.
2	.	4		4		4	4	.	.	.	.	.	.	.	.	.
3	.	.	.	.	.	.	.	.	2	2	2	2	2	2	2	2
4	.	.	4	.	.	.	2	2				4	2	2	.	.
5	.	.	4	.	.	.	2	2	.	.	4	.	2	2	.	.
6	.	2	.	2	2	.	.	2	2	.	2	.	.	2	2	.
7	.	2	.	2	2	.	.	2	.	2	.	2	2	.	.	2
8	.	.	.	.	4	4	.	.	.	.	.	.	2	2	2	2
9	.	.	.	.	4	4	.	.	.	.	.	.	2	2	2	2
a	.	.	.	.	.	4	4	.	2	2	2	2	.	.	.	.
b	.	4	.	4	.	.	.	.	.	.	.	.	2	2	2	2
c	.	.	4	.	.	.	2	2	4	.	.	.	.	.	2	2
d	.	.	4	.	.	.	2	2	.	4	.	.	.	.	2	2
e	.	2	.	2	2	.	.	2	.	2	.	2	.	2	2	.
f	.	2	.	2	2	.	.	2	2	.	2	.	2	.	.	2

References

1. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK Families of Lightweight Block Ciphers (2013). Cryptology ePrint Archive, Report 2013/404. http://eprint.iacr.org/
2. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: Cryptanalysis competition (2016). https://sites.google.com/site/skinnycipher/cryptanalysis-competition
3. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 123–153. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53008-5_5
4. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer, Heidelberg (1999). doi:10.1007/3-540-48910-X_2
5. Jean, J., Nikolić, I., Peyrin, T.: Tweaks and keys for block ciphers: the TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 274–288. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45608-8_15
6. Knudsen, L.: DEAL - A 128-bit Block Cipher. In: NIST AES Proposal (1998)
7. Liu, G., Ghosh, M., Ling, S.: Security Analysis of SKINNY under Related-Tweakey Settings (2016). Cryptology ePrint Archive, Report 2016/1108. http://eprint.iacr.org/2016/1108

8. Sadeghi, S., Mohammadi, T., Bagheri, N.: Cryptanalysis of Reduced round SKINNY Block Cipher (2016). Cryptology ePrint Archive, Report 2016/1120. http://eprint.iacr.org/2016/1120
9. Tolba, M., Abdelkhalek, A., Youssef, A.M.: Impossible differential cryptanalysis of reduced-round SKINNY. In: Joye, M., Nitaj, A. (eds.) AFRICACRYPT 2017. LNCS, vol. 10239, pp. 117–134. Springer, Cham (2017). doi:10.1007/978-3-319-57339-7_7

Faster Secure Multi-party Computation of AES and DES Using Lookup Tables

Marcel Keller, Emmanuela Orsini, Dragos Rotaru(✉), Peter Scholl,
Eduardo Soria-Vazquez, and Srinivas Vivek

Department of Computer Science, University of Bristol, Bristol, UK
{m.keller,emmanuela.orsini,dragos.rotaru,peter.scholl,
eduardo.soria-vazquez,sv.venkatesh}@bristol.ac.uk

Abstract. We present an actively secure protocol for secure multi-party computation based on lookup tables, by extending the recent, two-party 'TinyTable' protocol of Damgård et al. (ePrint 2016). Like TinyTable, an attractive feature of our protocol is a very fast and simple online evaluation phase. We also give a new method for efficiently implementing the preprocessing material required for the online phase using arithmetic circuits over characteristic two fields. This improves over the suggested method from TinyTable by at least a factor of 50.

As an application of our protocol, we consider secure computation of the Triple DES and the AES block ciphers, computing the S-boxes via lookup tables. Additionally, we adapt a technique for evaluating (Triple) DES based on a polynomial representation of its S-boxes that was recently proposed in the side-channel countermeasures community. We compare the above two approaches with an implementation. The table lookup method leads to a very fast online time of over 230,000 blocks per second for AES and 45,000 for Triple DES. The preprocessing cost is not much more than previous methods that have a much slower online time.

Keywords: Multi-party computation · Block cipher · Implementation

1 Introduction

Secure multi-party computation (MPC) protocols allow useful computations to be performed on private data, without the data owners having to reveal their

This work has been partially supported by EPSRC via grant EP/N021940/1; by the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under contract No. N66001-15-C-4070; by EPSRC via grant EP/M016803; and by the European Union's H2020 Programme under grant agreement number ICT-644209 (HEAT) and the Marie Skłodowska-Curie grant agreement No. 643161 (ECRYPT-NET).

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 229–249, 2017.
DOI: 10.1007/978-3-319-61204-1_12

inputs. The last decade has seen an enormous amount of progress in the practicality of MPC, with many works designing more efficient protocols and implementations. There has also been a growing interest in exploring the possible applications of MPC, with a number of works targeting specific computations such as auctions, statistics and stable matching [6,7,23,29].

One promising application area that has recently emerged is the use of secure computation to protect long-term secret keys, for instance, in authentication servers or to protect company secrets [4]. Here, the secret key, sk, is split up into n pieces, or shares, such that certain subsets of the n shares are needed to reconstruct sk, and each share is stored on a different server (possibly in a different location and/or managed by a separate entity). When the key is needed by an application, say for a user logging in, the servers run an MPC protocol to authenticate the user, without ever revealing sk. Typically, the type of computation required here can be performed using a symmetric primitive such as a block cipher or hash function.

Several previous works in secure computation have studied the above type of application, and the AES function is even considered a standard benchmark for new protocols [5,16,35,37,39]. A recent line of works has even looked at special-purpose symmetric primitives, designed to have low complexity when evaluated in MPC [1,2,27]. However, in industries such as banking and the wider financial sector, strict regulations and legacy systems mean that switching to new primitives can be very expensive, or even impossible. Indeed, most banking systems today are using AES or Triple DES (3DES) to secure their data [24], but may still benefit greatly from MPC technologies to prevent theft and data breaches.

1.1 Our Contributions

In this work, we focus on the task of secure multi-party computation of the AES and the (Triple) DES block ciphers, in the setting of active security against any number of corrupted parties. We present a new technique for the preprocessing phase of efficient, secure computation of *table lookup* (with a secret index), and apply this to evaluating the S-boxes of AES and DES. In addition, we describe a new method of secure MPC evaluation of the DES S-boxes based on evaluating polynomials over binary finite fields, which reduces the number of non-linear field multiplications.

Our protocol for secure table lookup builds upon the recent ‘TinyTable’ protocol for secure two-party computation by Damgård et al. [18]. This protocol requires a preprocessing phase, which is independent of the inputs, where randomly masked (or ‘scrambled’) lookup tables on random data are created. In the online phase, where the function is securely evaluated, each (one-time) masked table can be used to perform a single table lookup on a private index in the MPC protocol. The online phase of TinyTable is *very efficient*, as each party only needs to send $\log_2 N$ bits over the network, for a table of size N.

However, the suggested technique for creating the masked tables is far less efficient: for secure computation of AES, it would take at least *256 times longer* to create the masked lookup tables, compared with using standard methods with a slower online time.

We extend and improve upon the TinyTable approach in two ways. Firstly, we show that the technique can easily be generalized to the multi-party setting and used in any SPDZ-like MPC protocol based on secret-sharing and information-theoretic MACs. Secondly, we describe a new, general approach for creating the masked tables using finite field arithmetic, which significantly improves the preprocessing cost of the protocol. Concretely, for a lookup table of size N, we can create the masked table using an arithmetic circuit over $\mathbb{F}_{2^k}$ with fewer than $N/k + \log N$ multiplications. This provides a range of possible instantiations with either binary or arithmetic circuit-based protocols. When using binary circuits, we only require $N - 2$ multiplications. For arithmetic circuits over $\mathbb{F}_{2^8}$, an AES S-box can be preprocessed with 33 multiplications, improving on the method in [18], which takes 1792 multiplications, by more than 50 times. With current practical protocols, it turns out to be even more efficient to work over $\mathbb{F}_{2^{40}}$, with only 11 multiplications. We remark that standard methods for computing AES based on polynomials or Boolean circuits can obtain better overall running times, but with a much slower online phase. The main goal of this work is to reduce the preprocessing cost whilst preserving the very fast online phase of TinyTable.

We also consider a new method for secure multi-party computation of DES based on a masking side-channel countermeasure technique. The DES S-box can be viewed as a lookup table mapping 6 bits to 4 bits, or as a polynomial over $\mathbb{F}_{2^6}$. A naïve method requires 62 field multiplications to evaluate a DES S-box polynomial over $\mathbb{F}_{2^6}$. There were many recent works that reduced the number of non-linear multiplications required to evaluate polynomials over binary finite fields, including the DES S-box polynomials [10,13,14,38,41]. A recent proposal by Pulkus and Vivek [38] showed that the DES S-boxes, when represented over a different field, $\mathbb{F}_{2^8}$, can be evaluated with only 3 non-linear multiplications. This is better than the best-known circuit over $\mathbb{F}_{2^6}$, which needs 4 non-linear multiplications. Applying the Pulkus–Vivek method in our context, we show how 1 round of the DES block cipher can be computed with just 24 multiplications over $\mathbb{F}_{2^8}$. This compares favorably with previous methods based on evaluating polynomials over $\mathbb{F}_{2^6}$ and boolean circuits.

Analogous to the MPC protocols based on table lookups, there are also masking side-channel countermeasures based on random-table lookups [11,12]. This analogy should not come as a surprise since the masking technique is also based on secret-sharing. The state-of-the-art for (higher-order) masking seems to suggest that the schemes based on evaluation of S-box polynomials usually outperform table-lookups based schemes in terms of time, RAM memory and randomness. We perform a similar comparison in the MPC context too. To this end, we evaluate the complexity of the various methods for secure computation of AES and 3DES, and present some implementation results. We implemented the protocols using the online phase of the SPDZ [16,19] MPC protocol. The preprocessing additionally requires some random multiplication triples and shared bits, for which we estimated costs using MASCOT [30] for arithmetic circuits, and based on the recent optimized TinyOT protocol [35,43] for binary circuits.

Our experiments show that the fastest online evaluation is achieved using lookup tables. The preprocessing for this method costs much less when using arithmetic circuits over larger fields, compared with a binary circuit protocol such as TinyOT [35,43], despite the quadratic (in the field bit length) communication cost of [30]. The polynomial-based methods for AES and DES still perform slightly better in the preprocessing phase, but for applications where a low online latency is desired, the lookup table approach is definitely preferred. If an application is mainly concerned with the total running time, then the polynomial-based methods actually lead to runtimes for AES that are comparable with the fastest recent 2-PC implementations using garbled circuits.

Related Work. A recent, independent work by Dessouky et al. [22] presented two different protocols for lookup table-based secure two-party computation in the semi-honest security model. The first protocol, OP-LUT, offers an online phase very similar to ours (and [18]), while the preprocessing stage, that is implemented using 1-out-of-N oblivious transfer, is incomparable to ours as we must work much harder to achieve active security.

The second protocol, SP-LUT, proposes a more efficient preprocessing phase, which only requires random 1-out-of-N oblivious transfer computation, but a slower online evaluation; however this protocol has a much lower overall communication compared to the previous one. These two protocols are also compared with the OTTT (One-Time Truth-Table) protocol by Ishai et al. [28] with parallel circuit based preprocessing [20]. More detailed comparisons with our protocols are provided in Sect. 5.2.

This work also provides an FPGA-based synthesis tool that transforms a high level function representation to multi-input/multi-output table-lookup representation, which could also be used with our protocol.

2 Preliminaries

We denote by λ the computational security parameter and κ the statistical security parameter. We consider the sets $\{0, 1\}$ and $\mathbb{F}_2^k$ endowed with the structure of the fields $\mathbb{F}_2$ and $\mathbb{F}_{2^k}$, respectively. We denote by $\mathbb{F} = \mathbb{F}_{2^k}$ any finite field of characteristic two. Finally, we use $a \overset{\$}{\leftarrow} A$ as notation for a uniformly random sampling of a from a set A.

Note that by linearity we always mean $\mathbb{F}_2$-linearity, as we only consider fields of characteristic 2.

2.1 MPC Computation Model

Our protocol builds upon the *arithmetic black-box* model for MPC, represented by the functionality $\mathcal{F}_{\mathsf{ABB}}$ (shown in the full version). This functionality permits the parties to input and output secret-shared values and evaluate arbitrary binary circuits performing basic operations. This abstracts away the underlying details of secret sharing and MPC. Other than the standard **Add** and **Mult** commands, $\mathcal{F}_{\mathsf{ABB}}$ also has a **BitDec** command for generating the bit decomposition of a given secret-shared value, two commands **Random** and **RandomBit** for generating random values according to different distributions and an **Open** command which allows the parties and the adversary to output values. **BitDec** can be implemented in a standard manner by opening and then bit-decomposing $x + r$, where r is obtained using k secret random bits.

We use the notation $[\![x]\!]$ to denote an authenticated and secret-shared value x, which is stored by $\mathcal{F}_{\mathsf{ABB}}$. More precisely, this can be implemented with active security using the SPDZ protocol [16,19] based on additive secret sharing and unconditionally secure MACs. We also use the $+$ and $\cdot$ operators to denote calls to **Add** and **Mul** with the appropriate shared values in $\mathcal{F}_{\mathsf{ABB}}$.

More concretely, our protocols are in the so called *preprocessing model* and consist of two different phases: an *online* computation, where the actual evaluation takes place, and a *preprocessing* phase that is independent of the parties' inputs. During the online evaluation, linear operations only require local computations thanks to the linearity of the secret sharing scheme and MAC. Multiplications and bit decompositions require random preprocessed data and interactions. More generally, the main task of the preprocessing step is to produce enough random secret data for the parties to use during the online computation: other than multiplication triples, which allow parties to compute products, it also provides random shared values. The preprocessing phase can be efficiently implemented using OT-based protocols for binary circuits [8,25,43] and arithmetic circuits [30].

Security Model. We describe our protocols in the universal composition (UC) framework of Canetti [9], and assume familiarity with this. Our protocols work with n parties from the set $\mathcal{P} = \{P_1, \ldots, P_n\}$, and we consider security against malicious, static adversaries, i.e. corruption may only take place before the protocols start, corrupting up to $n - 1$ parties.

3 Evaluating AES and DES S-box Polynomials

In this section, we recollect some of the previously known methods that aim to reduce the number of non-linear operations to evaluate univariate polynomials over binary finite fields, particularly, the AES and the DES S-boxes represented in this form. Note here that, by a non-linear multiplication, we mean those multiplications of polynomials that are neither multiplication by constants nor squaring operations. Since squaring is a linear operation in binary fields, once a monomial is computed, it can be repeatedly squared to generate as many more monomials as possible without costing any non-linear multiplication.

Due to limited space, a more detailed discussion can found in the full version.

3.1 AES S-box

The AES S-box evaluation on a given input (as an element of $\mathbb{F}_{2^8}$) consists of first computing its multiplicative inverse in $\mathbb{F}_{2^8}$ (mapping zero to zero), and then applying a bijective affine transformation. For the inverse S-box, the inverse affine transformation is applied first and then the multiplicative inverse. Note that the polynomial representation of the inverse function in $\mathbb{F}_{2^8}$ is X^{254}.

BitDecompostion Method. This approach, described by Damgård et al. [15], computes the squares X^{2^i}, for $i \in [7]$, and then multiplies them to get X^{254}. This method needs 6 non-linear multiplications.

Rivain–Prouff Method. This method, as presented in Gentry et al. [26], is a variant of the method of Rivain–Prouff [40] to evaluate the AES S-box polynomial using only 4 non-linear multiplications in $\mathbb{F}_{2^8}[X]$: $\{X, X^2\} \xrightarrow{\times} \{X^3, X^{12}\} \xrightarrow{\times} \{X^{14}\} \xrightarrow{\times} \{X^{15}, X^{240}\} \xrightarrow{\times} X^{254}$.

3.2 Des S-boxes

Cyclotomic Class Method. Recall that DES has eight 6-to-4-bit S-boxes. In this naïve method given by Carlet et al. [10], the DES S-boxes are represented as univariate polynomials over $\mathbb{F}_{2^6}$. In particular, the 4-bit S-box outputs are padded with zeros in the most significant bits and then identified with the elements of $\mathbb{F}_{2^6}$. It turns out that these polynomials have degree at most 62 [41].

Over $\mathbb{F}_{2^m}[X]$, define $C_i^m := \left\{X^{i \cdot 2^j} : j = 0, 1, \ldots, m-1\right\}$ for $0 < i < 2^m$. Now we need to compute $C_0^6, C_1^6, C_3^6, C_5^6, C_7^6, C_9^6, C_{11}^6, C_{13}^6, C_{15}^6, C_{21}^6, C_{23}^6, C_{27}^6, C_{31}^6$, to cover all monomials up to degree 62, and this needs at most 11 non-linear multiplications. The target polynomial is then simply obtained as a linear combination of the computed monomials.

Pulkus–Vivek Method. This generic method to evaluate arbitrary polynomials over binary finite fields was proposed recently by Pulkus and Vivek [38] as an improvement over the method of Coron–Roy–Vivek [13,14]. In the PV method, the DES S-boxes are represented as polynomials over $\mathbb{F}_{2^8}$ instead of $\mathbb{F}_{2^6}$. The 6-bit input strings of the DES S-boxes are padded with zeroes in the two most significant positions and then naturally identified with the elements of $\mathbb{F}_{2^8}$. The four most significant coefficient bits of the polynomial outputs are *discarded* to obtain the desired 4-bit S-box output.

Firstly, a set of monomials $L = C_1^8 \cup C_3^8 \cup C_7^8$ in $\mathbb{F}_{2^8}[X]$ is computed. Then a polynomial, say $P(X)$, representing the given S-box is sought as $P(X) = p_1(X) \cdot q_1(X) + p_2(X)$, where $p_1(X)$, $q_1(X)$, and $p_2(X)$ have monomials only from the set L. In total, the PV method needs 3 non-linear multiplications in $\mathbb{F}_{2^8}[X]$ to evaluate each of the S-box polynomial.

3.3 MPC Evaluation of AES and DES S-box Polynomials

Here we detail the MPC evaluation of AES and DES S-boxes using the techniques described above. We recall that since the S-boxes, in both the ciphers we are considering, are the only non-linear components, they represent the only parts which actually need interactions in an MPC evaluation.

AES Evaluation. As we mention before in Sect. 3.1, the straightforward way to compute the S-box is using the BitDecomposition method, which requires 6 multiplications in $4+1$ rounds. We are considering the case of active security, so the AES evaluation is done in the field $\mathbb{F}_{2^{40}}$ instead of $\mathbb{F}_{2^8}$, via the embedding $\mathbb{F}_{2^8} \hookrightarrow \mathbb{F}_{2^{40}}$. This follows from the fact that we are using the SPDZ protocol which requires a field size of at least 2^κ, where κ is the statistical security parameter. This permits to have only one MAC per data item [15].

The evaluation proceeds as follow: first X is bit-decomposed so that all the squarings can be locally evaluated, and then X^{254} is obtained as described in [15]:

$$X^{254} = ((X^2 \cdot X^4) \cdot (X^8 \cdot X^{16})) \cdot ((X^{32} \cdot X^{64}) \cdot X^{128}).$$

This requires 4 rounds, out of which one is a call to BitDec. We also need an extra round for computing the inverse of the field embedding $\mathbb{F}_{2^8} \hookrightarrow \mathbb{F}_{2^{40}}$ to evaluate the S-box linear layer. We denote this method by `AES-BD`.

We denote by `AES-RP` the AES S-box evaluation that uses the Rivain–Prouff method (cf. Sect. 3.1). It requires $6+1$ rounds to compute the four powers $X^3, X^{14}, X^{15}, X^{254}$. Furthermore, this can be done with three calls to BitDec and four non-linear multiplications, but some of the openings can be done in parallel which yields to a depth-6 circuit. As before, we need an extra round to call BitDec and compute the S-box linear layer.

DES Evaluation. We denote by `DES-PV` the DES S-box evaluation using the Pulkus–Vivek method. Note that, although in side-channel world computing the squares is for free, since it is an $\mathbb{F}_2$-linear operation, in a secret-shared based MPC with MACs this is no longer true and we need to bit-decompose.

The squares from C_1^8, C_3^8, C_7^8, are obtained locally after X, X^3, X^7 are bit-decomposed. Here we need two multiplications, since $X^3 = X \cdot X^2$ and $X^7 = X^3 \cdot X^4$. The third multiplication occurs when computing the product $p_1(X) \cdot q_1(X)$, resulting in an S-box cost of only 3 triples, 24 bits and 5 communication rounds.

The number of rounds is due to 3 calls to BitDec (on X^3, X^7 and $p_1(X) \cdot q_1(X) + p_2(X)$) and 3 non-linear multiplications. Although at a first glance there seems to be six rounds, we have that $\mathsf{BitDec}(X^7)$ is independent of the $\mathsf{BitDec}(X^3)$, as we can compute X^7 without the call $\mathsf{BitDec}(X^3)$, resulting in only five rounds.

4 MPC Evaluation of Boolean Circuits Using Lookup Tables

In this section we describe an efficient MPC protocol for securely evaluating circuits over extension fields of $\mathbb{F}_2$ (including boolean circuits) containing additional 'lookup table' gates. This protocol is in the preprocessing model and follows the same approach proposed in [20], evaluating lookup table gates using preprocessed, masked lookup tables.

The functionality that we implement is $\mathcal{F}_{\mathsf{ABB-LUT}}$ (Fig. 1), which augments the standard $\mathcal{F}_{\mathsf{ABB}}$ functionality with a table lookup command. The concrete online cost of each table lookup is just $\log_2 N$ bits of communication per party, where N is the size of the table. Note that the functionality $\mathcal{F}_{\mathsf{ABB-LUT}}$ works over a finite field $\mathbb{F}_{2^k}$, and has been simplified by assuming that the size of the range and domain of the lookup table T is not more than 2^k. However, our protocol actually works for general table sizes, and $\mathcal{F}_{\mathsf{ABB-LUT}}$ can easily be extended to model this by representing a table lookup result with several field elements instead of one.

We now show how Protocol 1 implements the **Table Lookup** command of $\mathcal{F}_{\mathsf{ABB-LUT}}$, given the right preprocessing material. For any non-linear function T, with ℓ input and m output bits, it is well known that it can be implemented as a lookup table of 2^ℓ components of m bits each. To evaluate $\mathsf{T}(\cdot)$ on a secret authenticated value $\llbracket x \rrbracket, x \in \mathbb{F}_{2^\ell}$, the parties use a random authenticated T

Functionality $\mathcal{F}_{\mathsf{ABB-LUT}}$

This functionality has all the features of $\mathcal{F}_{\mathsf{ABB}}$, operating over $\mathbb{F}_{2^k}$, plus the following command.

Table Lookup: On command $(\mathsf{T}, \mathsf{id}_1, \mathsf{id}_2)$ from all parties, where $\mathsf{T} : \{0,1\}^\ell \rightarrow \{0,1\}^m$, for $\ell, m \leq k$, and id_1 is present in memory, retrieve (id_1, x) and store $(\mathsf{id}_2, \mathsf{T}(x))$.

Fig. 1. The ideal functionality for MPC using lookup tables

Functionality $\mathcal{F}_{\mathsf{Prep-LUT}}$

This functionality has all of the same features as $\mathcal{F}_{\mathsf{ABB}}$, with the following additional command.

Masked Table: On input $(\mathsf{MaskedTable}, \mathsf{T}, \mathsf{id})$ from all parties, where $\mathsf{T} : \{0,1\}^\ell \rightarrow \{0,1\}^m$ for $\ell, m \leq k$, sample a random value s, set $(\mathsf{Val}[\mathsf{id}_s], \mathsf{Val}[\mathsf{id}_{\mathsf{T}(s)}], \ldots, \mathsf{Val}[\mathsf{id}_{\mathsf{T}(s \oplus (2^\ell - 1))}]) \leftarrow (s, \mathsf{T}(s), \ldots, \mathsf{T}(s \oplus (2^\ell - 1))$, and return $(\mathsf{id}_s, (\mathsf{id}_{\mathsf{T}(s)}, \ldots, \mathsf{id}_{\mathsf{T}(s \oplus (2^\ell - 1))}))$.

Fig. 2. Ideal functionality for the preprocessing of masked lookup tables.

Protocol 1. Secure online evaluation of SBox using lookup tables

Table Lookup: On input $[\![x]\!]$ compute $[\![\mathsf{T}(x)]\!]$ as follows.

1. Call $\mathcal{F}_{\mathsf{Prep-LUT}}$ on input ($\mathsf{MaskedTable}, \mathsf{T}$), and obtain a precomputed masked table ($[\![s]\!], [\![\mathsf{Table}(s)]\!]$).
2. The parties open the value $h = x \oplus s$.
3. Locally compute $[\![\mathsf{T}(x)]\!] = [\![\mathsf{Table}(s)]\!][h]$, where $[\![\mathsf{Table}(s)]\!][h]$ is the hth component of $[\![\mathsf{Table}(s)]\!]$.

evaluation from $\mathcal{F}_{\mathsf{Prep-LUT}}$ (Fig. 2). More precisely, we would like the preprocessing to output values ($[\![s]\!], [\![\mathsf{Table}(s)]\!]$), where $[\![s]\!]$ is a random authenticated value unknown to the parties and $[\![\mathsf{Table}(s)]\!]$) is the table

$$[\![\mathsf{Table}(s)]\!] = \big(\, [\![\mathsf{T}(s)]\!], [\![\mathsf{T}(s \oplus 1)]\!], \ldots, [\![\mathsf{T}(s \oplus (2^\ell - 1))]\!] \,\big)\,,$$

so that $[\![\mathsf{Table}(s)]\!][j], 0 \leq j \leq 2^\ell - 1$, denotes the element $[\![\mathsf{T}(s \oplus j)]\!]$. Given such a table, evaluating $[\![\mathsf{T}(x)]\!]$ is straightforward: first the parties open the value $h = x \oplus s$ and then they locally retrieve the value $[\![\mathsf{Table}(s)]\!][h] = [\![\mathsf{T}(s \oplus h)]\!] = [\![\mathsf{T}(s \oplus s \oplus x)]\!] = [\![\mathsf{T}(x)]\!]$.

Correctness easily follows from the linearity of the $[\![\cdot]\!]$-representation and the discussion above. Privacy follows from the fact that the value s used in **Table Lookup** is randomly chosen and is used only once, thus it perfectly blinds the secret value x.

4.1 The Preprocessing Phase: Securely Generating Masked Lookup Tables

In this section we describe how to securely implement $\mathcal{F}_{\mathsf{Prep-LUT}}$ (see Fig. 2), and in particular how to generate masked lookup tables which can be used for the online phase evaluation.

Recall that the goal is to obtain the shared values:

$$[\![\mathsf{Table}(s)]\!] = ([\![\mathsf{T}(s)]\!], [\![\mathsf{T}(s \oplus 1)]\!], \ldots, [\![\mathsf{T}(s \oplus (2^\ell - 1))]\!]).$$

Protocol 2 begins by taking a secret, random ℓ-bit mask $[\![s]\!] = ([\![s_0]\!], \ldots, [\![s_{\ell-1}]\!])$. Then, the parties expand s into a secret-shared bit vector $(s'_0, \ldots, s'_{2^\ell - 1})$ which has a 1 in the s-th entry and is 0 elsewhere. We denote this procedure—the most expensive part of the protocol—by Demux, and describe how to perform it in the next section.

Once this is done, the parties can obtain the i-th entry of the masked lookup table by computing:

$$\mathsf{T}(i) \cdot [\![s'_0]\!] + \mathsf{T}(i \oplus 1) \cdot [\![s'_1]\!] + \cdots + \mathsf{T}(i \oplus (2^\ell - 1)) \cdot [\![s'_{2^\ell - 1}]\!],$$

which is clearly $[\![\mathsf{T}(i \oplus s)]\!]$ as required. Note that since the S-box is public, this is a local computation for the parties. In the following we give an efficient protocol for computing Demux.

4.2 Computing Demux with Finite Field Multiplications

We now present a general method for computing Demux using fewer than $N/k + \log N$ multiplications over $\mathbb{F}_{2^k}$, when k is any power of 2 and $N = 2^\ell$ is the table size. Launchbury et al. [32] previously described a protocol with $O(N)$ multiplications in $\mathbb{F}_2$, but our protocol has fewer multiplications than theirs for all choices of k.

As said before, Demux maps a binary representation $(s_0, \ldots, s_{\ell-1})$ of an integer $s = \sum_{i=0}^{\ell-1} s_i \cdot 2^i$ into a unary representation of fixed length 2^ℓ that contains a one in the position s and zeros elsewhere. A straightforward way to compute Demux is by computing, over $\mathbb{F}_{2^N}$[1]:

$$[\![s']\!] = \prod_{i=0}^{\ell-1} ([\![s_i]\!] \cdot X^{2^i} + (1 - [\![s_i]\!])).$$

Notice that if $s_i = 1$ then the i-th term of the product equals X^{2^i}, whereas the term equals 1 if $s_i = 0$. This means the entire product evaluates to $s' = X^s$, where s is the integer representation of the bits $(s_0, \ldots, s_{\ell-1})$. Bit decomposing s' obtains the demuxed output as required. Unfortunately, this approach does not scale well with N, the table size, as we must exponentially increase the size of the field.

We now show how to compute this more generally, using operations over $\mathbb{F}_{2^k}$, where k is a power of two. We will only ever perform multiplications between elements of $\mathbb{F}_2$ and $\mathbb{F}_{2^k}$, and will consider elements of $\mathbb{F}_{2^k}$ as vectors over $\mathbb{F}_2$. Define the partial products, for $j = 1, \ldots, \ell$:

$$p_j(X) = \prod_{i=0}^{j-1} (s_i \cdot X^{2^i} + (1 - s_i)) \in \mathbb{F}_{2^N}$$

and note that $p_{j+1}(X) = p_j(X) \cdot (s_j \cdot X^{2^j} + (1 - s_j))$, for $j < \ell$.

Note also that the polynomial $p_j(X)$ has degree $< 2^j$, so $p_j(X)$ can be represented as a vector in $\mathbb{F}_2^{2^j}$ containing its coefficients, and more generally, a

Protocol 2. Protocol to generate secret shared table lookup

Table: On input (Table, P_i) from all the parties, do the following:

1: Take ℓ random authenticated bits $[\![s_0]\!], \ldots, [\![s_{\ell-1}]\!]$, where each s_i is unknown to all the parties.

2: Compute $([\![s'_0]\!], \ldots, [\![s'_{2^\ell-1}]\!]) \leftarrow \mathsf{Demux}([\![s_0]\!], \ldots, [\![s_{\ell-1}]\!])$

3: $\forall i = 0, \ldots, 2^\ell - 1$, locally compute

$$[\![\mathsf{T}(i \oplus s)]\!] = \mathsf{T}(i) \cdot [\![s'_0]\!] + \mathsf{T}(i \oplus 1) \cdot [\![s'_1]\!] + \cdots + \mathsf{T}((2^\ell - 1) \oplus i) \cdot [\![s'_{2^\ell-1}]\!]$$

[1] A similar trick was used by Aliasgari et al. [3] for binary to unary conversion over prime fields.

Protocol 3. $(\llbracket s'_0 \rrbracket, \dots, \llbracket s'_{N-1} \rrbracket) \leftarrow \mathsf{Demux}(k, \llbracket s_0 \rrbracket, \dots, \llbracket s_{\ell-1} \rrbracket)$

Require: k a power of two, $u = N/k$, $\ell = \log_2 N$
Input: Bit decomposition of $s \in \{0, \dots, N-1\}$, with LSB first
Output: Satisfies $s'_s = 1$ and $s'_i = 0$ for all $i \neq s$

```
$\llbracket p \rrbracket = (1 - \llbracket s_0 \rrbracket, \llbracket s_0 \rrbracket)$                                // $p$ starts in $\mathbb{F}_2^2$
for $j = 1$ to $\ell - 1$ do
    $\llbracket t \rrbracket = \llbracket s_j \rrbracket \cdot \llbracket p \rrbracket$                              // $\mathbb{F}_2 \times \mathbb{F}_2^{2^j}$ multiplication, 1 round
    $\llbracket p \rrbracket = (0^{2^j} \| \llbracket t \rrbracket) + (\llbracket p \rrbracket - \llbracket t \rrbracket) \| 0^{2^j})$   // $p$ now in $\mathbb{F}_2^{2^{j+1}}$
Write $\llbracket p \rrbracket = (\llbracket b_0 \rrbracket, \dots, \llbracket b_{u-1} \rrbracket)$                      // $b_i \in \mathbb{F}_2^k$
for $i = 0$ to $u - 1$ do
    $(\llbracket s'_{ki} \rrbracket, \dots, \llbracket s'_{ki+k-1} \rrbracket) = \mathsf{BitDec}(\llbracket b_i \rrbracket)$   // 1 round
return $(\llbracket s'_0 \rrbracket, \dots, \llbracket s'_{N-1} \rrbracket)$
```

vector p_j containing $\lceil 2^j/k \rceil$ elements of $\mathbb{F}_2^k$. This is the main observation that allows us to emulate the computation of s' using only $\mathbb{F}_{2^k}$ arithmetic.

Given a sharing of p_j represented in this way, a sharing of $p_j(X) \cdot X^{2^j}$ can be seen as the vector (increasing the powers of X from left to right):

$$(0^{2^j} \| p_j) \in \mathbb{F}_2^{2^{j+1}}$$

and a vector representation of $p_{j+1}(X)$ is:

$$\Big((0^{2^j} \| s_j \cdot p_j) + ((1 - s_j) \cdot p_j \| 0^{2^j}) \Big) \in \mathbb{F}_2^{2^{j+1}}.$$

Thus, given $\llbracket p_j \rrbracket$ represented as $\lceil 2^j/k \rceil$ shared elements of $\mathbb{F}_{2^k}$, we can compute $\llbracket p_{j+1} \rrbracket$ in MPC with $\lceil 2^j/k \rceil$ multiplications between $\llbracket s_j \rrbracket$ and a shared $\mathbb{F}_{2^k}$ element, plus some local additions.

Starting with $p_1(X) = s_0 \cdot X + (1 - s_0)$ we can iteratively apply the above method to compute $p_\ell = s'$, as shown in Protocol 3. The overall complexity of this protocol is given by

$$\sum_{j=1}^{\ell-1} \lceil 2^j/k \rceil < N/k + \ell$$

multiplications between bits and $\mathbb{F}_{2^k}$ elements.

Table 1 illustrates this trade-off between the field size and number of multiplications for some example parameters. We note that the main factor affecting the best choice of k is the cost of performing a multiplication in $\mathbb{F}_{2^k}$ in the underlying MPC protocol, and this may change as new protocols are developed. However, we compare costs of some current protocols in Sect. 5.

4.3 MPC Evaluation of AES and DES Using Lookup Tables

We now show how to use the lookup table MPC protocol described above to evaluate AES and DES.

Table 1. Number of $\mathbb{F}_2 \times \mathbb{F}_{2^k}$ multiplications for creating a masked lookup table of size N, for varying k.

N	$k = 1$	8	40	64	128
64	62	9	5	5	5
256	254	33	11	8	7
1024	1022	129	31	20	13

AES Evaluation. We require an MPC protocol which performs operations in $\mathbb{F}_{2^8}$. In practice, we actually embed $\mathbb{F}_{2^8}$ in $\mathbb{F}_{2^{40}}$, since we use the SPDZ protocol which requires a field size of at least 2^κ, for statistical security parameter κ. We implement the AES S-box using the table lookup method from Protocol 2 combined with Demux (Protocol 3) over $\mathbb{F}_{2^{40}}$, since this yields a lower communication cost (see Table 4). Notice that the data sent is highly dependent on the number of bits, triples and the field size.

In a naive implementation of this approach, we would have call BitDec on $[\![\mathsf{Table}(s)]\!]$, in order to perform the embedding $\mathbb{F}_{2^8} \hookrightarrow \mathbb{F}_{2^{40}}$. This is required since the table output is not embedded, but the `MixColumns` step needs this to perform multiplication by $X \in \mathbb{F}_{2^8}$ on each state.

With a more careful analysis we can avoid the BitDec calls by locally embedding the bit shares inside Protocol 2. We store the masked S-box table in bit decomposed form and then its bits are multiplied (in the clear) with Demux's output (secret-shared). This trick reduces the online communication by a factor of 8, halves the number of rounds required to evaluate AES and gives a very efficient online phase with only 10 rounds and 160 openings in $\mathbb{F}_{2^{40}}$.

DES Evaluation. Using the fact that DES S-boxes have size 64, we chose to use the Demux Protocol 3 with multiplications in $\mathbb{F}_{2^{40}}$, based on the costs in Table 4. Like AES, we try to isolate the input-dependent phase as much as possible with no extra cost.

Every DES round performs only bitwise addition and no embedding is necessary here. The masked table can be bit-decomposed without interaction, exactly as described above for AES, by multiplying clear bits with secret shared values. This yields a low number of openings, one per S-box look-up, so the total online cost for `3DES` is 46 rounds with 384 openings.

5 Performance Evaluation

This section presents timings for `3DES` and `AES` using the methods presented in previous sections. We also discuss trade-offs and different optimizations which turn out to be crucial for our running-times. The setup we have considered is that both the key and message used in the cipher are secret shared across two parties. We consider the input format for each block cipher as already embedded into

$\mathbb{F}_{2^{40}}$ for AES, or as a list of shared bits for DES. We implemented the protocols using the SPDZ software,[2] and estimated times for computing the multiplication triples and random bits needed based on the costs of MASCOT [30].

The results, shown in Tables 2 and 3, give measurements in terms of *latency* and *throughput*. Latency indicates the online phase time required to evaluate one block cipher, whereas throughput (which we consider for both online and offline phases) shows the maximum number of blocks per second which can be evaluated in parallel during one execution. We also measure the number of rounds of interaction of the protocols, and the number of *openings*, which is the total number of secret-shared field elements opened during the online evaluation.

Benchmarking Environment. The experiments were ran across two machines each with Intel i7-4790 CPUs running at 3.60 GHz, 16 GB of RAM connected over a 1 GBps LAN with an average ping of 0.3 ms (roundtrip). For experiments with 3–5 parties, we used three additional machines with i7-3770 CPUs at 3.1 GHz. In order to get accurate timings each experiment was averaged over 5 executions, each with at least 1000 cipher calls.

Security Parameters and Field Sizes. Secret-sharing based MPC can be usually split into 2 phases—preprocessing and online. In SPDZ-like systems, the preprocessing phase depends on a computational security parameter, and the online phase a statistical security parameter which depends on the field size. In our experiments the computational security parameter is $\lambda = 128$. The statistical security κ is 40 for every cipher except for `3DES-Raw` which requires an embedding into a 42 bit field.

Results. The theoretical costs and practical results are shown in Tables 2 and 3, respectively. Timings are taken only for the encryption calls, excluding the key schedule mechanism.

`AES-BD` is implemented by embedding each block into $\mathbb{F}_{2^{40}}$, and then squaring the shares locally after the inputs are bit-decomposed. In this manner, each S-box computation costs 5 communication rounds and 6 multiplications. This method was described in [15].

`3DES-Raw` represents the `3DES` cipher with the S-box evaluated as a polynomial of degree 62 over the field $\mathbb{F}_{2^6} = \mathbb{F}_2[x]/(x^6 + x^4 + x^3 + x + 1)$. To make the comparisons relevant with other ciphers in terms of active security we chose to embed the S-box input in $\mathbb{F}_{2^{42}}$, via the embedding $\mathbb{F}_{2^6} \hookrightarrow \mathbb{F}_{2^{42}}$, where $\mathbb{F}_{2^{42}} = \mathbb{F}_2[y]/(y^{42} + y^{21} + 1)$ and $y = x^7 + 1$. The S-boxes used for interpolating are taken from the PyCrypto library [34]. `3DES-Raw` is implemented only for benchmarking purposes and it has no added optimizations. One S-box has a cost of 62 multiplications and 62 rounds.

`3DES-PV` is `3DES` implemented with the Pulkus-Vivek method from Section 3.2. Since it has only a few multiplications in $\mathbb{F}_{2^{40}}$, the amount of preprocessing data required is very small, close to `AES-BD`. It suffers in terms of both latency and throughput due to the high number of communication rounds (needed for bit decomposition to perform the squarings).

[2] https://github.com/bristolcrypto/SPDZ-2.

Table 2. Communication cost for AES and 3DES in MPC.

Cipher	Online cost			Preprocessing cost			
	Rounds	Openings	Field	Triples	Bits	Field	Comm. (MB)
AES-BD	50	2240	$\mathbb{F}_{2^{40}}$	960	2560	$\mathbb{F}_{2^{40}}$	4.3
AES-RP	70	1920	$\mathbb{F}_{2^{40}}$	640	5120	$\mathbb{F}_{2^{40}}$	2.9
AES-LT	10	160	$\mathbb{F}_{2^{40}}$	1760	42240	$\mathbb{F}_{2^{40}}$	8.4
3DES-Raw	2979	48024	$\mathbb{F}_{2^{42}}$	23808	2688	$\mathbb{F}_{2^{42}}$	112
3DES-PV	230	3456	$\mathbb{F}_{2^{40}}$	1152	9216	$\mathbb{F}_{2^{40}}$	5.2
3DES-LT	46	384	$\mathbb{F}_{2^{40}}$	1920	26880	$\mathbb{F}_{2^{40}}$	8.8

Table 3. 1 GBps LAN timings for evaluating AES and 3DES in MPC.

Cipher	Online (single-thread)			Online (multi-thread)			Preprocessing[a]
	Latency (ms)	Batch size	ops/s	Batch size	ops/s	Threads	ops/s
AES-BD	5.20	64	758	1024	3164	16	30.7
AES-RP	7.19	1024	940	64	3872	16	**46.1**
AES-LT	**0.928**	2048	53918	512	**236191**	32	16.79
3DES-Raw	270	512	130	-	-	-	1.24
3DES-PV	36.98	512	86	512	366	32	**25.6**
3DES-LT	**4.254**	1024	10883	512	**45869**	16	15.3

[a] Extrapolated from timings for a 128-bit field

Surprisingly, AES-RP (the polynomial-based method from Sect. 3.1) has a better throughput than AES-BD although it requires 20 more rounds and 2 times more shared bits to evaluate. The explanation for this is that in AES-RP there are fewer openings, thus less data sent between parties.

AES-LT and 3DES-LT are the ciphers obtained with the lookup table protocol from Sect. 4. AES-LT achieves the lowest latency and the highest throughput in the online phase. The communication in the preprocessing phase is roughly twice the cost of the previous method, AES-BD.

Packing Optimization. We notice that in the online phase of AES-LT each opening requires to send 8 bit values embedded in $\mathbb{F}_{2^{40}}$. Instead of sending 40 bits across the network we select only the relevant bits, which for AES-LT are 8 bits. This reduces the communication by a factor of 5 and gives a throughput of 236k AES/second over LAN and a multi-threaded MPC engine.

The same packing technique is applied for 3DES-LT since during the protocol we only open 6 bit values from Protocol 1. These bits are packed into a byte and sent to the other party. Here the multi-threaded version of 3DES-LT improves the throughput only by a factor of 4.2x (vs AES-LT 4.4x) due to the higher number of rounds and openings along with the loss of 2 bits from packing.

General Costs of the Table Lookup Protocol. In Table 4, we estimate the communication cost for creating preprocessed, masked tables for a range of table sizes, using our protocol from Sect. 4.1. This requires multiplication triples over $\mathbb{F}_{2^k}$, where k is a parameter of the protocol. When $k = 1$, we give figures using a recent optimized variant [43] of the two-party TinyOT protocol [35]. For larger choices of k, the costs are based on the MASCOT protocol [30]. We note that even though MASCOT has a communication complexity in $O(k^2)$, it still gives the lowest costs (with $k = 40$) for all the table sizes we considered.

Table 4. Total communication cost (kBytes) of the $\mathbb{F}_2 \times \mathbb{F}_{2^k}$ multiplications needed in creating a masked lookup table of size N, with two parties. The $k = 1$ estimates are based on TinyOT [43], the others on MASCOT [30].

N	$k = 1$	40	64	128
64	35.01	21.8	43.52	112.64
256	143.45	47.96	69.63	157.7
1024	577.17	135.16	174.08	292.86

5.1 Multiparty Setting

We also ran the AES-LT protocol with different numbers of parties and measured the throughput of the preprocessing and online phases. Figure 3 indicates that the preprocessing gets more expensive as the number of parties increases, whereas the online phase throughput does not decrease by much. This is likely to be because the bottleneck for the preprocessing is in terms of communication (which is $O(n^2)$ in total), whereas the online phase is more limited by the local computation done by each party.

5.2 Comparison with Other Works

We now compare the performance of our protocols with other implementations in similar settings. Table 5 gives an overview of the most relevant previous works. We see that our AES-LT protocol comes very close to the best online throughput of TinyTable, whilst having a far more competitive offline cost.[3] Our AES-RP variant has a slower online phase, but is comparable to the best garbled circuit protocols overall.

TinyTable Protocol. The original, 2-party TinyTable protocol [18] presented implementations of the online phase only, with two different variants. The fastest variant is based on table lookup and obtains a throughput of around 340 thousand AES blocks per second over a 1Gbps LAN, which is 1.51x faster than our

[3] The reason for the very large preprocessing cost of TinyTable is due to the need to evaluate the S-box 256 times per table lookup.

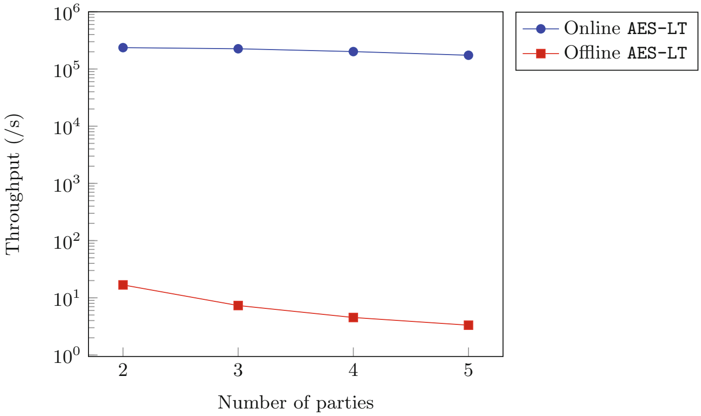


Fig. 3. Table lookup-based AES throughput for multiple parties.

Table 5. Performance comparison with other 2-PC protocols for evaluating AES in a LAN setting.

Protocol	Online		Comms. (total)	Notes
	Latency (ms)	Throughput (/s)		
TinyTable (binary) [18]	4.18	24500	3.07 MB	
TinyTable (optim.) [18]	1.02	339000	786.4 MB	
Wang et al. [43]	0.93	1075	2.57 MB	10 GBps
Rindal-Rosulek [39]	1.0	1000	1.6 MB	10 GBps
OP-LUT [22]	5	41670	0.103 MB	Passive
SP-LUT [22]	6	2208	0.044 MB	Passive
AES-LT	0.93	236200	8.4 MB	
AES-RP	7.19	940	2.9 MB	

online throughput. The latency (for sequential operations) is around 1ms, the same as ours. We attribute the difference in throughput to the additional local computation in our implementation, since we need to compute on MACs for every linear operation.

TinyTable does not report figures for the preprocessing phase. However, we estimate that using TinyOT and the naive method suggested in the paper would need would need over 1.3 million TinyOT triples for AES (34 ANDs for each S-box, repeated 256 times to create one masked table, for 16 S-boxes in 10 rounds). In contrast, our table lookup method uses around 160 thousand TinyOT triples, or just 2080 triples over $\mathbb{F}_{2^{40}}$ (cf. Table 1), per AES block.

Garbled Circuits. There are many implementations of AES for actively secure 2-PC using garbled circuits [33,36,39,42,43]. When measuring online throughput in a LAN setting, using garbled circuits gives much worse performance than methods based on table lookup, because evaluating a garbled circuit is much more expensive computationally. For example, out of all these works the lowest reported online time (even over a 10 GBps LAN) is 0.93 ms [43], and this does not improve in the amortized setting.

Some recent garbled circuit implementations, however, improve upon our performance in the preprocessing phase, where communication is typically the bottleneck. Wang et al. [43] require 2.57 MB of communication when 1024 circuits are being garbled at once, while Rindal and Rosulek [39] need only 1.6 MB. The runtime for both of these preprocessing phases is around 5 ms over a 10 GBps LAN; this would likely increase to at least 15–20 ms in a 1 GBps network, whereas our table lookup preprocessing takes around 60 ms using MASCOT. If a very fast online time is not required, our implementation of the Rivain–Prouff method would be more competitive, since this has a total amortized time of only 23 ms per AES block.

Secret-Sharing Based MPC. Other actively implementations of AES/DES using secret-sharing and dishonest majority based on secret sharing include those using SPDZ [15,31] and MiniMAC [17,21]. Our `AES-BD` method is the same as [15] and obtains faster performance than both SPDZ implementations. For DES, our TinyTable approach improves upon the times of the binary circuit implementation from [31] (which are for single-DES, so must be multiplied by 3) by over 100 times. Regarding MiniMAC, the implementation of [17] obtains slower online phase times than our work and TinyTable, and it is not known how to do the preprocessing with concrete efficiency.

OP-LUT and SP-LUT. The proposed 2-party protocols by Dessouky et al. [22] only offer security in the semi-honest setting. The preprocessing phase for both the protocols are based on 1-out-of-N oblivious transfer. In particular, the cost of the OP-LUT setup is essentially that of 1-out-of-N OT, while the cost of SP-LUT is the cost of 1-out-of-N *random* OT, which is much more efficient in terms of communication.

The online communication cost of OP-LUT is essentially the same as our online phase, since both protocols require each party to send $\log_2 N$ bits for a table of size N. However, we incur some additional local computation costs and a MAC check (at the end of the function evaluation) to achieve active security. The online phase of SP-LUT is less efficient, but the overall communication of this protocol is very low, only 0.055 MB for a single AES evaluation over a LAN setting with 1 GB network.

The work [22] reports figures for both preprocessing and online phase: using OP-LUT gives a latency of around 5 ms for 1 AES block in the LAN setting, and a throughput of 42000 blocks/s. These are both slower than our online phase figures using `AES-LT`. The preprocessing runtimes of both OP-LUT and

SP-LUT are much better than ours, however, achieving over 1000 blocks per second (roughly 80 times faster than AES-LT). This shows that we require a large overhead to obtain active security in the preprocessing, but the online phase cost is the same, or better.

References

1. Albrecht, M., Grassi, L., Rechberger, C., Roy, A., Tiessen, T.: MiMC: efficient encryption and cryptographic hashing with minimal multiplicative complexity. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 191–219. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53887-6_7
2. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 430–454. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46800-5_17
3. Aliasgari, M., Blanton, M., Zhang, Y., Steele, A.: Secure computation on floating point numbers. In: NDSS 2013. The Internet Society, February 2013
4. Araki, T., Furukawa, J., Lindell, Y., Nof, A., Ohara, K.: High-throughput semi-honest secure three-party computation with an honest majority. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 805–817. ACM Press, October 2016
5. Ben-Efraim, A., Lindell, Y., Omri, E.: Optimizing semi-honest secure multiparty computation for the internet. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 16, pp. 578–590. ACM Press, October 2016
6. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: a framework for fast privacy-preserving computations. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 192–206. Springer, Heidelberg (2008). doi:10.1007/978-3-540-88313-5_13
7. Bogetoft, P., Damgård, I., Jakobsen, T., Nielsen, K., Pagter, J., Toft, T.: A practical implementation of secure auctions based on multiparty integer computation. In: Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 142–147. Springer, Heidelberg (2006). doi:10.1007/11889663_10
8. Burra, S.S., Larraia, E., Nielsen, J.B., Nordholt, P.S., Orlandi, C., Orsini, E., Scholl, P., Smart, N.P.: High performance multi-party computation for binary circuits based on oblivious transfer. Cryptology ePrint Archive, Report 2015/472 (2015). http://eprint.iacr.org/2015/472
9. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, October 2001
10. Carlet, C., Goubin, L., Prouff, E., Quisquater, M., Rivain, M.: Higher-order masking schemes for S-boxes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 366–384. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34047-5_21
11. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999). doi:10.1007/3-540-48405-1_26
12. Coron, J.-S.: Higher order masking of look-up tables. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 441–458. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_25

13. Coron, J.-S., Roy, A., Vivek, S.: Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 170–187. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44709-3_10
14. Coron, J., Roy, A., Vivek, S.: Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. J. Cryptogr. Eng. **5**(2), 73–83 (2015). http://dx.doi.org/10.1007/s13389-015-0099-9
15. Damgård, I., Keller, M., Larraia, E., Miles, C., Smart, N.P.: Implementing AES via an actively/covertly secure dishonest-majority MPC protocol. In: Visconti, I., Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 241–263. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32928-9_14
16. Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical covertly secure MPC for dishonest majority – or: breaking the SPDZ limits. In: Crampton, J., Jajodia, S., Mayes, K. (eds.) ESORICS 2013. LNCS, vol. 8134, pp. 1–18. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40203-6_1
17. Damgård, I., Lauritsen, R., Toft, T.: An empirical study and some improvements of the MiniMac protocol for secure computation. In: Abdalla, M., Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 398–415. Springer, Cham (2014). doi:10.1007/978-3-319-10879-7_23
18. Damgård, I., Nielsen, J.B., Nielsen, M., Ranellucci, S.: Gate-scrambling revisited - or: the TinyTable protocol for 2-party secure computation. Cryptology ePrint Archive, Report 2016/695 (2016). http://eprint.iacr.org/2016/695
19. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_38
20. Damgård, I., Zakarias, R.W.: Fast oblivious AES a dedicated application of the MiniMac protocol. In: Progress in Cryptology - AFRICACRYPT 2016–Proceedings of 8th International Conference on Cryptology in Africa, Fes, Morocco, 13–15 April 2016, pp. 245–264 (2016). http://dx.doi.org/10.1007/978-3-319-31517-1_13
21. Damgård, I., Zakarias, S.: Constant-overhead secure computation of Boolean circuits using preprocessing. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 621–641. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36594-2_35
22. Dessouky, G., Koushanfar, F., Sadeghi, A.R., Schneider, T., Zeitouni, S., Zohner, M.: Pushing the communication barrier in secure computation using lookup tables. In: 24th Annual Network and Distributed System Security Symposium (NDSS 2017). The Internet Society, 26 February–1 March 2017 (to appear). http://thomaschneider.de/papers/DKSSZZ17.pdf
23. Doerner, J., Evans, D., Shelat, A.: Secure stable matching at scale. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 1602–1613. ACM Press, October 2016
24. EMVCo: EMVCo Security QA (2017). https://www.emvco.com/faq.aspx?id=38. Accessed Feb 2017
25. Frederiksen, T.K., Keller, M., Orsini, E., Scholl, P.: A unified approach to MPC with preprocessing using OT. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 711–735. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48797-6_29
26. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 850–867. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_49

27. Grassi, L., Rechberger, C., Rotaru, D., Scholl, P., Smart, N.P.: MPC-friendly symmetric key primitives. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, pp. 430–443. ACM Press, October 2016
28. Ishai, Y., Kushilevitz, E., Meldgaard, S., Orlandi, C., Paskin-Cherniavsky, A.: On the power of correlated randomness in secure computation. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 600–620. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36594-2_34
29. Jha, S., Kruger, L., Shmatikov, V.: Towards practical privacy for genomic computation. In: 2008 IEEE Symposium on Security and Privacy, pp. 216–230. IEEE Computer Society Press, May 2008
30. Keller, M., Orsini, E., Scholl, P.: MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 830–842. ACM Press, October 2016
31. Keller, M., Scholl, P., Smart, N.P.: An architecture for practical actively secure MPC with dishonest majority. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013, pp. 549–560. ACM Press, November 2013
32. Launchbury, J., Diatchki, I.S., DuBuisson, T., Adams-Moran, A.: Efficient lookup-table protocol in secure multiparty computation. In: ACM SIGPLAN International Conference on Functional Programming, ICFP 2012, Copenhagen, Denmark, 9–15 September 2012, pp. 189–200 (2012). http://doi.acm.org/10.1145/2364527.2364556
33. Lindell, Y., Riva, B.: Blazing fast 2PC in the offline/online setting with security for malicious adversaries. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015, pp. 579–590. ACM Press, October 2015
34. Litzenberger, D.C.: Pycrypto - the Python cryptography toolkit (2017). https://www.dlitz.net/software/pycrypto
35. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 681–700. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_40
36. Nielsen, J.B., Schneider, T., Trifiletti, R.: Constant round maliciously secure 2PC with function-independent preprocessing using LEGO. In: 24th NDSS Symposium. The Internet Society (2017). http://eprint.iacr.org/2016/1069
37. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10366-7_15
38. Pulkus, J., Vivek, S.: Reducing the number of non-linear multiplications in masking schemes. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 479–497. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53140-2_23
39. Rindal, P., Rosulek, M.: Faster malicious 2-party secure computation with online/offline dual execution. In: 25th USENIX Security Symposium, USENIX Security 2016, Austin, TX, USA, 10–12 August 2016, pp. 297–314 (2016). https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/rindal
40. Rivain, M., Prouff, E.: Provably secure higher-order masking of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 413–427. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15031-9_28
41. Roy, A., Vivek, S.: Analysis and improvement of the generic higher-order masking scheme of FSE 2012. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 417–434. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40349-1_24

42. Wang, X., Malozemoff, A.J., Katz, J.: Faster two-party computation secure against malicious adversaries in the single-execution setting. In: EUROCRYPT 2017 Proceedings (2017)
43. Wang, X., Ranellucci, S., Katz, J.: Authenticated garbling and communication-efficient, constant-round, secure two-party computation. IACR Cryptology ePrint Archive 2017, 30 (2017). http://eprint.iacr.org/2017/030

Cryptographic Primitives

An Experimental Study of the BDD Approach for the Search LWE Problem

Rui Xu[1(✉)], Sze Ling Yeo[2], Kazuhide Fukushima[1], Tsuyoshi Takagi[3],
Hwajung Seo[2], Shinsaku Kiyomoto[1], and Matt Henricksen[4]

[1] KDDi Research Inc., Fujimino, Japan
ru-xu@kddi-research.jp
[2] Institute for Infocomm Research (I2R), Singapore, Singapore
[3] Institute of Mathematics for Industry (IMI), Kyushu University, Fukuoka, Japan
[4] Huawei Technologies, Singapore, Singapore

Abstract. The proved hardness of the Learning With Errors (LWE) problem, assuming the worst case intractability of classic lattice problems, has made it a standard building block in the recent design of lattice based cryptosystems. Nonetheless, a thorough understanding of the security of these schemes from the perspective of existing attacks remains an open problem. In this manuscript, we report our implementation of the Bounded Distance Decoding (BDD) approach for solving the search LWE problem. We implement a parallel version of the pruned enumeration method of the BDD strategy proposed by Liu and Nguyen.

In our implementation we use the embarrassingly parallel design so that the power of multi-cores can be fully utilized. We let each thread take a randomized basis and perform independent enumerations to find the solution instead of parallelizing the enumeration algorithm itself. Other optimizations include fine-tuning the BKZ block size, the enumeration bound and the pruning coefficients and the optimal dimension of the LWE problem. Experiments are done using the TU Darmstadt LWE challenge. Finally we compare our implementation with a recent parallel BDD implementation by Kirshanova et al. [18] and show that our implementation is more efficient.

Keywords: Learning With Errors · Lattice based cryptography · Security evaluation · Bounded Distance Decoding

1 Introduction

Decades of development in the area of lattice-based cryptography have identified two important primitive hard problems, namely, the Shortest Integer Solution (SIS) problem [1] and the Learning With Errors (LWE) problem [24], to be standard building blocks of modern lattice-based cryptosystems.

In this work, we focus on the LWE problem proposed by Regev [24]. LWE has attracted more and more attention since its proposal. Initially LWE problem was reduced to the GAPSVP (the decision version of the shortest vector problem) or

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 253–272, 2017.
DOI: 10.1007/978-3-319-61204-1_13

SIVP (Shortest Independent Vector Problem) under the quantum setting. This means that LWE is considered hard if there are no algorithms to efficiently solve the GAPSVP or SIVP using a quantum computer. Subsequently, the hardness reduction as been sharpened to accept a classic reduction to these standard lattice problems [8]. As such, LWE-based schemes are widely studied as potential primitives in the post-quantum era.

LWE. Let n be a positive integer, denoting the dimension of the lattice related with the LWE problem, q an odd prime, and let $\mathcal{D}$ be an error distribution over the integer ring modulo q, $\mathbb{Z}_q$. Denote by $\mathbf{s}$ a fixed secret vector in $\mathbb{Z}_q^n$ (in this manuscript we adopt the row vector convention to be consistent with software implementation) selected according to the uniform distribution on its support. Let $\mathcal{L}_{n,q,\mathcal{D}}$ be the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ generated by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing an error e according to $\mathcal{D}$ and returning

$$(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$$

in $\mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\langle \cdot, \cdot \rangle$ denotes the inner product of two vectors in $\mathbb{Z}_q^n$. The search LWE problem is to find the secret vector $\mathbf{s}$ given a fixed number of m samples from $\mathcal{L}_{n,q,\mathcal{D}}$.

Although the intractability of LWE is well established by the reduction proofs, its concrete hardness is far from clear. In this work we follow the approach of Liu and Nguyen [21] to evaluate the performance of the BDD approach for solving LWE problem.

1.1 Our Contribution

In this manuscript, our main contributions include:

- We implement a parallel version of the BDD approach for solving the LWE problem. The implementation features an embarrassingly parallel design where each thread takes a randomized basis and performs an independent enumeration. The advantage of this design is that the power of multi-cores can be fully utilized.
- We give heuristic analysis on how to choose the optimal sub-dimension of the LWE instance. We use the Gaussian heuristic to estimate the cost of a pruned enumeration tree to find better sub-dimension which can reduce the time to solve an LWE instance.
- We compare our implementation with that of Kirshanova et al. [18] and show the advantages of our implementation. Specifically we show that the performance of our parallelization strategy is not limited by Amdahl's Law and the extreme pruning in our implementation brings huge speedup compared with the linear pruning used in the implementation of [18].
- We demonstrate that our implementation solves a couple of instances from the TU Darmstadt LWE challenge.

2 Preliminaries

2.1 Discrete Gaussian Distribution

We first describe the error distribution $\mathcal{D}$ in the LWE problem. In the general situation, any error distribution with small variance is fine for the LWE problem to be hard. However, in this work, similar to many other previous works regarding LWE, we focus on the discrete Gaussian distribution over the ring $\mathbb{Z}_q$ as the error distribution. Let $x \in \mathbb{Z}$. The discrete Gaussian distribution over $\mathbb{Z}$ with mean 0 and width parameter σ, denoted by $D_{\mathbb{Z},\sigma}$ assigns to each $x \in \mathbb{Z}$ the probability proportional to $\exp(-x^2/2\sigma^2)$. The error distribution we consider for the LWE problem is the discrete Gaussian distribution over $\mathbb{Z}_q$, denoted by $D_{\mathbb{Z}_q,\sigma}$, by accumulating the values of the probability mass function over all integers in each residue class mod q. In the original proposal of Regev, the width parameter associated with the moduli q is $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$, where α is the relative error rate. With a slight abuse of notation, we also denote the discrete Gaussian distribution as $\mathcal{D}_{\mathbb{Z}_q,\alpha q}$. When the error distribution of an LWE instance is $\mathcal{D}_{\mathbb{Z}_q,\alpha q}$, we express the LWE instance as $\mathcal{L}_{n,q,\alpha}$.

2.2 Lattice

A lattice in $\mathbb{R}^m$ is a discrete additive subgroup generated by a (non-unique) basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_m)^T$. Equivalently, the lattice $\Lambda(\mathbf{B})$ generated by $\mathbf{B}$ is given by $\Lambda(\mathbf{B}) = \{x | x = \sum_{i=1}^{m} z_i \mathbf{b}_i\}$, where z_i's are integers. Note that by our convention, the vector $\mathbf{b}_i$ in the basis matrix $\mathbf{B}$ is its row vector. The rank of the lattice $\Lambda(\mathbf{B})$ is defined as the rank of the basis matrix $\mathbf{B}$. If the rank of $\Lambda(\mathbf{B})$ equals m, we say that the lattice is full rank. A fundamental notion that lies in various lattice problems is the successive minimal $\lambda_k(\Lambda)$ which is defined to be the smallest real number r such that the lattice contains k linearly independent nonzero vectors of Euclidean length at most r. Specifically, $\lambda_1(\Lambda)$ is the length of the shortest nonzero vector of the lattice Λ.

The lattices we are interested in are a special type of lattices called q-ary lattices which are lattices satisfying $q\mathbb{Z}^m \subset \Lambda \subset \mathbb{Z}^m$. Fix positive integers $n \leq m \leq q$, where n serves as the main security parameter, and q is an odd prime. For any matrix $\mathbf{A} \in \mathbb{Z}^{m \times n}$, define the following two lattices.

$$\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{x}\mathbf{A} = 0 \ \mathtt{mod} \ q\},$$
$$\Lambda_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{x} = \mathbf{A}\mathbf{s} \ \mathtt{mod} \ q \ \mathtt{for\ some} \ \mathbf{s} \in \mathbb{Z}_q^n\}.$$

It is easy to check that both $\Lambda_q^{\perp}(\mathbf{A})$ and $\Lambda_q(\mathbf{A})$ are q-ary lattices [23].

2.3 Lattice Reduction

As we have noticed in Sect. 2.2 that a lattice can be generated from different bases, the property of the basis plays a central role in the difficulty of various hard lattice problems. Informally, the more orthogonal the basis is, the easier the corresponding lattice problems are. As such, many attempts to solve hard lattice problems try to alter (often called reduce in the literature) the given basis in order to get basis which generates the same lattice while at the same time achieves the highest orthogonality possible. We adopt the convention that the first vector $\mathbf{b}_1$ in a reduced basis has the smallest length among the (reduced) basis vectors. After the lattice reduction algorithm, we can use the vector $\mathbf{b}_1$ as an approximation of the shortest vector. Since the determinate of a lattice is invariant under lattice reduction, when the basis is reduced, the length of each basis vector decreases. The common measurement of the quality of a lattice basis is called Hermite factor δ^m defined as: $||\mathbf{b}_1|| = \delta^m \mathtt{vol}(\Lambda)^{1/m}$. We also refer to δ as the root-Hermite factor. A smaller root-Hermite factor typically implies a reduced basis with higher quality.

Lattice reduction algorithms can be viewed as a hierarchy of BKZ [26] based on the parameter blocksize β. The case when $\beta = 2$ is called LLL reduction, which was invented by Lenstra et al. [20]. LLL reduction is proven to run in polynomial time in the lattice dimension and outputs a short vector which is within an exponential factor of the minimal length of a lattice Λ, i.e., $\lambda_1(\Lambda)$. When $\beta = m$, i.e., the full size of the basis, the output basis is HKZ reduced [17] which implies solving the SVP. The situation when k lies in between 2 and m is known as the BKZ-β reduction which is the most referenced reduction algorithm in practice. Chen and Nguyen observed that the running time of BKZ reduction is mainly dominated by the root-Hermite factor δ and is less affected by the dimension m. See Chen and Nguyen [11] for a detailed analysis and their improvements over the standard BKZ as a collection of optimization known as BKZ 2.0. See also Albrecht et al. [2] for a thorough comparison of different estimations of the complexity of BKZ.

2.4 Pruned Enumeration

Gram-Schmidt Orthogonalization. Given a lattice basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_m)^T$, the Gram-Schmidt Orthogonalization of $\mathbf{B}$ is denoted as $\mathbf{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_m^*)$, where $\mathbf{b}_i^*$ is computed as $\mathbf{b}_i^* = \mathbf{b}_i - \sigma_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*$ for $i = 1, \ldots, m$, with $\mu_{i,j} =< \mathbf{b}_i, \mathbf{b}_j^* > / ||\mathbf{b}_j^*||^2$ for all $1 \leq j \leq i \leq m$. Denote by $\pi_i(\cdot)$ the orthogonal projection onto $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1})^{\perp}$. Then $\mathbf{b}_i^* = \pi_i(\mathbf{b}_i)$. Also $\pi_i(\mathbf{\Lambda}(\mathbf{B}))$ is an $(m+1-i)$-dimensional lattice generated by the basis $(\pi_i(b_1), \pi_i(b_2), \ldots, \pi_i(b_{i-1}))^T$.

Lattice Enumeration. Given a target vector $\mathbf{t}$, a lattice basis $\mathbf{B} = (\mathbf{b}_1, \cdots, \mathbf{b}_m)^T$ and a radius R, lattice enumeration algorithm enumerates over all lattice vectors $\mathbf{v} \in \mathcal{L}$ such that $||\mathbf{v} - \mathbf{t}|| \leq R$ and finds the closest one. The enumeration algorithm enumerates over a search tree leveled by the enumeration depth $k \in [m]$. The root of the search is levelled using $k = 0$ and it represents the target vector. And $k = m$ corresponds to the leaves. The nodes at level k of the search tree consist of all vectors $\mathbf{v} \in \mathbf{\Lambda}(\mathbf{B})$ such that $||\pi_{m+1-k}(\mathbf{t} - \mathbf{v})|| \leq R$. Gama et al. [14] use Gaussian heuristic to approximate the number of nodes in the k-th level of the enumeration tree as

$$H_k = \frac{V_k(R)}{\prod_{i=m+1-k}^{n} ||\mathbf{b}_i^*||^2}, \tag{1}$$

where $V_k(R)$ denotes the volume of a k-dimensional ball of radius R. Then the total number of nodes in the enumeration tree is $N = \sum_{k=1}^{m} H_k$.

Gamma et al. also suggest to use extreme pruning to accelerate the enumeration algorithm. The idea of extreme pruning is by deliberately setting the probability that the solution vector is in the tree after pruning to be very small to cut lots of branches in the enumeration tree. Though the success probability of finding the desired solution becomes quite low, it is compensated by the huge reduction of the enumeration time. Their experiments show an exponential speed up over full enumeration. Formally, pruned enumeration bounds the enumeration tree by limiting the k-level nodes to those vectors $\mathbf{v} \in \mathbf{\Lambda}(\mathbf{B})$ such that $||\pi_{m+1-k}(\mathbf{v} - \mathbf{t})|| \leq R_k$ with R_k denoting the pruned radius and $R_1 \leq R_2 \leq \ldots \leq R_m = R$.

3 Related Work

We consider the search version of the LWE problem in this work. There are mainly three ways to solve the search LWE problem.

1. BKW approach: Blum, Kalai and Wasserman proposed BKW algorithm for the LPN (learning with parity noise) problem. Since LWE can be viewed as a generalization of LPN problem, BKW was also adapted to solve LWE by Albrecht et al. [3].
2. Algebraic approach: Arora-Ge [6] proposed to set up a system of algebraic equations over integers to describe the LWE problem and solve the search problem by solving the equation system. Later, this method was improved by using Gröbner basis techniques [4].
3. BDD approach: This approach views the search LWE problem as a decoding problem in a lattice. We will explain this idea in more details in the following.

Bounded Distance Decoding (**BDD**): Given m samples $(\mathbf{a_i}, c_i)$ following the given LWE distribution $\mathcal{L}_{n,q,\mathcal{D}}$, we organize the input into a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ whose rows constitute the m samples of the vector $\mathbf{a}_i$, and a vector $\mathbf{c} \in \mathbb{Z}_q^m$ whose i-th element is c_i from the i-th sample. Note that $\mathbf{c} = \mathbf{As} + \mathbf{e}$, where $\mathbf{e}$ is the error vector which follows the distribution $\mathcal{D}^n$. When the error distribution of the LWE problem is the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}_q,\alpha q}$, we observe that the length of $\mathbf{e}$ is relatively small since each of its entries is distributed according to the discrete Gaussian. Consider the q-ary lattice

$$\Lambda_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{x} = \mathbf{As} \texttt{ mod } q \texttt{ for some } \mathbf{s} \in \mathbb{Z}_q^n\},$$

induced by $\mathbf{A}$. Then the vector $\mathbf{c}$ is bounded in distance from a vector $\mathbf{v} \in \Lambda_q(\mathbf{A})$. Finding the vector $\mathbf{v}$ from the q-ary lattice is called the **BDD** problem.

One approach to solve the BDD problem is to reduce the BDD problem to unique SVP problem by the embedding technique of Kannan [17] and solve the corresponding SVP problem. Another more common approach is to adapt the well-established Babai's algorithm to solve the BDD problem directly. In this regard, Lindner and Peikert [22] proposed a variant of Babai's nearest plane algorithm to solve the BDD problem. Bischof et al. [10] and Kirshanova et al. [18] have implemented parallel versions of this algorithm and investigated its practical performance. Liu and Nguyen [21] observed that Lindner and Peickert nearest plane algorithm can be viewed as a form of pruned enumeration where in the former, the pruning strategy bounds the coefficients instead of the usual way of bounding the projection lengths. Further, they propose to use the lattice enumeration with GNR extreme pruning strategy to accelerate the speed of finding the closest vector. This will be the approach we use in our experimental study. We refer the readers to the excellent survey by Albrecht et al. [2] for a comprehensive exploration of the concrete hardness of LWE.

4 Our Implementation

We choose to implement the Liu and Nguyen algorithm to study its practical performance. This algorithm uses enumeration with extreme pruning to solve the BDD problem. Given M samples $\{(\mathbf{a}_i, c_i)\}_{i=1,2,\ldots,M}$ from the LWE distribution $\mathcal{L}_{n,q,\alpha}$, we use the matrix representation to express the LWE problem as $\mathbf{As}+\mathbf{e} = \mathbf{c}$. We outline the algorithm steps as follows:

Algorithm 1. LWE solver using BDD approach

```
Require: Inputs are $n, M, \alpha, q, \mathbf{A} \in \mathcal{Z}_q^{M\times n}, \mathbf{c} \in \mathcal{Z}_q^M$. The inputs satisfy $\mathbf{As} + \mathbf{e} = \mathbf{c} \bmod q$ for some hidden secret vector $\mathbf{s} \in \mathcal{Z}_q^n$ and error vector $\mathbf{e} \in \mathcal{Z}^M$ with its coefficients chosen independently according to the discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}_q,\alpha q}$.
Ensure: Output the hidden secret vector $\mathbf{s} \in \mathcal{Z}_q^n$ with high probability.
1: Decide an appropriate sub-dimension $\underline{m}$ for the given LWE instance.
2: Choose $m$ random rows from the matrix $\mathbf{A}$ as $\mathbf{A}' \in \mathcal{Z}_q^{m\times n}$ and the corresponding
       $m$ elements from the vector $\mathbf{c}' \in \mathcal{Z}_q^m$.
3: Compute the basis of the $q$-ary lattice $\Lambda_q(\mathbf{A}')$ generated by $\mathbf{A}'$, denote it by $\mathbf{B}$.
4: Choose an appropriate block size $\underline{\beta}$ and use BKZ-$\beta$ to reduce the lattice basis $\mathbf{B}$.
5: Choose an appropriate enumeration radius $\underline{R}$.
6: Compute a set of pruning coefficients and use pruned enumeration to find the
       closest vector $\mathbf{v}$ to the target vector $\mathbf{c}'$ using the enumeration radius $R$.
7: If the closest vector $\mathbf{v}$ can be found, then compute the secret vector $\mathbf{s}$ by solving
       the equation $\mathbf{A}'\mathbf{s} = \mathbf{c}' - \mathbf{v} \bmod q$ and return $\mathbf{s}$. Otherwise, goto Step 2.
```

The red and underlined parameters sub-dimension $\underline{m}$, BKZ block size $\underline{\beta}$ and enumeration radius $\underline{R}$ in Algorithm 1 need to be optimized to achieve better performance for our LWE solver. However, it is easy to see that these parameters affect the running time and success probability of our LWE solver in an entangled way. Thus it is a multi-object optimization problem. In general, it is not trivial to solve such a multi-object optimization problem.

We give detailed explanation of the choice we make regarding each step of the algorithm in subsequent subsections.

4.1 Compute the Basis

This is easy linear algebra. Given a matrix $\mathbf{A}$ of size $m \times n$, we want to find the matrix $\mathbf{B}$ which is the basis of the q-ary lattice $\Lambda_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{x} = \mathbf{A}\mathbf{s} \texttt{ mod } q \texttt{ for some } \mathbf{s} \in \mathbb{Z}_q^n\}$. Recall that we use the convention of row vectors, so a lattice vector generated by the basis $\mathbf{B}$ can be represented by $\mathbf{zB}$ where $\mathbf{z}$ is a row vector. If we forget for a moment that we are working on the modular ring $\mathbb{Z}_q$, then the basis for $\Lambda_q(\mathbf{A})$ is simply $\mathbf{A}^T$, the transpose of $\mathbf{A}$ since all lattice vectors except for those in $q\mathbb{Z}^m$ can be expressed by an integer linear combination of rows in matrix $\mathbf{A}^T$. To include $q\mathbb{Z}^m$ in order to make it q-ary lattice, we further compute the Hermite normal form of $\left[\frac{\mathbf{A^T}}{q\mathbf{I}_m}\right]$ to get the basis of the q-ary lattice $\Lambda_q(\mathbf{A})$. In other words, $\mathbf{B} =$ HNF$(\left[\frac{\mathbf{A^T}}{q\mathbf{I}_m}\right])$, where HNF$(\mathbf{A})$ denotes the row Hermite normal form of a matrix $\mathbf{A}$ removing all zero rows. To see this, first note that $q\mathbb{Z}^m$ itself can be viewed as a lattice with the (m dimensional) identity matrix scaled by q as its basis. Thus, we get $\Lambda_q(\mathbf{A}) = \Lambda(\mathbf{A^T}) \cup \Lambda(q\mathbf{I}_m)$. The last step of our computation relies on the following fact:

Fact 1. Given two lattices $\Lambda(\mathbf{B}_1)$ and $\Lambda(\mathbf{B}_2)$ of the same dimension, the basis for the lattice generated by the union of $\Lambda(\mathbf{B}_1)$ and $\Lambda(\mathbf{B}_2)$ is HNF$(\left[\frac{\mathbf{B}_1}{\mathbf{B}_2}\right])$.

4.2 Enumeration Radius and Basis Randomization

The enumeration radius only affects the running time and success probability of the enumeration part. Consider an LWE instance $\mathcal{L}_{n,q,\alpha}$. Fix m samples from it to get the equation $\mathbf{A}'\mathbf{s} + \mathbf{e}' = \mathbf{c}' \bmod q$. The exact BDD radius is the length of the error vector $\mathbf{e}' \in \mathcal{Z}^m$. Though we do not know the exact value of $||\mathbf{e}'||^2$, we know that its coefficients are generated from the discrete Gaussian distribution $\mathcal{D}_{\mathcal{Z}_q,\alpha q}$. According to the acceptance criteria of the LWE challenge, the requirement is that $||\mathbf{e}'|| \leq 2\sqrt{m}\alpha q$ for an LWE instance $\mathcal{L}_{n,q,\alpha}$ with m samples. Thus one option is to take $R = 2\sqrt{m}\alpha q$ as the **BDD** bound. More generally, we set squared **BDD** bound R^2 as $c \cdot m\alpha^2 q^2$ for some fixed constant c. To approximate the **BDD** bound R, we sampled error vectors according to the discrete Gaussian distribution and record the squared length of the error vector $\mathbf{e}'$. See Fig. 1 for our experiment results. The figure shows the histograms of the distribution of the squared norm of error vector $\mathbf{e}'$ with the bins set as the multiplier between $||\mathbf{e}'||^2$ and $m\alpha^2 q^2$ (i.e., the scalar c). From Fig. 1 we can see that $c = 1.3$ is an appropriate choice for making sure that the closest vector can be found with overwhelming probability and the distribution of the scalar c does not depend on the parameters n, α and m. However, if one chooses $c = 1$ so as to reduce the running time of the enumeration algorithm, then with probability about half, the closest vector can not be found within this radius.

Typical applications of pruned enumeration will first randomize the lattice basis by multiplying the basis matrix with a random unimodular matrix and then apply pruned enumeration to find the desired shortest vector or closest vector. If we adopt this method in our LWE solver, two problems arise.

1. The randomized basis usually has larger entries than the initial basis, thereby adding some burden to the lattice reduction algorithm.
2. If we want to choose a smaller enumeration radius such as setting the scalar c less than 1.3, we might miss the opportunity of finding the closest vector. This is primarily due to the fact that we are working with different bases of a fixed lattice and hence, the error norm is fixed.

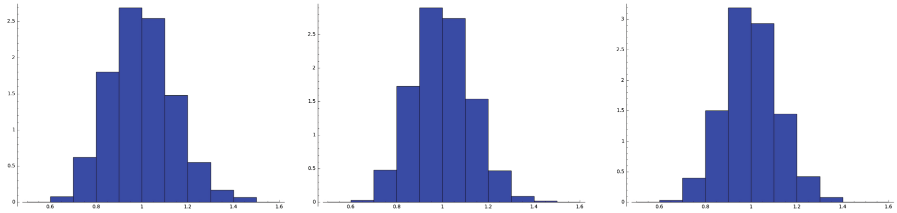
Our randomization is natural and effective. We do not bother to do randomization over a fixed lattice but instead we choose a different lattice each time. In most instances, the number of samples M is larger than the LWE dimension n. It follows that after deciding on a sub-dimension m with $n < m < M$, we can randomly choose m samples from the total M samples to form a different lattice each time. This simple trick solves the two problems discussed above. First the entries of the generated basis are all less than or equal than q. Second, since we randomize over the different m-combinations of the samples, the error vector $\mathbf{e}'$ changes every time. Then we can choose a lower enumeration bound R and be confident that a fixed portion of the trials contribute to error vectors within the bound. For example, according to Fig. 1, if we choose $R^2 = m\alpha^2q^2$ then the closest vector could be found within this bound with probability about 50 %.

Denote by p_{enum} the success probability of an enumeration algorithm given that the length of the error vector $||\mathbf{e}'||$ is indeed within the enumeration bound R. For simplicity we first consider $||\mathbf{e}'|| \leq 1.3{\cdot}m\alpha^2q^2$. Let $T_{enum}(c)$ be the time of the enumeration algorithm when setting the enumeration radius to be $c \cdot m\alpha^2q^2$. We can estimate the total time of enumeration to find the closest vector (when setting $c = 1.3$) as $T(1.3) = T_{enum}(1.3)/p_{enum}$. Further assuming that changing the enumeration radius does not affect p_{enum}, we can approximate the success probability of the enumeration algorithm using different enumeration scalars c. For example if we choose $c = 1$ the probability that the error vector is within $1 \cdot m\alpha^2q^2$ is about 0.5 so we get the total enumeration time of solving the BDD problem as $T(1) = 2{*}T_{enum}(1)/p_{enum}$. In particular, choosing $c = 1$ may lead to a faster algorithm if $T_{enum}(1) < T_{enum}(1.3)/2$. Thus, by analyzing the impact of the enumeration radius on the running time of enumeration algorithm, we can choose nearly optimal enumeration radius. Finally, we can use the Gaussian heuristic Eq. (1) to approximate $T_{enum}(c)$ (see below).

4.3 Choose Sub-dimension

In the typical setting of LWE problem, the number of total samples M is bounded by a polynomial of the LWE dimension n. When treating LWE as a lattice problem, an important decision concerns a suitable choice for the dimension of the lattice. The dimension of the lattice equals the number of samples we choose. How many of the total M samples do we use to form the generating matrix $\mathbf{A}'$?

First, we show that if the sub-dimension were chosen too small, the sub LWE problem may not have a unique solution. Consider the following equation $\mathbf{A}'\mathbf{s} + \mathbf{e}' = \mathbf{c}'$. For any choice of $\mathbf{s}$, we can find an error vector $\mathbf{e}'$ satisfying the above equation. However, the LWE problem restricts the length of $\mathbf{e}'$. More



(a) $n = 40$, $\alpha = 0.005$, $m = 100$. (b) $n = 50$, $\alpha = 0.010$, $m = 120$. (c) $n = 60$, $\alpha = 0.015$, $m = 140$.

Fig. 1. Histograms of square length of $\mathbf{e}'$ for different parameters.

precisely, each element of $\mathbf{e}'$ is chosen from the discrete Gaussian distribution with small variance and thus, it can not be too large. TU Darmstat University has held an LWE challenge website[1] similar to the famous SVP challenge. According to Buchmann et al. [9], the acceptance criteria for the correct answer of the LWE problem $\mathcal{L}_{n,q,\alpha}$ with M samples is that $||\mathbf{e}|| \leq 2\sqrt{M}\alpha q$. Based on this criteria, when we choose the sub-dimension to be m, we would also expect to find a secret vector $\mathbf{s}$ such that it leads to a error vector of length less than $2\sqrt{m}\alpha q$. Following the argument of Buchmann et al. [9], we calculate the probability that the sub LWE problem has more than one solution. For a chosen matrix $\mathbf{A}'$ of size $m \times n$, let $\Lambda_q(\mathbf{A}')$ denote the q-ary lattice generated by $\mathbf{A}'$. Recall that $\lambda_1(\Lambda_q(\mathbf{A}'))$ is the norm of the shortest non zero vector in $\Lambda_q(\mathbf{A})$. Assume that we have two solutions for the secret vector $\mathbf{s}_1$ and $\mathbf{s}_2$ satisfying the criteria $\mathbf{A}'\mathbf{s}_1 + \mathbf{e}'_1 = \mathbf{c}' = \mathbf{A}'\mathbf{s}_2 + \mathbf{e}'_2$ and $\mathbf{e}'_i \leq 2\sqrt{m}\alpha q$. Then by the triangle inequality, we have $||\mathbf{A}'(\mathbf{s}_1 - \mathbf{s}_2)|| \leq 4\sqrt{m}\alpha q$. Since $\mathbf{A}'(\mathbf{s}_1 - \mathbf{s}_2)$ is actually a vector in the q-ary lattice $\Lambda_q(\mathbf{A}')$, the fact that the sub LWE problem has more than one solution implies that $\lambda_1(\Lambda_q(\mathbf{A}')) \leq 4\sqrt{m}\alpha q$. On the other hand, Gaussian heuristic tells us that the expected length of the shortest vector of $\Lambda_q(\mathbf{A}')$ is $q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}$. In view of this, in our implementation we choose the sub-dimension m such that the corresponding Gaussian heuristic $q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}$ is larger than $4\sqrt{m}\alpha q$ so that the expected number of solutions is small.

The next question concerns how large the lattice sub-dimension m should be. Note that a large dimension invariably increases the time for the basis reduction. In [15], the authors experimentally showed that for a random input lattice, the root Hermite factor δ after a BKZ-beta reduction is independent of the lattice dimension. The following table shows the root Hermite factor obtained from various m random samples taken from the LWE challenge with $\alpha = 0.005$ under a BKZ-20 reduction averaged over 20 experiments for each pair of m and n (Table 1).

One sees that for each n, the value of δ is approximately 0.128 for the first values of m but deviates from this value for larger values of m. A closer examination reveals that in the latter case, the shortest vectors produced by the

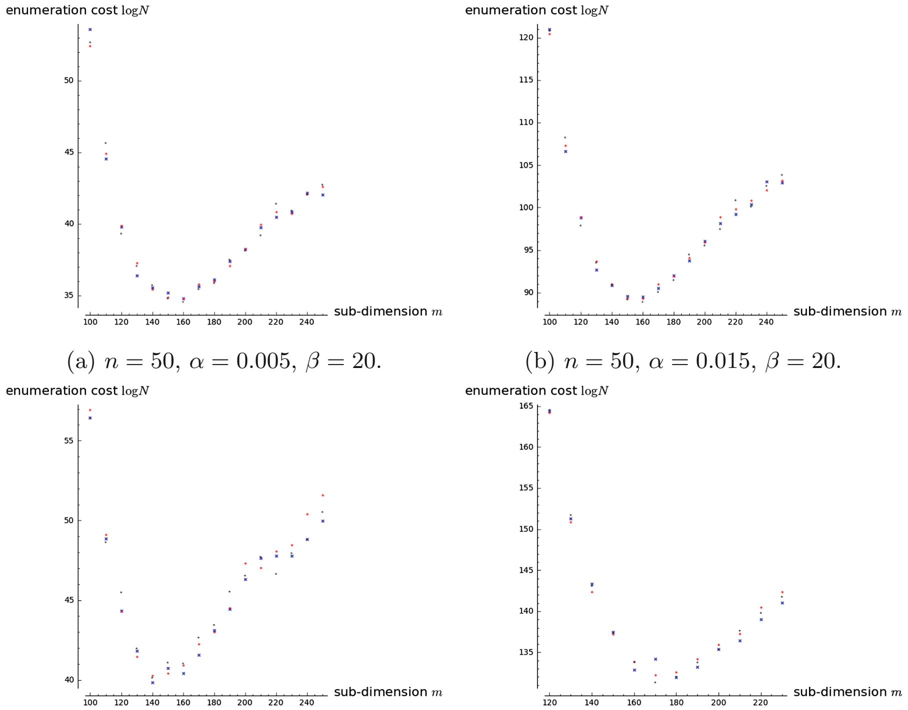
[1] https://www.latticechallenge.org/lwe_challenge/challenge.php.

Table 1. Average root hermite factor for LWE instances with sub-dimension m

n	m										
	100	110	120	130	140	150	160	170	180	190	200
40	1.01272	1.01312	1.01291	1.01289	1.01308	1.01296	1.01160	1.01026	1.00915	1.00821	1.00741
45	1.01290	1.01276	1.01291	1.01301	1.01290	1.01298	1.01309	1.01193	1.01063	1.00954	1.00860
50	1.01282	1.01290	1.01282	1.01291	1.01301	1.01289	1.01309	1.01298	1.01215	1.01090	1.00983
55	1.01285	1.01298	1.01291	1.01291	1.01296	1.01310	1.01296	1.01296	1.01311	1.01229	1.01109
60	1.01281	1.01279	1.01287	1.01291	1.01296	1.01304	1.01301	1.01305	1.01309	1.01312	1.01236
65	1.01286	1.01280	1.01300	1.01304	1.01297	1.01303	1.01298	1.01312	1.01304	1.01322	1.01300

reduction algorithm are the unit vectors scaled by q. In general, we will like to have the lattice reduction to produce vectors with length less than q which tends to suggest that the input vectors are more well-mixed by the reduction algorithm to produce short vectors. In view of this, for a given BKZ-beta reduction, we will select the sub-dimension m such that the predicted shortest vector has length less than q, namely, $\delta^m q^{1-n/m} \leq q$ or, $m \leq \sqrt{n \log q / \log \delta}$, where δ is the expected root Hermite factor. For $\delta = 1.0128$, one checks that this gives the pairs (n, m) to be $(40, 152)$, $(45, 164)$, $(50, 175)$, $(55, 186)$, $(60, 196)$ and $(65, 206)$.

Apart from the lattice reduction, the size of m also affects the enumeration cost. Here, we examine the impact of m for the full enumeration tree. We propose to use the Gaussian heuristic to estimate the enumeration cost, i.e., Eq. (1) to estimate the (full) enumeration cost and to decide the optimal sub-dimension. The total cost is $N = \sum_{k=1}^{m} H_k$. We however do not know how to systematically solve the equation to find an optimal m which minimizes the total cost N. Instead we use numerical calculation to determine the optimal sub-dimension m for fixed BKZ block size β. We plot the total cost $N = \sum_{k=1}^{m} H_k$ for an LWE instance by varying the sub-dimension m. Figure 2 shows the estimated (logarithm of) full enumeration cost for different parameters. We deploy a conversion that for an LWE instance $\mathcal{L}_{n,q,\alpha}$, q is set to be the next prime of n^2 which follows the parameter setting of the LWE Challenge. Comparing Fig. 2a and b, we see that for fixed n and BKZ block size β, the optimal sub-dimension m does not depend on the relative error rate α. Figure 2a and c show the impact of BKZ block size β on the optimal sub-dimension. As we can see, by increasing β, the optimal sub-dimension m also increases and the full enumeration cost decreases for larger β. However, the larger β requires more BKZ reduction time. There is still a need for a trade-off between the BKZ reduction time and enumeration time by setting an appropriate block size β. We discuss this in the next subsection. Finally combining Fig. 2b and d, we can further get the impression that for fixed relative error rate α and BKZ block size β, the larger the dimension of the LWE instance n is, the larger sub-dimension we need to get an optimal performance. One problem of the numerical method to decide on the optimal sub-dimension is that we did not consider the BKZ reduction part. In practice we need to consider the running time of BKZ reduction algorithm, so the actual optimal sub-dimension is usually less than that viewed from the plot. However, the plot can still act as a rough guide to find the optimal sub-dimension. Due to page



(a) $n = 50$, $\alpha = 0.005$, $\beta = 20$.

(b) $n = 50$, $\alpha = 0.015$, $\beta = 20$.

(c) $n = 50$, $\alpha = 0.005$, $\beta = 10$.

(d) $n = 60$, $\alpha = 0.015$, $\beta = 20$.

Fig. 2. Semi-log graph for full cost N. The parameters have the following reference: n is the LWE dimension, α is the relative error rate of the LWE instance and β is the block size for BKZ reduction algorithm used. Different colors stand for different trials. (Color figure online)

constraints, we defer the details of the estimation of enumeration cost and the relation between cost of full enumeration and that of enumeration with extreme pruning in Appendix A.

4.4 Balancing Reduction and Enumeration

Since we use enumeration to solve the BDD problem, we want to first reduce the lattice basis before applying enumeration. BKZ is now the de-facto standard of lattice reduction algorithm in cryptanalysis. We use the BKZ implementation in FPLLL [5] library to perform BKZ reduction.

The quality of the reduced basis and the running time of BKZ reduction algorithm highly depend on the block size β. The choice of an appropriate block size β affects the total running time of our LWE solver. Generally speaking, a larger block size β leads to longer running time of BKZ reduction algorithm but the highly reduced basis will decrease the running time of enumeration. So the folklore is that the optimal block size β should balance the running time

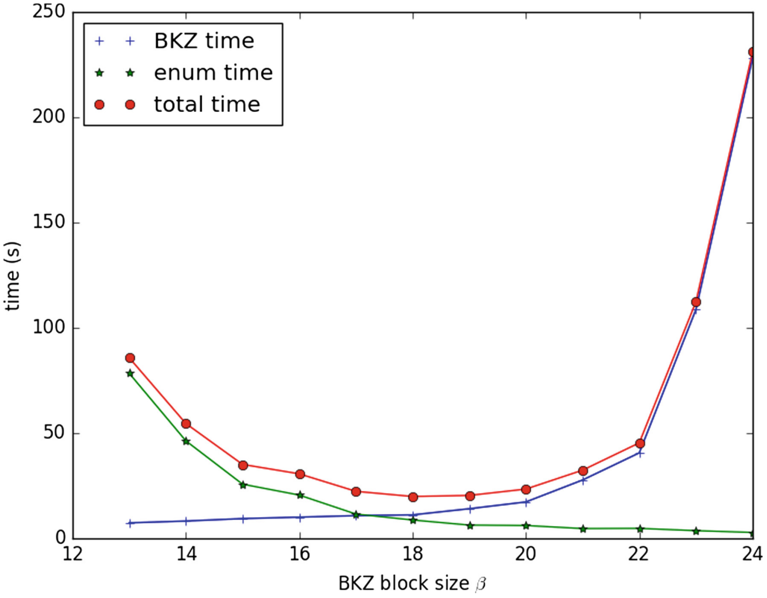


Fig. 3. Running time for BKZ reduction and pruned enumeration.

of BKZ reduction and enumeration. In other words, when the BKZ reduction time and the enumeration time are close to each other, the total running time is minimized. See Fig. 3 for an example. We plot the actual running time of BKZ reduction algorithm and pruned enumeration time for the LWE instance $\mathcal{L}_{40,1601,0.015}$ using sub-dimension $m = 120$. The enumeration radius is set to be $R^2 = 0.8m\alpha^2q^2$. The figure confirms the folklore that the optimal block size β should roughly balance the running time of BKZ reduction and enumeration. However, finding such optimal block size is not easy, especially for extreme pruning. In our experiments we manually tune the block size β by measuring the running time of the BKZ reduction part and pruned enumeration part.

4.5 Parallelization

Parallelism is ubiquitous in today's program design. We have multi-core CPUs even in our laptops. It is natural to implement the LWE solver algorithm in parallel. One option is to use parallel implementation of enumeration algorithm and parallel implementation of lattice reduction algorithm. Alternatively, One can use a sequential implementation of lattice reduction algorithm and enumeration algorithm but instead launch several threads to solve the BDD problem with different randomized basis. We choose the latter approach for its simplicity and its embarrassing parallelism. Although there are parallel implementations of lattice enumeration algorithms [12,13,16,19], we do not know any public available parallel implementation of BKZ reduction algorithm. Thus if we want to use parallel implementation of enumeration we might have to use a sequential implementation of BKZ reduction. Amdahl's Law sets a bound on the potential program speedup defined by the fraction of code (p) that can be parallelized as $speedup = \frac{1}{1-p}$. In using the combination of BKZ reduction and enumeration to solve the SVP or CVP problem, it is common knowledge that when the running time of BKZ reduction part and the enumeration part are roughly equal, the total running time is minimized (refer to the previous section and Fig. 3). If we want optimal performance then the fraction of parallelizable code would be about 1/2. It follows that regardless of how many threads are used, the speedup can be at most 2. We can circumvent this by using a small block size for the BKZ reduction, or plugging in the parallel enumeration into the BKZ reduction, but those methods are either complicated or do not achieve optimal performance gain.

In our implementation we use the embarrassingly parallel design to let each thread work on a different randomized basis, and thus there is no load balance issue. In order to achieve best performance we carefully choose the BKZ block size so that the BKZ reduction time is comparable to the enumeration time.

5 Experimental Results

Our implementation is written in C++, using the library FPLLL for BKZ reduction and lattice enumeration. Our program is compiled using gcc 5.4.0 on a desktop running Ubuntu 14.04 LTS. We test our LWE solver using the instances from the LWE challenge website. We use extreme pruning [14] for lattice enumeration as suggested by Liu and Nguyen [21]. Gamma et al. [14] suggest using numerical approximation to generate optimal pruning coefficients by fixing the successful probability and seeking for minimum overhead. Aono [7] also describes how to compute the optimal pruning coefficients. We follow Aono's approach to compute the optimal pruning coefficients in our implementation. We are preparing to release the source code after further optimization. At this moment, it is available upon request.

5.1 LWE Challenge

TU Darmstadt held a LWE challenge project. The challenge provides LWE instances with different parameters. The LWE challenge instance is identified by two parameters: the LWE dimension n and the relative error rate α. The other parameters of an LWE instance are set as follows:

- Moduli q is set as the next prime of n^2;
- Number of samples is set as $M = n^2$;
- Error distribution is set as the discrete Gaussian distribution with width parameter $\sigma = \alpha q$, i.e., the distribution $\mathcal{D}_{\mathbb{Z}_q,\sigma}$.

Using our implementation described in the preceding section, we solved several instances from the LWE Challenge website. Please refer to Table 2 for the

Table 2. Results on solving some instances from the LWE Challenge website

LWE parameters			BKZ reduction		Enumeration		#Trials	Total time
n	α	m	β	t	c	t		
40	0.005	100	10	-	1.3	-	2	4 s
40	0.01	120	10	4 s	1.3	4 s	2	16 s
40	0.015	120	18	12 s	0.8	10 s	819	18403 s
40	0.02	140	32	10.9 d	1.3	27.1 d	-	38 d
45	0.005	120	5	3 s	1.3	1 s	6	23 s
50	0.005	120	15	5 s	1.0	2 s	5	35 s
60	0.005	140	28	27 h	0.8	24 h	-	51 h

Note that the instances $\mathcal{L}_{40,0.02}$ and $\mathcal{L}_{60,0.005}$ take quite long time thus we use parallelized version of the solving algorithm, and we do not record the number of trials for these two instances.

detailed recording of the LWE parameters, the block size we used for BKZ and the running time for solving these instances. All the instances except two are run using a single thread on a desktop with a 3.60 GHz Intel Core i7 processor with eight cores and 32 GB 1600 MHz DDR3 memory. The instance ($n = 60, \alpha = 0.005$) was run on a cluster consisting of 20 c4.8xlarge instances, each having 36 cores with a 60 GB memory (720 threads in total), on the Amazon EC2 platform. The instance ($n = 40, \alpha = 0.02$) was solved on a cluster consisting of 8 desktops with a 3.60 GHz Intel Core i7 processor with eight cores and 32 GB 1600 MHz DDR3 memory (64 threads in total).

In the experiments we carefully choose the BKZ block size β to ensure the BKZ reduction time is comparable with the enumeration time so as to achieve the reduction on overall running time. Our experiments indeed confirm the folklore that when BKZ reduction time roughly equals that of enumeration time the total running time achieves the minimal. The squared BDD bound R^2 was set as $c \cdot m\alpha^2 q^2$. The successful probability in our pruning strategy is set to be 0.01. From the results of our experiments we find that the relative error rate α plays an important role in the hardness of the LWE problem.

5.2 Comparison with Other Implementations

Recently Kirshanova et al. [18] report a parallel implementation of BDD enumeration for solving LWE. They implement both the Lindner-Peikert [22] nearest planes algorithm and the pruned enumeration method[2] proposed by Liu and Nguyen [21]. They directly implement a pruned parallel enumeration algorithm. Their experiments show that the enumeration algorithm can be nicely parallelized. For example, they achieve a linear speedup by increasing the number of threads even until 10. However, the BKZ reduction is not parallelized. We can observe the impact of Amdahl's Law from their experimental results. For

[2] Kirshanova et al. use linear pruning instead of extreme pruning.

example in order to solve the LWE instance $\mathcal{L}_{80,4093,5}$ their serial implementation needs $4.3 + 13 = 17.3$ h. Their parallel implementation using 10 threads can reduce the enumeration time from 13 h to 1.5 h. Then the total running time is $4.3 + 1.5 = 5.8$ h. That is a 3x speedup by using 10 threads. Even when they increase the number of threads to 20, the total running time is $4.3 + 0.8 = 5.1$ h, which gives a 3.4x speedup by using 20 threads. Although one can circumvent this by using a very small block size for the BKZ reduction part, as we discussed in Sect. 4.4 this choice would increase the total running time of BKZ reduction and pruned enumeration.

On the contrary, our strategy to use extreme pruning and to use many threads working on different basis can scale quite well with respect to the number of threads. Moreover, using extreme pruning can highly reduce the time used by enumeration and thus reduce the total time needed for solving the LWE instance. We compare the running time of our implementation and that of Kirshanova et al. in Table 3. In the table, the time t for BKZ and enumeration stands for the total BKZ time and enumeration time for solving the corresponding LWE instance. In Kirshanova et al.'s setting they fixed the number of samples for the LWE instance and all their experiments use the fixed dimension. We try a different setting where the number of LWE samples are a polynomial of n, say n^2 so that we can use the optimal sub-dimension to reduce the difficulty of the LWE instance.

Table 3. Comparison between Kirshoanova et al. and ours results.

LWE			Kirshanova et al.					Ours implementation					
			Sub-dim	BKZ		Enum		Sub-dim	BKZ		Enum		#Trials
n	q	s	m	β	t	#Treads	t	m	β	t	c	t	
70	4093	6	140	20	65 min	1	44 min	140	20	19 min	1.0	36 s	18
			140	20	65 min	10	5 min	140	15	551 s	1.0	182 s	32
80	4093	5	150	25	4.3 h	1	13 h	150	15	2.8 h	0.8	2.4 h	347
			150	25	1.3 h	10	1.5 h	180	8	6 min	1.0	64 min	12
			150	25	1.3 h	20	50 min	180	10	417 s	1.0	117s	12

The first row of Table 3 shows that extreme pruning can indeed speedup the LWE solver. Kirshanova et al. need 109 min to solve the instance $\mathcal{L}_{70,4093,6}$ on a single thread, while our implementation can solve the instance using the same sub-dimension and block size $\beta = 20$ within 20 min. Further more, their implementation uses 70 min to solve the instance on 10 threads. Since our implementation uses 18 trials to solve the instance we can solve the instance within the time of two rounds if given 10 threads. Basically, we only need less than 4 min to solve the same instance given 10 threads. We note that the block size $\beta = 20$ is not optimal for our implementation. By changing β to 15, we solve the instance $\mathcal{L}_{70,4093,6}$ in 12 min on a single thread.

To further demonstrate the effectiveness of extreme pruning, we compare the performance of both our implementations for the instance $\mathcal{L}_{80,4093,5}$. When we use the same sub-dimension as $m = 150$, the running time of our implementation on a single thread is 5.2 h which is much smaller than 17.3 h of Kirshanova et al. But the advantage of our implementation lies also in another factor. Notice that the algorithm uses 347 trials to find the correct solution, which means a single trial uses on average only 1 min. Kirshanova et al. solve the instance in more than 2 h using 20 threads. Our implementation is expected to solve the instance in 347/20 = 18 min. If we have more than 400 threads, we can solve the instance within 1 min. Moreover, if we apply the optimal sub-dimension trick we do not need so many threads to achieve the speedup. For example when we use 180 samples and BKZ block size $\beta = 10$, the total of 12 trials take $417 + 117 = 534$ s. Then with 12 threads our implementation is expected to solve the instance $\mathcal{L}_{80,4093,5}$ using 45 s. On the contrary, the BKZ reduction of Kirshanova et al.'s implementation alone takes 1.3 h.

6 Conclusion and Future Work

This current work described our choice of strategy to solve the BDD problem, namely the details of our implementation and our experimental results on several LWE challenge instances. Our implementation features a embarrassingly parallel design and the use of extreme pruning shows advantages over existing implementations. Potential future work include:

- We choose the optimal BKZ block size β manually in our experiments. This would be impossible for LWE instances with large dimension and/or large relative error rate. Thus it would be useful to explore the relation between the BKZ reduction time and (pruned) enumeration time and use some heuristics to decide the optimal BKZ block size.
- The success probabilities from our experiments seem to be higher than those estimated by Aono's algorithm, thereby resulting in fewer threads. Since we are using parallel implementations of the LWE solver, we have more room for a lower success probability. Lower successful probability can reduce the running time while we can simply add more threads to compensate the low probability of success. In fact our current environment of 20 c4.8xlarge Amazon EC2 instances contains in total more than 700 threads. We can deal with this problem in two ways: first, we can reduce the successful probability for the pruning strategy; second, we can deploy a two-level parallelization by using the first level to run the LWE solver in parallel and using the second level to run the parallel enumeration algorithm.

A Estimate the Cost of BDD Enumeration

Recall that Gaussian heuristic suggests that the enumeration cost at level k of the full enumeration tree is

$$H_k = \frac{V_k(R)}{\prod_{i=m+1-k}^{m} ||\mathbf{b}_i^*||^2}$$

and the total cost is $N = \sum_{k=1}^{m} H_k$.

In the case of BDD enumeration, we set the enumeration bound $R = \sqrt{m}\alpha q$. As for the Gram-Schmidt vector $\mathbf{b}_i^*$, we use the GSA (Geometric Series Assumption) to approximate their lengths.

Geometric Series Assumption (GSA): Schnorr [25] introduced GSA which states that the Gram-Schmidt lengths $||\mathbf{b}_i^*||$ in a BKZ-reduced basis decrease geometrically with quotient r for some constant r related to the reduction algorithm. Specifically $||\mathbf{b}_{i+1}^*||/||\mathbf{b}_i^*|| = r$. Using Gaussian heuristic for the length of the shortest vector we approximate $||\mathbf{b}_1^*||$ as $\delta^m q^{\frac{m-n}{m}}$ where δ is the root Hermite factor achieved by a BKZ reduction algorithm.

Combining Gaussian heuristic and GSA together, we can reformulate Eq. (1) as

$$H_k = \frac{(\sqrt{m}\alpha q)^k \pi^{k/2}}{r^{(2m-k-1)k/2}\delta^{mk}q^{(m-n)k/m}\Gamma(k/2+1)}, \tag{2}$$

where r is the GSA constant, δ is the root Hermite factor achieved by the reduction algorithm and $\Gamma(\cdot)$ is the Gamma function. The total cost (number of nodes in the enumeration tree) is then calculated as

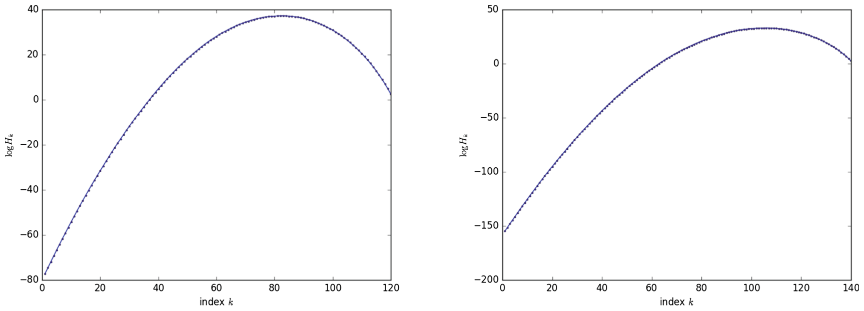
$$N = \sum_{k=1}^{m} H_k. \tag{3}$$

In the BDD case the enumeration cost behaves quite differently from that of the SVP enumeration. In the SVP case, the enumeration radius R is set roughly as $||\mathbf{b}_1^*||$ and the terms H_k reaches maximum when $k = m/2$. But for BDD enumeration, the enumeration radius[3] (bound) is set as $R = \sqrt{m}\alpha q$. This difference actually results in the fact that the cost may decrease as the sub-dimension m increases.

We numerically evaluated the leveled cost H_k and the results are shown in Fig. 4. Figure 4 displays the level cost H_k for different index k for an LWE instance $\mathcal{L}_{50,2503,0.005}$. The basis are reduced by the BKZ-20 reduction algorithm, while the enumeration radius is set as $R = \sqrt{m}\alpha q$. Figure 4a uses sub-dimension $m = 120$ and Fig. 4b uses sub-dimension $m = 140$. From the figures we can observe that the index which maximize H_k is no longer $m/2$ and for the smaller index the value of H_k may be less than 1. We next plot the total cost $N = \sum_{k=1}^{m} H_k$ for an LWE instance by varying the sub-dimension m.

In our experiments, we employ extreme pruning instead of the full enumeration tree. Here, we verify that extreme pruning does not affect the shape of the enumeration cost so that the estimation using full enumeration works effectively for choosing the optimal sub-dimension. Comparing Fig. 5 with Fig. 2, we can see that in the case of enumeration cost with extreme pruning the optimal sub-dimension m is usually a little bigger than that for full enumeration cost. However, the estimated costs do not differ too much. Moreover, when using a larger sub-dimension one has to take the cost of BKZ reduction into consideration. Our

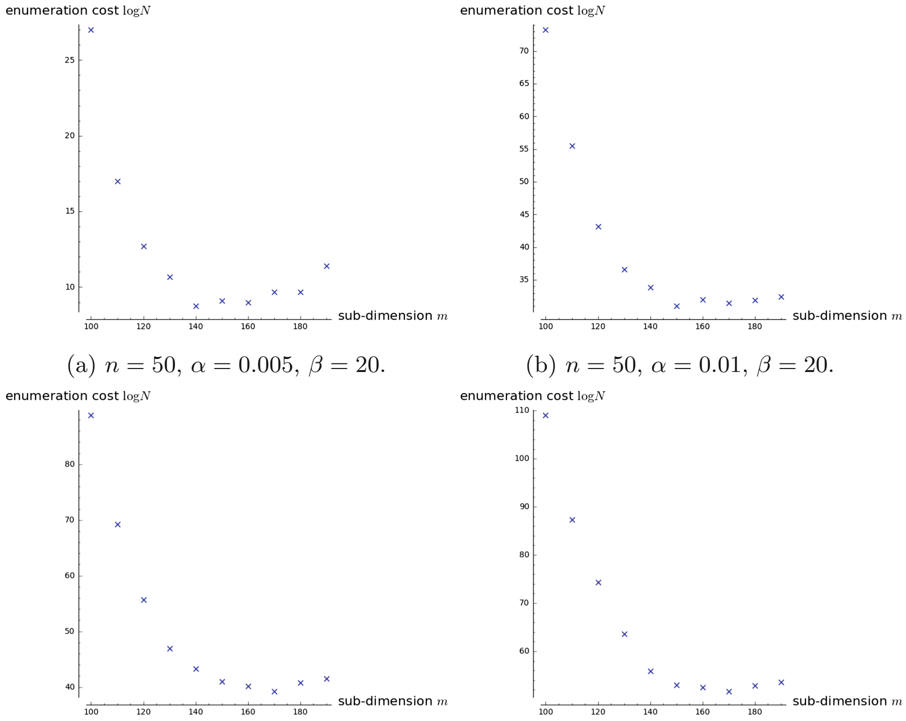
[3] In our implementation we set $R^2 = cm(\alpha q)^2$ for some bound scalar c ranging from 0.8 to 1.3.



(a) $n = 50$, $\alpha = 0.005$, $m = 120$.

(b) $n = 50$, $\alpha = 0.005$, $m = 140$.

Fig. 4. Semi-log graph for level cost H_k.



(a) $n = 50$, $\alpha = 0.005$, $\beta = 20$.

(b) $n = 50$, $\alpha = 0.01$, $\beta = 20$.

(c) $n = 50$, $\alpha = 0.012$, $\beta = 20$.

(d) $n = 50$, $\alpha = 0.015$, $\beta = 20$.

Fig. 5. Semi-log graph for cost N of pruned enumeration. The parameters have the following reference: n is the LWE dimension, α is the relative error rate of the LWE instance and β is the block size for BKZ reduction algorithm used.

experiments in solving the LWE challenge confirms that the estimation using full enumeration cost well serves our purpose to reduce the running time of the whole solving process.

References

1. Ajtai, M.: Generating hard instances of lattice problems. In: Proceedings of 28th Annual ACM Symposium on Theory of Computing, pp. 99–108. ACM (1996)
2. Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046 (2015)
3. Albrecht, M.R., Cid, C., Faugere, J., Fitzpatrick, R., Perret, L.: On the complexity of the BKW algorithm on LWE. Des. Codes Cryptogr. **74**(2), 325–354 (2015)
4. Albrecht, M.R., Cid, C., Faugere, J., Perret, L.: Algebraic algorithms for LWE. Cryptology ePrint Archive, Report 2014/1018 (2014)
5. Albrecht, M.R., Cadé, D., Pujol, X., Stehlé, D.: fplll-4.0, a floating-point LLL implementation. http://perso.ens-lyon.fr/damien.stehle
6. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6755, pp. 403–415. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22006-7_34
7. Aono, Y.: A faster method for computing Gama-Nguyen-Regev's extreme pruning coefficients (2014). arXiv preprint arXiv:1406.0342
8. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: Proceedings of 45th Annual ACM Symposium on Theory of Computing, STOC 2013, pp. 575–584. ACM, New York (2013)
9. Buchmann, J., Büscher, N., Göpfert, F., Katzenbeisser, S., Krämer, J., Micciancio, D., Siim, S., van Vredendaal, C., Walter, M.: Computation, creating cryptographic challenges using multi-party: the LWE challenge. In: Proceedings of 3rd ACM International Workshop on ASIA Public-Key Cryptography (AsiaCCS 2016), pp. 11–20 (2016)
10. Bischof, C., Buchmann, J., Dagdelen, Ö., Fitzpatrick, R., Göpfert, F., Mariano, A.: Nearest planes in practice. In: Ors, B., Preneel, B. (eds.) BalkanCryptSec 2014. LNCS, vol. 9024, pp. 203–215. Springer, Cham (2015). doi:10.1007/978-3-319-21356-9_14
11. Chen, Y., Nguyen, P.Q.: BKZ 2.0: better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25385-0_1
12. Detrey, J., Hanrot, G., Pujol, X., Stehlé, D.: Accelerating lattice reduction with FPGAs. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 124–143. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14712-8_8
13. Dagdelen, Ö., Schneider, M.: Parallel enumeration of shortest lattice vectors. In: D'Ambra, P., Guarracino, M., Talia, D. (eds.) Euro-Par 2010. LNCS, vol. 6272, pp. 211–222. Springer, Heidelberg (2010). doi:10.1007/978-3-642-15291-7_21
14. Gama, N., Nguyen, P.Q., Regev, O.: Lattice enumeration using extreme pruning. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 257–278. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_13
15. Gama, N., Nguyen, P.Q.: Predicting Lattice Reduction. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_3

16. Hermans, J., Schneider, M., Buchmann, J., Vercauteren, F., Preneel, B.: Parallel shortest lattice vector enumeration on graphics cards. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 52–68. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12678-9_4
17. Kannan, R.: Minkowski's convex body theorem and integer programming. Math. Oper. Res. **12**(3), 415–440 (1987)
18. Kirshanova, E., May, A., Wiemer, F.: Parallel implementation of BDD enumeration for LWE. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 580–591. Springer, Cham (2016). doi:10.1007/978-3-319-39555-5_31
19. Kuo, P.-C., Schneider, M., Dagdelen, Ö., Reichelt, J., Buchmann, J., Cheng, C.-M., Yang, B.-Y.: Extreme enumeration on GPU and in clouds. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 176–191. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23951-9_12
20. Lenstra, A., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Math. Ann. **261**(4), 515–534 (1982)
21. Liu, M., Nguyen, P.Q.: Solving BDD by enumeration: an update. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 293–309. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36095-4_19
22. Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19074-2_21
23. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Post-Quantum Cryptography, p. 147 (2009)
24. Regev, O.: On lattices, learning with errors, random linear codes, cryptography. J. ACM (JACM) **56**(6), 34:1–34:40 (2009)
25. Schnorr, C.P.: Lattice reduction by random sampling and birthday methods. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 145–156. Springer, Heidelberg (2003). doi:10.1007/3-540-36494-3_14
26. Schnorr, C.P., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Program. **66**(1–3), 181–199 (1994)

Efficiently Obfuscating Re-Encryption Program Under DDH Assumption

Akshayaram Srinivasan[1(✉)] and Chandrasekaran Pandu Rangan[2]

[1] University of California, Berkeley, USA
akshayaram@berkeley.edu, prangan55@gmail.com
[2] Indian Institute of Technology, Madras, India

Abstract. The re-encryption functionality transforms a ciphertext encrypted under a public key pk_1 to a ciphertext of the same message encrypted under a different public key pk_2. Hohenberger et al. (TCC 2007) proposed a pairing-based obfuscator for the family of circuits implementing the re-encryption functionality under a new notion of obfuscation called as *average-case secure obfuscation*. Chandran et al. (PKC 2014) proposed a lattice-based construction for the same.

The construction given by Hohenberger et al. could only support polynomial sized message space and the proof of security relies on strong assumptions on bilinear groups. Chandran et al.'s construction could only satisfy a relaxed notion of correctness.

In this work, we propose a *simple* and *efficient* obfuscator for the re-encryption functionality that satisfies the strongest notion of correctness, supports encryption of messages from an exponential sized domain and relies on the *standard* DDH-assumption. This is the *first* construction that does not rely on pairings or lattices. All our proofs are in the standard model.

Keywords: Re-encryption functionality · Average-case secure obfuscation · DDH assumption · Standard model

1 Introduction

The goal of program obfuscation [2,22] is to make a program "unintelligible" while preserving its functionality. This security goal was formalized in the seminal work of Barak et al. [2]. The strongest and the most intuitive notion of

A. Srinivasan—Work partially done while author was a student at IIT-Madras. Research supported in part from 2017 AFOSR YIP Award, DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, NSF CRII Award 1464397, and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley) of Prof. Sanjam Garg. The views expressed are those of the author and do not reflect the official policy or position of the funding agencies.

C. Pandu Rangan—Research partially supported by Information Security Education and Awareness Program of Ministry of Information Technology, Government of India and Microsoft Research, Bangalore, India.

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 273–292, 2017.
DOI: 10.1007/978-3-319-61204-1_14

security considered in their work is known as the *predicate black-box* (also known as virtual black-box) obfuscation. This definition requires any predicate that is efficiently computable given access to the obfuscated program must also be efficiently computable given black-box access to the same program. Unfortunately, Barak et al. showed that a *general* program obfuscator satisfying the predicate black-box property does not exist. In spite of the general impossibility result, program obfuscators satisfying this strong security notion have been constructed for simple function families in [10,12,29].

Average-Case Secure Obfuscation. Hohenberger et al. [24] noted that the predicate black-box property does not give meaningful security guarantees when the obfuscated functionality (like re-encryption) is a part of a larger cryptographic system (like the underlying encryption scheme). They discussed a scenario where having access to the obfuscated program could compromise the security of the larger cryptographic system even if the obfuscator satisfied the predicate black-box property.

To address this issue, Hohenberger et al. proposed a new definition of obfuscation which they termed as *average-case secure obfuscation.* This definition guarantees that any adversary against a cryptographic scheme having access to an obfuscated program can be transformed into an adversary with only black box access to the program given that the cryptographic scheme has *distinguishable attack property.* Informally, a cryptographic scheme is said to have distinguishable attack property if there exists a distinguisher that can "test" if a given algorithm can break the security of the scheme. Hohenberger et al. in fact showed that several natural cryptographic functionalities like semantically secure encryption and re-encryption have this property.

1.1 Prior Work

A cryptographic functionality that has been shown to be obfuscatable under the average-case security notion is the re-encryption function. The re-encryption functionality transforms a ciphertext encrypted under a public key pk_1 to a ciphertext of the same message encrypted under a different public key pk_2. Hohenberger et al. [24] designed an average case secure obfuscator for the re-encryption functionality under Decision Linear and a variant of 3-party Decisional Diffie-Hellman assumptions. However, this construction could only *support a message space of polynomial size.* The decryption algorithm in their work performs an exhaustive search on the message space by computing a pairing operation on each message and tests the output of the pairing against a specific value. Thus, it has to *compute a polynomial number of pairing operations in the worst case.* Moreover, the security of their construction is based on a strong assumption, namely, Strong 3-party DDH.

Remark 1. We note that it is possible to extend the system of [24] to a message space of arbitrary size by using their construction for the boolean space $\{0, 1\}$. For an arbitrary message space $\mathcal{M}$, one can encrypt each message bit by bit and thus incurring a $O(\log(|\mathcal{M}|))$ overhead on encryption and decryption. For an exponential (in the security parameter λ) sized message space, the overhead on encryption and decryption algorithms would be $\texttt{poly}(\lambda)$. But it is desirable to have a system that performs constant number of operations (i.e., have a constant overhead) in every algorithm.

Chandran et al. [13] designed average case secure obfuscators for the re-encryption circuit assuming interactability of certain lattice problems. Their construction could only satisfy certain relaxed notions of correctness. In particular, they considered three relaxations of the correctness property. The first relaxation guarantees that the output of the original circuit and the obfuscated circuit are statistically close only on a subset of the actual inputs. The next relaxation guarantees that the output of the obfuscated program on a subset of inputs is correct with respect to some algorithm (like decryption). The final relaxation guarantees that the output of the obfuscated circuit and the original circuit are computationally indistinguishable. A natural question that arises from the prior work is:

> *Is there an efficient obfuscator for the re-encryption program under milder assumptions that satisfies the strongest notion of correctness, has a constant overhead in every algorithm and supports an exponential sized message space?*

1.2 Our Contributions

We highlight the main contributions of this work.

Main Result. In this work, we propose a new encryption - re-encryption system that supports encryption of messages from an exponential (in security parameter) space, involves a constant number of group exponentiation operations in all algorithms. We also design an average-case secure obfuscator for the re-encryption program which achieves the strongest notion of correctness (as in [23,24]). We prove the average case secure obfuscator property of our obfuscator and the security of our encryption - re-encryption system under the standard DDH assumption. All our proofs are in the *standard model.* Informally, the main result in this work is:

Informal Theorem 1. *Under the DDH-assumption, there exists an average-case secure obfuscator for the family of circuits implementing the re-encryption functionality.*

Remark 2. We observe that our construction of obfuscator for the re-encryption program is not secure when the holder of sk_2 has access to the obfuscated circuit. This is the case with all prior constructions of average-case secure obfuscators for re-encryption [13,24] as well as encrypted signatures [23]. The construction is secure as long as the obfuscated circuit is run by some (possibly malicious) party other than delegatee. It would be interesting to investigate the possibility of constructing average-case secure obfuscators which have "insider security." That is, they remain secure even when the delegatee has access to the obfuscated circuit. We leave this as an open problem.

Strengthening the Black-Box Security Model. Recall that in the average-case obfuscation paradigm, we must first design an encryption - re-encryption system that is secure against adversaries that have black box access to the re-encryption program. The security model considered by Hohenberger et al. [24] for this purpose is as follows: the challenger samples two public key-secret key pairs (pk_1, sk_1) and (pk_2, sk_2) and then provides pk_1, pk_2 to the adversary. The adversary (with oracle access to re-encryption program from pk_1 to pk_2) chooses two messages m_0 and m_1 and also gives information about the public key as well as the ciphertext level on which it wishes to be challenged. More precisely, the adversary can choose either to be challenged on a first level ciphertext[1] under pk_1 or a first level ciphertext under pk_2 or a second level ciphertext[2] under pk_2. The scheme is secure if the adversary is unable to distinguish between the corresponding encryptions of m_0 and m_1.

We observe that the above security model is insufficient in capturing the full security notion of encryption - re-encryption system (See Remark 4). In particular, the above security model allows the following trivial but insecure encryption - re-encryption system to be secure. Consider any semantically secure encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$. To obtain a first level ciphertext of a message m under a public key pk, run the $\mathsf{Encrypt}$ algorithm on m and pk. The re-encryption program from pk_1 to pk_2 has sk_1, pk_1 and pk_2 hardwired into its description. When it is run with a first level ciphertext $c \leftarrow \mathsf{Encrypt}(m, pk_1)$, it decrypts the ciphertext using sk_1 and outputs $(\mathsf{Encrypt}(m, pk_1)||\mathsf{Encrypt}(sk_1, pk_2))$ where $||$ denotes concatenation. In order to decrypt a second level ciphertext, one can first decrypt the second component using sk_2 to obtain sk_1 and then decrypt the first component to obtain m. This system has an obvious drawback as it reveals sk_1 to the user with secret key sk_2. But one can prove that this system is secure under the security model considered in [24]. We also observe that it is possible to construct an average-case secure obfuscator for the above re-encryption program when one instantiates Π with a semantically secure encryption system which allows re-randomization of ciphertexts (e.g., the standard El-Gamal encryption).

To address this issue, we strengthen the security model for encryption - re-encryption system as follows. We consider the security of the system under two different security games.

1. The first game called as *Original Ciphertext Security* proceeds exactly as in [24] but the adversary is either challenged on a first level ciphertext under pk_1 or a first level ciphertext under pk_2.

[1] A first level ciphertext is the one which has not been re-encrypted. In other words, a first level ciphertext is given as input to the re-encryption program.

[2] A second level ciphertext under pk_2 is the output of re-encryption program from pk_1 to pk_2 on a first level ciphertext under pk_1.

2. In the second game called as the *Transformed Ciphertext Security*, in addition to (pk_1, pk_2) the adversary is also provided with sk_1. In the challenge phase, the adversary obtains a second level ciphertext under pk_2 as the challenge ciphertext. The goal of the adversary in both the games is to distinguish between encryptions of messages from encryptions of junk values.

Additionally, we require the encryption - re-encryption system to satisfy a special property called as *statistical independence.* Statistical independence requires that the output distribution of the re-encryption program (i.e., the distribution of the second level ciphertext) to be *statistically independent* of sk_1.

Note that the above trivial encryption - re-encryption system is not transformed ciphertext secure as the adversary with access to sk_1 can directly decrypt the first component of the challenge ciphertext to obtain the hidden message. We also stress that statistical independence property guarantees that the second level ciphertext cannot "leak" any information (in an information theoretic sense) regarding sk_1. This in particular, disallows other contrived examples which may reveal the secret key sk_1 to Bob but possibly is still transformed ciphertext secure.[3]

Remark 3. Though this security model was not explicitly considered, all prior works [13,14,24] satisfy this security notion.

Remark 4. We consider a stronger model for encryption - re-encryption security since the output of the re-encryption program (which we obfuscate in this work) does not have the same probability distribution as a fresh encryption of the message m under pk_2 (i.e., as an output of another encryption algorithm $\mathsf{Encrypt2}(m, pk_2)$ as in [24]). If the output was distributed identically to a fresh encryption under pk_2 then the security model given by Hohenberger et al. is sufficient for our purposes.[4] The above discussion regarding the issues with the security model is for the generalized case where the output distribution of the re-encryption program and distribution of a freshly encrypted ciphertext under pk_2 are not identical.

1.3 Related Work

Proxy Re-Encryption. A paradigm in cryptography which is closely related to re-encryption is *proxy re-encryption.* In a proxy re-encryption system, a semi-trusted proxy transforms ciphertexts intended for Alice (delegator) to a ciphertext of the same message for Bob (delegatee). Specifically, Alice provides the

[3] For example, we could consider the output of the re-encryption functionality to be $(\mathsf{Encrypt}(C, pk_2)||\mathsf{Encrypt}(sk_1, pk_2))$ where C is the input ciphertext. This is transformed ciphertext secure. However, the output of the re-encryption functionally is dependent on sk_1.

[4] The counter example discussed above will not work since the output of the re-encryption program is not identically distributed to a freshly generated ciphertext under pk_2.

proxy with a re-key $RK_{A \to B}$ which is a function of her secret key sk_1 and Bob's public key pk_2. The proxy runs a specific algorithm (called as the *re-encryption algorithm* in literature) which takes the ciphertext encrypted under Alice's public key and the re-key and outputs a ciphertext under Bob's public key. A (non-exhaustive) list of proxy re-encryption schemes under different notions of security can be found in [1,5,11,15,25]. The security goal is that given the re-encryption key, the proxy cannot learn any information about the underlying message from the first level ciphertext. This is formalized as a game between the challenger and the proxy. This must be contrasted with the simulation based security guarantee provided by obfuscation of re-encryption program. In particular, obfuscation of re-encryption circuit guarantees that no *non-black-box* information about the re-encryption circuit is "leaked" to the proxy. On the other hand, it is not directly evident if such guarantees can be given from proxy re-encryption systems as it may be possible that the re-key could leak some non-black-box information. For example, the re-key could reveal a function of the secret key that still does not contradict the semantic security of the encryption scheme.

FHE. Fully Homomorphic Encryption (FHE) [26] allows arbitrary computations to be performed on a ciphertext. The first construction of FHE was provided in the breakthrough work by Gentry [20]. Following this result, there have been several works constructing FHE from worst-case intractability of several lattice problems [7–9]. We note that using re-randomizable (or circuit private) FHE, it is possible to obfuscate the re-encryption program. However, all known constructions of FHE are from specific assumptions on lattices and there is still someway to go before they become "truly" practical. In contrast, we propose an obfuscation of re-encryption program from the standard DDH assumption and our construction is very efficient as it involves only a (small) constant number of group exponentiation operations.

Indistinguishability Obfuscation. Since the strongest notion of program obfuscation (namely, predicate black-box obfuscation) was shown to be impossible, Barak et al. [2] proposed a weakened notion of obfuscation called as *indistinguishability obfuscation* or iO. Indistinguishability obfuscation guarantees that for any two functionally equivalent circuits having the same size, obfuscations of the circuits are indistinguishable. The first candidate construction of iO was given in the recent breakthrough work of Garg et al. [17]. Subsequently, several cryptographic primitives like functional encryption [17], deniable encryption [27], non-interactive key exchange without a trusted setup [6], two-round multiparty computation protocols [16], hard instances of the complexity class PPAD [3,18] and trapdoor permutations [4,19] (to name a few) were constructed from iO and other assumptions like one-way functions. We note that indistinguishability guarantee provided by iO is strictly weaker than the security guarantee needed in this work. In addition, our goal is to obfuscate a specific functionality namely, the re-encryption functionality.

2 Preliminaries

A function $\mu(\cdot) : \mathbb{N} \rightarrow \mathbb{R}^+$ is said to be **negligible**, if for every positive polynomial $p(\cdot)$, there exists an N such that for all $n \geq N$, $\mu(n) < 1/p(n)$. Given a probability distribution D on a universe U, we denote $x \leftarrow D$ as the process of sampling an element x from U according to the distribution D. Given a finite set X, we use the notation $x \overset{\$}{\leftarrow} X$ for denoting the process of sampling x from the set X uniformly. If two probability distributions D and D' defined on a set X are identical, we denote it by $D \approx D'$. We use notation similar to [23] to denote the following randomized process: given n probability distributions $D_1, \cdots, D_n$, let $\{x_1 \leftarrow D_1; \cdots; x_n \leftarrow D_n : f(x_1, \ldots, x_n)\}$ be the probability distribution of a (possibly randomized) function f. PPT machines refer to Probabilistic Polynomial Time Turing machines. All PPT machines run in time polynomial in the security parameter denoted by λ. We consider the non-uniform model of computation to model the adversaries. These machines take an additional auxiliary input z of length polynomial in λ. If p is a prime number then let $\mathbb{Z}_p^*$ denote the set $\{1, 2, \ldots, p-1\}$.

We assume familiarity with the notion of computational indistinguishability and statistical distance (a.k.a. variation distance) and skip the standard definitions. We state the following simple lemma regarding statistical distance. The proof can be derived directly from the definition.

Lemma 1. *For all distributions X_n and Y_n, for all PPT distinguishers D that output a single bit and for all $z \in \{0,1\}^{poly(n)}$ we have,*

$$\Delta(D(X_n, z), D(Y_n, z)) = \left|Pr[b \leftarrow D(X_n, z) : b = 1] - Pr[b \leftarrow D(Y_n, z) : b = 1]\right|$$

The above lemma implies that $\{X_n\}_n \overset{c}{\approx} \{Y_n\}_n$ if and only if $D(X_n, z)$ and $D(Y_n, z)$ are statistically close for all PPT distinguishers D and for all auxiliary input z.

We now recall the Decisional Diffie-Hellman (DDH) assumption on prime order groups. Let Gen be an algorithm which takes 1^λ as input and randomly generates the parameters $(p, \mathbb{G}, g)$ where p is a λ bit prime, $\mathbb{G}$ is a multiplicative group of order p and g is a generator for $\mathbb{G}$.

Definition 1 (DDH assumption). *The DDH assumption states that the following distribution ensembles are computationally indistinguishable:*

$$\{(p, \mathbb{G}, g) \leftarrow \mathsf{Gen}(1^\lambda); a, b \overset{\$}{\leftarrow} \mathbb{Z}_p^* : (g, g^a, g^b, g^{ab})\}_\lambda \overset{c}{\approx}$$
$$\{(p, \mathbb{G}, g) \leftarrow \mathsf{Gen}(1^\lambda); a, b, c \overset{\$}{\leftarrow} \mathbb{Z}_p^* : (g, g^a, g^b, g^c)\}_\lambda$$

We assume the reader's familiarity with the syntax and security notion (multi message security) for a Public Key Encryption (PKE) system. We also assume familiarity with the El-Gamal encryption system. We recall the theorem regarding the multi-message security of El-Gamal encryption system and its variant.

Theorem 1 (Multi message security). *Assuming the DDH-assumption holds in the group $\mathbb{G}$, the El-Gamal encryption system is multi-message secure.*

We assume familiarity with of the concept of Pseudo Random Generator (PRG) and refer the reader to [21] for a formal definition.

2.1 Average-Case Secure Obfuscation

Let $\mathcal{C} = \{C_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of polynomial sized circuits. For a length parameter λ, let C_λ be the set of circuits in $\mathcal{C}$ with input length $p_{in}(\lambda)$ and output length $p_{out}(\lambda)$ where $p_{in}(\cdot)$ and $p_{out}(\cdot)$ are polynomials. The circuit family $\mathcal{C}$ has an associated sampling algorithm Samp which takes 1^λ as input and outputs a circuit C chosen uniformly at random from C_λ. We also assume that there exists efficient $(\mathsf{Encode}, \mathsf{Decode})$ algorithms which encodes and decodes a given circuit C into binary strings when used as input/output of Turing machines. We make implicit use of such encoding and decoding algorithms and do not mention them explicitly.

We use similar notations (with some minor changes) as in [23] to denote probabilistic circuits. A probabilistic circuit $C(x; r)$ takes two inputs. The first input is called as the *regular input* and the second input is termed as the *random input.* The output of a probabilistic circuit on a regular input (denoted by $C(x; \cdot)$) can be viewed as a probability distribution where the randomness in the distribution comes from the random choice of r. We say that a machine $\mathcal{A}$ has *oracle access* to a probabilistic circuit C (denoted by $\mathcal{A}^{\mathcal{O}(C)}$) if during the oracle queries, $\mathcal{A}$ can only specify the regular input x to the circuit and the random input r is chosen uniformly at random from the corresponding sample space by the oracle $\mathcal{O}$. The output of a probabilistic machine $\mathcal{A}$ having oracle access to a probabilistic circuit C (denoted by $\mathcal{A}^{\mathcal{O}(C)}(x_1, \cdots, x_n)$) is a probability distribution where the randomness in the distribution comes from the random coins used by $\mathcal{A}$ as well as the random coins used by $\mathcal{O}$ in answering $\mathcal{A}$'s oracle queries. We say that $\mathcal{B}$ *evaluates* a probabilistic circuit C (or in other words, $\mathcal{B}$ is an evaluator of C) on regular input x, if $\mathcal{B}$ supplies the regular input as well as the random input r chosen uniformly at random from the corresponding sample space and outputs $C(x; r)$. We use $|C|$ to denote the size of a circuit C.

We recall the notion of *average case secure obfuscation* given in [24].

Definition 2 [23,24]**.** *A PPT machine* Obf *that takes as input a (probabilistic) circuit and outputs a new (probabilistic) circuit is an average-case secure obfuscator for the circuit family* $\mathcal{C} = \{C_\lambda\}_{\lambda \in \mathbb{N}}$ *with an associated sampling algorithm* Samp *if it satisfies the following properties:*

1. *Preserving Functionality: For all length parameter* $\lambda \in \mathbb{N}$ *and for all* $C \in C_\lambda$*:*

$$Pr[C' \leftarrow \mathsf{Obf}(C) : \exists x \in \{0,1\}^{p_{in}(\lambda)}, \Delta\big(C'(x; \cdot), C(x; \cdot)\big) \neq 0] = 0$$

2. *Polynomial Slowdown: There exists a polynomial* $p(\cdot)$ *such that for sufficiently large length parameters* λ*, for any* $C \in C_\lambda$*, we have*

$$Pr[C' \leftarrow \mathsf{Obf}(C) : |C'| \leq p(|C|)] = 1$$

3. *Average-case Secure Virtual Black Box: There exists a PPT machine (simulator)* Sim *such that for every PPT distinguisher D, there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that for every length parameter λ and for every $z \in \{0,1\}^{poly(\lambda)}$:*

$$|Pr[C \leftarrow \mathsf{Samp}(1^\lambda); C' \leftarrow \mathsf{Obf}(C); b \leftarrow D^{\mathcal{O}(C)}(C', z) : b = 1] -$$
$$Pr[C \leftarrow \mathsf{Samp}(1^\lambda); C' \leftarrow \mathsf{Sim}^{\mathcal{O}(C)}(1^\lambda, z); b \leftarrow D^{\mathcal{O}(C)}(C', z) : b = 1]| \leq \mathsf{neg}(\lambda)$$

Remark 5. The definition given in [24] considers a relaxed notion of correctness. Specifically, it allows a the output distribution of the obfuscated circuit and the original circuit to have a negligible statistical distance with a negligible probability. Here, as given in [23], we consider a stronger notion of correctness where we require that the output distribution of the original circuit and the obfuscated circuit to be identical.

3 Obfuscator for Re-Encryption Functionality

In this section, we describe our new encryption system, the re-encryption functionality which is to be obfuscated and finally the construction of an average case secure obfuscator for the functionality.

New Encryption Scheme. The new encryption system under consideration is a variant of the El-Gamal system.

New Encryption Scheme

- $\mathsf{Setup}(1^\lambda)$: Let $(p, \mathbb{G}, g) \leftarrow \mathsf{Gen}(1^\lambda)$. Let H be a pseudo random generator which takes as input an element from $\mathbb{G}$ and outputs an element in $\mathbb{Z}_p^*$.[a] Output the public parameters as $params = (p, g, \mathbb{G}, H)$ with message space $\mathcal{M} = \mathbb{G}$.
- $\mathsf{KeyGen}(1^\lambda, params)$: Choose $x \xleftarrow{\$} \mathbb{Z}_p^*$ and set the public key pk to be (g, g^x) and the secret key $sk = x$.
- $\mathsf{Encrypt1}(m, pk)$: Parse pk as (g, g^x). Choose a random $r \xleftarrow{\$} \mathbb{Z}_p^*$ and output $(m \cdot g^r, (g^x)^r)$.
- $\mathsf{Decrypt1}(sk, [C_1, C_2])$: Parse the secret key sk as x. Output $m = (C_1) \cdot ((C_2)^{1/x})^{-1}$.

[a] The standard definition of pseudo random generator assumes the domain and the range to be bit strings. We note that it can be extended to any domain and range assuming efficient encoding and decoding functions from the domain to bit strings and from bit strings to range. The expansion factor of H depends on the actual encoding and decoding schemes used.

Re-encryption Functionality. Let (pk_1, sk_1) and (pk_2, sk_2) be two key-pairs which are obtained by running the KeyGen algorithm with independent random tapes. Let $h \xleftarrow{\$} \mathbb{G}$ be an element chosen uniformly and independently at random from the group $\mathbb{G}$. The PPT algorithm performing re-encryption from pk_1 to pk_2 (denoted by $\mathsf{Re-Enc}_{1\rightarrow 2}$) is described below.

```
$\mathsf{Re} - \mathsf{Enc}_{1 \to 2}$
Input: $c_1 = [Y_1, Y_2]$ or special symbol denoted by $\mathsf{keys}$[a]
Constants:[b] $sk_1 = x$, $pk_1 = (g, g^x)$, $pk_2 = (g, g^y)$ and $h$

1. If $input = \mathsf{keys}$, output $(pk_1, pk_2)$.
2. Else,
     – Compute $m = \mathsf{Decrypt1}(sk_1, c_1)$.
     – Choose $r', v, s \overset{\$}{\leftarrow} \mathbb{Z}_p^*$.
     – Output $[X_1, X_2, X_3, X_4, X_5] = [m \cdot g^{r'}, (g^{H(h)})^{r'} \cdot (g^y)^s, h \cdot (g^y)^v, g^v, g^s]$.
```

[a] $\mathsf{keys} \notin \mathbb{G} \times \mathbb{G}$. We need this for a technical part in the proof.

[b] Constants in a program denotes those values which are hardcoded in the program description.

Re-encryption Circuit Family. Let $C_{sk_1, pk_1, pk_2, h}$ be the description of a probabilistic circuit implementing the program $\mathsf{Re} - \mathsf{Enc}_{1 \to 2}$. We note that the constants in the above program are hardwired in the circuit description. These constants can be extracted when given the description of the circuit. Formally, the class of circuits implementing the re-encryption functionality for a given length parameter λ is,

$$C_\lambda = \{ C_{sk_1, pk_1, pk_2, h} : (pk_1, sk_1) \leftarrow \mathsf{KeyGen}(1^\lambda), (pk_2, sk_2) \leftarrow \mathsf{KeyGen}(1^\lambda), h \overset{\$}{\leftarrow} \mathbb{G} \}$$

The circuit family implementing the re-encryption functionality is given by $C = \{C_\lambda\}_{\lambda \in \mathbb{N}}$. The associated sampling algorithm Samp proceeds by choosing $(p, \mathbb{G}, g, H) \leftarrow \mathsf{Setup}(1^\lambda)$. It then samples $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $h \overset{\$}{\leftarrow} \mathbb{G}$. It finally outputs the circuit description of $C_{sk_1, pk_1, pk_2, h}$.

The evaluator of the circuit $C_{sk_1, pk_1, pk_2, h}$ supplies the regular input which is either the ciphertext $c_1 = [C_1, C_2]$ or the special symbol keys and also supplies the random input rand chosen uniformly at random from $\{0,1\}^{3\lambda}$ to the circuit for sampling r', v, s uniformly from $\mathbb{Z}_p^*$.

Decrypting the Circuit Output. The output of $\mathsf{Re} - \mathsf{Enc}_{1 \to 2}$ can be decrypted using the following algorithm $\mathsf{Decrypt2}$:

```
Decrypt2

Input: $sk_2, [X_1, X_2, X_3, X_4, X_5]$ :

1. Parse $sk_2$ as $y$.
2. Compute $h = X_3 \cdot ({X_4}^y)^{-1}$.
3. Compute $X_2' = X_2 \cdot ((X_5)^y)^{-1}$
4. Output $m = (X_1) \cdot ((X_2')^{1/(H(h))})^{-1}$.
```

Correctness. We note that correctness of $\mathsf{Decrypt1}$ algorithm directly follows from the correctness of El-Gamal encryption scheme and correctness of $\mathsf{Decrypt2}$ follows from inspection.

Obfuscator Construction. We now present the construction of an average-case secure obfuscator (denoted by Obf) for the re-encryption circuit family defined in Sect. 3.

```
Obf
Input: $C_{sk_1,pk_1,pk_2,h}$

1. Read $sk_1 = x$, $pk_1 = (g, g^x)$ , $pk_2 = (g, g^y)$ and $h$ from the description
   of the circuit $C_{sk_1,pk_1,pk_2,h}$.
2. Select $v \xleftarrow{\$} \mathbb{Z}_p^*$.
3. Compute $(Z_1, Z_2, Z_3) = (h \cdot (g^y)^v, g^v, H(h)/x)$.
4. Output the description of a circuit implementing the program
   $\mathsf{Re-Enc}'_{1\to 2}$ described below with $pk_1, pk_2, Z_1, Z_2, Z_3$ as the constants
   in the program.
```

```
$\mathsf{Re-Enc}'_{1\to 2}$
Input: $c_1 = [Y_1, Y_2]$ or special symbol denoted by $\mathsf{keys}$.
Constants: $pk_1, pk_2, Z_1, Z_2, Z_3$.

1. If $input = \mathsf{keys}$, output $(pk_1, pk_2)$.
2. Else,
     – Choose two re-randomization values $r', v' \xleftarrow{\$} \mathbb{Z}_p^*$.
     – Re-randomize the input as $X_1 = Y_1 \cdot g^{r'}$, $\overline{X_2} = (Y_2 \cdot (g^x)^{r'})$ and the
       hardwired values as $X_3 = Z_1 \cdot (g^y)^{v'}$, $X_4 = Z_2 \cdot g^{v'}$.
     – Compute $\overline{\overline{X_2}} = (\overline{X_2})^{Z_3}$.
     – Choose $s \xleftarrow{\$} \mathbb{Z}_p^*$.
     – Compute $X_2 = \overline{\overline{X_2}} \cdot (g^y)^s$ and $X_5 = g^s$
     – Output $[X_1, X_2, X_3, X_4, X_5]$.
```

Let C' denote the circuit implementing $\mathsf{Re-Enc}'_{1\to 2}$. The evaluator for the circuit C' provides either $c_1 = [C_1, C_2]$ or special symbol keys as the regular input and $\mathsf{rand} \xleftarrow{\$} \{0,1\}^{3\lambda}$ as the random input for sampling r', v', s uniformly from $\mathbb{Z}_p^*$.

Remark 6. The obfuscated circuit C' is generated by the owner of sk_1 but can be evaluated by anyone. We assume (as described in Remark 2) that the evaluator of C' and the owner of sk_2 do not collude.

4 Security of New Encryption Scheme

We now describe the security model for semantic security of the encryption scheme when the adversary is given black box access to re-encryption functionality. In view of discussion presented in Sect. 1.2, we modify the security model given in [24] as follows.

4.1 Security Model

Let $C \leftarrow \mathsf{Samp}(1^\lambda)$[5] be the re-encryption circuit from pk_1 to pk_2.

Original Ciphertext Security. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against the original ciphertext security.

Definition 3. *Let $\varPi$ be an encryption scheme and let $IND_{b,ori}(\varPi, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), \lambda, i)$ where $b \in \{0, 1\}$ and $i \in \{1, 2\}$, denote the following experiment:*

$$IND_{b,ori}(\varPi, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), \lambda, i)$$

1. *$params \leftarrow \mathsf{Setup}(1^\lambda)$. $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(params)$ and $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(params)$. Choose $h \xleftarrow{\$} \mathbb{G}$. Set $C = C_{sk_1,pk_1,pk_2,h}$* [a].
2. *$(m_0, m_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}(C)}(pk_1, pk_2, params)$.*
3. *$C^* \leftarrow \mathsf{Encrypt1}(m_b, pk_i, params)$.*
4. *$b' \leftarrow \mathcal{A}_2^{\mathcal{O}(C)}(C^*, state)$. Output b'*

[a] Note that setting C in this way is equivalent to sampling C using the Samp algorithm.

The scheme $\varPi$ is said to be ***original ciphertext secure*** *with respect to the oracle access to C if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and for all $i \in \{1, 2\}$, there exists a negligible function $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\varDelta\big(IND_{0,ori}(\varPi, \mathcal{A}, \lambda, i), IND_{1,ori}(\varPi, \mathcal{A}, \lambda, i)\big) \leq \mu(\lambda)$$

Transformed Ciphertext Security. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against the transformed ciphertext security.

Definition 4. *Let $\varPi$ be an encryption scheme and let $IND_{b,tran}(\varPi, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), \lambda)$ where $b \in \{0, 1\}$ denote the following experiment:*

$$IND_{b,tran}(\varPi, \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2), \lambda)$$

1. *$params \leftarrow \mathsf{Setup}(1^\lambda)$. $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(params)$ and $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(params)$. Choose $h \xleftarrow{\$} \mathbb{G}$. Set $C = C_{sk_1,pk_1,pk_2,h}$.*
2. *$(m_0, m_1, state) \leftarrow \mathcal{A}_1^{\mathcal{O}(C)}(params, pk_1, pk_2, sk_1)$.*
3. *$\mathsf{rand} \xleftarrow{\$} \{0, 1\}^{3\lambda}$. Compute $C^* \leftarrow C(\mathsf{Encrypt1}(m_b, pk_1, params); \mathsf{rand})$.*
4. *$b' \leftarrow \mathcal{A}_2^{\mathcal{O}(C)}(C^*, state)$. Output b'*

The scheme $\varPi$ is said to be ***transformed ciphertext secure*** *with respect to the oracle access to C if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\varDelta\big(IND_{0,tran}(\varPi, \mathcal{A}, \lambda), IND_{1,tran}(\varPi, \mathcal{A}, \lambda)\big) \leq \mu(\lambda)$$

[5] For the ease of exposition, we drop the subscripts sk_1, pk_1, pk_2, h.

Statistical Independence. Let us consider the following experiment.

$\mathsf{Stat}(\Pi, \lambda, m)$

1. $params \leftarrow \mathsf{Setup}(1^\lambda)$. $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(params)$ and $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(params)$. Choose $h \xleftarrow{\$} \mathbb{G}$. Set $C = C_{sk_1,pk_1,pk_2,h}$.
2. $\mathsf{rand} \xleftarrow{\$} \{0,1\}^{3\lambda}$. Compute $C^* \leftarrow C(\mathsf{Encrypt1}(m, pk_1, params); \mathsf{rand})$.
3. Output C^*

We require the output of $\mathsf{Stat}(\Pi, \lambda, m)$ to be statistically independent of sk_1.

4.2 Security Proof

We now show that the New Encryption Scheme is *original ciphertext secure* (in Theorem 2), *transformed ciphertext secure* (in Theorem 3) and has statistical independence property (in Lemma 2).

Theorem 2. *The New Encryption Scheme is* ***original ciphertext secure*** *with respect to the oracle $C_{sk_1,pk_1,pk_2,h}$ under the DDH-assumption.*

Proof. We give the proof of this theorem in the full version of our paper [28].

We now show the transformed ciphertext security of our construction.

Theorem 3. *The New Encryption Scheme is* ***transformed ciphertext secure*** *with respect to the oracle $C_{sk_1,pk_1,pk_2,h}$ under the multi-message security (2 messages) of El-Gamal encryption system (Theorem 1).*

Proof. The proof of this theorem appears in the full version of the paper [28].

We note that the statistical independence property of the re-encryption functionality directly follows from inspection of the output distribution of the re-encryption circuit. We record the following lemma.

Lemma 2. *The output distribution of $C_{sk_1,pk_1,pk_2,h}$ where $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}(params)$, $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}(params)$ and $h \xleftarrow{\$} \mathbb{G}$ is statistically independent of sk_1.*

5 Average-Case Virtual Black Box Property

We note that obfuscator construction preserves functionality (the formal proof appears in the full version). We note that the polynomial slowdown property of our construction can be easily verified. It is interesting to note that the obfuscated circuit computes seven exponentiations whereas the original circuit computes eight exponentiations.

We now show that Obf satisfies the average-case virtual black box property.

Lemma 3. Obf *satisfies the average case secure virtual black-box property.*

Proof. The proof techniques used here are similar to that of Hohenberger et al. in [24] and the details follow.

Let $C \leftarrow \mathsf{Samp}(1^\lambda)$ be a circuit chosen randomly from the set C_λ using the Samp algorithm. Let D be any distinguisher with oracle access to C.

We first describe our simulator Sim which has oracle access to the circuit C and takes as input the security parameter in unary form and auxiliary information string denoted by z.

$\mathsf{Sim}^{\mathcal{O}(C)}$

Input: $1^\lambda, z$

1. Query the oracle $\mathcal{O}(C)$ with the special symbol *keys* and obtain pk_1 and pk_2.
2. Parse pk_2 as (g, g^y).
3. Choose $h \xleftarrow{\$} \mathbb{G}$, $v \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $(Z_1', Z_2') = (h \cdot (g^y)^v, g^v)$. Choose Z_3' uniformly at random from $\mathbb{Z}_p^*$.
4. Construct a circuit C' implementing the program $\mathsf{Re}-\mathsf{Enc}'_{1\to 2}$ with values $(pk_1, pk_2, Z_1', Z_2', Z_3')$ hardcoded in the program description.
5. Output the circuit description of C'.

It remains to show that the output distribution of the simulator is computationally indistinguishable to the output distribution of Obf even to distinguishers having oracle access to C.

We define two distributions $\mathsf{Nice}(D^{\mathcal{O}(C)}, \lambda, z)$ and $\mathsf{Junk}(D^{\mathcal{O}(C)}, \lambda, z)$ as follows:

$\mathsf{Nice}(D^{\mathcal{O}(C)}, \lambda, z)$

- $(p, g, \mathbb{G}, H) \leftarrow \mathsf{Setup}(1^\lambda)$. Choose $x, y \xleftarrow{\$} \mathbb{Z}_p^*$. Set $pk_1 = (g, g^x)$ and $pk_2 = (g, g^y)$.
- Choose $h \xleftarrow{\$} \mathbb{G}$ and $v \xleftarrow{\$} \mathbb{Z}_p^*$.
- Compute $Z_1 = h \cdot (g^y)^v$, $Z_2 = g^v$ and $Z_3 = H(h)/x$.
- Output $D^{\mathcal{O}(C)}(pk_1, pk_2, Z_1, Z_2, Z_3, z)$.

$\mathsf{Junk}(D^{\mathcal{O}(C)}, \lambda, z)$

- $(p, g, \mathbb{G}, H) \leftarrow \mathsf{Setup}(1^\lambda)$. Choose $x, y \xleftarrow{\$} \mathbb{Z}_p^*$. Set $pk_1 = (g, g^x)$ and $pk_2 = (g, g^y)$.
- Choose $h \xleftarrow{\$} \mathbb{G}$ and $v \xleftarrow{\$} \mathbb{Z}_p^*$.
- Compute $Z_1' = h \cdot (g^y)^v$, $Z_2' = g^v$ and $Z_3' \xleftarrow{\$} \mathbb{Z}_p^*$.
- Output $D^{\mathcal{O}(C)}(pk_1, pk_2, Z_1', Z_2', Z_3', z)$.

We first observe that for all $z \in \{0,1\}^{poly(\lambda)}$ and for all distinguishers D,

$$\left\{C \rightarrow \mathsf{Samp}(1^\lambda); C' \leftarrow \mathsf{Obf}(C) : D^{\mathcal{O}(C)}(C', z)\right\} \approx \mathsf{Nice}(D^{\mathcal{O}(C)}, \lambda, z)$$

$$\left\{C \rightarrow \mathsf{Samp}(1^\lambda); C' \leftarrow \mathsf{Sim}^{\mathcal{O}(C)}(1^\lambda, z) : D^{\mathcal{O}(C)}(C', z)\right\} \approx \mathsf{Junk}(D^{\mathcal{O}(C)}, \lambda, z)$$

In order to show that Obf satisfies the average case virtual black box property it is enough to show that (from Lemma 1), for all PPT distinguishers D, there exists a negligible function $\mu(\cdot)$ such that for all $z \in \{0,1\}^{poly(\lambda)}$,

$$\varDelta\big(\mathsf{Nice}(D^{\mathcal{O}(C)},\lambda,z)\},\mathsf{Junk}(D^{\mathcal{O}(C)},\lambda,z)\big) \leq \mu(\lambda)$$

We show that for all PPT distinguishers D, there exists a negligible function $\mu(\cdot)$ such that for all $z \in \{0,1\}^{poly(\lambda)}$,

$$\varDelta\big(\mathsf{Nice}(D^{\mathcal{O}(C)},\lambda,z)\},\mathsf{Junk}(D^{\mathcal{O}(C)},\lambda,z)\big) \leq \mu(\lambda)$$

We start with an useful lemma.

Lemma 4.

$$\left\{\begin{array}{l} (p,g,\mathbb{G},H) \leftarrow \mathsf{Setup}(1^\lambda); \\ x,y \xleftarrow{\$} \mathbb{Z}_p^*; \\ pk_1 = (g,g^x); \\ pk_2 = (g,g^y); \\ h \xleftarrow{\$} \mathbb{G}; \\ Z_3', v \xleftarrow{\$} \mathbb{Z}_p^* : \\ \big(pk_1, pk_2, h\cdot(g^y)^v, g^v, Z_3'\big) \end{array}\right\}_\lambda \overset{c}{\approx} \left\{\begin{array}{l} (p,g,\mathbb{G},H) \leftarrow \mathsf{Setup}(1^\lambda); \\ x,y \xleftarrow{\$} \mathbb{Z}_p^*; \\ pk_1 = (g,g^x); \\ pk_2 = (g,g^y); \\ h \xleftarrow{\$} \mathbb{G}; \\ v \xleftarrow{\$} \mathbb{Z}_p^* : \\ \big(pk_1, pk_2, h\cdot(g^y)^v, g^v, H(h)/x\big) \end{array}\right\}_\lambda$$

Proof. The proof of this lemma appears in the full version of the paper [28].

First, we consider two distributions which are similar to Nice and Junk except that they consider a "dummy" distinguisher D^* which outputs whatever is given as input.

Proposition 1. $\big\{\mathsf{Nice}(D^*,\lambda,z)\big\}_\lambda \overset{c}{\approx} \big\{\mathsf{Junk}(D^*,\lambda,z)\big\}_\lambda$

Proof. The proof for the this proposition follows directly from the proof of Lemma 4. We note that left distribution in the lemma statement is identically distributed to $\mathsf{Junk}(D^*,\lambda,z)$ and the right distribution is identically distributed to $\mathsf{Nice}(D^*,\lambda,z)$. Hence,

$$\mathsf{Nice}(D^*,\lambda,z) \overset{c}{\approx} \mathsf{Junk}(D^*,\lambda,z)$$

□

We now consider two more distributions which proceed as Nice and Junk except that they consider distinguishers $D^{\mathcal{O}(R)}$ where R is a probabilistic circuit which on any input $[C_1, C_2]$, first checks if C_1, C_2 belong to $\mathbb{G}$ and if yes, outputs $[A,B,C,D,E]$ where A,B,C,D,E are chosen uniformly and independently from $\mathbb{G}$. Otherwise, it outputs $\perp$.

Note that input to $D^{\mathcal{O}(R)}$ is identically distributed to $\mathsf{Nice}(D^*,\lambda,z)$ in $\mathsf{Nice}(D^{\mathcal{O}(R)},\lambda,z)$ and its input is identically distributed to $\mathsf{Junk}(D^*,\lambda,z)$ in $\mathsf{Junk}(D^{\mathcal{O}(R)},\lambda,z)$. The following proposition is a direct consequence of Proposition 1.

Proposition 2. *For all PPT distinguihsers D, there exists a negligible function $\mu(\cdot)$ such that for all $z \in \{0,1\}^{poly(\lambda)}$ and for all $\lambda \in \mathbb{N}$, we have*

$$\Delta\big(\mathsf{Nice}(D^{\mathcal{O}(R)}, \lambda, z), \mathsf{Junk}(D^{\mathcal{O}(R)}, \lambda, z)\big) \leq \mu(\lambda)$$

Proof. Assume for the sake of contradiction that there exists a distinguisher $D^{\mathcal{O}(R)}$ which can distinguish between $\mathsf{Nice}(D^*, \lambda, z)$ and $\mathsf{Junk}(D^*, \lambda, z)$ with non-negligible advantage. We construct an distinguisher between D' (without the oracle access to R) which distinguishes between $\mathsf{Nice}(D^*, \lambda, z)$ and $\mathsf{Junk}(D^*, \lambda, z)$ with the same advantage.

D' runs D internally by giving its own input as input to D. When D requests an oracle access to R, D' can simulate the responses on its own (It will choose five independent random elements from the group and return as the response for any oracle query after checking whether the input belongs to $\mathbb{G} \times \mathbb{G}$). D' finally outputs what D outputs.

It is easy to see that D' as the same distinguishing advantage that D has and hence we have arrived at a contradiction to Proposition 1. □

Consider any distinguisher D. Let us define,

$$\alpha(\lambda, z) = \Delta\big(\mathsf{Nice}(D^{\mathcal{O}(C)}, \lambda, z), \mathsf{Junk}(D^{\mathcal{O}(C)}, \lambda, z)\big)$$

$$\beta(\lambda, z) = \Delta\big(\mathsf{Nice}(D^{\mathcal{O}(R)}, \lambda, z), \mathsf{Junk}(D^{\mathcal{O}(R)}, \lambda, z)\big)$$

Let q_D be the number of oracle queries that D makes during its execution. Since D runs in polynomial time, q_D is polynomial in λ.

Proposition 3. *There exists an algorithm $\mathcal{B}$ against the multi-message ($2q_D$ messages) security of El-Gamal encryption scheme with an advantage $|\alpha(\lambda, z) - \beta(\lambda, z)|/2$.*

Proof. We prove the proposition by constructing an adversary $\mathcal{B}$ against the El-Gamal challenger with advantage $|\alpha(\lambda, z) - \beta(\lambda, z)|/2$.

$\mathcal{B}$ receives the public key g^y from the El-Gamal challenger. It chooses $x \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $pk_1 = (g, g^x)$ and $pk_2 = (g, g^y)$. It chooses two message vectors $\boldsymbol{M_0}$ and $\boldsymbol{M}_1$ each of length $2q_D$ as follows. It sets $\boldsymbol{M_0} = \{1, 1, \ldots, 1\}$ (of length $2q_D$) and $\boldsymbol{M_1} = \{m_1, m_2, \ldots, m_{2q_D}\}$ where $m_1, \ldots, m_{2q_D}$ are chosen uniformly and independently at random from $\mathbb{G}$. It then receives the challenge ciphertext vector $\boldsymbol{C^*} = \{(g^{r_1}, Q_1), (g^{r_2}, Q_2), \ldots, (g^{r_{q_D}}, Q_{2q_D})\}$ and auxiliary information z. Note that for all $i \in \{1, 2, \ldots 2q_D\}$, r_i is a random element in $\mathbb{Z}_p^*$ and $Q_i = g^{yr_i}$ or an uniformly chosen element depending on whether $\boldsymbol{M_0}$ was encrypted or $\boldsymbol{M_1}$ was encrypted (due to the random choice of $m_1, \ldots, m_{2q_D}$).

$\mathcal{B}$ now uses D to determine whether the challenge ciphertext vector is an encryption of $\boldsymbol{M_0}$ or $\boldsymbol{M_1}$. It first generates the tuples which are distributed exactly as $\mathsf{Nice}(D^*, \lambda, z)$ and $\mathsf{Junk}(D^*, \lambda, z)$. It tosses a random coin c and runs D with input $\mathsf{Nice}(D^*, \lambda, z)$ if $c = 0$ and with input $\mathsf{Junk}(D^*, \lambda, z)$ if $c = 1$. $\mathcal{B}$ needs to answer the re-encryption oracle queries made by D. It uses the challenge ciphertext to answer those oracle queries. We show that if the challenge ciphertext is an encryption of $\boldsymbol{M_0}$, then the oracle responses given by $\mathcal{B}$ are identically distributed to the output of the re-encryption circuit C. If the challenge ciphertext was an encryption of $\boldsymbol{M_1}$, we show that the oracle responses are identically distributed to the output of R. The exact details follow.

$\mathcal{B}$ chooses $h \overset{\$}{\leftarrow} \mathbb{G}$ and $v \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and computes $Z_1 = h \cdot (g^y)^v$ and $Z_2 = g^v$. It then chooses $Z_3 = H(h)/x$ and $Z_3' \overset{\$}{\leftarrow} \mathbb{Z}_p^*$. It tosses a random coin and chooses $c \overset{\$}{\leftarrow} \{0,1\}$. If $c = 0$, it runs $D^{\mathcal{O}(X)}(pk_1, pk_2, Z_1, Z_2, Z_3, z)$. Else it runs, $D^{\mathcal{O}(X)}(pk_1, pk_2, Z_1, Z_2, Z_3', z)$ where X is the circuit description of the program $\mathsf{Re-Enc}''_{1\to 2}$ described below. Note that if $c = 0$, input to D is identical to $\mathsf{Nice}(D^*, \lambda, z)$. Else, it is identical to $\mathsf{Junk}(D^*, \lambda, z)$.

When D makes i^{th} oracle query $[C_1, C_2]$, $\mathcal{B}$ runs the following program and returns the output of the program to D as the response.

```
                    $\mathsf{Re-Enc}''_{1\to 2}$

Constants: $pk_1, pk_2, Z_1, Z_2, Z_3$
Input: $[C_1, C_2], i$
Choose $r' \overset{\$}{\leftarrow} \mathbb{Z}_p^*$.
Compute $C_1' = C_1 \cdot g^{r'}$, $C_2' = C_2 \cdot (g^x)^{r'}$.
Compute $Z_1' = Z_1 \cdot (Q_i)$, $Z_2' = Z_2 \cdot (g^{r_i})$.
Compute $C_2'' = C_2'^{Z_3}$.
Compute $D_2 = C_2'' \cdot Q_{i+q_D}$
Output $[C_1', D_2, Z_1', Z_2', g^{r_{i+q_D}}]$.
```

D finally outputs its guess. Let c' denote the output of D. If $c = c'$, $\mathcal{B}$ outputs 1. Else, it outputs 0.

We now prove the following two claims regarding the output of $\mathcal{B}$.

Claim. If $\boldsymbol{M_0}$ was encrypted, the probability that $\mathcal{B}$ outputs 1 is given by $1/2 + \alpha(\lambda, z)/2$.

Proof. We claim that if $\boldsymbol{M_0}$ was encrypted, then $\mathcal{B}$ perfectly simulates $D^{\mathcal{O}(C)}\big(pk_1, pk_2, h \cdot (g^y)^r, g^r, H(h)/x, z\big)$ or $D^{\mathcal{O}(C)}\big(pk_1, pk_2, h \cdot (g^y)^r, g^r, Z_3', z\big)$ depending upon the bit c. We already noted that the input to D is identically distributed to $\mathsf{Nice}(D^*, \lambda, z)$ or $\mathsf{Junk}(D^*, \lambda, z)$. It is enough to show that X simulates the circuit C perfectly. Since $Q_i = (g^y)^{r_i}$ for all $i \in [1, 2q_D]$ and Z_1, Z_2, Z_3 are properly generated as per the Obf algorithm, the output of X is given by,

$$(m \cdot g^{r+r'}, (g^{r+r'})^{H(h)} \cdot (g^y)^{r_{i+q_D}}, h \cdot (g^y)^{v+r_i}, g^{v+r_i}, g^{r_{i+q_D}})$$

which is identically distributed as the output of the re-encryption circuit since r', r_i, r_{i+q_D} are chosen uniformly at random from $\mathbb{Z}_p^*$. Hence, the probability that $\mathcal{B}$ outputs 1 in this case is same as the probability that $D^{\mathcal{O}(C)}$ outputs $c = c'$ which is same as $1/2 + \alpha(\lambda, z)/2$. □

Claim. If $\boldsymbol{M_1}$ was encrypted, the probability that $\mathcal{B}$ outputs 1 is given by $1/2 + \beta(\lambda, z)/2$.

Proof. We already noted that the input to D are perfectly generated according to either $\mathsf{Nice}(D^*, \lambda, z)$ or $\mathsf{Junk}(D^*, \lambda, z)$. We claim that the response given by $\mathcal{B}$ are same as the one given by R. The output of $\mathcal{B}$ is given by

$$(m \cdot g^{r+r'}, (g^{r+r'})^{H(h)} \cdot Q_{i+q_D}, h \cdot (g^y)^v \cdot Q_i, g^{v+r_i}, g^{r_{i+q_D}})$$

Since Q_i and Q_{i+q_D} are uniformly chosen random elements in $\mathbb{G}$ if $\boldsymbol{M_1}$ was encrypted and r', r_i, r_{i+q_D} are chosen uniformly at random from $\mathbb{Z}_p^*$, we can easily see that all elements in the above distribution are random and independent for every invocation of the oracle.

Hence, in this case $\mathcal{B}$ perfectly simulates $D^{\mathcal{O}(R)}\big(pk_1, pk_2, h \cdot (g^y)^r, g^r, H(h)/x, z\big)$ or $D^{\mathcal{O}(R)}\big(pk_1, pk_2, h{\cdot}(g^y)^r, g^r, Z_3', z\big)$ depending on the bit c. Thus, the probability that $\mathcal{B}$ outputs 1 in this case is same as the probability that D^R outputs $c = c'$ which is given by $\beta(\lambda, z)/2 + 1/2$. □

Hence the advantage of $\mathcal{B}$ in the multi message security game of the El-Gamal Encryption scheme is given by $|\alpha(\lambda, z) - \beta(\lambda, z)|/2$. □

We know from Proposition 2 that $\beta(\lambda, z)$ is negligible. Hence from Proposition 3 we can infer that $\alpha(\lambda, z)$ is also negligible.

Hence, Obf satisfies the average case secure virtual black box property and this concludes the proof of Lemma.

□

Since Obf satisfies the three requirements given in Definition 2, we conclude that Obf is an average-case secure obfuscator.

References

1. Ateniese, G., Kevin, F., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. (TISSEC) **9**(1), 1–30 (2006)
2. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_1
3. Bitansky, N., Paneth, O., Rosen, A.: On the cryptographic hardness of finding a nash equilibrium. In: IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17–20 October 2015, pp. 1480–1498 (2015)
4. Bitansky, N., Paneth, O., Wichs, D.: Perfect structure on the edge of chaos. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9562, pp. 474–502. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49096-9_20
5. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998). doi:10.1007/BFb0054122

6. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 480–499. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44371-2_27
7. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 868–886. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_50
8. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, 22–25 October 2011, pp. 97–106 (2011)
9. Brakerski, Z., Vaikuntanathan, V.: Lattice-based FHE as secure as PKE. In: Innovations in Theoretical Computer Science, ITCS 2014, Princeton, NJ, USA, 12–14 January 2014, pp. 1–12 (2014)
10. Canetti, R., Dakdouk, R.R.: Obfuscating point functions with multibit output. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 489–508. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_28
11. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Proceedings of 14th ACM Conference on Computer and Communications Security, pp. 185–194. ACM (2007)
12. Canetti, R., Rothblum, G.N., Varia, M.: Obfuscation of hyperplane membership. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 72–89. Springer, Heidelberg (2010). doi:10.1007/978-3-642-11799-2_5
13. Chandran, N., Chase, M., Liu, F.-H., Nishimaki, R., Xagawa, K.: Re-encryption, functional re-encryption, and multi-hop re-encryption: a framework for achieving obfuscation-based security and instantiations from lattices. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 95–112. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54631-0_6
14. Chandran, N., Chase, M., Vaikuntanathan, V.: Functional re-encryption and collusion-resistant obfuscation. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 404–421. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28914-9_23
15. Chow, S.S.M., Weng, J., Yang, Y., Deng, R.H.: Efficient unidirectional proxy re-encryption. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 316–332. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12678-9_19
16. Garg, S., Gentry, C., Halevi, S., Raykova, M.: Two-round secure MPC from indistinguishability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 74–94. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54242-8_4
17. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS), pp. 40–49. IEEE (2013)
18. Garg, S., Pandey, O., Srinivasan, A.: Revisiting the cryptographic hardness of finding a nash equilibrium. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 579–604. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53008-5_20
19. Garg, S., Pandey, O., Srinivasan, A., Zhandry, M.: Breaking the sub-exponential barrier in obfustopia. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10212, pp. 156–181. Springer, Cham (2017). doi:10.1007/978-3-319-56617-7_6
20. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, 31 May–2 June 2009, pp. 169–178 (2009)

21. Goldreich, O.: The Foundations of Cryptography - Basic Techniques, vol. 1. Cambridge University Press, Cambridge (2001)
22. Hada, S.: Zero-knowledge and code obfuscation. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 443–457. Springer, Heidelberg (2000). doi:10.1007/3-540-44448-3_34
23. Hada, S.: Secure obfuscation for encrypted signatures. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 92–112. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_5
24. Hohenberger, S., Rothblum, G.N., Shelat, A., Vaikuntanathan, V.: Securely obfuscating re-encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 233–252. Springer, Heidelberg (2007). doi:10.1007/978-3-540-70936-7_13
25. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78440-1_21
26. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Found. Secure Comput. **4**(11), 169–180 (1978)
27. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Symposium on Theory of Computing, STOC 2014, New York, NY, USA, 31 May–03 June 2014, pp. 475–484 (2014)
28. Srinivasan, A., Rangan, C.P.: Efficiently obfuscating re-encryption program under DDH assumption. IACR Cryptology ePrint Archive 2015:822 (2015)
29. Wee, H.: On obfuscating point functions. In: Proceedings of 37th Annual ACM Symposium on Theory of Computing, pp. 523–532. ACM (2005)

Lattice-Based Group Signatures: Achieving Full Dynamicity with Ease

San Ling, Khoa Nguyen(✉), Huaxiong Wang, and Yanhong Xu(✉)

Division of Mathematical Sciences, School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore, Singapore
{lingsan,khoantt,hxwang,xu0014ng}@ntu.edu.sg

Abstract. Lattice-based group signature is an active research topic in recent years. Since the pioneering work by Gordon et al. (Asiacrypt 2010), eight other schemes have been proposed, providing various improvements in terms of security, efficiency and functionality. However, most of the existing constructions work only in the static setting where the group population is fixed at the setup phase. The only two exceptions are the schemes by Langlois et al. (PKC 2014) that handles user revocations (but new users cannot join), and by Libert et al. (Asiacrypt 2016) which addresses the orthogonal problem of dynamic user enrollments (but users cannot be revoked).

In this work, we provide the first lattice-based group signature that offers full dynamicity (i.e., users have the flexibility in joining and leaving the group), and thus, resolve a prominent open problem posed by previous works. Moreover, we achieve this non-trivial feat in a relatively simple manner. Starting with Libert et al.'s fully static construction (Eurocrypt 2016) - which is arguably the most efficient lattice-based group signature to date, we introduce simple-but-insightful tweaks that allow to upgrade it directly into the fully dynamic setting. More startlingly, our scheme even produces slightly shorter signatures than the former. The scheme satisfies the strong security requirements of Bootle et al.'s model (ACNS 2016), under the Short Integer Solution (SIS) and the Learning With Errors (LWE) assumptions.

Keywords: Lattice-based group signatures · Full dynamicity · Updatable Merkle trees · Stern-like zero-knowledge protocols

1 Introduction

Group signature, introduced by Chaum and van Heyst [14], is a fundamental anonymity primitive which allows members of a group to sign messages on behalf of the whole group. Yet, users are kept accountable for the signatures they issue since a tracing authority can identify them should the need arise. There have been numerous works on group signatures in the last quarter-century.

Ateniese et al. [2] proposed the first scalable instantiation meeting the security properties that can be intuitively expected from the primitive, although

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 293–312, 2017.
DOI: 10.1007/978-3-319-61204-1_15

clean security notions were not available yet at that time. Bellare et al. [4] filled this gap by providing strong security notions for static groups, in which the group population is fixed at the setup phase. Subsequently, Kiayias and Yung [23] and Bellare et al. [5] established the models capturing the partially dynamic setting, where users are able to join the group at any time, but once they have done so, they cannot leave the group. Sakai et al. [44] strengthened these models by suggesting the notion of opening soundness, guaranteeing that a valid signature only traces to one user. Efficient schemes satisfying these models have been proposed in the random oracle model [16,40] and in the standard model [20,32].

One essential functionality of group signatures is the support for membership revocation. Enabling this feature in an efficient manner is quite challenging, since one has to ensure that revoked users are no longer able to sign messages and the workloads of other parties (managers, non-revoked users, verifiers) do not significantly increase in the meantime. Several different approaches have been suggested [6,10,11,39,46] to address this problem, and notable pairing-based constructions supporting both dynamic joining and efficient revocation were given in [30,31,37]. Very recently, Bootle et al. [7] pointed out a few shortcomings of previous models, and put forward stringent security notions for fully dynamic group signatures. They also demonstrated a construction satisfying these notions based on the decisional Diffie-Hellman (DDH) assumption, following a generic transformation from a secure accountable ring signature scheme [8].

For the time being, existing schemes offering full dynamicity all rely on number-theoretic assumptions which are vulnerable to quantum attacks. To avoid putting all eggs in one basket, it is thus encouraging to consider instantiations based on alternative, post-quantum foundations, e.g., lattice assumptions. In view of this, let us now look at the topic of lattice-based group signatures.

Lattice-based group signatures. Lattice-based cryptography has been an exciting research area since the seminal works of Regev [42] and Gentry et al. [18]. Along with other primitives, lattice-based group signature has received noticeable attention in recent years. The first scheme was introduced by Gordon et al. [19] whose solution produced signature size linear in the number of group users N. Camenisch et al. [12] then extended [19] to achieve anonymity in the strongest sense. Later, Laguillaumie *et al.* [24] put forward the first scheme with the signature size logarithmic in N, at the cost of relatively large parameters. Simpler and more efficient solutions with $\mathcal{O}(\log N)$ signature size were subsequently given by Nguyen et al. [41] and Ling et al. [34]. Libert et al. [28] obtained substantial efficiency improvements via a construction based on Merkle trees which eliminates the need for GPV trapdoors [18]. More recently, a scheme supporting message-dependent opening (MDO) feature [43] was proposed in [29]. All the schemes mentioned above are designed for static groups.

The only two known lattice-based group signatures that have certain dynamic features were proposed by Langlois et al. [25] and Libert et al. [26]. The former is a scheme with verifier-local revocation (VLR) [6], which means that only the verifiers need to download the up-to-date group information. The latter addresses the orthogonal problem of dynamic user enrollments (but users cannot be revoked).

To achieve those partially dynamic functionalities, both of the proposals have to incorporate relatively complicated mechanisms[1] and both heavily rely on lattice trapdoors.

The discussed above situation is somewhat unsatisfactory, given that the full dynamicity feature is highly desirable in most applications of group signatures (e.g., protecting the privacy of commuters in public transportation), and it has been achieved based on number-theoretic assumptions. This motivates us to work on fully dynamic group signatures from lattices. Furthermore, considering that the journey to partial dynamicity in previous works [25,26] was shown not easy, we ask ourselves an inspiring question: Can we achieve full dynamicity with ease? At the end of the day, it is good to solve an open research question, but it would be even better and more exciting to do this in a simple way. To make it possible, we will likely need some new and insightful ideas.

Our Results and Techniques. We introduce the first fully dynamic group signature from lattices. The scheme satisfies the strong security requirements put forward by Bootle et al. [7], under the Short Integer Solution (SIS) and the Learning With Errors (LWE) assumptions. As in all previous lattice-based group signatures, our scheme is analyzed in the random oracle model.

For a security parameter λ and maximum expected number of group users N, our scheme features signature size $\widetilde{\mathcal{O}}(\lambda \cdot \log N)$ and group public key size $\widetilde{\mathcal{O}}(\lambda^2 + \lambda \cdot \log N)$. The user's secret key has bit-size $\widetilde{\mathcal{O}}(\lambda) + \log N$. At each epoch when the group information is updated, the verifiers only need to download an extra $\widetilde{\mathcal{O}}(\lambda)$ bits in order to perform verification of signatures[2], while each active signer only has to download $\widetilde{\mathcal{O}}(\lambda \cdot \log N)$ bits. In Table 1, we give a detailed comparison of our scheme with known lattice-based group signatures, in terms of efficiency and functionality. The full dynamicity feature is achieved with a very reasonable cost and without having to rely on lattice trapdoors. Somewhat surprisingly, our scheme even produces shorter signatures than the scheme from [28] - which is arguably the most efficient lattice-based group signature known to date. Furthermore, these results are obtained in a relatively simple manner, thanks to three main ideas/techniques discussed below.

Our starting point is the scheme [28], which works in the static setting. Instead of relying on trapdoor-based ordinary signatures as in prior works, the LLNW scheme employs on a SIS-based Merkle tree accumulator. For a group of $N = 2^\ell$ users, the manager chooses uniformly random vectors $\mathbf{x}_0, \ldots, \mathbf{x}_{N-1}$; hashes them to $\mathbf{p}_0, \ldots, \mathbf{p}_{N-1}$, respectively; builds a tree on top of these hash values; and publishes the tree root $\mathbf{u}$. The signing key of user i consists of $\mathbf{x}_i$

[1] Langlois et al. considered users' "tokens" as functions of Bonsai signatures [13] and associated them with a sophisticated revocation technique, while Libert et al. used a variant of Boyen's signature [9] to sign users' public keys. Both underlying signature schemes require long keys and lattice trapdoors.

[2] We remark that in the DDH-based instantiation from [7] which relies on the accountable ring signature from [8], the verifiers have to periodically download public keys of active signers. Our scheme overcomes this issue, thanks to the use of an updatable accumulator constructed in Sect. 3.

Table 1. Comparison of known lattice-based group signatures, in terms of efficiency and functionality. The comparison is done based on two governing parameters: security parameter λ and the maximum expected number of group users $N = 2^\ell$. As for the scheme from [25], R denotes the number of revoked users at the epoch in question.

Scheme	Sig. size	Group PK size	Signer's SK size	Trap-door?	Model	Extra info per epoch
GKV [19]	$\widetilde{\mathcal{O}}(\lambda^2 \cdot N)$	$\widetilde{\mathcal{O}}(\lambda^2 \cdot N)$	$\widetilde{\mathcal{O}}(\lambda^2)$	yes	static	NA
CNR [12]	$\widetilde{\mathcal{O}}(\lambda^2 \cdot N)$	$\widetilde{\mathcal{O}}(\lambda^2)$	$\widetilde{\mathcal{O}}(\lambda^2)$	yes	static	NA
LLLS [24]	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\mathcal{O}(\lambda^2 \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2)$	yes	static	NA
LLNW [25]	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2 \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	yes	VLR	Sign: **no** Ver: $\widetilde{\mathcal{O}}(\lambda) \cdot R$
NZZ [41]	$\widetilde{\mathcal{O}}(\lambda + \ell^2)$	$\widetilde{\mathcal{O}}(\lambda^2 \cdot \ell^2)$	$\widetilde{\mathcal{O}}(\lambda^2)$	yes	static	NA
LNW [34]	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2 \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda)$	yes	static	NA
LLNW [28]	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2 + \lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	**FREE**	static	NA
LLM+ [26]	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2 \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda)$	yes	partially dynamic	NA
LMN [29]	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2 \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda)$	yes	MDO	NA
Ours	$\widetilde{\mathcal{O}}(\lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda^2 + \lambda \cdot \ell)$	$\widetilde{\mathcal{O}}(\lambda) + \ell$	**FREE**	fully dynamic	Sign: $\widetilde{\mathcal{O}}(\lambda \cdot \ell)$ Ver: $\widetilde{\mathcal{O}}(\lambda)$

and the witness for the fact that $\mathbf{p}_i$ was accumulated in $\mathbf{u}$. When issuing signatures, the user proves knowledge of a valid pair $(\mathbf{x}_i, \mathbf{p}_i)$ and of the tree path from $\mathbf{p}_i$ to $\mathbf{u}$. The user also has to encrypt the binary representation $\mathsf{bin}(i)$ of his identity i, and prove that the ciphertext is well-formed. The encryption layer is also lattice-trapdoor-free, since it utilizes the Naor-Yung double-encryption paradigm [38] with Regev's LWE-based encryption scheme. To upgrade the LLNW scheme directly into a fully dynamic group signature, we now let the user compute the pair $(\mathbf{x}_i, \mathbf{p}_i)$ on his own (for enabling non-frameability), and we employ the following three ideas/techniques.

First, we add a dynamic ingredient into the static Merkle tree accumulator from [28]. To this end, we equip it with an efficient updating algorithm with complexity $\mathcal{O}(\log N)$: to change an accumulated value, we simply update the values at the corresponding leaf and along its path to the root.

Second, we create a simple rule to handle user enrollment and revocation efficiently (i.e., without resetting the whole tree). Specifically, we use the updating algorithm to set up the system so that: (i) If a user has not joined the group or has been revoked, the value at the leaf associated with him is set as $\mathbf{0}$; (ii) When a user joins the group, that value is set as his public key $\mathbf{p}_i$. Our setup guarantees that only active users (i.e., who has joined and has not been revoked at the given epoch) have their *non-zero* public keys accumulated into the updated root. This rule effectively separates active users who can sign from those who cannot: when signing messages, the user proceeds as in the LLNW scheme, and is asked to additionally prove in zero-knowledge that $\mathbf{p}_i \neq \mathbf{0}$. In other words, the seemingly big gap between being fully static and being fully dynamic has been reduced to a small difference!

Third, the arising question now is how to additionally prove the inequality $\mathbf{p}_i \neq \mathbf{0}$ in the framework of the Stern-like [45] protocol from [28]. One would naturally expect that this extra job could be done without losing too much in terms of efficiency. Here, the surprising and somewhat unexpected fact is that we can actually do it while *gaining* efficiency, thanks to the following simple idea. Recall that, in [28], to prove knowledge of $\mathbf{p}_i \in \{0,1\}^{nk}$, an extension technique from [33] is employed, in which $\mathbf{p}_i$ is extended into a vector of dimension $2nk$. We note that, the authors of [33] also suggested a slightly modified version of their technique, that allows to simultaneously prove that $\mathbf{p}_i \in \{0,1\}^{nk}$ and $\mathbf{p}_i$ is non-zero while working only with dimension $2nk - 1$. This intriguing tweak enables us to obtain a zero-knowledge protocol with slightly lower communication cost, and hence, group signatures with slightly smaller size than in [28].

To summarize, we solve a prominent open question in the field of lattice-based group signatures. Moreover, our solution is simple and comparatively efficient. Our results, while not yielding a truly practical scheme, would certainly help to bring the field one step closer to practice.

Organization. In Sect. 2, we recall some background on fully dynamic group signatures and lattice-based cryptography. Section 3 develops an updatable Merkle tree accumulator. Our main scheme is constructed and analyzed in Sect. 4.

2 Preliminaries

2.1 Fully Dynamic Group Signatures

We recall the definition and security notions of fully dynamic group signatures (FDGS) presented by Bootle et al. [7]. A FDGS scheme involves the following entities: a group manager GM that determines who can join the group, a tracing manager TM who can open signatures, and a set of users who are potential group members. Users can join/leave the group at the discretion of GM. We assume GM will publish some information info_τ regularly associated with a distinct index τ (referred as epoch hereafter). Without loss of generality, assume there is one-to-one correspondence between information and the associated epoch. The information describes changes of the group, e.g., current group members or members that are excluded from the group. We assume the published group information is authentic. By comparing current group information with previous one, it allows any party to identify revoked users at current epoch. For simplicity assume $\tau_1 < \tau_2$ if info_{τ_1} is published before info_{τ_2}, i.e., the epoch preserves the order in which the corresponding group information was published. In existing models, the keys of authorities are supposed to be generated honestly; while in [7], Bootle et al. consider a stronger model where the keys of authorities can be maliciously generated by the adversary.

Syntax of Fully Dynamic Group Signatures. A FDGS scheme is a tuple of following polynomial-time algorithms.

$\mathsf{GSetup}(\lambda) \rightarrow \mathsf{pp}$. On input security parameter λ, this algorithm generates public parameters pp.

$\langle \mathsf{GKgen}_{\mathsf{GM}}(\mathsf{pp}), \mathsf{GKgen}_{\mathsf{TM}}(\mathsf{pp})\rangle$. This is an interactive protocol between the group manager GM and the tracing manager TM. If it completes successfully, algorithm $\mathsf{GKgen}_{\mathsf{GM}}$ outputs a manager key pair $(\mathsf{mpk}, \mathsf{msk})$. Meanwhile, GM initializes the group information info and the registration table $\mathbf{reg}$. The algorithm $\mathsf{GKgen}_{\mathsf{TM}}$ outputs a tracing key pair $(\mathsf{tpk}, \mathsf{tsk})$. Set group public key $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk}, \mathsf{tpk})$.

$\mathsf{UKgen}(\mathsf{pp}) \rightarrow (\mathsf{upk}, \mathsf{usk})$. Given public parameters pp, this algorithm generates a user key pair $(\mathsf{upk}, \mathsf{usk})$.

$\langle \mathsf{Join}(\mathsf{info}_{\tau_{\mathrm{current}}}, \mathsf{gpk}, \mathsf{upk}, \mathsf{usk}); \mathsf{Issue}(\mathsf{info}_{\tau_{\mathrm{current}}}, \mathsf{msk}, \mathsf{upk})\rangle$. This is an interactive algorithm run by a user and GM. If it completes successfully, this user becomes a group member with an identifier uid and the Join algorithm stores secret group signing key $\mathsf{gsk}[\mathsf{uid}]$ while Issue algorithm stores registration information in the table $\mathbf{reg}$ with index uid.

$\mathsf{GUpdate}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau_{\mathrm{current}}}, \mathcal{S}, \mathbf{reg}) \rightarrow \mathsf{info}_{\tau_{\mathrm{new}}}$. This is an algorithm run by GM to update group information while advancing the epoch. Given gpk,msk, $\mathsf{info}_{\tau_{\mathrm{current}}}$, registration table $\mathbf{reg}$, a set $\mathcal{S}$ of active users to be removed from the group, GM computes new group information $\mathsf{info}_{\tau_{\mathrm{new}}}$ and may update the table $\mathbf{reg}$. If there is no change to the group, GM outputs $\perp$.

$\mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}[\mathsf{uid}], \mathsf{info}_{\tau}, M) \rightarrow \varSigma$. This algorithm outputs a group signature $\varSigma$ on message M by user uid. It outputs $\perp$ if this user is inactive at epoch τ.

$\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_{\tau}, M, \varSigma) \rightarrow 0/1$. This algorithm checks the validity of the signature $\varSigma$ on message M at epoch τ.

$\mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_{\tau}, \mathbf{reg}, M, \varSigma) \rightarrow (\mathsf{uid}, \varPi_{\mathsf{trace}})$. This is an algorithm run by TM. Given the inputs, TM returns an identity uid of a group member who signed the message and a proof indicating this tracing result or $\perp$ if it fails to trace to a group member.

$\mathsf{Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_{\tau}, \varPi_{\mathsf{trace}}, M, \varSigma) \rightarrow 0/1$. This algorithm checks the validity of $\varPi_{\mathsf{trace}}$ outputted by the Trace algorithm.

Correctness and Security of Fully Dynamic Group Signatures. As put forward by Bootle et al. [7], a FDGS scheme must satisfy *correctness*, *anonymity*, *non-frameability*, *traceability* and *tracing soundness*.

Correctness demands that a signature generated by an honest and active user is always accepted by algorithm Verify, and that algorithm Trace can always identify that user, as well as produces a proof accepted by algorithm Judge.

Anonymity requires that it is infeasible for any PPT adversary to distinguish signatures generated by two active users of its choice at the chosen epoch, even if it can corrupt any user, can choose the key of GM, and is given access to the Trace oracle.

Non-Frameability makes sure that the adversary cannot generate a valid signature that traces to an honest user even if it can corrupt all other users, and can choose keys of both managers.
Traceability ensures that the adversary cannot produce a valid signature that cannot be traced to an active user at the chosen epoch, even if it can corrupt any user and can choose the key of TM.
Tracing Soundness requires that it is infeasible to produce a valid signature that traces to two different users, even if all group users and both managers are fully controlled by the adversary.

Formal definitions of correctness and security requirements are available in the full version.

2.2 Background on Lattices

We recall the average-case lattice problems SIS and LWE, together with their hardness results.

Definition 1 [1,18]**.** *The* $\mathsf{SIS}^{\infty}_{n,m,q,\beta}$ *problem is as follows: Given uniformly random matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$*, find a non-zero vector* $\mathbf{x} \in \mathbb{Z}^m$ *such that* $\|\mathbf{x}\|_\infty \leq \beta$ *and* $\mathbf{A}\cdot\mathbf{x} = \mathbf{0} \bmod q$.

If $m, \beta = \mathsf{poly}(n)$, and $q > \beta \cdot \widetilde{\mathcal{O}}(\sqrt{n})$, then the $\mathsf{SIS}^{\infty}_{n,m,q,\beta}$ problem is at least as hard as worst-case lattice problem SIVP_γ for some $\gamma = \beta\cdot\widetilde{\mathcal{O}}(\sqrt{nm})$ (see, e.g., [18,36]). Specifically, when $\beta = 1$, $q = \widetilde{\mathcal{O}}(n)$, $m = 2n\lceil \log q\rceil$, the $\mathsf{SIS}^{\infty}_{n,m,q,1}$ problem is at least as hard as SIVP_γ with $\gamma = \widetilde{\mathcal{O}}(n)$.

In the last decade, numerous SIS-based cryptographic primitives have been proposed. In this work, we will extensively employ 2 such constructions:

- Our group signature scheme is based on the Merkle tree accumulator from [28], which is built upon a specific family of collision-resistant hash functions.
- Our zero-knowledge argument systems use the statistically hiding and computationally binding string commitment scheme from [22].

For appropriate setting of parameters, the security of the above two constructions can be based on the worst-case hardness of $\mathsf{SIVP}_{\widetilde{\mathcal{O}}(n)}$.

In the group signature in Sect. 4, we will employ the multi-bit version of Regev's encryption scheme [42], presented in [21]. The scheme is based on the hardness of the LWE problem.

Definition 2 [42]**.** *Let* $n, m \geq 1$*,* $q \geq 2$*, and let* χ *be a probability distribution on* $\mathbb{Z}$*. For* $\mathbf{s} \in \mathbb{Z}_q^n$*, let* $\mathcal{A}_{\mathbf{s},\chi}$ *be the distribution obtained by sampling* $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ *and* $e \hookleftarrow \chi$*, and outputting* $(\mathbf{a}, \mathbf{s}^\top\cdot\mathbf{a} + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$*. The* $\mathsf{LWE}_{n,q,\chi}$ *problem asks to distinguish* m *samples chosen according to* $\mathcal{A}_{\mathbf{s},\chi}$ *(for* $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$*) and* m *samples chosen according to the uniform distribution over* $\mathbb{Z}_q^n \times \mathbb{Z}_q$*.*

If q is a prime power, χ is the discrete Gaussian distribution $D_{\mathbb{Z},\alpha q}$, where $\alpha q \geq 2\sqrt{n}$, then $\mathsf{LWE}_{n,q,\chi}$ is as least as hard as $\mathsf{SIVP}_{\widetilde{\mathcal{O}}(n/\alpha)}$ (see [35,36,42]).

2.3 Stern-Like Protocols for Lattice-Based Relations

The zero-knowledge (ZK) argument systems appearing in this paper operate in the framework of Stern's protocol [45]. Let us now recall some background. This protocol was originally proposed in the context of code-based cryptography, and was later adapted into the lattice setting by Kawachi et al. [22]. Subsequently, it was empowered by Ling et al. [33] to handle the matrix-vector relations associated with the SIS and LWE problems, and further developed to design several lattice-based schemes: group signatures [25,26,28,29,34], policy-based signatures [15] and group encryption [27].

Stern-like protocols are quite useful in the context of lattice-based privacy-preserving systems, when one typically works with modular linear equations of the form $\sum_i \mathbf{M}_i \cdot \mathbf{x}_i = \mathbf{v} \bmod q$ - where $\{\mathbf{M}_i\}_i$, $\mathbf{v}$ are public, and one wants to prove in ZK that secret vectors $\{\mathbf{x}_i\}_i$ satisfy certain constraints, e.g., they have small norms and/or have coordinates arranged in a special way. The high-level ideas can be summarized as follows. If the given constraints are invariant under certain type of permutations of coordinates, then one readily uses uniformly random permutations to prove those constraints. Otherwise, one performs some pre-processings with $\{\mathbf{x}_i\}_i$ to reduce to the former case. Meanwhile, to prove that the modular linear equation holds, one makes use of a standard masking technique.

The basic protocol consists of 3 moves: commitment, challenge, response. If the statistically hiding and computationally binding string commitment scheme from [22] is employed in the first move, then one obtains a statistical zero-knowledge argument of knowledge (ZKAoK) with perfect completeness, constant soundness error 2/3, and communication cost $\mathcal{O}(|w| \cdot \log q)$, where $|w|$ denotes the total bit-size of the secret vectors. In many applications, the protocol is repeated $\kappa = \omega(\log \lambda)$ times, for security parameter λ, to achieve negligible soundness error, and then made non-interactive via the Fiat-Shamir heuristic [17]. In the random oracle model, this results in a non-interactive zero-knowledge argument of knowledge (NIZKAoK) with bit-size $\mathcal{O}(|w| \cdot \log q) \cdot \omega(\log \lambda)$.

3 Updatable Lattice-Based Merkle Hash Trees

We first recall the lattice-based Merkle-tree accumulator from [28], and then, we equip it with a simple updating algorithm which allows to change an accumulated value in time logarithmic in the size of the accumulated set. This updatable hash tree will serve as the building block of our construction in Sect. 4.

3.1 Cryptographic Accumulators

An *accumulator scheme* is a tuple of polynomial-time algorithms defined below.

$\mathsf{TSetup}(\lambda)$. On input security parameter λ, output the public parameter pp.

$\mathsf{TAcc}_{\mathsf{pp}}$. On input a set $R = \{\mathbf{d}_0, \ldots, \mathbf{d}_{N-1}\}$ of N data values, output an accumulator value $\mathbf{u}$.

$\mathsf{TWitness}_{\mathsf{pp}}$. On input a data set R and a value $\mathbf{d}$, output $\perp$ if $\mathbf{d} \notin R$; otherwise output a witness w for the fact that $\mathbf{d}$ is accumulated in $\mathsf{TAcc}(R)$. (Typically, the size of w should be short (*e.g.*, constant or logarithmic in N) to be useful.)

$\mathsf{TVerify}_{\mathsf{pp}}$. On input accumulator value $\mathbf{u}$ and a value-witness pair $(\mathbf{d}, w)$, output 1 (which indicates that $(\mathbf{d}, w)$ is valid for the accumulator $\mathbf{u}$) or 0.

An accumulator scheme is called correct if for all $\mathsf{pp} \leftarrow \mathsf{TSetup}(\lambda)$, we have $\mathsf{TVerify}_{\mathsf{pp}}\big(\mathsf{TAcc}_{\mathsf{pp}}(R), \mathbf{d}, \mathsf{TWitness}_{\mathsf{pp}}(R, \mathbf{d})\big) = 1$ for all $\mathbf{d} \in R$.

A natural security requirement for accumulators, as considered in [3,11,28], says that it is infeasible to prove that a value $\mathbf{d}^*$ was accumulated in a value $\mathbf{u}$ if it was not. This property is formalized as follows.

Definition 3 [28]. *An accumulator scheme* $(\mathsf{TSetup}, \mathsf{TAcc}, \mathsf{TWitness}, \mathsf{TVerify})$ *is called secure if for all PPT adversaries* $\mathcal{A}$*:*

$$\Pr\big[\mathsf{pp} \leftarrow \mathsf{TSetup}(\lambda); (R, \mathbf{d}^*, w^*) \leftarrow \mathcal{A}(\mathsf{pp}) : \\ \mathbf{d}^* \notin R \wedge \mathsf{TVerify}_{\mathsf{pp}}(\mathsf{TAcc}_{\mathsf{pp}}(R), \mathbf{d}^*, w^*) = 1\big] = \mathsf{negl}(\lambda).$$

3.2 The LLNW Merkle-Tree Accumulator

Notations. Hereunder we will use the notation $x \xleftarrow{\$} S$ to indicate that x is chosen uniformly at random from finite set S. For bit $b \in \{0,1\}$, we let $\bar{b} = 1 - b$. The Merkle-tree accumulator scheme from [28] works with parameters $n = \mathcal{O}(\lambda)$, $q = \widetilde{\mathcal{O}}(n^{1.5})$, $k = \lceil \log_2 q \rceil$, and $m = 2nk$. The set $\mathbb{Z}_q$ is identified by $\{0, \ldots, q-1\}$. Define the "powers-of-2" matrix

$$\mathbf{G} = \begin{bmatrix} 1\ 2\ 4\ \ldots\ 2^{k-1} & & \\ & \ldots & \\ & & 1\ 2\ 4\ \ldots\ 2^{k-1} \end{bmatrix} \in \mathbb{Z}_q^{n \times nk}.$$

Note that for every $\mathbf{v} \in \mathbb{Z}_q^n$, we have $\mathbf{v} = \mathbf{G} \cdot \mathsf{bin}(\mathbf{v})$, where $\mathsf{bin}(\mathbf{v}) \in \{0,1\}^{nk}$ denotes the binary representation of $\mathbf{v}$. The scheme is built upon the following family of SIS-based collision-resistant hash functions.

Definition 4. *The function family* $\mathcal{H}$ *mapping* $\{0,1\}^{nk} \times \{0,1\}^{nk}$ *to* $\{0,1\}^{nk}$ *is defined as* $\mathcal{H} = \{h_{\mathbf{A}} \mid \mathbf{A} \in \mathbb{Z}_q^{n \times m}\}$*, where for* $\mathbf{A} = [\mathbf{A}_0 | \mathbf{A}_1]$ *with* $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times nk}$*, and for any* $(\mathbf{u}_0, \mathbf{u}_1) \in \{0,1\}^{nk} \times \{0,1\}^{nk}$*, we have:*

$$h_{\mathbf{A}}(\mathbf{u}_0, \mathbf{u}_1) = \mathsf{bin}\big(\mathbf{A}_0 \cdot \mathbf{u}_0 + \mathbf{A}_1 \cdot \mathbf{u}_1 \bmod q\big) \in \{0,1\}^{nk}.$$

Note that $h_{\mathbf{A}}(\mathbf{u}_0, \mathbf{u}_1) = \mathbf{u} \Leftrightarrow \mathbf{A}_0 \cdot \mathbf{u}_0 + \mathbf{A}_1 \cdot \mathbf{u}_1 = \mathbf{G} \cdot \mathbf{u} \bmod q$.

A Merkle tree with $N = 2^\ell$ leaves, where ℓ is a positive integer, then can be constructed based on the function family $\mathcal{H}$ as follows.

$\mathsf{TSetup}(\lambda)$. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, and output $\mathsf{pp} = \mathbf{A}$.

$\mathsf{TAcc}_{\mathbf{A}}(R = \{\mathbf{d}_0 \in \{0,1\}^{nk}, \ldots, \mathbf{d}_{N-1} \in \{0,1\}^{nk}\})$. For every $j \in [0, N-1]$, let $\mathsf{bin}(j) = (j_1, \ldots, j_\ell) \in \{0,1\}^\ell$ be the binary representation of j, and let $\mathbf{d}_j = \mathbf{u}_{j_1,\ldots,j_\ell}$. Form the tree of depth $\ell = \log N$ based on the N leaves $\mathbf{u}_{0,0,\ldots,0}, \ldots, \mathbf{u}_{1,1,\ldots,1}$ as follows:

1. At depth $i \in [\ell]$, the node $\mathbf{u}_{b_1,\ldots,b_i} \in \{0,1\}^{nk}$, for all $(b_1, \ldots, b_i) \in \{0,1\}^i$, is defined as $h_{\mathbf{A}}(\mathbf{u}_{b_1,\ldots,b_i,0}, \mathbf{u}_{b_1,\ldots,b_i,1})$.
2. At depth 0: The root $\mathbf{u} \in \{0,1\}^{nk}$ is defined as $h_{\mathbf{A}}(\mathbf{u}_0, \mathbf{u}_1)$.

The algorithm outputs the accumulator value $\mathbf{u}$.

$\mathsf{TWitness}_{\mathbf{A}}(R, \mathbf{d})$. If $\mathbf{d} \notin R$, return $\perp$. Otherwise, $\mathbf{d} = \mathbf{d}_j$ for some $j \in [0, N-1]$ with binary representation $(j_1, \ldots, j_\ell)$. Output the witness w defined as:

$$w = \big((j_1, \ldots, j_\ell), (\mathbf{u}_{j_1,\ldots,j_{\ell-1},\bar{j_\ell}}, \ldots, \mathbf{u}_{j_1,\bar{j_2}}, \mathbf{u}_{\bar{j_1}})\big) \in \{0,1\}^\ell \times \big(\{0,1\}^{nk}\big)^\ell,$$

for $\mathbf{u}_{j_1,\ldots,j_{\ell-1},\bar{j_\ell}}, \ldots, \mathbf{u}_{j_1,\bar{j_2}}, \mathbf{u}_{\bar{j_1}}$ computed by algorithm $\mathsf{TAcc}_{\mathbf{A}}(R)$.

$\mathsf{TVerify}_{\mathbf{A}}\big(\mathbf{u}, \mathbf{d}, w\big)$. Let the given witness w be of the form:

$$w = \big((j_1, \ldots, j_\ell), (\mathbf{w}_\ell, \ldots, \mathbf{w}_1)\big) \in \{0,1\}^\ell \times \big(\{0,1\}^{nk}\big)^\ell.$$

The algorithm recursively computes the path $\mathbf{v}_\ell, \mathbf{v}_{\ell-1}, \ldots, \mathbf{v}_1, \mathbf{v}_0 \in \{0,1\}^{nk}$ as follows: $\mathbf{v}_\ell = \mathbf{d}$ and

$$\forall i \in \{\ell-1, \ldots, 1, 0\} : \ \mathbf{v}_i = \begin{cases} h_{\mathbf{A}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}), & \text{if } j_{i+1} = 0; \\ h_{\mathbf{A}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}), & \text{if } j_{i+1} = 1. \end{cases} \tag{1}$$

Then it returns 1 if $\mathbf{v}_0 = \mathbf{u}$. Otherwise, it returns 0.

The following lemma states the correctness and security of the above Merkle tree accumulator.

Lemma 1 [28]. *The given accumulator scheme is correct and is secure in the sense of Definition 3, assuming the hardness of the* $\mathsf{SIS}^{\infty}_{n,m,q,1}$ *problem.*

3.3 An Efficient Updating Algorithm

Unlike the static group signature scheme from [28], our fully dynamic construction of Sect. 4 requires to regularly edit the accumulated values without having to reconstruct the whole tree. To this end, we equip the Merkle tree accumulator from [28] with a simple, yet efficient, updating algorithm: to change the value at a given leaf, we simply modify all values in the path from that leaf to the root. The algorithm, which takes as input a bit string $\mathsf{bin}(j) = (j_1, j_2, \ldots, j_\ell)$ and a value $\mathbf{d}^* \in \{0,1\}^{nk}$, is formally described below.

Given the tree in Sect. 3.2, algorithm $\mathsf{TUpdate}_{\mathbf{A}}((j_1, j_2, \ldots, j_\ell), \mathbf{d}^*)$ performs the following steps:

1. Let $\mathbf{d}_j$ be the current value at the leaf of position determined by $\mathsf{bin}(j)$, and let $((j_1, \ldots, j_\ell), (\mathbf{w}_{j,\ell}, \ldots, \mathbf{w}_{j,1}))$ be its associated witness.

2. Set $\mathbf{v}_\ell := \mathbf{d}^*$ and recursively compute the path $\mathbf{v}_\ell, \mathbf{v}_{\ell-1}, \ldots, \mathbf{v}_1, \mathbf{v}_0 \in \{0,1\}^{nk}$ as in (1).
4. Set $\mathbf{u} := \mathbf{v}_0$; $\mathbf{u}_{j_1} := \mathbf{v}_1; \ldots; \mathbf{u}_{j_1,j_2,\ldots,j_{\ell-1}} := \mathbf{v}_{\ell-1}$; $\mathbf{u}_{j_1,j_2,\ldots,j_\ell} := \mathbf{v}_\ell = \mathbf{d}^*$.

It can be seen that the provided algorithm runs in time $\mathcal{O}(\ell) = \mathcal{O}(\log N)$. In Fig. 1, we give an illustrative example of a tree with $2^3 = 8$ leaves.

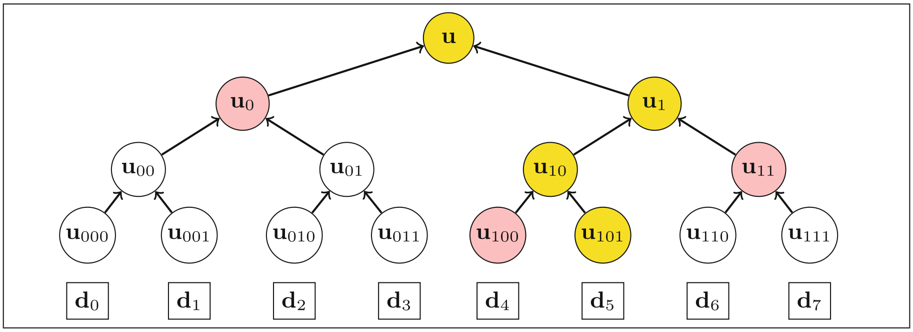


Fig. 1. A Merkle tree with $2^3 = 8$ leaves, which accumulates the data blocks $\mathbf{d}_0, \ldots, \mathbf{d}_7$ into the value $\mathbf{u}$ at the root. The bit string (101) and the pink nodes form a witness to the fact that $\mathbf{d}_5$ is accumulated in $\mathbf{u}$. If we replace $\mathbf{d}_5$ by a new value $\mathbf{d}^*$, we only need to update the yellow nodes. (Color figure online)

4 Our Fully Dynamic Group Signatures from Lattices

In this section, we construct our lattice-based fully dynamic group signature and prove its security in Bootle et al.'s model [7]. We start with the LLNW scheme [28], which works in the static setting.

While other constructions of lattice-based group signatures employ trapdoor-based ordinary signature schemes (e.g., [9,13]) to certify users, the LLNW scheme relies on a SIS-based Merkle tree accumulator which we recalled in Sect. 3.2. The GM, who manages a group of $N = 2^\ell$ users, chooses uniformly random vectors $\mathbf{x}_0, \ldots, \mathbf{x}_{N-1} \in \{0,1\}^m$; hashes them to $\mathbf{p}_0, \ldots, \mathbf{p}_{N-1} \in \{0,1\}^{nk}$, respectively; builds a tree on top of these hash values; and lets the tree root $\mathbf{u} \in \{0,1\}^{nk}$ be part of the group public key. The signing key of user i consists of $\mathbf{x}_i$ and the witness for the fact that $\mathbf{p}_i$ was accumulated in $\mathbf{u}$. When generating group signatures, the user proves knowledge of a valid pair $(\mathbf{x}_i, \mathbf{p}_i)$ and of the tree path from $\mathbf{p}_i$ to $\mathbf{u}$. The user also has to encrypt the binary representation $\mathsf{bin}(i)$ of his identity i, and prove that the ciphertext is well-formed. The encryption layer utilizes the Naor-Yung double-encryption paradigm [38] with Regev's LWE-based cryptosystem, and thus, it is also lattice-trapdoor-free.

To upgrade the LLNW scheme directly into a fully dynamic group signature, some tweaks and new ideas are needed. First, to enable the non-frameability feature, we let the user compute the pair $(\mathbf{x}_i, \mathbf{p}_i)$ on his own. The second problem we have to consider is that Merkle hash trees seem to be a static primitive. To this end, we equip the accumulator with an efficient updating algorithm (see Sect. 3.3). Now, the challenging question is how to handle user enrollment and revocation in a simple manner (i.e., without having to reset the whole tree). To tackle these issues, we associate each of the N potential users with a leaf in the tree, and then use the updating algorithm to set up the system so that:

1. If a user has not joined the group or has been revoked, the value at the leaf associated with him is set as $\mathbf{0}$;
2. When a user joins the group, that value is set as his public key $\mathbf{p}_i$.

Our setup guarantees that only active users (i.e., who has joined and has not been revoked at the given epoch) have their *non-zero* public keys accumulated into the updated root. This effectively gives us a method to separate active users who can sign from those who cannot: when signing messages, the user proceeds as in the LLNW scheme, and is asked to additionally prove in ZK that $\mathbf{p}_i \neq \mathbf{0}$.

At this point, the arising question is how to prove the inequality $\mathbf{p}_i \neq \mathbf{0}$ in the framework of the Stern-like [45] protocol from [28]. One would naturally hope that this extra job could be done without losing too much in terms of efficiency. Here, the surprising and somewhat unexpected fact is that we can actually do it while *gaining* efficiency, thanks to a technique originally proposed in [33].

To begin with, let B_t^L denote the set of all vectors in $\{0,1\}^L$ having Hamming weight exactly t. In Stern-like protocols (see Sect. 2.3), a common technique for proving in ZK the possession of $\mathbf{p} \in \{0,1\}^{nk}$ consists of appending nk "dummy" entries to it to obtain $\mathbf{p}^* \in \mathsf{B}_{nk}^{2nk}$, and demonstrating to the verifier that a random permutation of $\mathbf{p}^*$ belongs to the "target set" B_{nk}^{2nk}. This suffices to convince the verifier that the original vector $\mathbf{p}$ belongs to $\{0,1\}^{nk}$, while the latter cannot learn any additional information about $\mathbf{p}$, thanks to the randomness of the permutation. This extending-then-permuting technique was first proposed in [33], and was extensively used in the underlying protocol of the LLNW scheme. Now, to address our question, we will employ a modified version of this technique, which was also initially suggested in [33]. Let us think of another "target set", so that it is possible to extend $\mathbf{p} \in \{0,1\}^{nk}$ to an element of that set if and only if $\mathbf{p}$ is non-zero. That set is B_{nk}^{2nk-1}. Indeed, the extended vector $\mathbf{p}^*$ belongs to B_{nk}^{2nk-1} if and only if the original vector has Hamming weight at least $nk - (nk-1) = 1$, which means that it cannot be a zero-vector. When combining with the permuting step, this modification allows us to additionally prove the given inequality while working with smaller dimension. As a result, our fully dynamic scheme produces slightly shorter signatures than the original static scheme.

Finally, we remark that the fully dynamic setting requires a proof of correct opening, which boils down to proving correct decryption for Regev's encryption scheme. It involves modular linear equations with bounded-norm secrets, and can be easily handled using Stern-like techniques from [26,33].

4.1 Description of the Scheme

Our scheme is described as follows.

$\mathsf{GSetup}(\lambda)$. On input security parameter λ, this algorithm specifies the following:

- An expected number of potential users $N = 2^\ell = \mathsf{poly}(\lambda)$.
- Dimension $n = \mathcal{O}(\lambda)$, prime modulus $q = \widetilde{\mathcal{O}}(n^{1.5})$, and $k = \lceil \log_2 q \rceil$. These parameters implicitly determine the "powers-of-2" matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$, as defined in Sect. 3.
- Matrix dimensions $m = 2nk$ for the hashing layer, and $m_E = 2(n+\ell)k$ for the encryption layer.
- An integer $\beta = \sqrt{n} \cdot \omega(\log n)$, and a β-bounded noise distribution χ.
- A hash function $\mathcal{H}_{\mathsf{FS}} : \{0,1\}^* \rightarrow \{1,2,3\}^\kappa$, where $\kappa = \omega(\log \lambda)$, to be modelled as a random oracle in the Fiat-Shamir transformations [17].
- Let $\mathsf{COM} : \{0,1\}^* \times \{0,1\}^m \rightarrow \mathbb{Z}_q^n$ be the string commitment scheme from [22], to be used in our zero-knowledge argument systems.
- Uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$.

The algorithm outputs public parameters

$$\mathsf{pp} = \{\lambda, N, n, q, k, m, m_E, \ell, \beta, \chi, \kappa, \mathcal{H}_{\mathsf{FS}}, \mathsf{COM}, \mathbf{A}\}.$$

$\langle \mathsf{GKgen}_{\mathsf{GM}}(\mathsf{pp}), \mathsf{GKgen}_{\mathsf{TM}}(\mathsf{pp}) \rangle$. The group manager GM and the tracing manager TM initialize their keys and the public group information as follows.

- $\mathsf{GKgen}_{\mathsf{GM}}(\mathsf{pp})$. This algorithm samples $\mathsf{msk} \xleftarrow{\$} \{0,1\}^m$ and computes $\mathsf{mpk} = \mathbf{A} \cdot \mathsf{msk} \bmod q$, and outputs $(\mathsf{mpk}, \mathsf{msk})$. Here, we consider mpk as an identifier of the group managed by GM who has mpk as his public key. Furthermore, as in [7, Sect. 3.3, full version], we assume that the group information board is visible to everyone, but can only be edited by a party knowing msk.
- $\mathsf{GKgen}_{\mathsf{TM}}(\mathsf{pp})$. This algorithm initializes the Naor-Yung double-encryption mechanism with the ℓ-bit version Regev encryption scheme. It first chooses $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times m_E}$. For each $i \in \{1,2\}$, it samples $\mathbf{S}_i \xleftarrow{\$} \chi^{n \times \ell}$, $\mathbf{E}_i \hookleftarrow \chi^{\ell \times m_E}$, and computes $\mathbf{P}_i = \mathbf{S}_i^\top \cdot \mathbf{B} + \mathbf{E}_i \in \mathbb{Z}_q^{\ell \times m_E}$. Then, TM sets $\mathsf{tsk} = (\mathbf{S}_1, \mathbf{E}_1)$, and $\mathsf{tpk} = (\mathbf{B}, \mathbf{P}_1, \mathbf{P}_2)$.
- TM sends tpk to GM who initializes the following:
  - Table $\mathbf{reg} := (\mathbf{reg}[0][1], \mathbf{reg}[0][2], \ldots, \mathbf{reg}[N-1][1], \mathbf{reg}[N-1][2])$, where for each $i \in [0, N{-}1]$: $\mathbf{reg}[i][1] = \mathbf{0}^{nk}$ and $\mathbf{reg}[i][2] = 0$. Looking ahead, $\mathbf{reg}[i][1]$ will be used to record the public key of a registered user, while $\mathbf{reg}[i][2]$ stores the epoch at which the user joins.
  - The Merkle tree $\mathcal{T}$ built on top of $\mathbf{reg}[0][1], \ldots, \mathbf{reg}[N-1][1]$. (Note that $\mathcal{T}$ is an all-zero tree at this stage, but it will be modified when a new user joins the group, or when GM computes the updated group information.)
  - Counter of registered users $c := 0$.

  Then, GM outputs $\mathsf{gpk} = (\mathsf{pp}, \mathsf{mpk}, \mathsf{tpk})$ and announces the initial group information $\mathsf{info} = \emptyset$. He keeps $\mathcal{T}$ and c for himself.

$\mathsf{UKgen}(\mathsf{pp})$. Each potential group user samples $\mathsf{usk} = \mathbf{x} \xleftarrow{\$} \{0,1\}^m$, and computes $\mathsf{upk} = \mathbf{p} = \mathrm{bin}(\mathbf{A} \cdot \mathbf{x}) \bmod q \in \{0,1\}^{nk}$.

Without loss of generality, we assume that every honestly generated upk is a non-zero vector. (Otherwise, the user would either pick $\mathbf{x} = \mathbf{0}^m$ or accidentally find a solution to the $\mathsf{SIS}^{\infty}_{n,m,q,1}$ problem associated with matrix $\mathbf{A}$ - both happen with negligible probability.)

$\langle \mathsf{Join}, \mathsf{Issue} \rangle$. If the user with key pair $(\mathsf{upk}, \mathsf{usk}) = (\mathbf{p}, \mathbf{x})$ requests to join the group at epoch τ, he sends $\mathbf{p}$ to GM. If the latter accepts the request, then the two parties proceed as follows.

1. GM issues a member identifier for the user as $\mathsf{uid} = \mathsf{bin}(c) \in \{0,1\}^{\ell}$. The user then sets his long-term signing key as $\mathsf{gsk}[c] = (\mathsf{bin}(c), \mathbf{p}, \mathbf{x})$.
2. GM performs the following updates:
   – Update $\mathcal{T}$ by running algorithm $\mathsf{TUpdate}_{\mathbf{A}}(\mathsf{bin}(c), \mathbf{p})$.
   – Register the user to table $\mathbf{reg}$ as $\mathbf{reg}[c][1] := \mathbf{p}$; $\mathbf{reg}[c][2] := \tau$.
   – Increase the counter $c := c + 1$.

$\mathsf{GUpdate}(\mathsf{gpk}, \mathsf{msk}, \mathsf{info}_{\tau_{\mathrm{current}}}, \mathcal{S}, \mathbf{reg})$. This algorithm is run by GM to update the group information while also advancing the epoch. It works as follows.

1. Let the set $\mathcal{S}$ contain the public keys of registered users to be revoked. If $\mathcal{S} = \emptyset$, then go to Step 2.
   Otherwise, $\mathcal{S} = \{\mathbf{reg}[i_1][1], \ldots, \mathbf{reg}[i_r][1]\}$, for some $r \in [1, N]$ and some $i_1, \ldots, i_r \in [0, N{-}1]$. Then, for all $t \in [r]$, GM runs $\mathsf{TUpdate}_{\mathbf{A}}(\mathsf{bin}(i_t), \mathbf{0}^{nk})$ to update the tree $\mathcal{T}$.
2. At this point, by construction, each of the zero leaves in the tree $\mathcal{T}$ corresponds to either a revoked user or a potential user who has not yet registered. In other words, only active users who are allowed to sign in the new epoch τ_{new} have their non-zero public keys, denoted by $\{\mathbf{p}_j\}_j$, accumulated in the root $\mathbf{u}_{\tau_{\mathrm{new}}}$ of the updated tree.
   For each j, let $w_j \in \{0,1\}^{\ell} \times (\{0,1\}^{nk})^{\ell}$ be the witness for the fact that $\mathbf{p}_j$ is accumulated in $\mathbf{u}_{\tau_{\mathrm{new}}}$. Then GM publishes the group information of the new epoch as:
   $$\mathsf{info}_{\tau_{\mathrm{new}}} = \big(\mathbf{u}_{\tau_{\mathrm{new}}}, \{w_j\}_j\big).$$

We remark that the info_{τ} outputted at each epoch by GM is technically not part of the verification key. Indeed, as we will describe below, in order to verify signatures bound to epoch τ, the verifiers only need to download the first component $\mathbf{u}_{\tau}$ of size $\widetilde{\mathcal{O}}(\lambda)$ bits. Meanwhile, each active signer only has to download the respective witness of size $\widetilde{\mathcal{O}}(\lambda) \cdot \ell$.

$\mathsf{Sign}(\mathsf{gpk}, \mathsf{gsk}[j], \mathsf{info}_{\tau}, M)$**.** To sign message M using the group information info_{τ} at epoch τ, the user possessing $\mathsf{gsk}[j] = (\mathsf{bin}(j), \mathbf{p}, \mathbf{x})$ first checks if info_{τ} includes a witness containing $\mathsf{bin}(j)$. If this is not the case, return $\perp$. Otherwise, the user downloads $\mathbf{u}_{\tau}$ and the witness of the form $\big(\mathsf{bin}(j), (\mathbf{w}_{\ell}, \ldots, \mathbf{w}_1)\big)$ from info_{τ}, and proceeds as follows.

1. Encrypt vector $\mathsf{bin}(j) \in \{0,1\}^{\ell}$ twice using Regev's encryption scheme. Namely, for each $i \in \{1,2\}$, sample $\mathbf{r}_i \xleftarrow{\$} \{0,1\}^{m_E}$ and compute
   $$\begin{aligned}\mathbf{c}_i &= (\mathbf{c}_{i,1}, \mathbf{c}_{i,2})\\ &= \Big(\mathbf{B} \cdot \mathbf{r}_i \bmod q,\ \mathbf{P}_i \cdot \mathbf{r}_i + \big\lfloor \frac{q}{2} \big\rfloor \cdot \mathsf{bin}(j) \bmod q\Big) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{\ell}.\end{aligned}$$

2. Generate a $\mathsf{NIZKAoK}$ $\mathit{\Pi}_{\mathsf{gs}}$ to demonstrate the possession of a valid tuple

$$\zeta = (\mathbf{x}, \mathbf{p}, \mathsf{bin}(j), \mathbf{w}_\ell, \ldots, \mathbf{w}_1, \mathbf{r}_1, \mathbf{r}_2) \tag{2}$$

   such that:
   (i) $\mathsf{TVerify}_{\mathbf{A}}\big(\mathbf{u}_\tau, \mathbf{p}, \big(\mathsf{bin}(j), (\mathbf{w}_\ell, \ldots, \mathbf{w}_1)\big)\big) = 1$ and $\mathbf{A} \cdot \mathbf{x} = \mathbf{G} \cdot \mathbf{p} \bmod q$;
   (ii) $\mathbf{c}_1$ and $\mathbf{c}_2$ are both correct encryptions of $\mathsf{bin}(j)$ with randomness $\mathbf{r}_1$ and $\mathbf{r}_2$, respectively;
   (iii) $\mathbf{p} \neq \mathbf{0}^{nk}$.

   Note that statements (i) and (ii) were covered by the LLNW protocol [28]. Meanwhile, statement (iii) is handled using the technique described at the beginning of this Section. We thus obtain a Stern-like interactive zero-knowledge argument system which is a slight modification of the one from [28]. Due to space restriction, the details are presented in the full version.

   The protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via the Fiat-Shamir heuristic as a triple $\mathit{\Pi}_{\mathsf{gs}} = (\{\mathrm{CMT}_i\}_{i=1}^{\kappa}, \mathrm{CH}, \{\mathrm{RSP}\}_{i=1}^{\kappa})$, where

$$\mathrm{CH} = \mathcal{H}_{\mathsf{FS}}\big(M, (\{\mathrm{CMT}_i\}_{i=1}^{\kappa}, \mathbf{A}, \mathbf{u}_\tau, \mathbf{B}, \mathbf{P}_1, \mathbf{P}_2, \mathbf{c}_1, \mathbf{c}_2\big) \in \{1,2,3\}^{\kappa}.$$

3. Output the group signature

$$\mathit{\Sigma} = (\mathit{\Pi}_{\mathsf{gs}}, \mathbf{c}_1, \mathbf{c}_2). \tag{3}$$

$\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}_\tau, M, \mathit{\Sigma})$. This algorithm proceeds as follows:

1. Download $\mathbf{u}_\tau \in \{0,1\}^{nk}$ from info_τ.
2. Parse $\mathit{\Sigma}$ as $\mathit{\Sigma} = \big(\{\mathrm{CMT}_i\}_{i=1}^{\kappa}, (Ch_1, \ldots, Ch_\kappa), \{\mathrm{RSP}\}_{i=1}^{\kappa}, \mathbf{c}_1, \mathbf{c}_2\big)$.
   If $(Ch_1, \ldots, Ch_\kappa) \neq \mathcal{H}_{\mathsf{FS}}\big(M, (\{\mathrm{CMT}_i\}_{i=1}^{\kappa}, \mathbf{A}, \mathbf{u}_\tau, \mathbf{B}, \mathbf{P}_1, \mathbf{P}_2, \mathbf{c}_1, \mathbf{c}_2\big)$, then return 0.
3. For each $i = 1$ to κ, run the verification phase of the protocol presented in the full version to check the validity of RSP_i with respect to CMT_i and Ch_i. If any of the conditions does not hold, then return 0.
4. Return 1.

$\mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{info}_\tau, \mathbf{reg}, M, \mathit{\Sigma})$. This algorithm parses tsk as $(\mathbf{S}_1, \mathbf{E}_1)$, parses $\mathit{\Sigma}$ as in (3), and performs the following steps.

1. Use $\mathbf{S}_1$ to decrypt $\mathbf{c}_1 = (\mathbf{c}_{1,1}, \mathbf{c}_{1,2})$ to obtain a string $\mathbf{b}' \in \{0,1\}^\ell$ (i.e., by computing $\lfloor(\mathbf{c}_{1,2} - \mathbf{S}_1^\top \cdot \mathbf{c}_{1,1})/(q/2)\rceil$.
2. If info_τ does not include a witness containing $\mathbf{b}'$, then return $\perp$.
3. Let $j' \in [0, N-1]$ be the integer having binary representation b'. If the record $\mathbf{reg}[j'][1]$ in table $\mathbf{reg}$ is $\mathbf{0}^{nk}$, then return $\perp$.
4. Generate a $\mathsf{NIZKAoK}$ $\mathit{\Pi}_{\mathsf{trace}}$ to demonstrate the possession of $\mathbf{S}_1 \in \mathbb{Z}^{n \times \ell}$, $\mathbf{E}_1 \in \mathbb{Z}^{\ell \times m_E}$, and $\mathbf{y} \in \mathbb{Z}^\ell$, such that:

$$\begin{cases} \|\mathbf{S}_1\|_\infty \leq \beta;\ \|\mathbf{E}_1\|_\infty \leq \beta;\ \|\mathbf{y}\|_\infty \leq \lceil q/5 \rceil; \\ \mathbf{S}_1^\top \cdot \mathbf{B} + \mathbf{E}_1 = \mathbf{P}_1 \bmod q; \\ \mathbf{c}_{1,2} - \mathbf{S}_1^\top \cdot \mathbf{c}_{1,1} = \mathbf{y} + \lfloor q/2 \rfloor \cdot \mathbf{b}' \bmod q. \end{cases} \tag{4}$$

As the statement involves modular linear equations with bounded-norm secrets, we can obtain a statistical zero-knowledge argument by employing the Stern-like interactive protocol from [26]. The protocol is repeated $\kappa = \omega(\log \lambda)$ times to achieve negligible soundness error and made non-interactive via the Fiat-Shamir heuristic as a triple $\Pi_{\mathsf{trace}} = (\{\mathrm{CMT}_i\}_{i=1}^{\kappa}, \mathrm{CH}, \{\mathrm{RSP}\}_{i=1}^{\kappa})$, where

$$\mathrm{CH} = \mathcal{H}_{\mathsf{FS}}\big((\{\mathrm{CMT}_i\}_{i=1}^{\kappa}, \mathsf{gpk}, \mathsf{info}_\tau, M, \Sigma, \mathbf{b}'\big) \in \{1,2,3\}^{\kappa}. \tag{5}$$

5. Set $\mathsf{uid} = \mathbf{b}'$ and output $(\mathsf{uid}, \Pi_{\mathsf{trace}})$.

$\mathsf{Judge}(\mathsf{gpk}, \mathsf{uid}, \mathsf{info}_\tau, \Pi_{\mathsf{trace}}, M, \Sigma)$. This algorithm consists of verifying the argument Π_{trace} w.r.t. common input $(\mathsf{gpk}, \mathsf{info}_\tau, M, \Sigma, \mathsf{uid})$, in a similar manner as in algorithm Verify.

If Π_{trace} does not verify, return 0. Otherwise, return 1.

4.2 Analysis of the Scheme

Efficiency. We first analyze the efficiency of the scheme described in Sect. 4.1, with respect to security parameter λ and parameter $\ell = \log N$.

- The public key gpk contains several matrices, and has bit-size $\widetilde{\mathcal{O}}(\lambda^2 + \lambda \cdot \ell)$.
- For each $j \in [0, N-1]$, the signing key $\mathsf{gsk}[j]$ has bit-size $\ell + nk + m = \widetilde{\mathcal{O}}(\lambda) + \ell$.
- At each epoch, the signature verifiers downloads $nk = \widetilde{\mathcal{O}}(\lambda)$ bits, while each active signer downloads $\widetilde{\mathcal{O}}(\lambda \cdot \ell)$ bits.
- The size of signature Σ is dominated by that of the Stern-like $\mathsf{NIZKAoK}$ Π_{gs}, which is $\mathcal{O}(|\zeta| \cdot \log q) \cdot \omega(\log \lambda)$, where $|\zeta|$ denotes the bit-size of the witness-tuple ζ in (2). Overall, Σ has bit-size $\widetilde{\mathcal{O}}(\lambda \cdot \ell)$.
- The Stern-like $\mathsf{NIZKAoK}$ Π_{trace} has bit-size $\widetilde{\mathcal{O}}(\ell^2 + \lambda \cdot \ell)$.

Correctness. We now demonstrate that the scheme is correct with overwhelming probability, based on the perfect completeness of Stern-like protocols, and the correctness of Regev's encryption scheme.

First, note that a signature $\Sigma = (\Pi_{\mathsf{gs}}, \mathbf{c}_1, \mathbf{c}_2)$ generated by an active and honest user j is always accepted by algorithm Verify. Indeed, such a user can always compute a tuple $\zeta = (\mathbf{x}, \mathbf{p}, \mathsf{bin}(j), \mathbf{w}_\ell, \ldots, \mathbf{w}_1, \mathbf{r}_1, \mathbf{r}_2)$ satisfying conditions (i),(ii) and (iii) in the Sign algorithm. The completeness of the underlying argument system then guarantees that Σ is always accepted by algorithm Verify.

Next, we show that algorithm Trace outputs $\mathsf{bin}(j)$ with overwhelming probability, and produces a proof Π_{trace} accepted by algorithm Judge. Observe that, the decryption algorithm essentially computes

$$\mathbf{e} = \mathbf{c}_{1,2} - \mathbf{S}_1^T \mathbf{c}_{1,1} = \mathbf{E}_1 \cdot \mathbf{r}_1 + \lfloor q/2 \rceil \cdot \mathsf{bin}(j) \mod q,$$

and sets the j-th bit of $\mathbf{b}'$ to be 0 if j-th entry of $\mathbf{e}$ is closer to 0 than to $\lfloor q/2 \rceil$ and 1 otherwise. Note that our parameters are set so that $\|\mathbf{E}_1 \cdot \mathbf{r}_1\|_\infty < q/5$, for $\mathbf{E}_1 \hookleftarrow \chi^{\ell \times m_E}$ and $\mathbf{r}_1 \xleftarrow{\$} \{0,1\}^{m_E}$. This ensures that $\mathbf{b}' = \mathsf{bin}(j)$ with overwhelming probability.

Further, as the user is active, info_{τ} must contain $w = (\mathsf{bin}(j), \mathbf{w}_{\ell}, \ldots, \mathbf{w}_1)$ and $\mathbf{reg}[j][1]$ in table $\mathbf{reg}$ is not $\mathbf{0}^{nk}$. Therefore, algorithm Trace will move to the 4-th step, where it can always obtain the tuple $(\mathbf{S}_1, \mathbf{E}_1, \mathbf{y})$ satisfying the conditions (4). By the completeness of the argument system, Π_{trace} will be accepted by the algorithm Judge.

SECURITY. In Theorem 1, we prove that our scheme satisfies the security requirements of the Bootle et al.'s model [7].

Theorem 1. *Assume that the Stern-like argument systems used in Sect. 4.1 are simulation-sound. Then, in the random oracle model, the given fully dynamic group signature satisfies the anonymity, traceability, non-frameability and tracing soundness requirements under the* $\mathsf{LWE}_{n,q,\chi}$ *and* $\mathsf{SIS}^{\infty}_{n,m,q,1}$ *assumptions.*

In the random oracle model, the proof of Theorem 1 relies on the following facts:

1. The Stern-like zero-knowledge argument systems being used are simulation-sound;
2. The underlying encryption scheme, which is obtained from Regev cryptosystem [42] via the Naor-Yung transformation [38], is IND-CCA2 secure;
3. The Merkle tree accumulator we employ is secure in the sense of Definition 3;
4. For a properly generated key-pair $(\mathbf{x}, \mathbf{p})$, it is infeasible to find $\mathbf{x}' \in \{0,1\}^m$ such that $\mathbf{x}' \neq \mathbf{x}$ and $\mathsf{bin}(\mathbf{A} \cdot \mathbf{x}' \bmod q) = \mathbf{p}$.

Due to space restriction, details of the proof of Theorem 1 are provided in the full version of the paper.

Acknowledgements. The authors would like to thank Benoît Libert and the anonymous reviewers for helpful comments and discussions. The research was supported by Research Grant TL-9014101684-01 and the Singapore Ministry of Education under Research Grant MOE2013-T2-1-041.

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: STOC 1996, pp. 99–108. ACM (1996)
2. Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000). doi:10.1007/3-540-44598-6_16
3. Barić, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_33
4. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003). doi:10.1007/3-540-39200-9_38

5. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: the case of dynamic groups. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 136–153. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30574-3_11
6. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM CCCS 2004, pp. 168–177. ACM (2004)
7. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J.: Foundations of fully dynamic group signatures. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 117–136. Springer, Cham (2016). doi:10.1007/978-3-319-39555-5_7. Full version: https://eprint.iacr.org/2016/368.pdf
8. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J., Petit, C.: Short accountable ring signatures based on DDH. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 243–265. Springer, Cham (2015). doi:10.1007/978-3-319-24174-6_13
9. Boyen, X.: Lattice mixing and vanishing trapdoors: a framework for fully secure short signatures and more. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13013-7_29
10. Bresson, E., Stern, J.: Efficient revocation in group signatures. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 190–206. Springer, Heidelberg (2001). doi:10.1007/3-540-44586-2_15
11. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002). doi:10.1007/3-540-45708-9_5
12. Camenisch, J., Neven, G., Rückert, M.: Fully anonymous attribute tokens from lattices. In: Visconti, I., Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 57–75. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32928-9_4
13. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_27
14. Chaum, D., Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991). doi:10.1007/3-540-46416-6_22
15. Cheng, S., Nguyen, K., Wang, H.: Policy-based signature scheme from lattices. Des. Codes Cryptogr. **81**(1), 43–74 (2016)
16. Delerablée, C., Pointcheval, D.: Dynamic fully anonymous short group signatures. In: Nguyen, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 193–210. Springer, Heidelberg (2006). doi:10.1007/11958239_13
17. Fiat, A., Shamir, A.: How To prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). doi:10.1007/3-540-47721-7_12
18. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC 2008, pp. 197–206. ACM (2008)
19. Gordon, S.D., Katz, J., Vaikuntanathan, V.: A group signature scheme from lattice assumptions. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 395–412. Springer, Heidelberg (2010). doi:10.1007/978-3-642-17373-8_23
20. Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76900-2_10
21. Kawachi, A., Tanaka, K., Xagawa, K.: Multi-bit cryptosystems based on lattice problems. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 315–329. Springer, Heidelberg (2007). doi:10.1007/978-3-540-71677-8_21

22. Kawachi, A., Tanaka, K., Xagawa, K.: Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 372–389. Springer, Heidelberg (2008). doi:10.1007/978-3-540-89255-7_23
23. Kiayias, A., Yung, M.: Secure scalable group signature with dynamic joins and separable authorities. Int. J. Secur. Netw. **1**(1), 24–45 (2006)
24. Laguillaumie, F., Langlois, A., Libert, B., Stehlé, D.: Lattice-based group signatures with logarithmic signature size. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 41–61. Springer, Heidelberg (2013). doi:10.1007/978-3-642-42045-0_3
25. Langlois, A., Ling, S., Nguyen, K., Wang, H.: Lattice-based group signature scheme with verifier-local revocation. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 345–361. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54631-0_20
26. Libert, B., Ling, S., Mouhartem, F., Nguyen, K., Wang, H.: Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 373–403. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53890-6_13
27. Libert, B., Ling, S., Mouhartem, F., Nguyen, K., Wang, H.: Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 101–131. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53890-6_4
28. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: logarithmic-size ring signatures and group signatures without trapdoors. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 1–31. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49896-5_1
29. Libert, B., Mouhartem, F., Nguyen, K.: A lattice-based group signature scheme with message-dependent opening. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 137–155. Springer, Cham (2016). doi:10.1007/978-3-319-39555-5_8
30. Libert, B., Peters, T., Yung, M.: Group signatures with almost-for-free revocation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 571–589. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_34
31. Libert, B., Peters, T., Yung, M.: Scalable group signatures with revocation. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 609–627. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29011-4_36
32. Libert, B., Peters, T., Yung, M.: Short group signatures via structure-preserving signatures: standard model security from simple assumptions. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 296–316. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48000-7_15
33. Ling, S., Nguyen, K., Stehlé, D., Wang, H.: Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 107–124. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36362-7_8
34. Ling, S., Nguyen, K., Wang, H.: Group signatures from lattices: simpler, tighter, shorter, ring-based. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 427–449. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46447-2_19
35. Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 465–484. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22792-9_26

36. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 21–39. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_2
37. Nakanishi, T., Fujii, H., Hira, Y., Funabiki, N.: Revocable group signature schemes with constant costs for signing and verifying. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 463–480. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00468-1_26
38. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC 1990, pp. 427–437. ACM (1990)
39. Nguyen, L.: Accumulators from bilinear pairings and applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30574-3_19
40. Nguyen, L., Safavi-Naini, R.: Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 372–386. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30539-2_26
41. Nguyen, P.Q., Zhang, J., Zhang, Z.: Simpler efficient group signatures from lattices. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 401–426. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46447-2_18
42. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC 2005, pp. 84–93. ACM (2005)
43. Sakai, Y., Emura, K., Hanaoka, G., Kawai, Y., Matsuda, T., Omote, K.: Group signatures with message-dependent opening. In: Abdalla, M., Lange, T. (eds.) Pairing 2012. LNCS, vol. 7708, pp. 270–294. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36334-4_18
44. Sakai, Y., Schuldt, J.C.N., Emura, K., Hanaoka, G., Ohta, K.: On the security of dynamic group signatures: preventing signature hijacking. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 715–732. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30057-8_42
45. Stern, J.: A new paradigm for public key identification. IEEE Trans. Inf. Theory **42**(6), 1757–1768 (1996)
46. Tsudik, G., Xu, S.: Accumulating composites and improved group signing. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 269–286. Springer, Heidelberg (2003). doi:10.1007/978-3-540-40061-5_16

Breaking and Fixing Mobile App Authentication with OAuth2.0-based Protocols

Ronghai Yang(✉), Wing Cheong Lau, and Shangcheng Shi

Department of Information Engineering, The Chinese University of Hong Kong,
Hong Kong, China
{yr013,wclau,ss016}@ie.cuhk.edu.hk

Abstract. Although the OAuth2.0 protocol was originally designed to serve the authorization need for websites, mainstream identity providers like Google and Facebook have made significant changes on this protocol to support authentication for mobile apps. Prior research mainly focuses on how the features of mobile operating systems can affect the OAuth security. However, little has been done to analyze whether these significant modifications of the protocol call-flow can be well understood and implemented by app developers. Towards this end, we report a field-study on the Android OAuth2.0-based single-sign-on systems. In particular, we perform an in-depth static code analysis on three identity provider apps including Facebook, Google and Sina as well as their official SDKs to understand their OAuth-related transactions. We then dynamically test 600 top-ranked US and Chinese Android apps. Apart from various types of existing vulnerabilities, we also discover three previously unknown security flaws among these first-tier identity providers and a large number of popular 3rd-party apps. For example, 41% apps under study are susceptible to a newly discovered profile attack, which unlike prior works, enables remote account hijacking without any need to trick or interact with the victim. The prevalence of vulnerabilities further motivates us to propose/implement an alternative, fool-proof OAuth SDK for one of the affected IdPs to automatically prevent from these vulnerabilities. To facilitate the adoption of our proposed fixes, our solution requires minimal code changes by the 3rd-party-developers of the affected mobile apps.

Keywords: OAuth2.0 · OpenID Connect · Mobile app authentication

1 Introduction

The OAuth2.0 protocol was originally designed to serve the authorization need for 3rd-party websites. However, many major Identity Providers (IdPs) such as Facebook, Google and Sina, have recently adapted the OAuth2.0-based protocols to support Single-Sign-On (SSO) services for 3rd-party mobile apps (which take the role of Relying Party under the context of OAuth2.0). When OAuth2.0 is used as a SSO scheme, a user can log into the mobile Relying Party (RP), *e.g.*,

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 313–335, 2017.
DOI: 10.1007/978-3-319-61204-1_16

IMDB, via the IdP without sharing the identity credential with the RP. In this paper, we will focus on OAuth2.0 and OpenID Connect (which is built on top of OAuth2.0), since they are the *de facto* SSO standards.

To support SSO services with 3rd-party mobile apps, the security of the adapted OAuth2.0 protocol has been evaluated by the literature. Chen *et al.* [12] first point out that some operating-system-provided components (*e.g.*, Intent/WebView for Android) are required to implement OAuth2.0 for mobile platforms. As further shown by Chen, the features of these components, if not well understood by mobile app developers, can be leveraged to compromise mobile SSO systems. Following this work, Ye *et al.* [34] apply model checking method to theoretically evaluate this modified protocol and Wang *et al.* [28,29] summarize those known vulnerabilities among 15 Chinese IdPs over different platforms.

Prior studies mainly focus on how the differences of mobile systems (*i.e.*, vulnerabilities in the system-provided components) can compromise mobile SSO systems. However, the security implications resulting from the major protocol changes, when adapting the OAuth2.0-based protocols to mobile platforms, are often left out. For example, the standard OAuth2.0 implicit flow has shown to be insecure for authentication and thus a revised version, *i.e.*, a variant of OpenID Connect (OIDC), is recommended. Nevertheless, they do not consider that the SSO results, even in the revised version, are passed through the user device. Consequently, these security-critical results are subject to tampering and further enable an adversary to infer the program logic of the RP server.

Towards this end, the goal of this research is to further the understanding of (1) how the changes of OAuth[1] protocol flow, if not well implemented, can lead to nontrivial security flaws, (2) the overall security quality of mobile SSO systems by checking whether existing vulnerabilities have been fixed or not. Our work consists of two pillars: (1) We perform a standard static code analysis of three first-tier Android IdP apps (Facebook, Google and Sina) and their corresponding SDKs widely used by the RP to understand their client-side program logic. (2) We develop a tool to dynamically test Top-600 US and Chinese Android apps to see how well the RP/IdP servers perform the OAuth transactions. From these studies, we have made the following technical contributions in this paper:

- We have identified the security-critical changes of the OAuth protocol call-flow.
- We have examined the implementations of 3 first-tier IdPs and 600 top-ranked US/Chinese Android Apps. In addition to different types of existing vulnerabilities, we have discovered three previously unknown vulnerabilities resulting from the incorrect implementations of the protocol call-flow modification.
- We have designed and implemented a foolproof solution which prevents future 3rd party app developers from committing the same mistakes. To ease the transition, our solution only requires minimal changes in the 3rd-party developed codes of a vulnerable mobile app.

[1] We use OAuth to denote OAuth2.0 and OpenID Connect, if not specified otherwise.

2 Background

In the OAuth ecosystem, four parties are involved to support SSO for 3rd-party mobile apps, namely, the backend server of the 3rd-party mobile app (RP server, for short), the backend server of the IdP (IdP server), the 3rd-party client-side mobile app (RP app) and the client-side mobile app of the IdP (IdP app). For ease of presentation, in the rest of this paper, we use notations in the parentheses to denote these four parties and use OAuth to denote OAuth2.0 as well as OIDC, if not specified otherwise.

The ultimate goal of SSO is for the IdP server to issue an identity proof, *e.g.*, the access token for OAuth2.0 and the id_token for OIDC, to the RP server. With this identity proof, the RP server can determine the user's identity and then log the user in.

2.1 The Implicit Flow of OAuth 2.0 for Mobile Platforms

OAuth2.0 [18] defines four types of authorization flows, out of which the implicit flow and authorization code flow[2] are widely used by the mobile platform and the website, respectively. Thanks to the demystification by Chen *et al.* [12], the standard implicit flow has proven to be insecure for authentication under mobile platforms. After revising the standard protocol, one believed-to-be-secure realization is illustrated in Fig. 1, although neither the RFC nor the IdP provides a complete call-flow diagram.

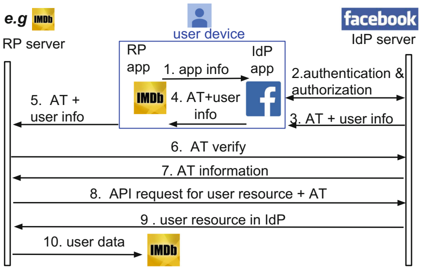


Fig. 1. The implicit flow of OAuth2.0 for mobile platforms

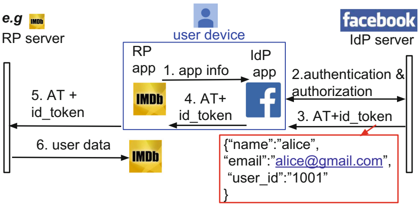


Fig. 2. The implicit flow of OpenID Connect

1. A user attempts to log into the RP with the IdP. The RP app sends its app information (*e.g.*, requested permissions) to the IdP app via a secure channel provided by the mobile operating system, for example the *Intent* channel in Android.

[2] In fact, the authorization code flow can also be securely used for mobile apps, but with the cost of worse performance.

2. Thanks to the secure channel, the IdP app can verify whether the RP app information is correct or not. If so, the IdP app then sends such information, as the authorization request, to the IdP server.
3. The IdP server compares information of the authorization request and that pre-registered by the RP developer. If it matches, the IdP server would believe the validity of the authorization request and thus issue an access token (AT) together with optional user information (*e.g.*, uid) to its own client-side app.
4. The IdP app returns the access token to the RP app via the secure channel maintained by the operating system.
5. The RP app sends the access token and user information to the RP server.
6. The RP server should call the security-focused SSO-API provided by the IdP to verify the access token.
7. If the access token is valid, the IdP server should respond the RP server with authorization information including which RP this access token is issued to.
8. Only if the access token belongs to this RP can the RP server accept it. Thereafter, the RP server can retrieve the user data with this verified access token.
9. The IdP server returns the user data associated with the access token.
10. The RP server can then identify the user and return his sensitive data to RP app.

2.2 The OpenID Connect Protocol

Since OAuth2.0 was originally designed to support authorization, to adapt it for authentication, it involves multiple high-latency round trips, *i.e.*, Step 6–Step 9 of Fig. 1. To support authentication more efficiently, IdPs like Google and Facebook have developed the OpenID Connect (OIDC) protocol [25] and its variants, on top of OAuth2.0. In addition to the authorization code flow and implicit flow, OIDC further supports another type of authorization flow, *i.e.*, hybrid flow. Regardless, *ALL* the real-world OIDC-enabled apps under study only implement the implicit flow. As such, we will focus the implicit flow in the rest of this paper.

Regarding the implicit flow, OIDC is backward compatible with OAuth2.0. The only difference is that, apart from the access token, IdPs also introduce a new parameter, *i.e.*, *id_token*, which is digitally signed by the IdP server. As illustrated in Fig. 2, the *id_token*, which consists of the user profile, is then sent to the RP server along with the original access token. Since the signature cannot be tampered/forged by an attacker, the RP server can now directly identify the user by extracting the user profile from the *id_token* without the trouble to retrieve the user profile from the IdP server.

2.3 Threat Model

The goal of an adversary is to break the mobile app authentication, *i.e.*, log into the RP app as the victim. Here, we trust the mobile operating system and assume it cannot be compromised. We assume the adversary can install the RP app and IdP app in her own device so as to communicate with the RP server and IdP server. The adversary, Eve, can act as a normal user and monitor/tamper the network traffic through her own device. In addition, we consider that an attacker can trick a user into installing a malicious app on her device. This app does not have any permission considered to be dangerous.

3 Major Protocol Changes that Affect Mobile OAuth Security

Although the above OAuth protocol seems simple, some security-related changes (when adapting the protocol from the website) are often overlooked by mobile app developers.

3.1 Untrusted Identity Proof

Websites typically employ the authorization code flow to support SSO services. In this case, the identity proof is transmitted via a secure HTTPS channel between the RP server and IdP server. On the contrary, mobile developers advocate the implicit flow, which passes the identity proof through the user device. Since the mobile device is untrustworthy (the attacker has full control of her own device), the identity proof is subject to tampering. Therefore, such an identity proof can only be accepted by the RP server for two reasons:

1. The RP server makes a direct server-to-server call to the IdP server to verify the identity proof (*i.e.*, Step 6–Step 9 in Fig. 1);
2. The identity proof is signed by the IdP server (*i.e.*, *id_token* in Fig. 2).

In other words, the RP server should establish a direct trust relationship with the IdP server to correctly process the identity proof.

Furthermore, there are different types of the so-called identity proof including the access token, the id_token, the user profile (returned at Step 9 of Fig. 1) and even the authorization code (if the authorization code flow is used). Since this notion is neither covered by the protocol specification nor research studies, RP developers need to determine which identity proof to use in which way, based on their own understanding of the protocol.

3.2 Heavy Client-Side Logic

Another major difference is about the user-agent. In the web-based SSO services, the user operates on the end-user's browser. By contrast, in the mobile SSO systems, the user interacts with the RP app and IdP app (the browser is split into two parts). Since mobile apps are more powerful, the RP app and IdP app are often responsible for more message exchange which instead is managed by the backend server in the case of website. This seems reasonable at the first sight. But some messages can only be processed at the server side, such as the user resource retrieved at Step 9 of Fig. 1. Otherwise, it can lead to the user-profile attack, as presented in Sect. 5.

Meanwhile, both the IdP app and RP app store much more data on the client side. For example, the IdP app keeps the user's identity information and the RP app stores the authorization information (*e.g.*, the RP's name), which is displayed for the user to check/grant the permissions. Note that these security-critical data need to be retrieved from the server during SSO transactions. However, with such information in the client side, it may be tempting for an app developer to retrieve it from the phone directly. This can lead to the so-called user-profile attack and inconsistent RP app identity as presented in Sect. 5.

4 Our Approach

To evaluate the security implications resulting from the above protocol changes, we first perform dynamic testing on every RP/IdP app. The testing helps to understand the program logic on the server side. Secondly, to better understand the security practices on the client side, we conduct an in-depth static code analysis on the IdP apps (*i.e.*, Facebook, Google and Sina) and their corresponding SDKs (used by RP apps). Given the limited number of IdP apps and SDKs, we can afford for the manual code examination.

4.1 Dynamic Testing

We design a tool to automatically fuzz every OAuth-related message. As shown in Fig. 3, we first set up a man-in-the-middle (MITM) proxy so that we can observe and tamper the network traffic going into and leaving our phone. We then manually operate the phone as a normal user and mimic the SSO process to generate a series of OAuth requests. For every request, our tool replaces each parameter with that from a different RP and user. Finally, the tool sends the fuzzed request to the receiver and checks whether the response is normal or not. Note that since the network traffic is typically protected by HTTPS, we employ the SSL-enabled proxy like *mitmproxy* [3].

The analysis of the communication between the IdP app and IdP server is straightforward, since such interactions share the same format for the same IdP. Unfortunately, fuzzing the messages between the RP app and RP server, *i.e.*, Step 5 (ii) in Fig. 3, is more challenging than expected. Firstly, there are numerous interactions between the user mobile device and the RP server. It is therefore difficult to identify which request is used by the RP server to authenticate the user. Secondly, besides being protected by HTTPS, the message exchanges between the RP app and its backend server are often further encrypted or signed by the RP developer. Although it is possible to extract the cryptographic key from the Android app, such a practice may not be scalable. Therefore, it is usually easier to tamper the response from the IdP server to the IdP app, *i.e.*, Step 3 (ii) instead. After all, all OAuth-related information received by the RP server can only be derived from the IdP server[3].

[3] One exception is Google's Android account management, as presented in Sect. 5.1.

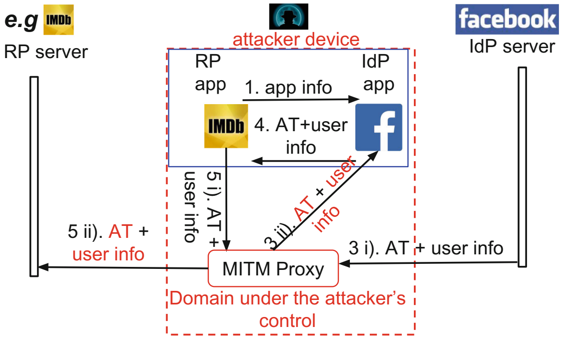


Fig. 3. The platform to analyze the implementations on the IdP/RP server side

4.2 Static Code Analysis

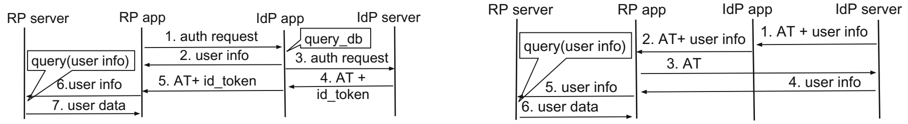
To understand the client-side logic, we decompile the binary code of the latest IdP app, and the official SDK (if it is compiled) widely used by the RP app. Although the IdP app and SDK usually are heavily obfuscated, the names of special activities and system APIs (*e.g.*, startActivity, getCallingPackage, *etc.*) are not changed. As such, we can identify the entry point of SSO services and then build a partial call flow graph. This provides an opportunity for us to only focus on the relatively small number of SSO-related security-critical activities. We then manually examine these activities to identify potential vulnerabilities. For example, we find that the SSO entry of Sina (*i.e.*, SSOActivity) by default verifies the received information, except when the information is from Sina itself. This practice is seemingly sound at first. However, it may be leveraged by a malicious RP app to bypass the security checking, if there is a "next Intent" [31] in Sina app. Therefore, during the code examination process, we will try to check the existence of the "next Intent". To confirm these vulnerabilities, we then build a toy RP app with the official SDK and launch the corresponding attacks on this toy app. By this way, we have identified the problem of inconsistent RP identity, as presented in Sect. 5.2.

5 Vulnerability Analysis

In addition to various existing vulnerabilities, the above method also helps to discover three unknown security flaws resulting from the inaccurate understanding/implementations of the protocol changes.

5.1 Profile Vulnerability

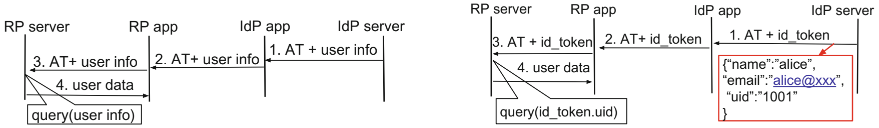
The identity proof is often incorrectly processed by the RP server and has led to the profile vulnerability, which enables an attacker to log into a susceptible RP as the victim by leveraging the victim's public user profile only. Note that all the



(a) RP app returns user information obtained via Android account management system

(b) The RP app returns user information obtained via API

Fig. 4. The RP app does not return correct identity proof of the user to the RP server



(a) No verification of the access token by the RP server

(b) No verification of the signature by the RP server

Fig. 5. The RP server does not verify the identity proof of the user

prior findings [12,29,34] require different types of interactions with the victim. In contrast, this newly discovered vulnerability can be exploited remotely and solely by the attacker without any need to trick or interact with the victim, for example via phishing attacks.

Different Types of Incorrect Implementations. We present two types of common but widespread mistakes (but the real-world misuses do not limited to these two types).

Return Incorrect Identity Proof to the RP Server. Some RP apps can directly retrieve the user information from the mobile device it is running on, regardless of the OAuth access token obtained from the IdP. Without the access token, the RP app only sends the user profile (*e.g.*, uid, email) to the RP server as the identity proof. As a consequence, the RP server has no way to correctly identify the user.

One interesting misuse is caused by Android Account Management System (AMS) [2,15] when using Google as the identity provider. Android AMS provides a centralized database (*i.e.*, */data/system/users/0/accounts.db*) for storing user accounts. While the main goal of AMS is to support seamless access user data via background synchronization, Google has integrated it to support SSO service. Specifically, when a user logs into his/her Google account, Google Login Service (*i.e.*, IdP app) will store the user’s Google account information in the *accounts* table as shown below.

```
INSERT INTO "accounts" VALUES(1, ’alice@gmail.com’, ’com.
    google’, ’password’, NULL);
```

With the Android permission of *GET_ACCOUNTS*, an app can easily get the user's Google account (*i.e.*, email address) by calling *getAccounts()* method. As shown in Fig. 4(a), some RP developers (*e.g.*, one antivirus app Psafe with 50+ million installs) directly return this email address (retrieved from the database) to the RP server as the identity proof, regardless of the access token (or id_token). As such, an adversary can insert a forged entry, *i.e.*, *victim's* email address, into the database on the *adversary's* mobile device.

Another typical example is shown in Fig. 4(b) where an RP app immediately retrieves the user data by calling the IdP API with the access token. However, this RP app only sends the user profile to its server as the identity proof. Although such a proof is protected by encryption/signature techniques in addition to TLS, an attacker can simply feed incorrect user information at Step 4 of Fig. 4(b).

The RP Server Does Not Verify the Identity Proof. As shown in Fig. 5(a), when the IdP servers return the user identity information (*e.g.*, user id/email address) along with the access token via OAuth, many RP servers (*e.g.*, Sohu with 80+ million monthly-active-user *etc.*) simply and incorrectly return sensitive user information to its own client-side app based on the received user-id WITHOUT verifying whether the received user-id is indeed bound to the issued access token (*i.e.*, lack of Step 6–Step 9 in Fig. 1).

Figure 5(b) shows another case where Facebook and Google adopt the OIDC-like protocol and digitally sign the user identity information. However, most RP servers just ignore this signature and insist on the traditional OAuth protocol. Worse still, some RP servers (*e.g.*, a free call/text app DingTong with 10+ million installs) even do not verify the signature but simply extract the user-id from the payload of the signature and accept the user-id as is the way without any authentication/validation.

Exploiting the Profile Vulnerability. Leveraging the same system setup of Fig. 3, an attacker can log into a susceptible app as the victim by exploiting the victim's profile with the following steps[4]:

1. The attacker setups a SSL-enabled MITM proxy for her own mobile device to monitor and tamper network traffic going into and leaving from her device.
2. The attacker installs the vulnerable RP app in her own mobile device.
3. The attacker signs into the vulnerable RP app with the attacker's own IdP login name and password.
4. When the IdP server returns the user profile to its client-side app, *i.e.*, Step 3 (ii) of Fig. 3, the attacker substitutes her own user-id (public user id for the case of Google+ and Sina users or guessable email address) with the victim's one using the MITM proxy. Although Facebook has started to issue private per-app user-id for each RP since May 2014, for backward compatibility reasons, to-date, Facebook still uses the public user-id to identify early adopters

[4] For the case of Google Android AMS, an attacker just needs to insert a forged entry in her own device and then follows the normal steps to complete the SSO procedure.

of a RP app. As such, a user of a vulnerable RP of Facebook is still susceptible to our attack as long as he/she has signed into the RP app via Facebook before May 2014.

5. Since the RP server directly uses the user identity information returned by its client-side app to identify the user WITHOUT further validation, the attacker can therefore successfully sign into the RP as the victim.

Additional Challenges for the Exploit. The above exploit involves additional challenges when the IdP client-side app, e.g., the one by Facebook, applies the certificate pinning. In this case, the message sent by the IdP server (and then tampered by the attacker's MITM proxy) to its client-side app will not be accepted by the latter. As a workaround, the attacker can simply uninstall the IdP app so that the IdP SDK (widely used by RP apps) would automatically downgrade to carry out OAuth authentication via the built-in WebView browser. Being a general built-in browser, the WebView does not support the certificate pinning for a specific IdP.

But some IdPs do not support WebView. In this case, the attacker cannot just use off-the-shelf tools like Xposed SSLUnpinning [6] module, since the IdPs often use customized methods (instead of the native Android framework) to implement the certificate pinning. For such IdPs, the attacker has to reverse engineer the IdP app and manually remove the certificate pinning. To demonstrate the feasibility of this approach, we have successfully implemented a proof-of-concept hack on the Facebook app by reverse engineering the apk. But the RP app (more precisely, the IdP SDK) will also compare the certificate of Facebook with a previously hard-coded one (*i.e.*, the real certificate of Facebook). As such, we also need to bypass the certificate comparison function.

Vendor Responses. All of three IdPs under study acknowledged the security issue and pledged to help to notify the affected third-party app developers. In particular, Sina already sent a specific notification to ALL RP developers on its platform to inform them about the problem. The company also granted us the maximum amount of reward credits allowed by their bug-bounty program. It has also updated the Single-Sign-On section of its programming guide accordingly. Google has acknowledged our finding via their Google Security Hall of Fame and indicated that they will modify the corresponding documentation for their 3rd party app developers. Facebook has informed us that they are seeking a way to make their RP developers aware of this problem.

5.2 Inconsistent RP App Identity for the User

After authenticating the user at Step 2 of Fig. 1, the IdP app should retrieve the authorization information including the RP name, the requested permissions from its backend server. From the perspective of the user, the IdP app will pop up a dialog to indicate which RP requests what permissions from which user. Here, the name of the RP represents the identity of the RP which is verified by the IdP.

Since this authorization information is also stored by the RP app on the mobile device, we find that Google immediately retrieves such information from the device. Specifically, Google presents the name of the RP according to the value of *android:label* in the *AndroidManifest* file, when an RP app adopts the Intent[5] scheme (which is the default method used by Google's official SDK) to support SSO services. On one hand, Google in fact can correctly learn the identity of the RP. On the other hand, the *AndroidManifest* file can be arbitrarily defined by the RP app. As such, the displayed authorization page is inconsistent with Google's understanding.

Taking the advantage of this inconsistence, a malicious RP app can convince the user that the interacting RP, as verified by Google, is a benign and privileged app like IMDB. Due to the great trust placed on Google, the user is willing to grant the permissions if Google verifies the RP app as IMDB rather than some random app. As such, it would be easier for the malicious RP to obtain an access token. This access token enables an attacker to retrieve the victim's data hosted by Google. When combined with other attacks, *e.g.*, token hijacking, the attacker can also log into the victim's account on other Google-enabled benign apps.

Vendor Response. Google acknowledged this security bug. But as Google security team claims, this security issue was found independently by another concurrent work (but this issue is not fixed yet). Unfortunately, only the first report is in the scope of Google vulnerability reward program.

5.3 Treat the IdP App as a Special RP

Some IdPs including Google and Sina treat their client-side app as a special RP. If the user can successfully authenticate with the IdP, the IdP server would issue an access token to its client-side app. Note that this access token has higher privileges. It enables the IdP app to make sensitive transactions on behalf of the user, for example, to issue access token for any other RPs, *etc.* Since the adversary (acting as a normal user) can also obtain this access token, the adversary can easily launch application impersonation attack [20]. For example, the adversary can utilize this highly privileged access token to invoke sensitive APIs (*e.g.*, query the user information in a batch) with higher API quota. Note that such APIs/quotas originally are not allowed by the adversary or users.

Worse still, this privileged access token is not protected by HTTPS for Sina app. Therefore, an adversary can easily obtain it via eavesdropping. Using this special access token, the attacker can pretend to be the victim and make any transactions, for example, signing into any RP app on the Sina platform as the victim.

Vendor Response. We have reported this issue to Sina. Sina directly acknowledged this problem and have applied the corresponding fixes for their Android app.

[5] When an RP app chooses to use the WebView scheme, Google can correctly get the RP name from its own server.

6 Empirical Evaluation

We have studied the Android IdP apps and OAuth SDKs provided by three top-tier IdPs, *i.e.*, Facebook, Google and Sina. The number of registered users in these IdPs ranges from more than 800 million to over 2.5 billion as depicted in Table 1. We then comprehensively test the implementations of 600 top-ranked Android applications in US and China. Since more Chinese RPs support OAuth, we only select top 100 RPs (in overall category) and another top 100 apps in different categories for Sina from one major Chinese app store [4]. By contrast, we select 300 top-ranked RPs (in overall category) and 100 top-ranked RPs (in different categories) for Facebook and Google from Google Play. The top 100 apps in different categories is selected as follows: top 30 free apps in social, top 30 free apps in travel and local, top 30 free apps in fitness, top 10 free apps in communication. Out of these 600 apps, we identify that 182 apps use OAuth authentication service provided by one or more of the 3 IdPs mentioned above.

Table 1. Statistics for the usage of the protocol

IdPs (# of third-party RP)	# of IdP users (in Millions)	OAuth2.0		OpenID Connect		
		Insecure usage	Correct usage	Ignore id_token	Not verify id_token	Correct usage
Facebook (59)	>1,500	9	10	35 (20*)	2	3
Google (40)	>2,500	24	14	0^+	0	2
Sina (83)	>8,00	78	5	N.A	N.A	N.A

– *: 20 out of 35 RPs are incorrectly implemented.

– $^+$: Google customizes the OIDC protocol where typically only the id_token is issued to the RP. Therefore, this id_token cannot be ignored. Otherwise, there is not a valid identity proof.

In addition to different types of vulnerabilities, our studies also present first-hand information regarding the adoption rate as well as the misuse rate of OAuth2.0 and OIDC. As shown in Table 1, all of three IdPs support the OAuth2.0 protocol. Facebook and Google additionally develop and advocate OIDC-like protocols for SSO services. Since OIDC by default is supported by Facebook SDK, 68% apps of Facebook, as opposed to only 2 RP apps of Google, employ the OIDC protocol. Regardless, there are two types of misuses for these OIDC-enabled RPs:

- Ignore *id_token*: Some RPs ignore the *id_token*, in which case these RPs revert to the OAuth2.0 protocol and rely on the access token to authenticate the user. As such, these RPs (20 out of 35) share the same security issues of OAuth2.0 as illustrated in Table 2.
- Not verify *id_token*: Some RPs indeed rely on the *id_token* to verify the user. But 29% of them do not check whether the *id_token* is well signed.

Table 2. Statistics of the vulnerabilities for OAuth

IdPs (# of 3rd-party RP)	Profile attack	Token hijacking	Improper user agent	Access token disclosure	App secret disclosure	# of vulnerable RPs[a]
Facebook (59)	9 (15%)	27	1	2	0	31 (53%)
Google (40)	8 (20%)	20	1	1	0	24 (60%)
Sina (83)	58 (70%)	15	7	13	4	78 (94%)
Summary (182)	75 (41%)	62	9	16	4	133 (73%)

– [a]: One RP may be susceptible to multiple vulnerabilities, *e.g.*, the profile attack and token hijacking, at the same time.

These observations show that OAuth2.0 is still the most popular SSO protocol. Unfortunately, the implementations of OAuth2.0 (including the ignore *id_token* case of OIDC) are also more susceptible: 75% OAuth2.0-enabled RPs have at least one vulnerability whereas 29% OIDC-supported RPs are vulnerable. Different OAuth security vulnerabilities are summarized in Table 2. Below we first discuss the implication of the profile attack and then demonstrate the pervasiveness of existing vulnerabilities.

6.1 The Implication of the Profile Attack

As illustrated in Table 2, 41% of the RP under test are found to be vulnerable to the newly discovered profile attack. Table 3 depicts a partial list of the vulnerable mobile apps we have identified so far. Notice that the total number of downloads for this incomplete list of popular but vulnerable apps already exceeds 2.4 billion. Based on the SSO-user-adoption-rate of 51% according to the recent survey by Janrain [7], we conservatively estimate that more than one billion of different types of mobile app accounts are susceptible to the profile attack as of this writing.

After signing into the victim's vulnerable RP app account using our exploit, the attacker will have, in many cases, full access to the victim's sensitive and private information which is hosted by the vulnerable RP server. Just for the vulnerable apps listed in Table 3 alone, a massive amount of extremely sensitive personal information is wide-open for grab: this includes detailed travel itineraries, personal/intimate communication archives, family/private photos, personal finance records, as well as the viewing or shopping history of the victim. For some RPs, the online-currency/service credits associated with the victim's account are also at the disposal of the attacker.

Although our current attack is demonstrated over the Android platform, the exploit itself is platform-agnostic: any iOS or Android user of the vulnerable mobile app is affected as long as he/she has used the OAuth SSO service with the app before. As a proof of concept, we have conducted the same attack on two vulnerable iOS apps.

6.2 Re-Discover Known Vulnerabilities

Table 2 shows that our testing also rediscovers different types of existing security issues.

Table 3. A Partial list of vulnerable apps and the sensitive information exposed

Type of apps	IdP supported	# of app downloads (in Millions)	Type of private/sensitive information exposed	Feasible transactions by the attacker
Travel plan app	Sina	>270	Travel itineraries	-
Hotel booking app	Facebook, Google	>5	Lodging history	Pay for room bookings
Private chat app	Sina	>10	Private message/album	Send forged messages
Dating app	Google, Sina	>5	Dating history, preferences	Purchase gifts
Finance app1	Sina	>25	Personal income/expenses	-
Finance app2	Sina	>50	Stock list of interest	-
Call app	Facebook	>10	Contact list and call history	Call for free
Live video app	Sina	>15	The host the victim likes	Purchase gifts
Download app	Sina	>60	Download history	Enjoy VIP speed
Shopping apps	Facebook, Google	>100	Shopping history	-
Browser	Sina	>40	Browsing history	-
Video apps	Sina	>700	Video watching history	Purchase videos
Music apps	Google, Sina	>800	Playlist	Purchase sound-tracks
News apps	Sina	>350	News-reading history	-

1. Token hijacking [1]: At Step 6–Step 7 of Fig. 1, the RP server must check that the received access token is granted to the same RP. Unfortunately, 34% RPs fail to do so, which enables an adversary to sign onto a victim's benign RP account by leveraging an access token issued to a malicious RP.
2. Improper user agent [12]: In addition to the infeasibility to identify the RP app, the WebView, as a custom webkit browser embedded in the app, is also untrustworthy to be a OAuth user-agent. Since the WebView is under the control of the RP app, a malicious RP app is capable of stealing any information submitted by the user in the WebView (*e.g.*, the user's IdP password) and modifying authorization information displayed by the WebView. Unfortunately, most RP apps support WebView, and worse still, 5% RP apps only support this problematic scheme.
3. Access token disclosure [27]: An access token should be transmitted securely. Thanks to the higher adoption rate of TLS, only 9% mobile RPs disclose their access token whereas 32% 3rd-party websites made this mistake [27].
4. App secret disclosure [29]: The confidential app secret should only be shared between the RP server and IdP server. There are 15 mobile RP apps deploy the authorization code flow rather than the implicit flow. Unfortunately, 27% of these apps inadvertently pass this secret through the user device. This allows an attacker to make operations on behalf of the RP, *e.g.*, changing the RP's security setting.

7 Plausible Root Causes

Surprised by the prevalence of various incorrect implementations, we also try to analyze the underlying reason by examining the SDKs and the OAuth APIs provided by IdPs.

7.1 Unclear Developer Documentation

We found that many authentication-related security issues are caused by the lack of clear guidelines. Since OAuth2.0 (RFC 6749 [18]) was not designed for mobile app authentication, various IdPs have developed different home-brewed extensions of OAuth2.0-based APIs and SDKs to support SSO for mobile apps. Unfortunately, the implicit security assumptions and operational requirements of such home-brewed adaptations are often not clearly documented or well-understood by RP developers. For example, when Sina returns the user profile to the RP app at Step 4 of Fig. 1, the only purpose is to allow the RP app to display the user info (*e.g.*, user name, avatar, etc.). Despite of this specific intention, Sina makes the following confusing claim[6] in its programming guide [5]:

> For the convenience of app developers, the returned user information can avoid calling the user-profile API, *i.e.*, *users/show*.

As pointed out in Sect. 3, a server-to-server call is inevitable for the standard OAuth2.0 protocol to verify the untrusted identity proof. However, the above claim can mislead RP servers not to make the user-profile API call to the IdP server at Step 8 and Step 9. Instead, the RP server may directly use the returned user information from its client-side app as the identity proof and thus be vulnerable to the profile attack.

For another example, Google assumes the RP developers would adopt the OIDC protocol, and thus only shows how an RP app can authenticate with its backend server using the *id_token*. Unfortunately, the majority of apps use the OAuth2.0 protocol instead. For these apps, Google does not define the interactions between the RP app and RP server. Therefore, the RP developers without adequate security expertise have to implement the error-prone authentication services by themselves.

After reporting the security issues to the three IdPs, all of them recognize the need to improve their documentation by explicitly pointing out the implicit security requirements. For example, Sina now updates its claim [5] as follows:

> The third-party apps should not use *uid* to identify the logged-in user. Note that the access token is the only valid identity proof.

[6] Since Sina developers are not native English speakers, the statements are translated by the author.

7.2 Poor API Design of Sina

Since more Sina apps are vulnerable to the profile attack, we further examine its SDK and API design. We find that the poor API design of Sina may compound this problem. As an open social network, Sina by design allows any RP in possession of a valid access token, no matter whom the access token is issued to, to retrieve *any* user's basic profile via the *users/show* API: https://api.weibo.com/2/users/show.json?access_token=x&uid=x. Even if the RP server does not trust the user id and would like to use the access token to identify the user, it may incorrectly use the above API. In this case, Sina server would return the victim's user profile only based on the value of *uid* in the URL. Without realizing the exact semantic of the returned information, the RP developer may incorrectly interpret that the returned user profile is bound to the access token, and thus, log the user in.

In fact, Sina also provides a correct API (*i.e.*, *account/get_uid*) to get the user-id of the one, whom the access token is bound to. But Sina never specifies which API to use[7]. Due to the richer information provided by the inappropriate *users/show* API[8], we believe RP developers prefer the incorrect API. By contrast, Facebook and Google use the *people/me* API, which is more self-explained, and more importantly, is typically handled by IdP SDKs. For example, the PHP SDK of Facebook hard-encodes the user id *me*, and as such the *uid* fed by the attacker would be ignored by the SDK.

8 Defense

The community has proposed the following Current Best Practices (CBPs):

1. IdPs should provide clearer, and more security-focused developer guidelines.
2. The RP server should not trust any information even if it is signed by its own app. Trust should be anchored on the IdP server directly.
3. To implement OAuth2.0 for mobile apps, RP developers should use the authorization code flow instead of implicit flow by strictly following [14].
4. Use OIDC for authentication whenever possible.

If these CBPs can be correctly followed, then most vulnerabilities will not exist. Unfortunately, most RP developers never adhere these CBPs at all. Towards this end, the crux of the defense is to enforce the defined security checks, which cannot simply rely on the RP developers, but instead must be strongly enforced by the IdPs, since the latter has adequate security expertise. Therefore, we develop a general and foolproof solution, from the perspective of the IdP, to help the RP developers to automatically handle the error-prone SSO services. Our solution should achieve three goals:

[7] Worse still, Sina seems to recommend the incorrect one in their unclear developer documentation.

[8] The other API only returns the *uid* of the user.

1. All the identified vulnerabilities (including existing ones) can be prevented.
2. To ease the transition, the code changes by the RP developers should be as few as possible.
3. The solution should be backward compatible in a sense that the RP server can still support those users who use the older version of the RP app. After all, it is difficult for every user to upgrade his/her RP app.

With these goals in mind, we first review the architecture of existing SSO systems. As shown in Fig. 6, both the RP server and RP app can be split into two modules: the SDKs provided by the IdP serve the interactions with the IdP app/server and the upper layer codes implemented by the RP developer manage its own business logic. Currently, the identity proof is also (incorrectly) handled by the upper layer code. To prevent from SSO errors, we migrate such functionality to the lower layer SDKs, with the belief that the SDKs provided by IdPs can correctly process this security-critical identity proof. Specifically, upon the reception of the identity proof at Step 4 of Fig. 1, the client-side SDK can send it to the server-side SDK. The latter then performs the real authentication task by utilizing this identity proof. Below we will discuss more details.

8.1 Defense on the Client-Side SDK

As shown in Fig. 4, when the RP apps only return user information, there is no way for the RP servers to correctly identify the user. Therefore, the enforced version of the client-side SDK should automatically send required information including the access token (or the id_token for OIDC) to its backend server. With due consideration to the ease of transition, below we first discuss the design of the existing client-side SDK.

API Design of the Existing Client-Side SDK. We only take Sina, [9] one OAuth2.0 IdP, as the example. When authorization succeeds, the client-side SDK (used by the RP app) will utilize Android API *onActivityResult* to receive an access token along with a user id from Sina app. Before forwarding this result to the upper layer, *authorizeCallBack* API is first invoked to check whether the access token is in the correct format.

```
protected void onActivityResult(...){
  mSsoHandler.authorizeCallBack(requestCode,resultCode,result);
}
```

Initial Attempt Using Cookie-Based Scheme. Without affecting the upper layer behavior, we manage to authenticate the user in the SDK. Referring to the scheme in websites, one natural attempt, as shown in Fig. 6, is to use *cookie*.

[9] Our solution is applicable to OIDC protocol or other IdPs since they follow the same flavor.

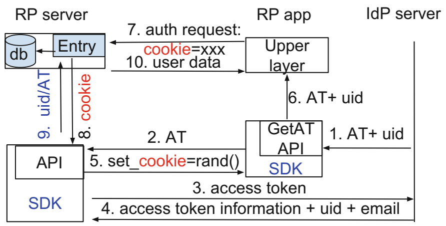


Fig. 6. The cookie-based defense

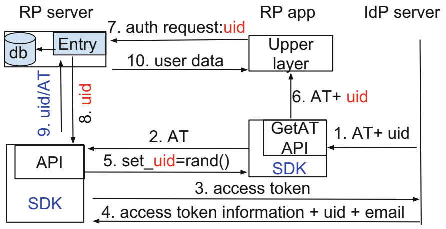


Fig. 7. The refined defense

1. After the client-side SDK (*i.e.*, *authorizeCallBack*) checks the format of the access token, instead of forwarding it to the upper layer, the client-side SDK first delivers the access token to the server-side SDK.
2. The server-side SDK exactly follows Step 3–Step 4 in Fig. 6 (corresponding to Step 6–Step 9 in Fig. 1) to identify the user via the access token.
3. If the verification succeeds, the server-side SDK then sets a *cookie* to the client-side app with a random nonce at Step 5 of Fig. 6. Only then would the client-side SDK forward the access token and the user profile to the upper layer code.
4. From the view of the upper layer, everything remains the same. Thus it can follow its original logics to interact with the RP server. The only difference is that a cookie would be automatically attached to the authentication request at Step 7 of Fig. 6.
5. Regardless of other information sent by the RP app, the server-side SDK only relies on the *cookie* to identify the user.

Unfortunately, this cookie-based solution is not applicable to every RP app: Unlike websites where a central browser can help to manage cookies, the Android app developers need to manage the cookies by themselves, for example using the *CookieManager*. Therefore, the *cookie* set by the underlying SDK may not be automatically used by the upper layer, if the latter adopts a customized *CookieStore*.

Refined Defense. Nevertheless, the cookie-based scheme still provides great insights. Note that the cookie is used to bind the requests of Step 2 and Step 7 in Fig. 6. Towards this end, the client-side SDK and its upper layer code must share some information like the *cookie*. Furthermore, the shared information must be a *secret*. Otherwise, an adversary can easily guess/compute this information and pretend to be anyone else.

Given these requirements, we revise the cookie-based solution. Referring to the practice of Facebook which issues private user-id on a per-app basis, we will randomly generate a one-time user id, instead of the *cookie*, as the secret to bind these two requests. More specifically, if authentication succeeds, the server-side SDK would return a randomly generated *uid* to its client-side SDK at Step 5 of Fig. 7. This *uid* plays the same role as *cookie*. Note that this *uid* can only be used for once so that an adversary cannot guess/compute its value. The old *uid* will be deleted once expired or used.

8.2 Defense on the Server-Side SDK

At Step 2 of Fig. 7, the server-side SDK can follow Step 3 and Step 4 in Fig. 7 (*i.e.*, correspond to Step 6–Step 9 in Fig. 1) to identify the user. Once authentication succeeds, the server-side SDK randomly generates a one-time *uid* with a specific prefix and maintains the mapping of $< uid, uid_{real} >$. At Step 7 of Fig. 7, the server-side upper layer should forward the authentication request to its SDK. Our newly added function in the SDK can then handle this request according to its content.

1. If the request only contains the user id, *i.e.*, *uid*, we check whether this *uid* starts with a specific prefix. If so, we check the mapping of $< uid$, $uid_{real} >$ and then get the user information uid_{real}. Otherwise, just abort the request.
2. If the request only contains the access token, we follow Step 3 and Step 4 in Fig. 7 to identify the user.
3. If the request contains the access token and user id, we first follow Case (1) to process the user id. If it fails, we then follow Case (2) to process the access token.

Remark. The correctness of our defense is based on two assumptions. Firstly, a correct implementation of OAuth, as shown in Figs. 1 and 2, is automatically immune to all the existing attacks. In fact, a formal proof is presented by [34]. More precisely, this work utilizes the model checking method to analyze the implementation-level protocol of Facebook and can only discover unauthorized storage access if the malicious app has root privileges. However, such a strong threat model is not considered in this paper. Secondly, the IdPs (*i.e.*, SDKs) with enough security expertise can accurately implement the protocol (as opposed to the RP developers who often make mistakes). Therefore, the crux of our solution is to enforce the IdP developers to correctly implement the OAuth protocol for the RP developers.

Although the design of the defense seems complicated, we only add two more requests (*i.e.*, Step 2 and Step 5 at Fig. 7) into the current system. Note that the other steps (*e.g.*, Step 6, Step 8, *etc.*) already reside in the existing systems. For the ease of presentation, we intentionally hide these detailed (SDK-level) steps when discussing the protocol flow in Fig. 1. Note also that the proposed remedies does not affect the authorization code flow, although we can use the same idea to improve its security.

8.3 Implementation and Evaluation of the Proposed Defense

To demonstrate the feasibility of the proposed remedies, we implement the solution on a sample app provided by Sina. Like examples from the other IdPs, the sample app is built on top of the Android SDK. Note that this SDK is in the form of executable Jar file. To modify it, we thereby need to decompile the Jar file into Java code. While there are off-the-shelf Java decompiler tools like CFR or JAD, the extracted source code is not well structured and contains various errors, which requires non-trivial manual resolve.

We then follow the common practices to build a backend server on top of the official PHP SDK. The only change in the server-side SDK is to add a new function which performs the real authentication task. Meanwhile, we only add 5 lines of code in the server-side upper layer, which demonstrates the least programming efforts from the RP developers.

Table 4. The average running time under three different settings

Two types of app	User info (in ms)	Access token (in ms)	Access token & user info (in ms)
Vulnerable app	4490	7604	5092
Fixed app	6414	9667	6009

Evaluation. To demonstrate the effectiveness, we have launched different attacks listed in Table 2 on the sample app. It turns out that our solution can prevent from (or alert for insecure transmission, *e.g.*, token disclosure) all these attacks. For example, the profile vulnerability becomes impossible since an adversary cannot guess the randomly-generated one-time *uid*. Take token hijacking as another example. Since our defense exactly follows Fig. 1, our server-side SDK will verify whether this access token is issued to itself or not at Step 3 of Fig. 7 (corresponds to Step 6 of Fig. 1). Thus an access token of another RP will not be accepted.

To show the efficiency of our solution, we measure the running time of a complete SSO process. Specifically, we enumerate all three possible cases of the authentication request, as mentioned in Sect. 8.2. For each case, we operate the sample app to complete the OAuth2.0 process for 20 times under the same phone and network environment. The average time is presented in Table 4. It shows that the fixed app has only a small impact on the user performance.

The State-of-the-Art OAuth Defense. The Current Best Practices (CBP) [14] suggests the usage of authorization code flow to prevent from many existing problems. However, a correct implementation of these CBPs is still challenging for the RP developers (as is the case for the revised implicit flow). Furthermore, it requires lots of efforts to migrate from implicit flow (the current *de fact* standard) to authorization code flow. Xing *et al.* [32] develop invariants of HTTP parameters to protect third-party web service integrations like OAuth2.0. However, it cannot discover many attacks such as the profile attack. Another defense [11] uses program verifier to check whether the sequence of method calls satisfies the defined predicate. Despite its power, this method requires significant programming efforts for *all* involved parties. Compared to the state of the art, our defense not only can prevent from all the existing attacks, but also requires the minimal programming efforts from the RP developers.

9 Related Work

Security Analyses from the Protocol Perspective. IETF has presented comprehensive security considerations and threat models in RFC6749 [18] and RFC6819 [22] for OAuth2.0 protocol from the initial design. Additionally, the authorization code flow has proven to be secure cryptographically [10] as long as the TLS is properly used. Hu *et al.* [20] present the App Impersonation attack. These works mainly prove the authorization security for the web from the protocol design standpoint. However, we consider whether the protocol is securely implemented on mobile platforms for authentication.

Formal Security Proof for OAuth. The model checking method is extensively used to analyze the protocol specification [18] by numerous works including [8,9,16,24], just to name a few. Researchers also attempt to use the same method to reason about the protocol implementations by modeling the complex runtime platform, *e.g.*, browser [16,17]. All of these works assume a correct implementation of the protocol call-flow. However, we show that protocols are often implemented incorrectly.

Real-World Study on the Web-Based SSO Systems. More efforts have been contributed to the vulnerability detection of the website-based real-world systems, including [13,23,27,30]. Another research direction is to conduct large-scale testing on the web SSO systems, including SSOScan [26,35], OAuthTester [21,33]. Nevertheless, all the works mentioned so far only study the OAuth/OIDC specification/implementations on the website. In contrast, we focus on the mobile platforms.

Analyses of Mobile OAuth SSO Systems. There are relatively few security analyses on OAuth under mobile environment. Chen *et al.* [12] shows how real-world OAuth systems can fall into the common pitfalls when leveraging the operating-system-provided components (*e.g.*, Intent, WebView, *etc.*). Ye *et al.* [34] utilize the model checking method to evaluate the OIDC-like protocol implemented by Facebook on Android platform. Summarizing these works, Wang *et al.* [19,29] collect the statistics of these known vulnerabilities. Prior works mainly focus on how the classic vulnerabilities (or features) of mobile systems can be leveraged to compromise SSO systems. In contrast, we analyze how the modified protocol call-flow itself can be incorrectly implemented.

10 Conclusion

In this paper, we report a field-study of mobile OAuth2.0-based SSO systems. We perform an in-depth static code analysis on three first-tier IdP apps and their official SDKs as well as dynamic testing on Top-600 Android apps in China and US. Besides the discovery of three previously unknown security flaws, we also demonstrate the prevalence of different types of existing vulnerabilities. The pervasiveness of these loopholes motivates us to design and implement a foolproof defense for those susceptible RPs with the aim of minimizing the programming efforts of RP developers. Our discoveries show that it is urgent for the various parties to re-examine their OAuth implementations and apply the fixes accordingly.

Acknowledgements. This project is supported in part by the Innovation and Technology Commission of Hong Kong (project no. ITS/216/15) and NSFC Grant (No. 61572415).

References

1. Access token hijacking. https://developers.facebook.com/docs/facebook-login/security#tokenhijacking
2. Android account manager. http://developer.android.com/reference/android/accounts/AccountManager.html
3. Man in the middle proxy. https://mitmproxy.org/
4. One major Chinese App store. http://sj.qq.com/myapp/category.htm
5. Sina access token API. http://open.weibo.com/wiki/OAuth2/access_token
6. SSL unpinning. https://github.com/ac-pm/SSLUnpinning_Xposed
7. Social login continues strong adoption (2014). http://janrain.com/blog/social-login-continues-strong-adoption/
8. Bai, G., Lei, J., Meng, G., Venkatraman, S.S., Saxena, P., Sun, J., Liu, Y., Dong, J.S.: AUTHSCAN: automatic extraction of web authentication protocols from implementations. In: NDSS (2013)
9. Bansal, C., Bhargavan, K., Maffeis, S.: Discovering concrete attacks on website authorization by formal analysis. In: IEEE CSF (2012)
10. Chari, S., Jutla, C.S., Roy, A.: Universally composable security analysis of OAuth v2.0. Cryptology ePrint Archive, Report 2011/526 (2011)
11. Chen, E.Y., Chen, S., Qadeer, S., Wang, R.: Securing multiparty online services via certification of symbolic transactions. In: IEEE S&P (2015)
12. Chen, E.Y., Pei, Y., Chen, S., Tian, Y., Kotcher, R., Tague, P.: OAuth demystified for mobile application developers. In: ACM CCS (2014)
13. Mainka, C., Vladislav Mladenov, J.S., Wich, T.: SoK: Single Sign-On security- an evaluation of OpenID Connect. In: IEEE EuroS&P (2017)
14. Denniss, W., Bradley, J.: OAuth 2.0 for native apps (2016)
15. Elenkov, N.: Android Security Internals: An In-Depth Guide to Android's Security Architecture. No Starch Press, San Francisco (2014)
16. Fett, D., Küsters, R., Schmitz, G.: An expressive model for the web infrastructure: definition and application to the browser ID SSO system. In: IEEE S&P (2014)
17. Fett, D., Küsters, R., Schmitz, G.: A comprehensive formal security analysis of OAuth 2.0. In: ACM CCS (2016)
18. Hardt, D.: The OAuth 2.0 authorization framework (2012)
19. Homakov, E.: The Achilles Heel of OAuth or Why Facebook Adds Special Fragment (2013)

20. Hu, P., Yang, R., Li, Y., Lau, W.C.: Application impersonation: problems of OAuth and API design in online social networks. In: ACM Conference on Online Social Networks, COSN (2014)
21. Li, W., Mitchell, C.J.: Analysing the security of Google's implementation of OpenID Connect. In: SIG SIDAR Conference on Detection of Intrusions and Malware & Vulnerability Assessment, DIMVA (2016)
22. Lodderstedt, T., McGloin, M., Hunt, P.: OAuth 2.0 threat model and security considerations (2013)
23. Mladenov, V., Mainka, C., Krautwald, J., Feldmann, F., Schwenk, J.: On the security of modern Single Sign-On protocols: OpenID Connect 1.0. CoRR abs/1508.04324 (2015)
24. Pai, S., Sharma, Y., Kumar, S., Pai, R.M., Singh, S.: Formal verification of OAuth 2.0 using Alloy framework. In: IEEE International Conference on Communication Systems and Network Technologies, CSNT (2011)
25. Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C.: OpenID Connect core 1.0. The OpenID Foundation (2014)
26. Shernan, E., Carter, H., Tian, D., Traynor, P., Butler, K.: More guidelines than rules: CSRF vulnerabilities from noncompliant OAuth 2.0 implementations. In: Almgren, M., Gulisano, V., Maggi, F. (eds.) DIMVA 2015. LNCS, vol. 9148, pp. 239–260. Springer, Cham (2015). doi:10.1007/978-3-319-20550-2_13
27. Sun, S., Beznosov, K.: The devil is in the (implementation) details: an empirical analysis of OAuth SSO systems. In: ACM CCS (2012)
28. Wang, H., Zhang, Y., Li, J., Gu, D.: The achilles heel of OAuth: a multi-platform study of OAuth-based authentication. In: ACM ACSAC (2016)
29. Wang, H., Zhang, Y., Li, J., Liu, H., Yang, W., Li, B., Gu, D.: Vulnerability assessment of OAuth implementations in Android applications. In: ACM ACSAC (2015)
30. Wang, R., Chen, S., Wang, X.: Signing me onto your accounts through Facebook and Google: a traffic-guided security study of commercially deployed Single-Sign-On web services. In: IEEE S&P (2012)
31. Wang, R., Xing, L., Wang, X., Chen, S.: Unauthorized origin crossing on mobile platforms: threats and mitigation. In: ACM CCS (2013)
32. Xing, L., Chen, Y., Wang, X., Chen, S.: InteGuard: toward automatic protection of third-party web service integrations. In: NDSS (2013)
33. Yang, R., Lee, G., Lau, W.C., Zhang, K., Hu, P.: Model-based security testing: an empirical study on OAuth 2.0 implementations. In: ACM ASIACCS (2016)
34. Ye, Q., Bai, G., Wang, K., Dong, J.S.: Formal analysis of a Single Sign-On protocol implementation for Android. In: International Conference on Engineering of Complex Computer Systems, ICECCS (2015)
35. Zhou, Y., Evans, D.: SSOScan: automated testing of web applications for Single Sign-On vulnerabilities. In: USENIX (2014)

Adaptive Proofs Have Straightline Extractors (in the Random Oracle Model)

David Bernhard, Ngoc Khanh Nguyen(✉), and Bogdan Warinschi

Computer Science Department, University of Bristol, Bristol, England
{csxdb,nn14160,csxbw}@bristol.ac.uk

Abstract. The concept of *adaptive security* for proofs of knowledge was recently studied by Bernhard et al. They formalised adaptive security in the ROM and showed that the non-interactive version of the Schnorr protocol obtained using the Fiat-Shamir transformation is not adaptively secure unless the one-more discrete logarithm problem is *easy*. Their only construction for adaptively secure protocols used the Fischlin transformation [11] which yields protocols with *straight-line extractors*. In this paper we provide two further key insights. Our main result shows that any adaptively secure protocol must have a straight-line extractor: even the most clever rewinding strategies cannot offer any benefits against adaptive provers. Then, we show that any Fiat-Shamir transformed Σ-protocol is not adaptively secure unless a related problem which we call the Σ-one-wayness problem is *easy*. This assumption concerns not just Schnorr but applies to a whole class of Σ-protocols including e.g. Chaum-Pedersen and representation proofs. We also prove that Σ-one-wayness is hard in an extension of the generic group model which, on its own is a contribution of independent interest. Taken together, these results suggest that the highly efficient proofs based on the popular Fiat-Shamir transformed Σ-protocols should be used with care in settings where adaptive security of such proofs is important.

1 Introduction

Noninteractive zero knowledge proofs are a useful tool widely deployed in modern cryptographic constructions. They allow a prover (e.g. the creator of a ciphertext) to send a single message which will convince a verifier of the veracity of a certain statement (e.g. that the plaintext underlying a ciphertext respects certain constraints), without revealing any further information. A particularly useful variant are the so called "proofs of knowledge" where there exists an extractor who can efficiently recover from the prover a witness that the statement is true. Over parties in the system (who should obtain no information from the witness), the extractor benefits from setup assumptions like the common reference string model or the random oracle mode [3]. Our focus is on the latter model which yields by far the most efficient noninteractive proofs of knowledge known to date. In a nutshell, in this paper we study different extraction strategies afforded by

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 336–353, 2017.
DOI: 10.1007/978-3-319-61204-1_17

the random oracle model and identify fundamental efficiency limitations that such proofs need to face in practically relevant scenarios.

Background. Recall that in the random oracle setting the extractor learns the RO answers/queries that the prover makes. Here, we distinguish between several extraction strategies. [1] An extractor is *straight-line* if it only sees a single execution of the prover. An extractor is *rewinding* if it is allowed to launch and interact with further copies of the prover (with the same coins used to produce the statement) before returning a witness.

The distinction between straightline and rewinding extractors may be crucial in applications since for a rewinding extractor it is not clear how many times does it have to rewind to extract all witnesses from a prover who makes a sequence of n proofs. Shoup and Gennaro [20] first encountered this problem in the context of proving CCA security of a particular public-key encryption scheme; the "obvious" approach ends up rewinding 2^n times which leads to an inefficient reduction. Clearly, this problem disappears for a straight-line extractor.

The notion of *adaptive* proofs which lie somewhere between proofs with inefficient rewinding strategies and straight-line PoKs has been recently proposed by Bernhard et al. [5]. A proof scheme is adaptively secure if there is an extractor that can rewind, but must efficiently extract even from provers who make sequences of proofs. The notion is called adaptive because the extractor must return a witness for the first proof to the prover before the prover makes the second one, and so on.

As an application, Bernhard et al. study the Fiat-Shamir-Schnorr proofs: a non-interactive proof of knowledge obtained by applying the popular Fiat-Shamir [10] transform to the Schnorr [17] protocol for proving knowledge of discrete logarithm. The main theorem of [5] shows the Fiat-Shamir-Schnorr proof scheme is *not* an adaptive proof unless the one-more discrete logarithm problem is easy. This result essentially separates the usual PoK notion from adaptive proofs, but the separation has several shortcommings: (i) it relies on an inefficient interactive assumption, (ii) it is specific to a proof system for a specific problem (discrete logarithm), and (iii) it is specific to a class of proofs (those obtained from Σ-protocols via the Fiat-Shamir transform). That is, it does not pinpoint precisely the source of the difficulty and, in particular, it leaves open the question whether any adaptive proofs exist that are not also straight-line[2].

Our Contribution. In this paper we obtain a full characterization of adaptive proofs in the random oracle model and we leverage this result to provide more general results regarding the limitations of the Fiat-Shamir transform. Below we outline our contributions.

Adaptive proofs ≡ proofs with straightline extractors. Our main contribution is a negative answer to Bernhard et al.'s open question. It holds for

[1] Recall that in the random oracle model hash functions can only be computed by calling an oracle available to all parties (and which on a fresh input returns a truly random output).

[2] Straight-line proofs are trivially adaptively secure.

all non-interactive proof schemes in the ROM, whether or not they are derived from Σ-protocols:

Theorem 1 (Informal). *Consider an arbitrary non-interactive proof of knowledge system in the ROM. If the proof system has an efficient adaptive extractor against adaptive provers then it also has a straight-line extractor.*

The immediate consequence of this theorem is that when designing PoKs for an adaptive setting, one cannot rely on rewinding and instead one should ensure the existence of a straightline extractor. While a general strategy is to employ Fischlin's transformation [11], one may still want to rely on the more efficient construction that uses the Fiat-Shamir transformation whenever possible – the impossibility result of Bernhard et al. [5] only applies to Fiat-Shamir-Schnorr proofs.

Limitations of the Fiat-Shamir transform. We show that the Fiat-Shamir transformation has intrinsic limitations. In particular, we generalize the results of [5] in two distinct directions. On the one hand, we show that it holds for arbitrary Σ-protocols for proving knowledge of preimages of linear functions (including Schnorr, Chaum–Pedersen and representation proofs). More interestingly, we weaken the condition under which these proofs are not adaptively secure from one-more discrete logarithm (resp. one-more one-wayness [1]) to the following assumption: a dishonest verifier in a single execution of the Σ-protocol cannot extract the witness. We call this assumption *Σ-one-wayness.* Our result thus improves from a "q-type" assumption, which does not admit an efficient game, to an efficient game with only three rounds.

This theorem answers the main open question of [5] and hints at a shortcoming of proofs based on the Fiat–Shamir transform: if used in a setting where the prover gets to adaptively chosen statements, an extractor would have to be (more or less) straight-line. However, if this is the case, the proof may not be that interesting anyway as the underlying witness is not well-protected:

Theorem 2 (Informal). *Suppose there is a straight-line extractor for a Fiat-Shamir transformed Σ-protocol Σ (to prove knowledge of a preimage of some linear function f). Then a dishonest verifier can extract a witness (a preimage under f) in a single run of Σ.*

Taken together, these results imply that Fiat-Shamir transformed Σ-protocols are not adaptively secure in any setting in which they might be useful. Take the Schnorr protocol as an example: if Fiat-Shamir-Schnorr is adaptively secure then either discrete logarithms are easy in the relevant group, in which case the Schnorr protocol is redundant, or the Schnorr protocol provably helps a dishonest verifier to extract the discrete log of the statement — in which case Schnorr is certainly not zero-knowledge.

The theorem of Bernhard et al. [5] contains an adaptive prover who makes a sequence of n proofs such that each one depends on all previous ones. The straightforward rewinding strategy — rewind on every proof to extract — ends up rewinding 2^n times since the rewound provers make new proofs which again have to be rewound. A combinatorial argument then shows that any strategy that rewinds fewer than 2^n times must have taken a discrete logarithm to find out the witness for one of the proofs output by the prover. The problem is that we do not know where, and also if we inject a challenge in one proof then we end up having to simulate all other proofs in the experiment. So far the solution to this problem was to reduce to the one-more discrete logarithm problem.

Our proof technique for Theorem 2 (formally, Theorem 14) is to take any prover and turn it into an adaptive prover who makes a chain of n proofs, together with some bookkeeping. We then show that any adaptive extractor against this prover must either take exponential time or reduce to a straight-line extractor against the original prover. Applying this theorem to the honest prover, we get a reduction to Σ-one-wayness. We summarize these results in the following table (FSS = Fiat-Shamir-Schnorr, DLOG = discrete logarithm, OMDL = one-more discrete logarithm).

Property	Breaks if FSS has
One-way (e.g. DLOG)	Straight line extractor [18]
Σ-one-way	Adaptive extractor (new)
One-more one-way (e.g. OMDL)	Adaptive extractor [5]

In conclusion, we suggest that adaptive proofs are not a new class of proofs but rather another description of the class of straight-line extractable proofs; and Fiat-Shamir transformed Σ-protocols for "useful" functions are *not* in this class.

Weaker assumptions. The obvious next question is whether one can improve our results even further and only rely on one-wayness (e.g. in the case of Schnorr, DLOG) rather than Σ-one-wayness. We show using a meta-metareduction that no algebraic metareduction [16] from a programming extractor to one-wayness (e.g. DLOG) can exist, unless one-wayness is already easy. All previous metareductions in this area [5,18] including ours to Σ-one-wayness are algebraic: the only operations they perform on elements of the target group are group operations.

A generalization of the Generic Group Model (GGM). To strengthen trust in the Σ-one-wayness hypothesis on which our impossibility relies we provide a justification in the generic group model. Interestingly, the first problem that one needs to face here is that the existing approaches to formalizing and using GGM is not suitable: in brief, an adversary that interacts with the Σ protocol for some problem gets to see not only group elements but also some information related to the exponents of these group elements – this ability is not considered the standard GGM formalizations. We suggest one approach to deal with this issue and use the resulting model to formally justify Σ-one-wayness.

Related Work. One-more type assumptions were introduced by Bellare et al. [1]. Their value in proving schemes secure is subject to some debate, as explained by Koblitz and Menezes [13] who also gave the first "weakened" one-more assumption. Problems with forking-based proofs were first noted by Shoup and Gennaro [20]; Paillier and Vergnaud [16] developed separation results for Schnorr-based signatures using metareductions that formed the first formal proof of a limitation of Schnorr-based techniques. Both Brown [8] and Bresson et al. [9] concurrently applied separation techniques to one-more problems. Fischlin and Fleischhacker [12] were the first to consider limitations of metareductions via meta-metareductions. The most recent results that motivated this paper are Seurin and Treger [18] who gave a very simple metareduction from a non-programming extractor for Schnorr proofs to discrete log; and Bernhard et al. [5] who introduced adaptive proofs.

2 Preliminaries

Notation. $f : A \rightarrow B$ is a function with domain A and range B; $\mathcal{A} : A \twoheadrightarrow B$ is a randomised algorithm on the same domain and range. We write security games in a language based on Bellare and Rogaway's code-based game-playing [2]. $y \leftarrow f(x)$ is assignment, $x \twoheadleftarrow R$ is uniform random sampling. $T[i]$ is the element at index i of table T.

An interactive, randomised algorithm $\mathcal{A}$ has access to a random string r and an input/output interface. It maintains its state between calls. A security game is such an algorithm that may at some point output "win" or "lose", which terminates the entire execution. We say that a security property is given by a game, to mean that the property holds if no efficient adversary can cause the game to output "win" with more than negligible probability in some underlying security parameter.

Σ-protocols — Let k be a field. Let $\mathcal{W}, \mathcal{X}$ be k-vector spaces and let $\phi : \mathcal{W} \rightarrow \mathcal{X}$ be a k-linear map, (i.e. linear map w.r.t. the field k). Suppose further that one can sample uniformly from k and $\mathcal{W}$.

Definition 3. *The Σ-protocol Σ_ϕ is the following interactive protocol for a prover P to prove knowledge of a preimage under ϕ to a verifier V.*

$$\begin{array}{rcl} P(w \in \mathcal{W}, x = \phi(w)) & & V \\ r \twoheadleftarrow \mathcal{W}; a \leftarrow \phi(r) & \xrightarrow{(x,a)} & \\ & \xleftarrow{c} & c \twoheadleftarrow k \\ s \leftarrow r + c \cdot w & \xrightarrow{s} & \phi(s) \overset{?}{=} a + c \cdot x \end{array}$$

We choose to let P transmit the statement x to V as part of the first round of the proof — of course, V may also know x in advance. The verifier accepts if the equation $\phi(s) = a + c \cdot x$ holds in $\mathcal{X}$, in which case we call (x, a, c, s) an accepting transcript. Instances of this template protocol include:

- Schnorr: $\mathcal{W} = k = GF(p)$, $\mathcal{X}$ is some group G of order p with a generator g (e.g. over an elliptic curve) and $\phi(w) = g^w$.

- Chaum-Pedersen: $\mathcal{W} = k$, $\mathcal{X} = G \times G$ for a group as above and $\phi(w) = (g^w, h^w)$ for two different generators g, h of G.
- Representation: $\mathcal{W} = k^n, \mathcal{X} = G$ and $\phi(w_1, \ldots, w_n) = \prod_{i=1}^n {g_i}^{w_i}$ for some known set of generators $\{g_i\}_{i \in I}$ of G.

Σ-protocols according to our definition automatically satisfy:

- Special soundness: if (x, a, c, s) and (x, a, c', s') are accepting transcripts with $c \neq c'$ then $1/(c - c') \cdot (s - s')$ is a preimage[3] of x under ϕ.
- Soundness: if $x' \in \mathcal{X} \setminus \mathrm{Im}[\phi]$, then a cheating prover gets a verifier to accept with probability at most $1/|k|$.
- Honest-verifier zero-knowledge: a verifier who chooses c as prescribed (at least, independently of a) gains no information from the protocol beyond the fact that the prover knows a preimage of x under ϕ.

Proof schemes — A (non-interactive) proof scheme for a relation ρ on sets X, W consists of a proof space Π and a pair of algorithms `prove` $: X \times W \twoheadrightarrow \Pi$ and `verify` $: X \times \Pi \rightarrow \{0, 1\}$ (i.e. `verify` is deterministic). An element $\pi \in \Pi$ satisfying `verify`$(x, \pi) = 1$ is called a valid proof for x. For any (x, w) satisfying ρ, if $\pi \leftarrow$ `prove`(x, w) then we require `verify`$(x, \pi) = 1$. We further assume that there is an algorithm `sample` $:\twoheadrightarrow X \times W$ that produces elements uniformly distributed in ρ (as a subset of $X \times W$). In the random oracle model (ROM), both `prove` and `verify` may call a function H that is modelled as a random oracle in security proofs. The relation ρ itself does not use H.

Fiat-Shamir — The Fiat-Shamir transformation turns Σ-protocols into non-interactive proof schemes that are full zero-knowledge proofs of knowledge in the random oracle model. The idea is simply to replace the verifier's challenge c by a hash over the statement x and the commitment a.

Definition 4. *Let $\phi : \mathcal{W} \rightarrow \mathcal{X}$ be a k-linear function where $\mathcal{W}$ and k are efficiently sampleable. Suppose that H is a function with domain (including) $\mathcal{X} \times \mathcal{X}$ and range k. Then the Fiat-Shamir transformed Σ-protocol $\mathcal{F}_\phi$ is the following proof scheme for sets $(\mathcal{X}, \mathcal{W})$ and relation $\rho(x, w) = 1 \iff \phi(w) = x$. The proof space is $\Pi = \mathcal{X} \times \mathcal{W}$ and the algorithms are*

- `sample()`*: pick $w \twoheadleftarrow \mathcal{W}$, set $x \leftarrow \phi(w)$ and return (x, w).*
- `prove(`x, w`)`*: pick $r \twoheadleftarrow \mathcal{W}$, set $a \leftarrow \phi(r)$, $c \leftarrow H(x, a)$ and $s \leftarrow r + cw$. The proof is $\pi = (a, s)$.*
- `verify(`$x, (a, s)$`)`*: check that $\phi(s) = a + H(x, a) \cdot x$.*

[3] The inversion $1/(c - c')$ is in the field k where it exists due to $c \neq c'$; the dot in this formula is field-vector multiplication.

3 Variations on the Theme of One-Wayness

Σ-protocols are only useful when the function ϕ is hard to invert: otherwise, they are trivially zero-knowledge proofs of knowledge, but so is the protocol in which the prover just sends the statement x to the verifier. For the same reasons, if ϕ is easy to invert then $\mathcal{F}_\phi$ is adaptively secure too. This shows that we cannot hope for a theorem of the form "Fiat-Shamir Schnorr is not adaptively secure", since it is adaptively secure e.g. in the group $(\mathbb{Z}_p, +)$ where taking discrete logarithms is easy. Consequently, limitation theorems take the form "if Schnorr is adaptively secure then some property (e.g. OMDL) is easy to break".

We discuss some possible security properties of the function ϕ and introduce Σ-one-wayness. Later on we will show that Fiat-Shamir proofs cannot be adaptively secure unless Σ-one-wayness of ϕ is easy to break (in which case, their use within protocols would be already questionable).

Σ-ONE-WAYNESS. One-wayness is the first obvious candidate property. Recall that a property defined by a game means that the property (in this case one-wayness of ϕ) holds if it is hard to make the game output "win". The game gives the adversary a uniformly chosen image x; the adversary wins by recovering any preimage w' s.t. $\phi(w') = x$.

Definition 5. *The one-wayness property for a function $\phi : \mathcal{W} \to \mathcal{X}$ is given by the following game.*

```
$w \twoheadleftarrow \mathcal{W}; x \leftarrow \phi(w)$
output $x$; input $w' \in \mathcal{W}$
if $\phi(w') = x$ then return "win" else return "lose" end
```

We propose a new security notion that we call Σ-one-wayness (for linear functions). This says that even a dishonest verifier (who chooses c arbitrarily, maybe depending on x and a) cannot extract w from a single run of the protocol. We do not claim that Σ-one-wayness is a sufficient security notion for Σ protocols (it says nothing about extracting partial information on w) but we postulate that it is only deployed if this condition is satisfied. Consequently, we are not proposing a new scheme that is secure under the Σ-one-wayness assumption, but we will claim that if the Fiat-Shamir proof scheme for a function ϕ is adaptively secure then the Σ-one-wayness property for ϕ is easy to break too. Σ-one-wayness is clearly stronger than one-wayness of the function ϕ, but it will turn out to be weaker than one-more one-wayness (which we define in a moment).

Definition 6. *The Σ-one-wayness property for a linear function $\phi : \mathcal{W} \to \mathcal{X}$ is defined by the following game.*

```
$w \twoheadleftarrow \mathcal{W}; x \leftarrow \phi(w)$
$r \twoheadleftarrow \mathcal{W}; a \leftarrow \phi(r)$
output $(x, a)$; input $c \in k$
$s \leftarrow r + c \cdot w$
output $s$; input $w' \in \mathcal{W}$
if $\phi(w') = x$ then return "win" else return "lose" end
```

If Σ-one-wayness of a function ϕ is easy to break but one-wayness is hard to break, then running the protocol Σ_ϕ could leak a preimage to a dishonest verifier, that said verifier could otherwise not compute by herself. In this case we would discourage the use of the protocol Σ_ϕ (among other things it is certainly not zero-knowledge). If both Σ-one-wayness and one-wayness of ϕ are easy to break then Σ_ϕ is both harmless and useless. So, we think that Σ-one-wayness of ϕ is a necessary condition for the protocol Σ_ϕ to be deployed. Under this condition, we will show that the Fiat-Shamir transformed Σ_ϕ is not adaptively secure.

Weak one-more one-wayness. For Schnorr, the one-wayness property of the function $\phi(x) = g^x$ is the discrete logarithm property. Bernhard et al. [5] use a stronger assumption known as one-more discrete logarithm, which generalises to one-more one-wayness [1,9].

One-more one-wayness assumes an invertible function ϕ. The adversary is given two oracles which she can call in any order: a sampling oracle that picks a random preimage $w_i \in \mathcal{W}$ and reveals $x_i = \phi(w_i)$ and an opening oracle that on input x outputs $\phi^{-1}(x)$. To win the game, the adversary must recover the preimages of all samples x_i with fewer calls to the opening oracle than to the sampling oracle.

One-more one-wayness was first discussed by Bellare et al. [1] although the name first appears in a later paper [9]. Unlike the other properties here, it does not admit an efficient security game: the game itself needs to be able to invert ϕ on arbitrary inputs. Koblitz and Menezes [13] discussed a variant that only allows the adversary to open challenges themselves; this is insufficient for our applications. Instead we propose a new property: weak one-more one-wayness fixes this problem by restricting the adversary to asking linear combinations of the sampled challenges. Your task is still to recover all w_i with fewer queries to the linear combination oracle than to the sampling oracle. The requirement for ϕ to be invertible can also be dropped again.

Definition 7. *The weak and normal one-more one-wayness properties for a bijection $\phi : \mathcal{W} \rightarrow \mathcal{X}$ are given by the following games. In each game the adversary can call the `sample` and `open` oracles many times, in any order. The adversary wins the game if it can provide preimages under ϕ for all samples that it had obtained and yet it made fewer opening queries than sample queries.*

weak one-more one-wayness:

```
sample():
    $n \leftarrow n + 1$
    $w[n] \twoheadleftarrow \mathcal{W}$
    return $\phi(w[n])$

open($c_1, \ldots, c_n \in k^n$):
    return $\sum_{i=1}^{n} c_i \cdot w_i$
```

one-more one-wayness:

```
sample():
    (same as weak version)



open($x \in \mathcal{X}$):
    return $\phi^{-1}(x)$
```

The strong problem clearly reduces to the weak one. A weak adversary can still obtain a preimage of a particular sample by submitting the vector with 1 at the appropriate position and 0 elsewhere.

The point of the weak one-more one-wayness property is that the theorem by Bernhard et al. [5] can trivially be strengthened to show that Fiat-Shamir-Schnorr is not adaptively secure even under the weak one-more discrete logarithm property: their reduction only ever makes opening queries on elements that are linear combinations of samples with known coefficients. This is not surprising since their reduction is trying to "simulate" Schnorr proofs on sample elements.

Σ-one-wayness reduces to weak one-more one-wayness, even to a weaker version where the number of samples is additionally bounded at 2 and only a single linear combination query is allowed. Thus, we end up with a hierarchy of one-wayness/Σ-one-wayness/weak one-more one-wayness/one-more one-wayness, in order of increasing strength.

4 Adaptive Proofs

In this section we recall the notion of adaptive proofs and introduce templates for a couple of provers that form the basis of the results we prove in the next section.

Definition 8. *A prover is an algorithm $\mathcal{P}$ that outputs a statement/proof pair (x, π); in the ROM a prover may make random oracle calls. We assume that there is a uniformly sampleable space of random strings R associated to each prover and we write $\mathcal{P}(r)$ to mean running prover $\mathcal{P}$ on random string $r \in R$. In the ROM, we write $(x, \pi, l) \leftarrow \mathcal{P}(r)$ to mean that we also return the list l of random oracle queries made by this execution of the prover on random string r.*

A proof scheme is sound with error ε if for any prover $\mathcal{P}$, the probability of producing a pair (x, π) such that $\texttt{verify}(x, \pi) = 1$ but no w exists making $\rho(x, w)$ hold, is at most ε.

A proof scheme in the random oracle model is straight-line extractable with error ε if there is an extractor $\mathcal{K}$ as follows. For any prover $\mathcal{P}$, pick $r \twoheadleftarrow R$ and execute $(x, \pi, l) \leftarrow \mathcal{P}(r)$. If $\texttt{verify}(x, \pi) = 1$ w.r.t.[4] l then with probability at least $1 - \varepsilon$, $\mathcal{K}(x, \pi, l)$ returns a w such that $\rho(x, w)$ holds. It follows immediately that an extractable proof scheme with error ε is also sound with at most the same error.

A straight-line extractor, has black-box access to further copies of the prover in the following sense: it may start these copies and control all interaction with them, including picking the random string and answering all random oracle queries. As motivation for this (established) notion of straightline extractors, consider an extractor who is trying to extract from an honest prover. The code

[4] We say $\texttt{verify}(x, \pi) = 1$ w.r.t. l if all elements on which verify queries the oracle on input (x, π) are contained in l, and verify outputs 1 on these inputs if given the appropriate responses in l.

of the honest prover is known, so the extractor can always simulate the honest prover on inputs of its choice. The extractor cannot see the random coins of the "main" copy of the honest prover from which it is trying to extract, however.

A proof scheme in the ROM has a *programming* straight-line extractor with error ε if there is a straight-line extractor $\mathcal{K}$ as follows. Let any prover $\mathcal{P}$ interact with $\mathcal{K}$ in the sense that $\mathcal{K}$ answers $\mathcal{P}$'s random oracle queries. If $\mathcal{P}$ outputs (x, π) such that $\texttt{verify}(x, \pi) = 1$ w.r.t. the oracle queries made by $\mathcal{P}$, then with probability at least $1 - \varepsilon$, $\mathcal{K}$ outputs a w such that $\rho(x, w)$ holds.

A rewinding extractor can, in addition to the capabilities of a straight-line extractor, launch further copies of the prover $\mathcal{P}$ with the *same random string* as the one that the extractor is trying to extract from, and answer their random oracle queries. The difference between straight-line and rewinding extractors is thus that rewinding extractors can run further copies of the prover that behave identically to the "main" one, as long as they receive the same inputs and outputs, and thus "fork" the prover instance from which they are trying to extract.

Adaptive proofs. We present the adaptive proof game of Bernhard et al. [5]. The game is an interactive algorithm with two interfaces for an adaptive prover and an extractor. An adaptive prover for a proof scheme $(\varPi, \texttt{prove}, \texttt{verify})$ w.r.t. (X, W, ρ) is an algorithm that can repeatedly output pairs $(x, \pi) \in X \times \varPi$ and expect witnesses $w \in W$ in return. After a number of such interactions, the adaptive prover halts. In the random oracle model, an adaptive prover may also ask random oracle queries.

An adaptive extractor is an algorithm $\mathcal{K}$ that can interact with an adaptive prover: it repeatedly takes pairs $(x, \pi) \in X \times \varPi$ such that $\texttt{verify}(x, \pi) = 1$ as input and returns witnesses $w \in W$ such that $\rho(x, w) = 1$. In addition, an adaptive extractor may be rewinding, i.e. launch further copies of the adaptive prover that run on the same random string as the main one (managed by the game) and answer all their queries. The adaptive proof game does not check the correctness of witnesses for the other copies of the prover.

Definition 9. *A proof scheme $(\varPi, \texttt{prove}, \texttt{verify})$ is an n-proof (with error ε) in the random oracle model if there is an adaptive extractor $\mathcal{K}$ such that for any adaptive prover $\mathcal{P}$, the extractor wins the game in Fig. 1 (with probability at least $1 - \varepsilon$). The adaptive extractor $\mathcal{K}$ may launch and interact with further copies of the adaptive prover $\mathcal{P}$ on the same random string r as the main one, without the game mediating between them.*

In the n-proof game in Fig. 1, the prover $\mathcal{P}$ is trying to find a claim (x, π) that verifies, but from which the extractor $\mathcal{K}$ cannot extract a witness w. The extractor is trying to extract witnesses from all claims made by the prover.

The game uses three global variables. K stores the number of witnesses that the extractor has found so far. If this counter reaches n, the extractor wins and the scheme is an n-proof. Q stores a list of all the prover's random oracle queries so far. These are provided[5] to the extractor along with each of the prover's

[5] The original definition gave the extractor an extra `list` oracle to query the prover's random oracle list. Our presentation is equivalent.

```
$initialise$:
    $Q \leftarrow [\ ]$
    $K \leftarrow 0$
    $r \twoheadleftarrow R$
    run $\mathcal{P}(r)$

$\mathcal{P}$ asks ro($x$):
    $y \leftarrow$ ro($x$)
    $Q \leftarrow Q :: (x, y)$
    send $y$ to $\mathcal{P}$

$\mathcal{P}$ outputs $(x, \pi)$:
    if not verify($x, \pi$) then
        $\mathcal{K}$ wins; halt.
    end
    $\Xi \leftarrow x$
    send $(x, \pi, Q)$ to $\mathcal{K}$
$\mathcal{K}$ asks ro($x$):
    $y \leftarrow$ ro($x$)
    send $y$ to $\mathcal{K}$

$\mathcal{K}$ outputs $w$:
    if $\rho(\Xi, w)$ then
        $K \leftarrow K + 1$
        if $K = n$ then
            $\mathcal{K}$ wins; halt.
        end
        send $w$ to $\mathcal{P}$
    else
        $\mathcal{P}$ wins; halt.
    end
$\mathcal{P}$ halts:
    $\mathcal{K}$ wins; halt.
```

Fig. 1. The adaptive proof game. The extractor also has access to further copies of $\mathcal{P}(r)$ – to avoid cluttered notation we do not show this access explicitly.

claims. Ξ stores the last statement that appeared in one of the prover's claims; it is used to check the validity of a witness returned by the extractor.

A n-proof for $n = 1$ is simply a proof of knowledge in the ROM: the prover makes a single claim (a pair containing a statement x and a proof π) and the extractor wins if it can obtain a witness. A proof scheme is an adaptive proof if there is an adaptive extractor that works for any polynomially bounded parameter n.

Canonical provers. The canonical prover $\mathcal{P}_C$ samples a statement/witness pair and creates a proof. We write R_C for the randomness space of the canonical prover and $\mathcal{P}_C(r)$ to denote running the canonical prover on the random string $r \in R_C$.

Definition 10 (canonical prover). *Let* $(\Pi, \texttt{prove}, \texttt{verify})$ *be a proof scheme for* (X, W, ρ) *where* r *is uniformly sampleable via an algorithm* `sample`*. The canonical prover* $\mathcal{P}_C$ *for this scheme is the following algorithm.*

$$(x, w) \leftarrow \texttt{sample();}\ \ \pi \leftarrow \texttt{prove}(x, w);\ \ \texttt{return}\ (x, \pi)$$

If required, the canonical prover can also return the list l *of all random oracle queries made during its execution (by* `prove`*).*

Since an extractor is supposed to work against any (efficient) prover, to argue that an extractor cannot exist it is enough to show that one cannot extract from the canonical prover. To deal with adaptive extractors, we propose the following construction of an adaptive chain prover $\mathcal{P}^n$ from any prover $\mathcal{P}$. It follows the idea of Shoup and Gennaro [20] in making a chain of "challenges" (in this case proofs) where each challenge depends on all previous ones and then asking queries on them *in reverse order*. This way, the obvious rewinding extractor using special soundness will take exponential time.

To make each challenge depend on previous ones, we use a function F to update the random string for each of the n contained copies of $\mathcal{P}$ based on the (random oracle) state of the previous copy. The final parameter l returned by $\mathcal{P}$ is the list of all random oracle queries made by this copy. Recall that $\mathcal{P}(r)$ means run prover $\mathcal{P}$ on random string $r \in R$ where R is the randomness space for this prover.

Definition 11 (adaptive chain prover). *Let* $(\Pi, \texttt{prove}, \texttt{verify})$ *be a proof scheme for* (X, W, ρ)*. Let* $\mathcal{P}$ *be any prover (in the ROM) and let* R *be its randomness space. Let* L *be the space of possible random oracle input/output transcripts. Let* $F : R \times L \rightarrow R$ *be a function (which does not depend on the random oracle). The adaptive chain prover* $\mathcal{P}^n$ *of order* n *w.r.t. function* F *is the algorithm in Fig. 2, taking an* $r \in R$ *as input.*

```
$\mathcal{P}^n(r)$:
for $i$ = 1, $n$ do
    $(x_i, \pi_i, l) \leftarrow \mathcal{P}(r)$
    $r \leftarrow F(r, l)$
end
for $i$ = $n$, 1, -1 do
    output $(x_i, \pi_i)$
    input $w'$
    if not $\rho(x_i, w')$ then
        halt
    end
end
```

Fig. 2. The adaptive chain prover $\mathcal{P}^n$. The provder depends on function F which here we assumed fixed. When F is a pseudorandom function, a key for F is sampled at the beginning of the execution of $\mathcal{P}^n$.

Later on, we will take F to be a (pseudo-)random function. This has the effect that two copies of $\mathcal{P}^n$ that get identical answers to their random oracle queries will behave identically, but two copies of $\mathcal{P}^n$ that "fork" will behave as copies of $\mathcal{P}$ with *independent* random strings from the forking point onwards.

The intuition behind this construction is that having access to copies of $\mathcal{P}$ on some uniformly random string r' cannot help you extract from a copy $\mathcal{P}(r)$, as long as r and r' are independent — certainly, an extractor could always simulate such copies herself if the code of $\mathcal{P}$ is known. We will use this idea to show that forking a copy of $\mathcal{P}^n$ is no help in extracting from the proofs made later on by another copy.

5 Limitations of the Fiat-Shamir Transformation

We recall the hierarchy of security definitions for functions (the last is the strongest): one-way/Σ-one-way/weak one-more one-way/one-more one-way.

Known limitations. Seurin and Treger [18] proved that Fiat-Shamir-Schnorr cannot have a non-programming straight-line extractor unless the underlying function is not one-way (i.e. one can take discrete logarithms). The following theorem generalizes this result to the case of arbitrary Σ-protocols.

Theorem 12. *Suppose there is a non-programming straight-line extractor K for the proof scheme $\mathcal{F}_\phi$. Then ϕ is not one-way. Specifically, there is an algorithm breaking one-wayness with approximately the same running time and success probability as the extractor K has against the canonical prover $\mathcal{P}_C$.*

Proof. Let x be a challenge from the one-way game for ϕ; we need to find a w' such that $\phi(w') = x$. We simulate a proof: pick $s \twoheadleftarrow \mathcal{W}$, $c \twoheadleftarrow k$ and $a \leftarrow \phi(s) - c \cdot x$. Then we give the extractor the statement x, proof (a, s) and a list of random oracle queries consisting of the entry $RO(x, a) = c$. These elements are identically distributed to what the extractor would see in an execution with the canonical prover $\mathcal{P}_C$ for $\mathcal{F}_\phi$. We pass any witness w' returned by the extractor on to the challenger to win with the same success probability. □

Bernhard et al. [5] showed that substituting an adaptive extractor for a straight-line one gets a similar result for the one-more one-wayness assumption on the function ϕ — the proof of this theorem is nontrivial however. While their original proof only concerned Fiat-Shamir-Schnorr, a close inspection of the proof shows that it works for any Σ protocol and that the weak one-more assumption is sufficient too. In summary:

Theorem 13. *Suppose that there is an efficient adaptive extractor for $\mathcal{F}_\phi$. Then ϕ is not weak one-more one-way.*

We will not re-prove the theorem here. As to running time, as long as the extractor makes fewer than 2^n queries when running against a particular prover then the reduction to weak one-more one-wayness runs in the same time as the extractor, but its success probability is the inverse of the number of copies of the prover that the extractor causes to be invoked. Although Bernhard et al. [5] only prove the theorem for the case of the Fiat-Shamir-Schnorr protocol, their reduction is "black box" in the sense that it only needs to be able to sample and open instances of the underlying Σ-protocol. The exact same proof will work for our generalisation. We can write the prover in the cited theorem as $(\mathcal{P}_C)^n$, the adaptive chain prover derived from the canonical prover. This will allow us to conclude that any extractor against the chain prover implies a straight-line extractor against the canonical prover $\mathcal{P}_C$.

New Results. If we switch to a programming straight-line extractor, we can show a separation result under the Σ-one-wayness assumption.

Theorem 14. *Suppose there is a programming straight-line extractor K for $\mathcal{F}_\phi$. Then ϕ is not Σ-one-way. Specifically, there exists a reduction with approximately the same running time and the same success probability as the extractor K against the canonical prover $\mathcal{P}_C$.*

Proof. We simulate the canonical prover towards the extractor. Receive x, a from the Σ-one-way challenger and ask the random oracle query $c \leftarrow RO(x, a)$ (which the extractor answers). Then send c to the Σ-one-way challenger to get s and send (x, a, s) to the extractor. Again, whenever the extractor provides the correct w', we win against the challenger. □

This result is new, but not surprising — Fiat-Shamir transformed Σ-protocols are not supposed to be straight-line extractable and the Σ-one-wayness property is constructed exactly to make this reduction work. The value of said property is that we can also use it for adaptive extractors.

Our main contribution in this paper is a new theorem that says all adaptive proofs in the ROM admit a straight-line extractor.

Theorem 15. *Consider any non-interactive ROM proof scheme with an adaptive extractor $\mathcal{K}$. Suppose that, running against any n-prover $\widehat{\mathcal{P}}$, the extractor $\mathcal{K}$ causes at most $f(n) < 2^n$ copies of the prover to be run in the experiment and answers all extraction queries of the main run correctly with probability at least $p(n) > 0$. Then there is a programming straight-line extractor against any non-adaptive prover $\mathcal{P}$ with success probability $p(n)/(n \cdot f(n))$.*

Remark 16. The proof, which we provide in the full version, is information theoretic. It relies only on the number of copies of P instantiated by $\mathcal{K}$ and, in particular, it makes no assumptions on the efficiency of $\mathcal{P}$ and $\mathcal{K}$. It does not establish a relationship between the running time of $\mathcal{K}$ and the success probability; it is simply an application of the pigeonhole principle to derive a contradiction that whenever $f(n) < 2^n$ then $\mathcal{K}$ must essentially have "guessed correctly" rather than computed the preimage through interacting with $\mathcal{P}$. For example, if launching a new copy of $\mathcal{P}$ costs $\mathcal{K}$ one unit of time then the theorem provides negative results even for subexponential-time extractors.

Applying this theorem to protocols obtained via the Fiat-Shamir transform from Σ-protocols yields the following insight.

Corollary 17. *Suppose that the Fiat-Shamir transformed Σ-protocol $\mathcal{F}_\phi$ is adaptively secure. Then ϕ is not Σ-one-way secure.*

For the corollary, note that we have a programming straight-line extractor against the canonical prover $\mathcal{P}_C$ by applying Theorem 15 to the prover $(\mathcal{P}_C)^n$. The result then follows from Theorem 14.

We sketch the proof here and provide the full argument in the full version of this paper [7]. Let $\mathcal{P}$ be a non-adaptive prover. We construct a simulator $\mathcal{S}^n$ that is indistinguishable from the black box providing "rewinding" access for multiple copies of $\mathcal{P}^n$. The point of the simulator is that it shares state "between the copies". We then guess which instance of the prover (specifically, which proof) the extractor is going to answer without "forking" and inject the Σ-one-wayness challenge into it. The same combinatorial argument as in the proof of Bernhard et al. [5] shows that such an instance must exist if the extractor launches fewer than 2^n copies of the prover.

The core of such a simulation argument is to keep track of a history of each instance of the prover, since two copies of the prover with identical histories must behave identically towards the extractor. In $(\mathcal{P}_C)^n$, this history is implicitly tracked in the randomness r used for each copy of $\mathcal{P}_C$, however a collision in the random oracle could lead to two copies with different histories "merging". Our simulator computes an explicit history instead, namely the list of all random oracle queries so far.

As in Bernhard et al. [5] we define an event E that occurs whenever a copy of the prover gets its extraction query answered without having a "partner" (another copy, from which the witness was extracted by forking and special soundness). The novelty in our proof is that because we have cast the prover as a chain $(\mathcal{P}_C)^n$ with suitable state tracking, we can show that event E implies not only a "break" of the chain prover but also of one of the contained canonical provers $\mathcal{P}_C$. We then show that, if the simulator guessed correctly, event E implies that the simulator can solve its Σ-one-wayness challenge. This is a much weaker assumption than one-more one-wayness.

6 Generic Hardness of Σ-One-Wayness

In this section we show that Fiat-Shamir transformed Σ-protocol $\mathcal{F}_\phi$ is not adaptively secure in the generic setting. Thus if we want to build a protocol where we need an adaptively secure proof (such as to get CCA encryption), we would not use $\mathcal{F}_\phi$. By Corollary 17 we just need to prove that ϕ is $\Sigma-$one-way secure in the generic group model (GGM). Again, let $\mathcal{X}$, $\mathcal{W}$ be vector spaces over k and $\phi : \mathcal{W} \to \mathcal{X}$ be a k-linear map. We assume that k is a finite field and $\mathcal{X}, \mathcal{W}$ are finite dimensional, since sampling uniformly from an infinite set does not make much sense. Let $b_1, b_2, ..., b_n$ be basis vectors of $\mathcal{X}$, where $\dim(\mathcal{X}) = n$ and $\{b_1, ..., b_s\}$ be a basis for $\mathrm{Im}(\phi)$. For each $1 \leq i \leq s$ denote a_i to be an element of $\mathcal{W}$ so that $\phi(a_i) = b_i$.

The generic group model is a model which analyses success of algorithms against representations of groups which do not reveal any information to adversary. There are many ways to formalise this idea [14,15,19]. We will follow the definition provided by Shoup [19]. Here, adversary is given access to images of elements of a group under a random injective map $\sigma : \mathcal{X} \to S \subset \{0,1\}^*$, called an *encoding function.* Group operations can be computed by making oracle queries. The adversary is given access to two oracles ADD and INV:

$$ADD(\sigma(x), \sigma(y)) = \sigma(x+y), \; INV(\sigma(x)) = \sigma(-x).$$

Note that the adversary cannot get any information from the representation $\sigma(x)$ of element x. A *generic algorithm* $\mathcal{A}$ for $\mathcal{X}$ on S is a probabilistic algorithm that takes as input an encoding list $(\sigma(x_1), \sigma(x_2), ..., \sigma(x_l))$ where each $x_i \in \mathcal{X}$ and σ is an encoding function of $\mathcal{X}$ on S. As the algorithm executes, it makes queries to ADD or INV oracles and then appends outputs of the queries to the encoding list. The output of $\mathcal{A}$ is a bit string denoted as $\mathcal{A}(\sigma; x_1, ..., x_l)$. We also want to extend the interface of the model and introduce the $FSS_{r,w}$ oracle:

$FSS_{r,w}(c) = r + cw$ for some $r, w \in \mathcal{W}$ and $c \in k$. We may assume that when oracles encounter some $\sigma_1 \notin Im(\sigma)$, they return an error message.

The next theorem establishes generic hardness of $\Sigma-$one-wayness; we provide the proof in the full version of this paper [7].

Theorem 18. *Let w, r be random elements of $\mathcal{W}$ and $\mathcal{A}$ be a generic algorithm for $\mathcal{X}$ on $S \subset \{0,1\}^*$ that makes at most m queries to ADD and INV oracles and exactly one query to $FSS_{r,w}$ oracle. Then the probability that $\phi(\mathcal{A}(\sigma; b_1, ..., b_n, x, y)) = x$ is $O((m+n)^2/|\mathcal{X}| + |ker(\phi)|/|\mathcal{W}|)$, where $x = \phi(w)$ and $y = \phi(r)$.*

Note that Theorem 18 implies that every generic algorithm $\mathcal{A}$, which wins $\Sigma-$one-wayness with high probability, must perform at least $\Omega(\alpha\sqrt{|\mathcal{X}|})$ group operations, where $\alpha = \sqrt{1 - |\ker(\phi)|/|\mathcal{W}|}$. In particular, if $\mathcal{W}$ is large then we get the lower bound $\Omega(\sqrt{|\mathcal{X}|})$ for IES (described in [6]) by choosing $\phi(w) = g^w$.

7 Reducing to DLOG?

Given a non-programming straight line extractor in the ROM for a Fiat-Shamir transformed Σ-protocol $\mathcal{F}_\phi$ we can break one-wayness of ϕ; for a programming extractor or an adaptive extractor we can break Σ-one-wayness. This raises the question, can we break one-wayness given a programming extractor? Our answer is negative. We give the argument for the case of Schnorr proofs where one-wayness is the discrete logarithm (DLOG) problem; this also implies that there can be no generic metareduction to one-wayness for any Σ-protocol.

The metareductions in the theorems of Seurin and Treger [18], Bernhard et al. [5] and this paper are all algebraic (in the sense of Paillier and Vergnaud) [16] over the vector space[6] $\mathcal{X}$, the range of the function ϕ. We therefore consider it a meaningful result to show that no algebraic metareduction to DLOG can exist (unless DLOG is already easy).

Theorem 19. *If there is an algebraic metareduction from a programming straight-line extractor for Fiat-Shamir-Schnorr proofs to the DLOG problem then there is also a meta-metareduction breaking the DLOG problem directly with approximately the same success probability.*

The proof is in [7]. The idea is that a metareduction M gets to see two bases in the group: the generator g and the challenge h from its DLOG challenger. Since we assumed a programming extractor, M must ask its random oracle queries to its extractor interface where our meta-metareduction will answer them. Any statement output by M (to the extractor) therefore has the form $(g^a h^b)$ for some (a, b) which are available to our meta-metareduction by use of the algebraic model. Proofs of the form $(a, 0)$ are independent of the challenge h; intuitively they should not help to compute the discrete logarithm of h so we just return

[6] Paillier and Vergnaud defined the algebraic model for groups; one can interpret a $GF(p)$ vector space as an Abelian group to use their definition of the algebraic model.

the witness a. The first time M outputs a proof with a statement of the form (a, b) with $b \neq 0$, we fork M on the relevant random oracle query and use special soundness to find the discrete logarithm of h to basis g.

8 Conclusions

Bernhard et al. introduced adaptive proofs, setting up a hierarchy of (1) proofs of knowledge (2) adaptive proofs and (3) straight-line extractable proofs, with a separation between (1) and (2). While useful for proving limitations of Σ-protocols, we have showed that adaptive proofs are not a new class of proof after all: all adaptively secure proofs admit a straight-line extractor against the canonical prover.

Along the way we have generalised previous results from Schnorr's protocol to Σ-protocols. In addition, we have weakened the counter-assumption from one-more one-wayness, which is a "q-type" interactive assumption (adversary gets an unbounded number of sample queries) and is not efficiently realisable to Σ-one-wayness, which both has a constant number of steps and an efficient security game.

Our result shows that the Fiat-Shamir transformation and Σ-protocols in general may be even weaker than previously thought. Namely, the non-interactive proof scheme $\mathcal{F}_\phi$ only achieves adaptive security if a single execution of the interactive protocol Σ_ϕ against a dishonest verifier already leaks the secret witness with non-negligible probability.

In essence, this shows that using proofs derived from the Fiat-Shamir scheme for some problem ϕ in a setting where adaptive security of such proofs is necessary requires care: these should be replaced with proofs that have straightline extractors. From a practice-oriented perspective, our results show that improving the efficiency of proofs that admit straightline extraction is an important line of future research.

References

1. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. eprint 2001/002. Originally appeared as "The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme. In: Financial Cryptography. LNCS, vol. 2339, pp. 319–338. Springer, Heidelberg (2001)
2. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006). doi:10.1007/11761679_25. The title cited is from the latest version on eprint at http://eprint.iacr.org/2004/331
3. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, CCS 1993, pp. 62–73 (1993)
4. Bernhard, D.: Zero-knowledge proofs in theory and practice. Ph.D. thesis, University of Bristol (2014). www.cs.bris.ac.uk/bernhard/papers.html

5. Bernhard, D., Fischlin, M., Warinschi, B.: Adaptive proofs of knowledge in the random oracle model. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 629–649. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46447-2_28
6. Bernhard, D., Fischlin, M., Warinschi, B.: On the hardness of proving CCA-security of signed ElGamal. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016. LNCS, vol. 9614, pp. 47–69. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49384-7_3
7. Bernhard, D., Nguyen, N.K., Warinschi, B.: Adaptive Proofs have Straightline Extractors (in the Random Oracle Model). Full version on eprint 2015/712
8. Brown, D.: Irreducibility to the One-More Evaluation Problems: More May Be Less. eprint 2007/435
9. Bresson, E., Monnerat, J., Vergnaud, D.: Separation results on the "one-more" computational problems. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 71–87. Springer, Heidelberg (2008). doi:10.1007/978-3-540-79263-5_5
10. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). doi:10.1007/3-540-47721-7_12
11. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg (2005). doi:10.1007/11535218_10
12. Fischlin, M., Fleischhacker, N.: Limitations of the meta-reduction technique: the case of schnorr signatures. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 444–460. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38348-9_27. eprint 2013/140
13. Koblitz, N., Menezes, A.: Another look at non-standard discrete log and Diffie-Hellman problems. J. Math. Cryptol. **2**(4), 311–326 (2008). eprint 2007/442
14. Maurer, U.: Abstract models of computation in cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005). doi:10.1007/11586821_1
15. Nechaev, V.I.: Complexity of a determinate algorithm for the discrete logarithm. Math. Notes **55**(2), 165–172 (1994)
16. Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 1–20. Springer, Heidelberg (2005). doi:10.1007/11593447_1
17. Schnorr, C.P.: Efficient signature generation for smart cards. J. Cryptol. **4**, 161–174 (1991). Springer
18. Seurin, Y., Treger, J.: A robust and plaintext-aware variant of signed ElGamal encryption. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 68–83. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36095-4_5
19. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_18
20. Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 1–16. Springer, Heidelberg (1998). doi:10.1007/BFb0054113

More Efficient Construction of Bounded KDM Secure Encryption

Kaoru Kurosawa(✉) and Rie Habuka

Ibaraki University, Hitachi, Japan
kaoru.kurosawa.kk@vc.ibaraki.ac.jp

Abstract. Let sk_i be the secret-key of user i for $i = 1, \ldots, \ell$, and pk_j be the public-key of user $j \in \{1, \ldots, \ell\}$. A bounded Key Dependent Message (KDM) secure encryption scheme $\mathcal{E}_{\mathrm{b-KDM}}$ provides security even when one encrypts $f(sk_1, \ldots, sk_\ell)$ under pk_j for any function f which has arbitrarily fixed circuit size. An $\mathcal{E}_{\mathrm{b-KDM}}$ is known to be constructed from projection KDM seucrity. In this paper, we first show that it can be obtained from much weaker KDM security than the projection KDM security. We next present more efficient $\mathcal{E}_{\mathrm{b-KDM}}$ than before under various assumptions.

Keywords: KDM · Key dependent message · Encryption · Garbling scheme

1 Introduction

1.1 Background

Key dependent message (KDM) security has been studied by many researchers recently. A KDM-secure encryption scheme provides security even when one encrypts the secret-key sk under the corresponding public-key pk. More generally, a KDM-secure encryption scheme with respect to a set of functions $\mathcal{F}$ provides security even when one encrypts $f(sk_1, \ldots, sk_\ell)$ under pk_j for any function $f \in \mathcal{F}$, where sk_i is the secret-key of user i for $i = 1, \ldots, \ell$, and pk_j is the public-key of user $j \in \{1, \ldots, \ell\}$.

Boneh et al. [5] showed the first KDM-secure public-key encryption scheme in the standard model which is called BHHO encryption scheme. It is KDM-secure with respect to the set of all "affine in the exponent" functions under the DDH assumption.

Then a natural question is "can we construct a KDM-secure encryption scheme with respect to the set of all functions ?" Barak et al. [4] affirmatively solved this problem by showing that there exists a KDM-secure encryption scheme with respect to the set of all functions which have arbitrarily fixed circuit size. Such encryption schemes are called *bounded* KDM secure. Barak et al. constructed a bounded KDM secure encryption scheme under either the DDH assumption or the LWE assumption.

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 354–372, 2017.
DOI: 10.1007/978-3-319-61204-1_18

On the other hand, a function is called projection if each output bit depends on at most one input bit. An encryption scheme is called *projection* KDM secure if it is KDM secure with respect to the set of all projection functions. Applebaum [1] showed that a bounded KDM secure encryption scheme can be obtained from a projection KDM secure encryption scheme.

Bellare et al. [6] then showed a clean method to construct a bounded KDM secure encryption scheme from a garbling scheme Ga and a projection KDM secure encryption scheme $\mathcal{E}$. Their scheme is more efficient than the schemes of [1,4]. (See [6, Fig. 20] for comparison.)[1] However, the size of ciphertexts is still very large.

To summarize, it is known that bounded KDM security can be obtained from projection KDM security. However, the size of ciphertexts is very large.

1.2 Our Contribution

In this paper, we first show that bounded KDM security can be obtained from much weaker KDM security than the projection KDM security. Based on this, we next present a more efficient construction of bounded KDM secure encryption schemes than Bellare et al. [6] under various assumptions.

In the scheme of Bellare et al. [6], a ciphertext of $f(sk_1, \ldots, sk_\ell)$ consists of (F, Y), where F is a garbled circuit and Y is a ciphertext of a projection KDM secure encryption scheme $\mathcal{E}$. Our scheme has the same structure, but $\mathcal{E}$ is required to be more weakly KDM secure. (More precisely, our weak KDM security is related to the underlying garbling scheme Ga.)

As a result, the size of our Y is k times smaller than that of Bellare et al. [6], where k is the security parameter. Further suppose that the number of gates of $f(sk_1, \ldots, sk_\ell)$ is $O(k\ell)$. Then the total size of our ciphertexts is $O(k^3\ell)$ under the DDH assumption while it is $O(k^4\ell)$ in the scheme of Bellare et al. [6], where BHHO encryption scheme is used in both schemes.

A projection KDM secure encryption scheme can be constructed from any KDM secure encryption scheme with respect to the set of affine (in the exponent) functions, where the secret-key is viewed as a bit string.[2] We can also construct our weakly KDM secure encryption scheme from such encryption schemes.

- As stated above, BHHO encryption scheme is KDM-secure with respect to the set of “affine in the exponent functions” under the DDH assumption.
- Brakerski and Goldwasser [3] showed a BHHO-like encryption scheme under the subgroup indistinguishablity assumptions. In particular, they presented a KDM secure encryption scheme with respect to the set of affine functions under the Paillier's decisional composite residuosity (DCR) assumption.
- Applebaum et al. [2] constructed a *symmetric-key* encryption scheme which is KDM-secure with respect to the set of affine functions under the LPN assumption.

[1] In [6], Bellare et al. mainly formalized Yao's garbled circuits as garbling schemes.

[2] Therefore we cannot use the KDM secure encryption schemes of [2, Sect. 3] [7].

Hence we can construct a more efficient bounded KDM secure encryption scheme than Bellare et al. [6] under the DDH assumption, the DCR assumption and the LPN assumption respectively.

We also point out that the security proof of Bellare et al. [6] is not complete, and show how to fix it.

2 Preliminaries

PPT means probabilistic polynomial time, and PT means polynomial time. If A is a PPT algorithm, then $y \leftarrow A(x_1, \ldots, x_n; r)$ represents the act of running the algorithm A with inputs $x_1, \ldots, x_n$ and coins r to get an output y, and $y \leftarrow A(x_1, \ldots, x_n)$ represents the act of picking r at random and letting $y \leftarrow A(x_1, \ldots, x_n; r)$.

If X is a set, then $x \overset{\$}{\leftarrow} X$ represents the act of choosing x randomly from X. $|X|$ denotes the cardinality of X. If X is a string, then $|X|$ denotes the bit length of X, and $lsb(X)$ denotes the least significant bit of X. If X and Y are bit strings, $X \| Y$ denotes the concatenation.

Let k be a security parameter.

2.1 DDH Assumption

Let $\mathbb{G}$ be a group of prime order p. The DDH assumption (on $\mathbb{G}$) is that the distributions (g, g^x, g^y, g^{xy}) and (g, g^x, g^y, g^z) are computationally indistinguishable, where g is a random generator for $\mathbb{G}$ and $x, y, z \overset{\$}{\leftarrow} Z_q$.

2.2 KDM Security

Let $\mathcal{E} = (K, E, D)$ be a public key encryption scheme, $\mathcal{S} = \{S_k\}$ be the space of secret keys and $\mathcal{M} = \{M_k\}$ be the space of messages. For an integer $\ell > 0$, define

$$ALL_k^{(\ell)} = \{f \mid f : (S_k)^\ell \to M_k\}.$$

We then define the KDM$^{(\ell)}$ attack game between a challenger and an adversary $\mathcal{A}$ with respect to a function class $\mathcal{F} = \{\mathcal{F}_k\}$ such that $\mathcal{F}_k \subseteq ALL_k^{(\ell)}$ as follows.

Initialize. The challenger chooses $b \overset{\$}{\leftarrow} \{0, 1\}$ and generates key pairs $(sk_i, pk_i) \leftarrow K(1^k)$ for all $i \in \{1, \ldots, \ell\}$. The challenger then sends $(pk_1, \ldots, pk_\ell)$ to the adversary $\mathcal{A}$.

Query. The adversary $\mathcal{A}$ makes adaptive queries of the form $(j, f) \in \{1, \ldots, \ell\} \times \mathcal{F}_k$. For each query, the challenger computes $x = f(sk_1, \ldots, sk_\ell)$ and sends the following ciphertext to the adversary $\mathcal{A}$.

$$c = \begin{cases} E_{pk_j}(x) & if\ b = 0 \\ E_{pk_j}(0^{|x|}) & if\ b = 1 \end{cases}$$

Finish. The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.

Define $\mathtt{Adv}^{\mathrm{kdm}}_{\mathcal{E},\ell}(\mathcal{A}) = |2\Pr[b' = b] - 1|$. We say that $\mathcal{E}$ is KDM$^{(\ell)}$ secure with respect to $\mathcal{F}$ if $\mathtt{Adv}^{\mathrm{kdm}}_{\mathcal{E},\ell}(\mathcal{A})$ is negligible for any PPT adversary $\mathcal{A}$.

The KDM security of symmetric-key encryption schemes is defined similarly.

2.3 BHHO Encryption Scheme

Boneh et al. showed a KDM-secure public-key encryption scheme w.r.t. a class of affine in the exponent functions under the DDH assumption [5]. Let $\mathbb{G}$ be a group of prime order p and g be a generator of $\mathbb{G}$.

Key Generation. Let $t = \lceil 3 \log_2 p \rceil$. Choose $(g_1, \ldots, g_t) \overset{\$}{\leftarrow} \mathbb{G}^t, (s_1, \cdots, s_t) \overset{\$}{\leftarrow} \{0,1\}^t$. Let $h \leftarrow (g_1^{s_1} \cdots g_t^{s_t})^{-1}$ and define the public and secret keys as

$$pk = (g_1, \ldots, g_t, h) \,\text{and}\, sk = (s_1, \ldots, s_t).$$

Encryption. For a plaintext $m \in \mathbb{G}$, choose $r \overset{\$}{\leftarrow} Z_p$ and output a ciphertext

$$(g_1^r, \ldots, g_t^r,\, h^r \cdot m).$$

Decryption. For a ciphertext $(c_1, \ldots, c_t, d)$, output $m = d \cdot (c_1^{s_1} \cdots c_t^{s_t})$.

We say that f is an affine in the exponent function if $f(x_1, \ldots, x_n) = g^{a_0 + \sum_{i=1}^n a_i x_i}$, where $x_i \in \{0,1\}^n$ and $a_i \in Z_p$. Define $\mathcal{F}_{\text{affine}} = \{\mathcal{F}_k\}$ in such a way that $\mathcal{F}_k = ALL_k^{(\ell)} \cap \{f \mid f \text{ is an affine in the exponent function}\}$.

Proposition 1 [5]**.** *The above encryption scheme is KDM$^{(\ell)}$ secure w.r.t. $\mathcal{F}_{\text{affine}}$ under the DDH assumption.*

2.4 Bounded-KDM Security

Let $q(k)$ be a polynomial in k. A function f is $q(k)$-bounded if it can be expressed as a boolean circuit such that the number of gates is $q(k)$. Define $\mathcal{F}_{\text{q}-\text{gates}} = \{\mathcal{F}_k\}$, where $\mathcal{F}_k = ALL_k^{(\ell)} \cap \{f \mid f \text{ is } q(k)\text{-bounded}\}$. Then an encryption scheme $\mathcal{E}$ is q-bounded KDM$^{(\ell)}$ secure if it is KDM$^{(\ell)}$ secure w.r.t. $\mathcal{F}_{\text{q}-\text{gates}}$.

2.5 Projection KDM Security

A function f is called a *projection* if each output bit depends on at most one input bit. Define $\mathcal{F}_{\text{proj}} = \{\mathcal{F}_k\}$ in such a way that $\mathcal{F}_k = ALL_k^{(\ell)} \cap \{f \mid f \text{ is a projection}\}$. We then say that an encryption scheme $\mathcal{E}$ is projection KDM$^{(\ell)}$ secure if it is KDM$^{(\ell)}$ secure w.r.t. $\mathcal{F}_{\text{proj}}$.

3 Garbling Scheme [6]

3.1 Circuits

A boolean circuit is a 5-tuple $f = (n, m, q, A, B, G)$. Here $n \geq 2$ is the number of inputs, $m \geq 1$ is the number of outputs, and $q \geq 1$ is the number of gates. We let Inputs = $\{1, ..., n\}$, Gates = $\{n+1, ..., n+q\}$, Wires = $\{1, ..., n+q\}$ and Outputs = $\{n + q - m + 1, \ldots, n + q\}$. Then A: Gates → Wires\Outputs is a function to identify each gate's first incoming wire, and B : Gates → Wires\Outputs is a function to identify each gate's second incoming wire. We require $A(g) < B(g) < g$ for each gate $g \in$ Gates. Finally G : Gates $\times\{0,1\}^2 \rightarrow \{0,1\}$ is a function that determines the functionality of each gate.

Each gate has two inputs, one output and arbitrary functionality. The wires are numbered 1 to $n+q$. The ith bit of the input is presented along wire i. Every non-input wire is the outgoing wire of some gate. The outgoing wire of each gate serves as the name of that gate. We denote the output of f on input $x \in \{0,1\}^n$ by $y = f(x)$. See Appendix A for an example.

Let

$$\Phi_{topo}(f) = (n, m, q, A, B),\ \Phi_{size}(f) = (n, m, q)$$

We say that $\Phi_{topo}(f)$ and $\Phi_{size}(f)$ are side information functions.

3.2 Garbling Scheme

A garbling scheme is a three-tuple of algorithms $\mathsf{Ga} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev})$ such as follows.

- $(F, e) \leftarrow \mathsf{Gb}(1^k, f)$, where $f = (n, m, q, A, B, G)$ is a boolean circuit, F is its garbled circuit and e is a key of En.
- $X \leftarrow \mathsf{En}(e, x)$, where $x \in \{0,1\}^n$ is a real input and X is its garbled input.
- $y \leftarrow \mathsf{Ev}(F, X)$, where y is the output of F on input X.

The correctness condition requires that if $(F, e) \leftarrow \mathsf{Gb}(1^k, f)$, then

$$\mathsf{Ev}[F, \mathsf{En}(e, x)] = f(x)$$

for any $x \in \{0,1\}^n$.

3.3 Security of Garbling Schemes

For a garbling scheme $\mathsf{Ga} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev})$ and a side information function Φ, we consider a game between a challenger and an adversary $\mathcal{A}$ as follows.

1. $\mathcal{A}$ chooses (f_0, x_0) and (f_1, x_1) such that $f_0(x_0) = f_1(x_1)$ and $\Phi(f_0) = \Phi(f_1)$, and sends them to the challenger.
2. The challenger chooses $b \overset{\$}{\leftarrow} \{0,1\}$, and computes $(F, e) \leftarrow \mathsf{Gb}(1^k, f_b)$ and $X \leftarrow \mathsf{En}(e, x_b)$. He then sends (F, X) to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a bit b'.

Define the advantage as $\mathtt{Adv}^{garble}_{\mathsf{Ga},\Phi}(\mathcal{A}) = |2\Pr(b' = b) - 1|$. We say that Ga is Φ-secure if $\mathtt{Adv}^{garble}_{\mathsf{Ga},\Phi}(\mathcal{A})$ is negligible for every PPT adversary $\mathcal{A}$.

Bellare et al. [6] showed a Φ_{topo}-secure garbling scheme $\mathsf{Ga} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev})$ which is called Garble1. We illustrate it in Figs. 11 and 12 of Appendix B. We also show how to construct Φ_{size}-secure garbling schemes in Appendix C.

4 Bounded KDM Secure Encryption by Bellare et al.

4.1 Generic Construction

Bellare et al. [6] constructed a q-bounded KDM$^{(\ell)}$ secure encryption scheme $\mathcal{E}' = (K', E', D')$ from

- a Φ_{size}-secure garbling scheme $\mathsf{Ga} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev})$ and
- a projection KDM$^{(\ell)}$ secure encryption scheme $\mathcal{E} = (K, E, D)$.

A ciphertext of $f(sk_1, \ldots, sk_\ell)$ under pk_j is given by $(F, E_{pk_j}(X))$, where

- F is a garbled circuit of an identity circuit ID, and
- X is a garbled input of a real input $f(sk_1, \ldots, sk_\ell)\|0^{\ell|sk|-L}$, where $L = |f(sk_1, \ldots, sk_\ell)|$.

The decryption is given by

$$F(X) = ID(f(sk_1, \ldots, sk_\ell)\|0^{\ell|sk|-L}) = f(sk_1, \ldots, sk_\ell)\|0^{\ell|sk|-L}.$$

More precisely, the key generation algorithm K' is the same as K, and it outputs $(pk, sk) \leftarrow K(1^k)$. The encryption algorithm E' and the decryption algorithm D' are given in Fig. 1. There ID is a circuit such that the number of gates is $q + n$, and $ID(z) = z$ for any $z \in \{0,1\}^n$, where $n = \max(|sk| \cdot \ell, |x|)$.

```
**proc** $E'_{pk}(x)$
$n \leftarrow \max(|sk| \cdot \ell, |x|)$
$z \leftarrow x\|0^{n-|x|}$
ID$\leftarrow$ConstructID$(n)$
$(F, e) \leftarrow \mathsf{Gb}(1^k, \mathrm{ID})$
$X \leftarrow \mathsf{En}(e, z)$
$Y \leftarrow E_{pk}(X)$
**return** $c = (F, Y)$

**proc** $D'_{sk}(c)$
$(F, Y) \leftarrow c$
$X \leftarrow D_{sk}(Y)$
$y \leftarrow \mathsf{Ev}(F, X)$
**return** the first $|x|$ bits of $z$

**proc** ConstructID$(n)$
**for** $i \in \{n+1, \ldots, n+q+1\}$ **do**
  $A(i) \leftarrow 1, B(i) \leftarrow i - 1,$
  $G_i(a, b) \leftarrow \{$ **return** $a\}$
**for** $i \in \{n+q+2, \ldots, q+2n\}$ **do**
  $A(i) \leftarrow 1, B(i) \leftarrow i - n - q,$
  $G_i(a, b) \leftarrow \{$ **return** $b\}$
ID $\leftarrow (n, n, q+n, A, B, G)$
**return** ID
```

Fig. 1. Generic construction of q-bounded KDM$^{(\ell)}$ secure encryption scheme.

Proposition 2 [6]. *Suppose that*

- $\mathsf{Ga} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev})$ *is* Φ_{size}*-secure and* $\mathsf{En}(e, \cdot)$ *is a projection function.*
- $\mathcal{E} = (K, E, D)$ *is projection KDM*$^{(\ell)}$ *secure.*

Then the above encryption scheme $\mathcal{E}'$ *is q-bounded KDM*$^{(\ell)}$ *secure. (See Fig. 2.)*

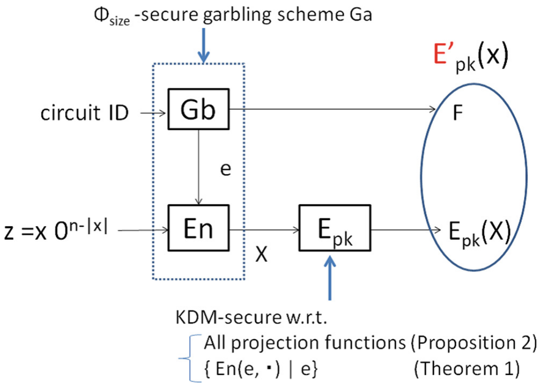

Φsize -secure garbling scheme Ga
E'pk(x)
circuit ID
Gb
F
e
z =x 0n-|x|
En
Epk
Epk(X)
X
KDM-secure w.r.t.
All projection functions (Proposition 2)
{ En(e, ・) | e} (Theorem 1)

Fig. 2. q-bounded KDM secure $\mathcal{E}'(\mathsf{Ga}, \mathcal{E})$.

4.2 Instantiation Under DDH

Bellare et al. [6, Sect. 7.2] instantiated Proposition 2 by using Garble1 [6, Sect. 5][3] and BHHO encryption scheme [5]. (See Fig. 3.)

$$\left.\begin{array}{rcl} (\Phi_{topo}\text{-secure) Garble1} & \rightarrow & (\Phi_{size}\text{-secure) } \widehat{Garble}1 \\ \text{BHHO} & \rightarrow & \text{projection KDM secure } \mathcal{E} \end{array}\right\} \rightarrow \mathcal{E}'$$

Fig. 3. q-bounded KDM$^{(\ell)}$ secure public-key encryption scheme $\mathcal{E}'$ of Bellare et al.

First construct a Φ_{size}-secure garbling scheme $\widehat{Garble}1 = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev})$ from (Φ_{topo}-secure) Garble1$=$ $(\mathsf{Gb1}, \mathsf{En1}, \mathsf{Ev1})$ by using the method of Appendix C. (See the comments of Fig. 20 of [6].) We need to show that $\mathsf{En}(e, \cdot)$ is a projection. In Garble1,

$$e = (X_1^0, X_1^1, \ldots, X_n^0, X_n^1),\ X = \mathsf{En1}(e, x) = (X_1^{x_1}, \ldots, X_n^{x_n}),$$

[3] See Appendix B for Garble1.

where $X_i^j \in \{0,1\}^k$, $x = (x_1, \ldots, x_n)$ and $x_i \in \{0,1\}$. Therefore $\mathsf{En1}(e, \cdot)$ is a projection. Hence $\mathsf{En}(e, \cdot)$ is also a projection because $\mathsf{En} = \mathsf{En1}$ the method of Appendix C.

Note that

$$|X| = |X_1^{x_1}| + \ldots + |X_n^{x_n}| = kn. \tag{1}$$

Next construct a projection KDM$^{(\ell)}$ secure encryption scheme $\mathcal{E} = (K, E, D)$ as follows. To compute $Y = E_{pk}(X)$, encrypt each bit of X by using BHHO encryption scheme. For more details, the key generation algorithm K is the same as that of BHHO encryption scheme, and for $X = (\epsilon_1, \ldots, \epsilon_{kn})$, let

$$E_{pk}(X) = (\mathrm{BHHO}_{pk}(g^{\epsilon_1}), \ldots, \mathrm{BHHO}_{pk}(g^{\epsilon_{kn}})), \tag{2}$$

where $\epsilon_i \in \{0,1\}$ and $\mathrm{BHHO}_{pk}(g^{\epsilon_i})$ denotes a BHHO ciphertext of g^{ϵ_i} for $i = 1, \ldots, kn$. This $\mathcal{E}$ is projection KDM$^{(\ell)}$ secure under the DDH assumption.

The obtained encryption scheme $\mathcal{E}'$ from $\widehat{Garble}1$ and the above $\mathcal{E}$ is q-bounded KDM$^{(\ell)}$ secure under the DDH assumption from Proposition 2.

Now suppose that we want to encrypt $f(sk_1, \ldots, sk_\ell)$. In Fig. 1, $X = \mathsf{En}(e, z)$ at line 05 has bit length kn from Eq. (1), where

$$n = \max(|sk| \cdot \ell, |f(sk_1, \ldots, sk_\ell)|) \tag{3}$$

Therefore $Y = E_{pk}(X)$ at line 06 consists of kn cihertexts of BHHO encryption scheme. The total ciphertext consists of a garbled circuit F of ID and kn ciphertexts of BHHO encryption scheme. (See the third column of Fig. 20 of [6].)

5 How to Prove Proposition 2

In this section, we first point out that the proof of Proposition 2 by Bellare et al. [6] is not complete. We next show how to fix it.

5.1 Proof by Bellare et al.

Bellare et al. [6] proved Proposition 2 as follows. Let $\mathcal{A}$ be an adversary attacking $\mathcal{E}'$. To simplify the exposition, they first consider the case $\mathcal{A}$ makes only a single query. Then they sketch how to extend it to the general case.

Single query case. Suppose that $\mathcal{A}$ makes a single query (j, f). Let C be a circuit of n input wires, n output wires, and $q + n$ gates such that

$$C(x) = f(\text{the first } |sk|\ell \text{ bits of } x) \| 0^{n - |f(sk_1, \cdots, sk_\ell)|},$$

where $n = \max(|sk| \cdot \ell, |f(sk_1, \ldots, sk_\ell)|)$. Then we have

$$C(\mathbf{K}) = f(sk_1 \| \cdots \| sk_\ell) \| 0^{n - |f(sk_1, \cdots, sk_\ell)|} \tag{4}$$

for $\mathbf{K} = sk_1 \| \cdots \| sk_\ell \| 0^{n - |sk| \cdot \ell}$.

We consider a series of Games $G_0, \ldots, G_4$ as shown in Fig. 4, where G_0 is equivalent to the KDM attack game with $b = 0$ because $C(\mathbf{K})$ is given by Eq. (4). Each game changes lines 04–05 of Fig. 1 step by step. Let $p_i = \Pr(b' = 1 \text{ in Game } G_i)$.

- $|p_0 - p_1|$ is negligible because $\mathrm{ID}(C(\mathbf{K})) = C(\mathbf{K})$, $\Phi_{size}(\mathrm{ID}) = \Phi_{size}(C)$ and the garbling scheme is Φ_{size}-secure.
- $|p_1 - p_2|$ is negligible because $\mathsf{En}(e, \cdot)$ is a projection function, and $\mathcal{E}$ is KDM$^{(\ell)}$ secure w.r.t. {all projection functions}.
- $p_2 = p_3$ because $(F, E_{pk_j}(X))$ is the same in each game.
- $|p_3 - p_4|$ is negligible because $\mathrm{ID}(C(0^n)) = C(0^n)$, $\Phi_{size}(\mathrm{ID}) = \Phi_{size}(C)$ and the garbling scheme is Φ_{size}-secure.

Their proof stops at this point. However, Game G_4 is not the KDM attack game with $b = 1$.

```
Game $G_0$
$(F, e) \leftarrow \mathsf{Gb}(1^k, \mathrm{ID})$,
$X \leftarrow \mathsf{En}(e, C(\mathbf{K}))$
return $(F, E_{pk_j}(X))$
```

```
Game $G_1$
$(F, e) \leftarrow \mathsf{Gb}(1^k, \mathrm{C})$,
$X \leftarrow \mathsf{En}(e, \mathbf{K})$
return $(F, E_{pk_j}(X))$
```

```
Game $G_2$
$(F, e) \leftarrow \mathsf{Gb}(1^k, \mathrm{C})$,
$X \leftarrow \mathsf{En}(e, \mathbf{K})$
return $(F, E_{pk_j}(0^{|X|}))$
```

```
Game $G_3$
$(F, e) \leftarrow \mathsf{Gb}(1^k, \mathrm{C})$,
$X \leftarrow \mathsf{En}(e, 0^n)$
return $(F, E_{pk_j}(0^{|X|}))$
```

```
Game $G_4$
$(F, e) \leftarrow \mathsf{Gb}(1^k, \mathrm{ID})$,
$X \leftarrow \mathsf{En}(e, C(0^n))$
return $(F, E_{pk_j}(0^{|X|}))$
```

Fig. 4. Proof of Proposition 2.

5.2 How to Fix the Proof

To complete the proof, we use the following proposition.

Proposition 3 [5]. *Suppose that the function class $\mathcal{F}$ contains all constant functions. Then the KDM security w.r.t. $\mathcal{F}$ implies IND-CPA security.*

Any constant function is a projection function. Therefore the set of all projection functions contain all constant functions.

Now we add Game G_5 and Game G_6 which are shown in Fig. 5. Then

```
Game $G_5$
$(F, e) \leftarrow \mathsf{Gb}(1^k, \mathrm{ID})$,
$X \leftarrow \mathsf{En}(e, 0^n)$
return $(F, E_{pk_j}(0^{|X|}))$
```

```
Game $G_6$
$(F, e) \leftarrow \mathsf{Gb}(1^k, \mathrm{ID})$,
$X \leftarrow \mathsf{En}(e, 0^n)$
return $(F, E_{pk_j}(X))$
```

Fig. 5. Game G_5 and Game G_6.

- $p_4 = p_5$ because $(F, E_{pk_j}(0^{|X|}))$ is the same in each game.
- $|p_5 - p_6|$ is negligible bacause $\mathcal{E}$ is IND-CPA secure from Proposition 3.

Game G_6 is the KDM attack game with $b = 1$ because the output of Game G_6 is $\mathcal{E}'_{pk_j}(0^{|f(sk_1\|\cdots\|sk_\ell)|})$. Finally $|p_0 - p_6|$ is negligible form the above discussion. This completes the proof.

6 Our Main Theorem

As shown above, bounded KDM security can be obtained from projection KDM security. However, the size of ciphertexts is still very large. In this section, we show that it can be obtained from a much weaker notion than the projection KDM security.

The encryption scheme $\mathcal{E}' = (K', E', D')$ given in Fig. 1 consists of

- a garbling scheme $\mathsf{Ga} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev})$ and
- an encryption scheme $\mathcal{E} = (K, E, D)$ such that $K' = K$.

We denote such $\mathcal{E}'$ by $\mathcal{E}'(\mathsf{Ga}, \mathcal{E})$.

For a garbling scheme $\mathsf{Ga} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev})$, first define $\widehat{\mathsf{En}}$ as

$$\widehat{\mathsf{En}}_{n,e}(x) = \mathsf{En}(e, x \| 0^{n-|x|}). \tag{5}$$

Next define the set of functions $\mathcal{F}(\mathsf{Ga}) = \{\mathcal{F}_k\}$ as follows.

$$\mathcal{F}_k = ALL_k^{(\ell)} \cap \{\widehat{\mathsf{En}}_{n,e} \mid (F, e) \leftarrow \mathsf{Gb}(1^k, C), \text{where } \Phi_{size}(C) = (n, n, n+q)\}.$$

Then our main Theorem is as follows. (See Fig. 2.)

Theorem 1. *$\mathcal{E}'(\mathsf{Ga}, \mathcal{E})$ is q-bounded KDM$^{(\ell)}$ secure if Ga is Φ_{size}-secure, and $\mathcal{E}$ is IND-CPA secure and KDM$^{(\ell)}$-secure w.r.t. $\mathcal{F}(\mathsf{Ga})$.*

The proof is the same as that of Proposition 2 which we fixed. In particular,

- $|p_1 - p_2|$ is negligible because $\mathcal{E}$ is KDM$^{(\ell)}$ secure w.r.t. $\mathcal{F}(\mathsf{Ga})$.
- $|p_5 - p_6|$ is negligible because $\mathcal{E}$ is IND-CPA secure.

Let $\widehat{\mathsf{Ga}}$ denote the Φ_{size}-secure garbling scheme which is obtained from a Φ_{topo}-secure garbling scheme Ga by using the method of Appendix C. Then we have the following corollary.

Corollary 1. *$\mathcal{E}'(\widehat{\mathsf{Ga}}, \mathcal{E})$ is q-bounded KDM$^{(\ell)}$ secure if Ga is Φ_{topo}-secure, and $\mathcal{E}$ is IND-CPA secure and KDM$^{(\ell)}$-secure w.r.t. $\mathcal{F}(\mathsf{Ga})$.*

Proof. Let $\widehat{\mathsf{Ga}} = (\mathsf{Gb}', \mathsf{En}', \mathsf{Ev}')$ and $\mathsf{Ga} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev})$. Then $\mathsf{En}' = \mathsf{En}$ in the method of Appendix C. Hence we obtain this corollary from the definition of $\mathcal{F}(\mathsf{Ga})$. □

Let's compare Theorem 1 with Proposition 2.

- In Proposition 2, $\mathcal{E}$ needs to be KDM$^{(\ell)}$ secure w.r.t. the set of "all projection functions". Therefore Bellare et al. had to encrypt each bit of $X = \mathsf{En}(e, z)$ by using BHHO encryption scheme.
- In our theorem, on the other hand, $\mathcal{E}$ needs to be KDM$^{(\ell)}$ secure w.r.t. $\mathcal{F}(\mathsf{Ga})$ only. Therefore we can construct a more efficient $\mathcal{E}$ with an appropriate garbling scheme Ga as shown in the following sections.

Another difference is that in Theorem 1, $\mathcal{E}$ needs to be IND-CPA secure. However, it is not a matter because any encryption scheme must be IND-CPA secure anyway. See Table 1.

Table 1. Underlying primitive for bounded KDM security.

Scheme	$\mathcal{E}$ must be KDM secure w.r.t	En of Ga
BHR [6]	The set of all projection functions	Projection
Proposed	$\mathcal{F}(\mathsf{Ga})$	$--$

7 Our Instantiation Under DDH

In this section, we show a more efficient bounded KDM secure encryption scheme $\mathcal{E}'(\widehat{\mathsf{Ga}}, \mathcal{E})$ than Bellare et al. [6] under the DDH assumption based on Corollary 1. The encryption scheme $\mathcal{E}$ is based on BHHO encryption scheme [5], and the garbling scheme Ga is a variant of Garble1 [6] which we call Garble1+.

Let $\mathbb{G}$ be a group of prime order p and g be a generator of $\mathbb{G}$ as required in BHHO encryption scheme.

7.1 Garble1+

We define $(\mathsf{Gb}, \mathsf{Ev})$ of Garble1+ as shown in Figs. 6 and 7. Namely

$$e = (u_1^0, u_1^1, \ldots, u_n^0, u_n^1), \tag{6}$$

$$\mathsf{En}(e, x) = (g^{u_1^{x_1}}, \ldots, g^{u_n^{x_n}}), \tag{7}$$

where $u_i^j \in Z_p$, and

$$X_i^0 \leftarrow H(g^{u_i^0}),\ X_i^1 \leftarrow H(g^{u_i^1}) \tag{8}$$

for $i = 1, \ldots, n$ in Gb, where $H : \mathbb{G} \rightarrow \{0,1\}^k$ is a hash function such that $H(v)$ is uniformly distributed over $\{0,1\}^k$ when $v \overset{\$}{\leftarrow} \mathbb{G}$.

The rest is the same as Garble1 [6] which is shown Appendix B. Then Garble1+ is Φ_{topo}-secure similarly to Garble1.

```
proc Gb$(1^k, f)$
$(n, m, 1, A, B, G) \leftarrow f$
for $i \in$ Inputs $= \{1, \ldots, n\}$ do
    $(u_i^0, u_i^1) \overset{\$}{\leftarrow} (Z_p)^2$ such that $lsb(H(g^{u_i^0})) \neq lsb(H(g^{u_i^1}))$
    $X_i^0 \leftarrow H(g^{u_i^0}), X_i^1 \leftarrow H(g^{u_i^1})$.
$e \leftarrow (u_1^0, u_1^1, \ldots, u_n^0, u_n^1)$
for $i \in$ Wires/Outputs do
    $(X_i^0, X_i^1) \overset{\$}{\leftarrow} (\{0,1\}^k)^2$ such that $lsb(X_i^0) \neq lsb(X_i^1)$.
The rest is the same as Garble1.
```

Fig. 6. Garble1+ (1).

```
proc $\mathsf{Ev}(F, X)$
$(P, \Phi_{topo}(f)) \leftarrow F$
$(n, m, q, A, B) \leftarrow \Phi_{topo}(f)$
$(g^{u_1}, \ldots, g^{u_n}) \leftarrow X$
for $i \in$ Inputs $= \{1, \ldots, n\}$ do
    $X_i \leftarrow H(g^{u_i})$.
The rest is the same as Garble1.
```

```
proc $\mathsf{En}(e, x)$
$(u_1^0, u_1^1, \ldots, u_n^0, u_n^1) \leftarrow e$
$(x_1, \ldots, x_n) \leftarrow x$
$X \leftarrow (g^{u_1^{x_1}}, \ldots, g^{u_n^{x_n}})$
return $X$
```

Fig. 7. Garble1+ (2).

7.2 KDM-Secure Encryption w.r.t. $\mathcal{F}(Garble1+)$

We next show an efficient encryption scheme $\mathcal{E} = (K, E, D)$ which is KDM$^{(\ell)}$-secure w.r.t. $\mathcal{F}(Garble1+)$.

- The key generation algorithm K is the same as that of BHHO encryption scheme. Let $(sk, pk) \leftarrow K(1^k)$.
- For a message $(m_1, \ldots, m_n) \in \mathbb{G}^n$, let

$$E_{pk}(m_1, \ldots, m_n) = (\mathrm{BHHO}_{pk}(m_1), \ldots, \mathrm{BHHO}_{pk}(m_n)). \tag{9}$$

We stress that $m_i \in \mathbb{G}^n$ in the above encryption scheme while $m_i \in \{1, g\}$ in that of Bellare et al. [6].

Theorem 2. *The above encryption scheme $\mathcal{E} = (K, E, D)$ is IND-CPA secure and KDM$^{(\ell)}$ secure w.r.t. $\mathcal{F}(Garble1+)$ under the DDH assumption.*

Proof. $\mathcal{E}$ is IND-CPA secure because BHHO encryption scheme is IND-CPA secure. We will prove that it is KDM$^{(\ell)}$ secure w.r.t. $\mathcal{F}(Garble1+)$.

In the KDM$^{(\ell)}$ attack game, the challenger generates key pairs $(sk_i, pk_i) \leftarrow K(1^k)$ for all $i \in \{1, \ldots, \ell\}$. Define $L = \ell|sk|$ and let

$$(sk_1, \ldots, sk_\ell) = (s_1, \ldots, s_L),$$

where $s_i \in \{0, 1\}$ for $i = 1, \ldots, L$.

Then for $e = (u_1^0, u_1^1, \ldots, u_n^0, u_n^1)$ of Eq. (6), we have

$$\begin{aligned}\widehat{\mathsf{En}}_{n,e}(sk_1, \ldots, sk_\ell) &= \widehat{\mathsf{En}}_{n,e}(s_1, \ldots, s_L)\\ &= (g^{u_1^{s_1}}, \ldots, g^{u_L^{s_L}}, g^{u_{L+1}^0}, \ldots, g^{u_n^0})\end{aligned}$$

from Eqs. (7) and (5). Further from Eq. (9), we have

$$\begin{aligned}E_{pk}(\widehat{\mathsf{En}}_{n,e}(sk_1, \ldots, sk_\ell)) &= E_{pk}(g^{u_1^{s_1}}, \ldots, g^{u_L^{s_L}}, g^{u_{L+1}^0}, \ldots, g^{u_n^0})\\ &= (\mathrm{BHHO}_{pk}(g^{u_1^{s_1}}), \ldots, \mathrm{BHHO}_{pk}(g^{u_L^{s_L}}),\\ &\quad\ \mathrm{BHHO}_{pk}(g^{u_{L+1}^0}), \ldots, \mathrm{BHHO}_{pk}(g^{u_n^0})).\end{aligned}$$

We note that

$$u_i^0 + (u_i^1 - u_i^0) \times s_i = \begin{cases} u_i^0 \; if \; s_i = 0 \\ u_i^1 \; if \; s_i = 1 \end{cases}$$

Therefore we have

$$u_i^{s_i} = u_i^0 + (u_i^1 - u_i^0) \times s_i.$$

This means that $u_i^{s_i}$ is an affine function of $(s_1, \ldots, s_L)$ for fixed $e = (u_1^0, u_1^1, \ldots, u_n^0, u_n^1)$. More precisely, let $a_0 = u_i^0$ and

$$a_j = \begin{cases} u_i^1 - u_i^0 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}$$

Then $u_i^{s_i}$ is written as

$$u_i^{s_i} = a_0 + \sum_{j=1}^{L} a_j s_j.$$

Let

$$\mathsf{Exp}_e(s_1, \ldots, s_L)_i = g^{u_i^{s_i}} = g^{a_0 + \sum_{j=1}^{L} a_j s_j}. \tag{10}$$

Then $\mathsf{Exp}_e(s_1, \ldots, s_L)_i$ belongs to $\mathcal{F}_{\text{affine}}$ of Sect. 2.3 for $i = 1, \ldots, L$. Further for $i = L+1, \ldots, n$, $g^{u_i^0}$ is a constant function. Hence they also belong to $\mathcal{F}_{\text{affine}}$.

Now let $\mathcal{A}$ be an adversary who attacks $\mathcal{E}$ w.r.t. $\mathcal{F}(Garble1+)$. We construct an adversary $\mathcal{B}$ who attacks BHHO encryption scheme w.r.t. $\mathcal{F}_{\text{affine}}$ as follows.

Upon receiving $(pk_1, \ldots, pk_\ell)$ from the challenger, $\mathcal{B}$ sends them to $\mathcal{A}$. Suppose that $\mathcal{A}$ queries $(j, \widehat{\mathsf{En}}_{n,e})$ such that $e = (u_1^0, u_1^1, \ldots, u_n^0, u_n^1)$. Then $\mathcal{B}$ does the following.

1. For $i = 1, \ldots, L$, $\mathcal{B}$ queries $(j, \mathsf{Exp}_e(\cdot)_i)$ to the challenger, and receives a ciphertext c_i, where $\mathsf{Exp}_e(\cdot)_i$ is defined by Eq. (10).
2. For $i = L+1, \ldots, n$, $\mathcal{B}$ queries $(j, g^{u_i^0})$ to the challenger, and receives a ciphertext c_i.
3. $\mathcal{B}$ returns $C = (c_1, \ldots, c_n)$ to $\mathcal{A}$.

It is easy to see that C is a right challenge ciphertext for $(j, \widehat{\mathsf{En}}_{n,e})$. Finally $\mathcal{B}$ outputs whatever $\mathcal{A}$ does. Then we have $\mathtt{Adv}_{\mathcal{E},\ell}^{\text{kdm}}(\mathcal{A}) = \mathtt{Adv}_{\text{BHHO},\ell}^{\text{kdm}}(\mathcal{B})$. Therefore $\mathtt{Adv}_{\mathcal{E},\ell}^{\text{kdm}}(\mathcal{A})$ is negligible under the DDH assumption because $\mathtt{Adv}_{\text{BHHO},\ell}^{\text{kdm}}(\mathcal{B})$ is negligible under the DDH assumption. □

7.3 Final Construction

Finally, our q-bounded KDM$^{(\ell)}$-secure encryption scheme $\mathcal{E}'$ is obtained by substituting Garble1+ and the encryption scheme $\mathcal{E}$ of Sect. 7.2 into Corollary 1 (Fig. 8).

Corollary 2. *Let $\mathcal{E}$ be the encryption scheme given by Sect. 7.2. Then $\mathcal{E}'(\widehat{Garble1}+, \mathcal{E})$ is q-bounded KDM$^{(\ell)}$-secure under the DDH assumption.*

Proof. From Corollary 1 and Theorem 2. □

$$\left.\begin{array}{ccc} (\Phi_{topo}\text{-secure})\ \text{Garble1+} & \to & (\Phi_{size}\text{-secure})\ \widehat{Garble}1+ \\ \text{BHHO} & \to & \mathcal{F}(Garble1+)\text{-KDM secure } \mathcal{E} \end{array}\right\} \to \mathcal{E}'$$

Fig. 8. Proposed q-bounded KDM$^{(\ell)}$ secure public-key encryption scheme $\mathcal{E}'$.

8 Comparison

Let's compare two q-bounded KDM$^{(\ell)}$-secure encryption schemes under the DDH assumption (namely, the schemes in Sects. 4.2 and 7). In both schemes, $\mathcal{E}'_{pk}(x_1, \ldots, x_n)$ consists of (F, Y), where F is a garbled circuit of ID. We compare $Y = E_{pk}(X)$ in Tables 2 and 3.

Table 2. $Y = E_{pk}(X)$ (1).

Scheme	$\mathcal{E} = (K, E, D)$	X
BHR [6]	Projection KDM secure	$(X_1^{x_1}, \ldots, X_n^{x_n})$, where $X_i^{x_i} \xleftarrow{\$} \{0,1\}^k$
Proposed	$\mathcal{F}(Garble1+)$-KDM secure	$(g^{u_1^{x_1}}, \ldots, g^{u_n^{x_n}})$, where $u_i^{x_i} \xleftarrow{\$} Z_p$

Table 3. $Y = E_{pk}(X)$ (2).

Scheme	$Y = E_{pk}(X)$
BHR [6]	BHHO encryption of each bit of $X_i^{x_i} \in \{0,1\}^k$ for $i = 1, \ldots, n$
Proposed	BHHO encryption of $g^{u_i^{x_i}}$ for $i = 1, \ldots, n$

Our $\mathcal{E}$ is KDM secure w.r.t. only $\mathcal{F}(Garble1+)$ while it must be KDM secure w.r.t. the set of all projection functions in the scheme of Bellare et al. Therefore

- $E_{pk}(X)$ encrypts $g^{u_i^{x_i}}$ for $i = 1, \ldots, n$ in our scheme
- while it must encrypt each bit of $X_i^{x_i} \in \{0,1\}^k$ for $i = 1, \ldots, n$ in the scheme of Bellare et al.

The size of each ciphertext of BHHO encryption scheme is $O(k^2)$ from Sect. 2.3, where $\log p = O(k)$. Therefore

- $|Y| = O(k^2 n)$ in our scheme and
- $|Y| = O(k^3 n)$ in the scheme of Bellare et al.

Thus $|Y|$ is reduced to $1/k$ in our scheme.

The circuit ID has gate size $r = q + n$ from lines 101–105 of Fig. 1. Therefore the universal circuit which realizes ID has the gate size $O(r \log r)$ from Eq. (12). Hence $|F| = O(kr \log r)$ in both schemes because each gate of F has size $O(k)$. See Table 4 for the total size of ciphertexts $|F| + |Y|$.

Table 4. Size of ciphertexts (k is the security parameter.)

Scheme	Garbled circuit F	$Y = E_{pk}(X)$	Total size
BHR [6]	$O(kr \log r)$	$O(k^3 n)$	$O(kr \log r) + O(k^3 n)$
Proposed	$O(kr \log r)$	$O(k^2 n)$	$O(kr \log r) + O(k^2 n)$

When $f(sk_1, \ldots, sk_\ell)$ is encrypted, $n = \max(|sk| \cdot \ell, |f(sk_1, \ldots, sk_\ell)|)$. Suppose that $|f(sk_1, \ldots, sk_\ell|) = O(|sk| \cdot \ell)$ and $q = O(n)$. Then $n = O(|sk| \cdot \ell) = O(k\ell)$ and $r = q + n = O(n) = O(k\ell)$.

In this case, $O(k^2 n) = O(k^3 \ell)$, $O(k^3 n) = O(k^4 \ell)$ and $O(kr \log r) = O(k^2 \ell \log k)$ because $n = poly(k)$. Hence the total size of ciphertexts is $O(k^3 \ell)$ in our scheme, and $O(k^4 \ell)$ in the scheme of Bellare et al. Thus it is k times smaller in our scheme (Table 5).

Table 5. Size of ciphertexts for $q = O(|sk| \cdot \ell)$ and $|f(sk_1, \ldots, sk_\ell| = O(|sk| \cdot \ell)$.

Scheme	Garbled circuit F	$Y = E_{pk}(X)$	Total size
BHR [6]	$O(k^2 \ell \log k)$	$O(k^4 \ell)$	$O(k^4 \ell)$
Proposed	$O(k^2 \ell \log k)$	$O(k^3 \ell)$	$O(k^3 \ell)$

9 Generalization

In general, we can construct

- a KDM secure encryption scheme w.r.t. $\mathcal{F}(Garble1)$ from any KDM secure encryption scheme w.r.t. the set of affine functions, and
- a KDM secure encryption scheme w.r.t. $\mathcal{F}(Garble1+)$ from any KDM secure encryption scheme w.r.t. the set of affine in the exponent functions,

where the secret-key is viewed as a bit string. By using each of them, we can construct a bounded KDM secure encryption scheme based on Corollary 1.

If the message space of our weakly KDM secure encryption scheme is large enough, then the size of our Y is k times smaller than that of Bellare et al. [6]. In this section, we show such examples.

9.1 Symmetric-Key Encryption Scheme

Proposition 2 holds for *symmetric-key* encryption schemes also as stated in [6]. Similarly, our Theorem 1 and Corollary 1 hold for symmetric-key encryption schemes too. The proofs are the same.

Applebaum et al. [2] showed a *symmetric-key* encryption scheme which is KDM$^{(\ell)}$ secure w.r.t. the set of affine functions (not in the exponent) under the LPN assumption. We call it ACPS encryption scheme. We can construct a bounded KDM$^{(\ell)}$ secure *symmetric-key* encryption scheme under the LPN assumption as follows. We use Garble1. Then

$$e = (X_1^0, X_1^1, \ldots, X_n^0, X_n^1),\ X = \mathsf{En}(e, x) = (X_1^{x_1}, \ldots, X_n^{x_n}),$$

where $X_i^j \in \{0,1\}^k$, $x = (x_1, \ldots, x_n)$ and $x_i \in \{0,1\}$. To compute $Y = E_{pk}(X)$ at line 06 of Fig. 1, encrypt each X_i^j by using ACPS encryption scheme. Namely

$$E(X) = (\text{ACPS}(X_1^{x_1}), \ldots, \text{ACPS}(X_n^{x_n})),$$

where ACPS$(X_i^{x_i})$ denotes an ACPS ciphertext of $X_i^{x_i}$. Now $X_i^{x_i}$ is written as

$$X_i^{x_i} = X_i^0 + (X_i^1 - X_i^0) \times x_i. \tag{11}$$

This means that $X_i^{x_i}$ is an affine function of $(x_1, \ldots, x_n)$. Therefore we can show that E is KDM$^{(\ell)}$ secure w.r.t. $\mathcal{F}(Garble1)$. Hence $\mathcal{E}'(\widehat{Garble}1, \mathcal{E})$ is q-bounded KDM$^{(\ell)}$ secure under the LPN assumption from Corollary 1 (Fig. 9).

On the other hand, if we use Proposition 2, we must encrypt *each bit* of X_i^j by ACPS encryption scheme.

$$\left.\begin{array}{ccc} (\Phi_{topo}\text{-secure) Garble1} & \rightarrow & (\Phi_{size}\text{-secure) } \widehat{Garble}1 \\ \text{ACPS} & \rightarrow & \mathcal{F}(Garble1)\text{-KDM secure } \mathcal{E} \end{array}\right\} \rightarrow \mathcal{E}'$$

Fig. 9. Proposed q-bounded KDM$^{(\ell)}$ secure symmetric-key encryption scheme $\mathcal{E}'$.

9.2 Subgroup Indistinguishability Assumptions

Brakerski and Goldwasser [3] showed a BHHO-like encryption scheme under the subgroup indistinguishablity assumptions. In particular, they presented a KDM secure encryption scheme with respect to the set of affine functions under the Paillier's decisional composite residuosity (DCR) assumption. In this encryption scheme, the message space is Z_N, where $N = pq$.

Now we can construct a bounded KDM$^{(\ell)}$ secure public-key encryption scheme under the DCR assumption by applying our technique to this encryption scheme and Garble1.

A Example of Boolean Circuit

In an example of Fig. 10, $n = 4, m = 1, q = 3$, Inputs $= \{1, \ldots, 4\}$, Gates $= \{5, \ldots, 7\}$, Wires $= \{1, \ldots, 7\}$, Outputs $= \{7\}$, $G_5(x, y) = x \wedge y$ and

$$A(5) = 1, A(6) = 3, A(7) = 5,$$

$$B(5) = 2, B(6) = 4, B(7) = 6.$$

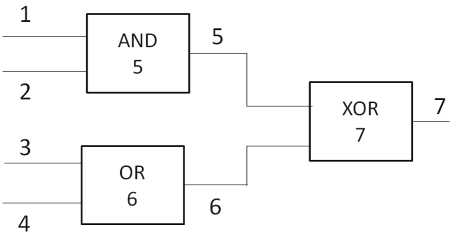


Fig. 10. A boolean circuit f.

B Garble1

Bellare et al. [6] showed a Φ_{topo}-secure garbling scheme $\mathsf{Ga} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev})$ which is called Garble1.

For a k bit string A, let $A[1:k-1]$ denote the first $k-1$ bits of A. Define $B[1:k-1]$ similarly. For a pseudo-random function F, let

$$\mathbb{E}^T_{A,B}(M) = \mathsf{F}_{A[1:k-1]}(T) \oplus \mathsf{F}_{B[1:k-1]}(T) \oplus M,$$
$$\mathbb{D}^T_{A,B}(C) = \mathsf{F}_{A[1:k-1]}(T) \oplus \mathsf{F}_{B[1:k-1]}(T) \oplus C,$$

Then Garble1 is given in Figs. 11 and 12.

```
**proc** $\mathsf{Gb}(1^k, f)$
$(n, m, 1, A, B, G) \leftarrow f$
**for** $i \in$ Wires/Outputs **do**
  Choose $X_i^0$ and $X_i^1$ such that $lsb(X_i^0) \neq lsb(X_i^1)$ randomly from $\{0,1\}^k$.
$e \leftarrow (X_1^0, X_1^1, \ldots, X_n^0, X_n^1)$
**for** $i \in$ Outputs **do**
  Choose $X_i^0$ and $X_i^1$ such that $lsb(X_i^0) = 0$ and $lsb(X_i^1) = 1$ randomly from $\{0,1\}^k$.
**for** $(g, a, b) \in$ Gates $\times \{0,1\} \times \{0,1\}$ **do**
  $i \leftarrow A(g), \alpha \leftarrow lsb(X_i^a)$,
  $j \leftarrow B(g), \beta \leftarrow lsb(X_j^b)$,
  $T \leftarrow g \,\|\, \alpha \,\|\, \beta$,
  $P[g, \alpha, \beta] \leftarrow \mathbb{E}^T_{A,B}(X_g^{G_g(a,b)})$
$F \leftarrow (P, \Phi_{topo}(f))$
**return** $(F, e)$
```

Fig. 11. Garble1 (1).

C How to Achieve Φ_{size}-Security

In a topological circuit, only (n, m, q', A, B) is specified, but not the gate function G. A universal circuit $\mathsf{UC}_{(n,m,q)}$ is a topological circuit (n, m, q', A, B) which

```
**proc** $\mathsf{Ev}(F, X)$
$(P, \Phi_{topo}(f)) \leftarrow F$
$(n, m, q, A, B) \leftarrow \Phi_{topo}(f)$
$(X_1, \ldots, X_n) \leftarrow X$
**for** $g \leftarrow n+1$ **to** $n+q$ **do**
  $i \leftarrow A(g), \alpha \leftarrow lsb(X_i),$
  $j \leftarrow B(g), \beta \leftarrow lsb(X_j),$
  $T \leftarrow g \,\|\, \alpha \,\|\, \beta,$
  $X_g \leftarrow \mathbb{D}^T_{A,B}(P[g, \alpha, \beta])$
**for** $i \in$ Outputs **do**
  $y_g \leftarrow lsb(X_g)$
**return** $(y_{n+q-m+1}, \ldots, y_{n+q})$
```

```
**proc** $\mathsf{En}(e, x)$
$(X_1^0, X_1^1, \ldots, X_n^0, X_n^1) \leftarrow e$
$(x_1, \ldots, x_n) \leftarrow x$
$X \leftarrow (X_1^{x_1}, \ldots, X_n^{x_n})$
**return** $X$
```

Fig. 12. Garble1 (2).

can realize any boolean circuit f such that $\Phi_{size}(f) = (n, m, q)$ by using an appropriate gate function G. Valiant [8] constructed a universal circuit which has asymptotically optimal gate size

$$q' = O(q \log q). \tag{12}$$

Let $\mathsf{UC}_{(n,m,q)}(G)$ denote a boolean circuit which is obtained from $\mathsf{UC}_{(n,m,q)}$ and G.

A Φ_{size}-secure garbling scheme $\mathsf{Ga} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev})$ can be obtained from a universal circuit and any Φ_{topo}-secure garbling scheme $(\mathsf{Gb1}, \mathsf{En1}, \mathsf{Ev1})$ as follows.

- $\mathsf{Gb}(1^k, f)$: Let $\Phi_{size}(f) = (n, m, q)$. Find a gate function G such that $\mathsf{UC}_{(n,m,q)}(G)$ realizes f. Then output $(F, e) \leftarrow \mathsf{Gb1}(1^k, \mathsf{UC}_{(n,m,q)}(G))$.
- $(\mathsf{En}, \mathsf{Ev}) = (\mathsf{En1}, \mathsf{Ev1})$.

References

1. Applebaum, B.: Key-dependent message security: generic amplification and completeness. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 527–546. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20465-4_29
2. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03356-8_35
3. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14623-7_1
4. Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 423–444. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_22
5. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision Diffie-Hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85174-5_7

6. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: ACM Conference on Computer and Communications Security, pp. 784–796 (2012). Cryptology ePrint Archive, Report 2012/265
7. Malkin, T., Teranishi, I., Yung, M.: Efficient circuit-size independent public key encryption with KDM security. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 507–526. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20465-4_28
8. Valiant, L.G.: Universal circuits (preliminary report). In: STOC, pp. 196–203 (1976)

Signature Schemes with Randomized Verification

Cody Freitag[1], Rishab Goyal[1](✉), Susan Hohenberger[2], Venkata Koppula[1],
Eysa Lee[1], Tatsuaki Okamoto[3], Jordan Tran[4], and Brent Waters[1]

[1] University of Texas at Austin, Austin, USA
{cody.freitag,eysa.lee}@utexas.edu,
{rgoyal,kvenkata,bwaters}@cs.utexas.edu
[2] Johns Hopkins University, Baltimore, USA
susan@cs.jhu.edu
[3] NTT Secure Platform Laboratories, Musashino, Japan
okamoto.tatsuaki@lab.ntt.co.jp
[4] Princeton University, Princeton, USA

Abstract. A signature scheme consists of a setup, signing and verification algorithms. In most existing works, the verification algorithm is assumed to be deterministic. However, there could be signature schemes where the verification algorithm is randomized. In this work, we study signature schemes with randomized verification. Our results can be summarized as follows.

First, we present a security definition for signature schemes with randomized verification. The standard EUFCMA notion of security for signature schemes with deterministic verification is very restrictive when we consider randomized verification. Therefore, we propose a new security definition called χ-EUFCMA which captures a broad class of signature schemes with randomized verification.

Next, we analyse the security of Naor's transformation from Identity Based Encryption to signature schemes. Such a transformation results in a scheme with randomized verification. We show that this transformation can be proven χ-EUFCMA secure by choosing χ appropriately.

Finally, we show how a scheme with randomized verification can be generically transformed to one with deterministic verification.

1 Introduction

Digital signatures [6] are one of the central primitives of modern cryptography, both as an application, and as a building block for other primitives. A signature scheme consists of three algorithms: a setup algorithm that outputs a secret signing key and a public verification key, a signing algorithm that computes

S. Hohenberger—Supported by the National Science Foundation (NSF) CNS-1228443 and CNS-1414023, the Office of Naval Research under contract N00014-14-1-0333, and a Microsoft Faculty Fellowship.

B. Waters—Supported by NSF CNS-1228599 and CNS-1414082, DARPA SafeWare, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 373–389, 2017.
DOI: 10.1007/978-3-319-61204-1_19

signatures on messages using the signing key, and a verification algorithm that uses the verification key to check if a signature corresponding to a message is valid. Clearly, the setup algorithm must be probabilistic. The signing algorithm can be transformed to a deterministic one by using a pseudorandom function (PRF). In such a transformation, a PRF key can be included as part of the secret key, and to sign a message, this PRF key can be used to generate the randomness for the signing algorithm. However, such a transformation cannot be applied to the verification algorithm, since verification is public.

Indeed, as noted by [8], "no generic transformation outside the random oracle model is known that makes the verification deterministic". As a result, when describing or using signatures, the verification algorithm is often implicitly or explicitly *assumed* to be deterministic. This is especially true for works which use signatures as part of a larger protocol. Moreover, in certain applications [8,9], it is important for the verification algorithm to be deterministic. This assumption regarding the determinism of the verification algorithm is often justified by the fact that most existing schemes have deterministic verification.

At the same time there are circumstances where a signature construction with a randomized verification will naturally arise. The most popular one is the transformation from identity based encryption (IBE) schemes to digital signatures suggested by Naor (sketched in [2]). This transformation, popularly referred to as Naor transformation, can be described as follows. The signing key consists of the IBE master secret key and the verification key is the IBE public key. To sign a message m, the signing algorithm generates an IBE secret key corresponding to the identity m. This signature can be verified by first choosing a random message x from the IBE message space, encrypting x for identity m and checking if the secret key can decrypt this ciphertext correctly. We would like to point out that the random coins used by verification algorithm consists of the IBE message x and the randomness required to encrypt x.[1]

The signature schemes derived from IBE schemes have some nice properties (for instance shorter signatures) as compared to the traditional signature schemes. Fortunately, for many IBE schemes, there are naturally derived signature schemes where the verification is deterministic. Boneh, Lynn and Shacham showed how to modify the Boneh-Franklin IBE scheme to get a signature scheme with deterministic verification [3]. Waters, in [10], showed an IBE scheme in the standard model, and then showed how it can be modified to get a signature scheme with deterministic verification. However, such signature scheme counterparts with deterministic verification do not exist for all IBE schemes. For example, the dual system based Waters IBE scheme [11] did not give an immediate signature with deterministic verification, and the generic Naor transformation was the only known approach. More generally, in future, there could be signature schemes with desirable properties, but where the verification algorithm is randomized.

[1] In this work we define take the "Naor transformation" to be how it was originally described in the Boneh-Franklin [2] paper. We empahsize that in this case there is a *single* message x that is encrypted and tested for verification.

Our Goals and Contributions. With this motivation, we explore a *broad* notion of signature schemes with randomized verification. We note that a few different works have considered randomized verification with varying degrees of formality (e.g., [7,8,11]). The main contributions of our work can be summarized as follows. First, we propose a formal security definition for signature schemes with randomized verification security. This security notion, called χ-Existential Unforgeability under Chosen Message Attack (χ-EUFCMA) security, captures a broad class of signature schemes with randomized verification. Next, we give a formal proof of Naor's IBE to signatures transformation. The proof of Naor transformation also illustrates the importance of our proposed security definition. Third, we show how to amplify the security of a χ-EUFCMA secure scheme. Finally, we show generic transformations from a signature scheme with randomized verification to one with deterministic verification. Each of these contributions is described in more detail below.

Defining a broad notion of security for signature schemes with randomized verification. Our first goal is to define a broad notion of security for signature schemes with randomized verification. Intuitively, a signature scheme is said to be secure if no polynomial time adversary can produce a forgery that gets verified with non-negligible probability. For deterministic verification, this can be easily captured via a security game [6] between a challenger and an adversary, where the adversary can query for polynomially many signatures, and eventually, must output a forgery σ^* on a fresh (unqueried) message m^*. The adversary wins if the verification algorithm accepts σ^* as a signature on message m^*, and we want that no polynomial time adversary wins with non-negligible probability.

Now one can achieve a standard notion of security for randomized verification if we just take the same game and make the attacker's success probability be over the coins of the challenger's verification algorithms (as well as the other coins in the game). While this is the most natural extension of the security to randomized verification, there are signature schemes that do not meet this definition, yet still offer some meaningful notion of security. Looking ahead, signature schemes that arise from the Naor transformation applied to IBE schemes with *small* message space does not satisfy this strong notion but still provide some type of security that we would like to be able to formally capture.

In order to capture a broad class of signature schemes with randomized verification, we introduce a weaker notion called χ-EUFCMA security. In this security game, the adversary receives a verification key from the challenger. It is allowed to make polynomially many signature queries, and must finally output a forgery σ^* on message m^* (not queried during the signature query phase). Informally, a signature scheme is said to be χ-EUFCMA secure if for any forgery σ^* on message m^* produced by a polynomially bounded adversary, the fraction of random coins that accept σ^*, m^* is at most χ. Ideally, we would want $\chi = 0$ (or negligible in the security parameter). However, as we will show below, this is not possible to achieve for certain Naor transformed signature schemes.

Naor Transformation for IBE schemes with small message spaces. For simplicity, let us consider the Naor transformation from IBE schemes with one bit message space (for example the Cocks IBE scheme [4]). Here, the adversary can send a malformed signature such that the adversary wins the game, but the signature-to-IBE reduction algorithm has 0 advantage. In more detail, consider an adversary that, with probability 1/2, outputs a valid IBE key σ^* for ID m^*, and with probability 1/2, outputs a malformed IBE key σ^* for m^*. This malformed key always outputs the incorrect decryption of the ciphertext. Such an adversary would break the signature scheme security, but if a reduction algorithm uses such a forgery directly to break the IBE security, then it has advantage 0. To get around this problem, we use a counting technique which is inspired by the artificial abort technique introduced by Waters [10]. The main idea behind Waters' artificial abort was to reduce the dependence between the event in which adversary wins and the one where simulation aborts. However, we require a technique which could efficiently test whether the forgery (i.e. IBE key) is malformed or not. To this end, the reduction algorithm runs the IBE decryption multiple times, each time on freshly generated ciphertext, and counts the number of correct decryptions. This helps the reduction in estimating whether the key is malformed or not. If the key is malformed, then the reduction algorithm simply aborts, otherwise it uses the forgery to break IBE security. The analysis is similar to that in [10].

The above issue with small message space IBE schemes was also noted by the work of Cui et al. [5]. They outlined a sketch of proof for large message spaces. However, even in the case of large message spaces, the issue of artificial abort needs to be addressed (although, unlike the small message case, the reduction algorithm does not suffer a loss in the winning advantage).

Amplifying the soundness from $\chi = 1 - 1/\mathsf{poly}(\cdot)$ *to* $\chi = 0$. Next, we show how to amplify the soundness from $\chi = 1 - 1/\mathsf{poly}(\lambda)$ to $\chi = 0$. Such a problem closely resembles that of transforming weak OWFs to strong OWFs. Therefore, the direct product amplification technique [12] could be used. So, if the verification algorithm is run sufficiently many times, each time with fresh randomness, and the signature is accepted only if all verifications succeed, then we could achieve $\chi = 0$-EUFCMA security.

Derandomizing the verification algorithm. Finally, we show how to transform any $\chi = 0$-EUFCMA secure signature scheme with randomized verification algorithm to a signature scheme with deterministic verification. We show two such transformations, one in the random oracle model (ROM), and another in the standard model. The transformation in the random oracle model is fairly straightforward. Here, the verification algorithm uses a hash function H which will be modeled as a random oracle in the proof. To verify a signature σ on message m, the verification algorithm computes $r = H(m, \sigma)$ and uses r as the randomness for the verification. To prove security, we first model the hash function H as a random oracle. Suppose there exists an attacker on this modified signature scheme. After the hash function is replaced with a random oracle, the attacker's forgery can be directly used as a forgery for the original $\chi = 0$-EUFCMA secure signature scheme. Here, note that it is crucial that the starting signature scheme with randomized verification is $\chi = 0$-EUFCMA secure, else the adversary's forgery may not be useful for the reduction as it will not translate to a forgery for the original signature scheme.

In our standard model transformation, the setup algorithm chooses a sufficiently large number of random strings $r_1, \ldots, r_\ell$ and includes them as part of the public verification key. The verification algorithm uses each of these strings as its randomness, and checks if m, σ verify in all cases. If so, it outputs 1, else it outputs 0. For security, suppose the adversary $\mathcal{A}$ sends a message m^* and a forgery σ^* such that the verification algorithm accepts m^*, σ^* when using $r_1, \ldots, r_\ell$ as the random strings. The reduction algorithm simply forwards m^*, σ^* as a forgery to the original signature scheme's challenger. To compute the winning probability of the reduction algorithm, we need to consider the case when the adversary $\mathcal{A}$ wins, but the reduction algorithm does not. This, in particular, can happen if the set of 'bad' random strings that result in (m^*, σ^*) being accepted is a negligible fraction of the total number of random strings. Suppose there exists a message m^* and forgery σ^* such that the number of 'bad' randomness strings that cause the verification algorithm to accept m^*, σ^* is a negligible fraction of the total number of randomness strings, and all strings $r_1, \ldots, r_\ell$ are in the bad set. We argue that since $r_1, \ldots, r_\ell$ are chosen uniformly at random, and since ℓ is a suitably large polynomial in the security parameter, this probability is negligible in the security parameter.

Conclusions. Many times in cryptography and security there is a concept that is thought to be well understood, but has never been rigorously proven or explored. Often when such an exploration takes place some new twists or facts will be discovered. In this case when we first decided to dig into the IBE to signature transformation we had an initial expectation that there would be a lossless connection from IBE to signatures. We actually were surprised that our initial attempts did not work out and that an artificial abort type mechanism was required for the small message spaces. Besides formally proving the IBE to signature transformation, we also show how to derandomize any signature scheme with randomized verification, both in the random oracle model as well as the standard model.

2 Definitions and Preliminaries

2.1 Signature Schemes with Deterministic Verification

A signature scheme $\mathcal{S} = (\mathsf{Setup}_{\mathrm{Det}}, \mathsf{Sign}_{\mathrm{Det}}, \mathsf{Verify}_{\mathrm{Det}})$ with message space $\mathcal{M}$ consists of three algorithms, as follows:

1. $\mathsf{Setup}_{\mathrm{Det}}(1^\lambda)$ is a randomized algorithm that takes security parameter λ as input and returns a pair of keys $(\mathsf{sk}, \mathsf{vk})$, where sk is the signing key and vk is the verification key.
2. $\mathsf{Sign}_{\mathrm{Det}}(\mathsf{sk}, m)$ is a possibly randomized algorithm that takes as input the signing key sk, and a message m, and returns a signature σ.
3. $\mathsf{Verify}_{\mathrm{Det}}(\mathsf{vk}, m, \sigma)$ is a determistic algorithm that takes as input the verification key vk, a message m, and a signature σ, and outputs 1 (accepts) if verification succeeds, and 0 (rejects) otherwise.

Definition 1. *A pair of polynomial time algorithms* ($\mathsf{Setup}_{\mathrm{Det}}, \mathsf{Sign}_{\mathrm{Det}}, \mathsf{Verify}_{\mathrm{Det}}$) *is a EUFCMA secure signature scheme if it satisfies the following conditions*

1. *(Correctness) For all* $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Verify}_{\mathrm{Det}}(\mathsf{vk}, m, \sigma) = 1 : \begin{array}{c} (\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{Setup}_{\mathrm{Det}}(1^\lambda) \\ \sigma \xleftarrow{\$} \mathsf{Sign}_{\mathrm{Det}}(\mathsf{sk}, m) \end{array}\right] = 1.$$

2. *(Security) For every PPT attacker* $\mathcal{A}$ *there exists a negligible function* negl($\cdot$) *such that for all* $\lambda \in \mathbb{N}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{EUFCMA}}(\lambda) \le \mathit{negl}(\lambda)$, *where advantage of* $\mathcal{A}$ *is defined as*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{EUFCMA}}(\lambda) = \Pr\left[\mathsf{Verify}_{\mathrm{Det}}(\mathsf{vk}, m^*, \sigma^*) = 1 : \begin{array}{c} (\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathsf{Setup}_{\mathrm{Det}}(1^\lambda) \\ (m^*, \sigma^*) = \mathcal{A}^{\mathsf{Sign}_{\mathrm{Det}}(\mathsf{sk}, \cdot)}(1^\lambda, \mathsf{vk}) \end{array}\right],$$

and $\mathcal{A}$ *should never have queried* m^* *to* $\mathsf{Sign}_{\mathrm{Det}}$ *oracle.*

2.2 Identity-Based Encryption

An Identity-Based Encryption (IBE) scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{IBE}}, \mathsf{KeyGen}_{\mathrm{IBE}}, \mathsf{Enc}_{\mathrm{IBE}}, \mathsf{Dec}_{\mathrm{IBE}})$ with message space $\mathcal{M}_{\mathrm{IBE}}$ and identity space $\mathcal{I}$ consists of the following polynomial time algorithms:

1. $\mathsf{Setup}_{\mathrm{IBE}}(1^\lambda)$ is a randomized polynomial time algorithm that takes security parameter λ as input and returns $(\mathsf{pp}, \mathsf{msk})$, where pp are public parameters and msk is the master secret key.
2. $\mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{pp}, \mathsf{msk}, \mathsf{ID})$ is a randomized polynomial time algorithm that takes as inputs the public parameters pp, the master secret key msk, and an identity ID and returns a secret key $\mathsf{sk}_{\mathsf{ID}}$ for ID.
3. $\mathsf{Enc}_{\mathrm{IBE}}(\mathsf{pp}, m, \mathsf{ID})$ is a randomized polynomial time algorithm that takes as inputs the public parameters pp, a message m, and an identity ID and returns a ciphertext ct.
4. $\mathsf{Dec}_{\mathrm{IBE}}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct})$ is an polynomial time algorithm that takes as inputs the public parameters pp, a secret key $\mathsf{sk}_{\mathsf{ID}}$, and a ciphertext ct and returns a message m or $\perp$.

Definition 2. *A pair of polynomial time algorithms* ($\mathsf{Setup}_{\mathrm{IBE}}, \mathsf{KeyGen}_{\mathrm{IBE}}, \mathsf{Enc}_{\mathrm{IBE}}, \mathsf{Dec}_{\mathrm{IBE}}$) *is an adaptively-secure IBE scheme if it satisfies the following conditions*

1. *(Correctness) For all* $\lambda \in \mathbb{N}$, $m \in \mathcal{M}_{\mathrm{IBE}}$, $\mathsf{ID} \in \mathcal{I}$,

$$\Pr\left[\mathsf{Dec}_{\mathrm{IBE}}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) = m : \begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}_{\mathrm{IBE}}(1^\lambda) \\ \mathsf{sk}_{\mathsf{ID}} \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{IBE}}(\mathsf{pp}, \mathsf{msk}, \mathsf{ID}) \\ \mathsf{ct} \xleftarrow{\$} \mathsf{Enc}_{\mathrm{IBE}}(\mathsf{pp}, m, \mathsf{ID}) \end{array}\right] = 1.$$

2. *(Security) For every PPT attacker* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *there exists a negligible function negl*$(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$, $\mathsf{Adv}^{\mathrm{IBE}}_{\mathcal{A}}(\lambda) \leq$ *negl*(λ)*, where advantage of* $\mathcal{A}$ *is defined as*

$$\left| \Pr\left[\mathcal{A}_1^{\mathcal{O}_{\mathsf{ID}^*}}(\mathsf{st}, \mathsf{ct}_b) \;= b : \begin{array}{c} (\mathsf{pp}, \mathsf{msk}) \xleftarrow{\$} \mathit{Setup}_{\mathrm{IBE}}(1^\lambda); \quad b \xleftarrow{\$} \{0,1\} \\ (\mathsf{ID}^*, m_0^*, m_1^*, \mathsf{st}) = \mathcal{A}_0^{\mathcal{O}}(1^\lambda, \mathsf{pp}) \\ \mathsf{ct}_b \xleftarrow{\$} \mathit{Enc}_{\mathrm{IBE}}(\mathsf{pp}, m_b^*, \mathsf{ID}^*) \end{array} \right] - \frac{1}{2} \right|,$$

Here oracle $\mathcal{O}$ *behaves same as* $\mathit{KeyGen}_{\mathrm{IBE}}(\mathsf{pp}, \mathsf{msk}, \cdot)$*,* $\mathcal{A}$ *should not have queried* $\mathcal{O}$ *for* ID^* *in the pre-challenge phase, and* $\mathcal{O}_{\mathsf{ID}^*}$ *behaves same as* $\mathcal{O}$*, except on input* ID^* *it outputs* $\perp$*.*

3 Signatures Schemes with Randomized Verification

In traditional signature schemes (as defined in Sect. 2.1), the verification algorithm is assumed to be deterministic. However, in certain scenarios, it might be useful to consider verification algorithms that are randomized. In this section, we propose a EUFCMA equivalent definition that captures a broad class of signature schemes with randomized verification. Let χ be a fixed parameter. Intuitively, a signature scheme with randomized verification is said to be χ secure if any forgery produced by a PPT adversary is accepted by the verification algorithm with probability at most χ. For example, if $\chi = 1/2$, then in the security game, the adversary first gets the verification key, then queries for q signatures, and finally sends a forgery σ^* on message m^*. The scheme is χ secure if (σ^*, m^*) is accepted by the verification algorithm with probability at most $1/2$.

More formally, a randomized verification signature scheme $\mathcal{S}_{\mathrm{RV}} = (\mathsf{Setup}_{\mathrm{RV}}, \mathsf{Sign}_{\mathrm{RV}}, \mathsf{Verify}_{\mathrm{RV}})$ with message space $\mathcal{M}(\lambda)$ and verification coin space $\mathcal{R}(\lambda)$ consists of three algorithms, as follows:

1. $\mathsf{Setup}_{\mathrm{RV}}(1^\lambda)$ is a randomized polynomial time algorithm that takes security parameter λ as input and returns a pair of keys $(\mathsf{sk}, \mathsf{vk})$, where sk is the signing key and vk is the verification key.
2. $\mathsf{Sign}_{\mathrm{RV}}(\mathsf{sk}, m)$ is a possibly randomized polynomial time algorithm that takes as input the signing key sk, and a message m, and returns a signature σ.
3. $\mathsf{Verify}_{\mathrm{RV}}(\mathsf{vk}, m, \sigma; r)$ is a randomized polynomial time algorithm that takes as input the verification key vk, a message m, and a signature σ. It also uses random coins $r \in \mathcal{R}$, and outputs 1 (accepts) if verification succeeds, and 0 (rejects) otherwise.

Successful Verification Probability. We define a function over the tuple $(\lambda, \mathsf{vk}, m, \sigma)$ for a randomized verification signature scheme which captures the fraction of message-signature pairs (m, σ) which get verified under verification key vk as follows.

$$\mathrm{VerifyProb}(\lambda, \mathsf{vk}, m, \sigma) = \frac{|\{r \in \mathcal{R} : \mathsf{Verify}_{\mathrm{RV}}(\mathsf{vk}, m, \sigma; r) = 1\}|}{|\mathcal{R}|}.$$

In our security proofs, we will use the notation $\mathcal{R}_{m,\sigma}$ to denote the set $\{r \in \mathcal{R} : \mathsf{Verify}_{\mathrm{RV}}(\mathsf{vk}, m, \sigma; r) = 1\}$.

Definition 3. *A pair of polynomial time algorithms* $(\mathit{Setup}_{\mathrm{RV}}, \mathit{Sign}_{\mathrm{RV}}, \mathit{Verify}_{\mathrm{RV}})$ *is a* $\chi(\lambda)$*-EUFCMA secure signature scheme with randomized verification if it satisfies the following conditions*

1. *(Correctness) For all* $\lambda \in \mathbb{N}$*,* $m \in \mathcal{M}$*,* $r \in \mathcal{R}$*,*

$$\Pr\left[\mathit{Verify}_{\mathrm{RV}}(\mathsf{vk}, m, \sigma; r) = 1 : \begin{array}{c}(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathit{Setup}_{\mathrm{RV}}(1^\lambda)\\ \sigma \xleftarrow{\$} \mathit{Sign}_{\mathrm{RV}}(\mathsf{sk}, m)\end{array}\right] = 1.$$

2. *(Security) For every PPT attacker* $\mathcal{A}$ *there exists negligible functions* $\mathit{negl}_1(\cdot)$ *and* $\mathit{negl}_2(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*,* $\mathsf{Adv}^{\mathrm{RV}}_{\mathcal{A}}(\lambda) \leq \mathit{negl}_1(\lambda)$*, where advantage of* $\mathcal{A}$ *is defined as*

$$\Pr\left[\mathrm{VerifyProb}(\lambda, \mathsf{vk}, m^*, \sigma^*) > \chi(\lambda) + \mathit{negl}_2(\lambda) : \begin{array}{c}(\mathsf{sk}, \mathsf{vk}) \xleftarrow{\$} \mathit{Setup}_{\mathrm{RV}}(1^\lambda)\\ (m^*, \sigma^*) = \mathcal{A}^{\mathit{Sign}_{\mathrm{RV}}(\mathsf{sk},\cdot)}(1^\lambda, \mathsf{vk})\end{array}\right],$$

and $\mathcal{A}$ *should never have queried* m^* *to* $\mathit{Sign}_{\mathrm{RV}}$ *oracle.*

4 Signature Scheme with Randomized Verification from IBE: Naor's Transformation

In this section, we construct signature schemes with randomized verification from IBE schemes using Naor's transformation. Naor suggested a generic method to construct signature schemes from IBE schemes, and a sketch of this transformation was given in [1]. At a high level, this transformation can be described as follows. The message space of the signature scheme is the identity space of the IBE scheme. To generate the signing key/verification key, the setup algorithm runs the IBE setup algorithm to compute an IBE public key/master secret key pair. The public key is set to be the verification key, while the master secret key is the signing key. To sign a message m, the signing algorithm generates a secret key corresponding to identity m. The verification algorithm chooses a uniformly random message x, encrypts it using the verification key for identity m. It then decrypts the encryption using the signature, and checks if the decryption is correct.

Intuitively, it appears that this signature scheme should be secure if the underlying IBE scheme is secure. However, there are certain subtleties, especially when the message space of the IBE scheme is small (see Sect. 1 for more details).

4.1 Construction

Let $\mathcal{E} = (\mathsf{Setup}_{\mathrm{IBE}}, \mathsf{KeyGen}_{\mathrm{IBE}}, \mathsf{Enc}_{\mathrm{IBE}}, \mathsf{Dec}_{\mathrm{IBE}})$ be an IBE scheme with message space $\mathcal{M}_{\mathrm{IBE}}$ and ID space $\mathcal{I}$. We define a randomized verification signature scheme $\mathcal{S}_{\mathrm{RV}} = (\mathsf{Setup}_{\mathrm{RV}}, \mathsf{Sign}_{\mathrm{RV}}, \mathsf{Verify}_{\mathrm{RV}})$ with message space $\mathcal{M} = \mathcal{I}$ as follows:

1. $\mathsf{Setup}_{\rm RV}(1^\lambda)$: It computes $(\mathsf{pp}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}_{\rm IBE}(1^\lambda)$ and returns $(\mathsf{sk}_{\rm RV}, \mathsf{vk}_{\rm RV})$ where $\mathsf{sk}_{\rm RV} = \mathsf{msk}$ and $\mathsf{vk}_{\rm RV} = \mathsf{pp}$.
2. $\mathsf{Sign}_{\rm RV}(\mathsf{sk}_{\rm RV}, m)$: Let $\mathsf{vk}_{\rm RV} = \mathsf{pp}$, $\mathsf{sk}_{\rm RV} = \mathsf{msk}$, and $m = \mathsf{ID}$. It computes $\mathsf{sk}_{\mathsf{ID}} \xleftarrow{\$} \mathsf{KeyGen}_{\rm IBE}(\mathsf{pp}, \mathsf{msk}, \mathsf{ID})$ and returns $\sigma = \mathsf{sk}_{\mathsf{ID}}$.
3. $\mathsf{Verify}_{\rm RV}(\mathsf{vk}_{\rm RV}, m, \sigma; r)$: Let $\mathsf{vk}_{\rm RV} = \mathsf{pp}$, $m = \mathsf{ID}$, $\sigma = \mathsf{sk}_{\mathsf{ID}}$, and $r = d||r'$. It computes $\mathsf{ct} = \mathsf{Enc}_{\rm IBE}(\mathsf{pp}, d, \mathsf{ID})$ passing r' as the random coins to $\mathsf{Enc}_{\rm IBE}$. It outputs 1 if $\mathsf{Dec}_{\rm IBE}(\mathsf{pp}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) = d$ and 0 otherwise.

4.2 Correctness and Security

Theorem 1. *If $\mathcal{E} = (\mathit{Setup}_{\rm IBE}, \mathit{KeyGen}_{\rm IBE}, \mathit{Enc}_{\rm IBE}, \mathit{Dec}_{\rm IBE})$ is an adaptively-secure IBE scheme (Definition 2) with message space $\mathcal{M}_{\rm IBE}$ such that $|\mathcal{M}_{\rm IBE}| = \mathrm{M}$, then above construction forms a $1/\mathrm{M}$-secure randomized verification signature scheme as per Definition 3.*

Proof. It is straightforward to verify the correctness as it follows directly from the correctness of IBE scheme. To prove 1/M-EUFCMA security, consider a PPT adversary $\mathcal{A}$ that wins the 1/M-EUFCMA security game described in Definition 3 with non-negligible advantage ϵ_1. This could be equivalently stated as that with probability ϵ_2, $\mathcal{A}$ outputs (m^*, σ^*) such that $\mathrm{VerifyProb}(\lambda, \mathsf{vk}, m^*, \sigma^*) > 1/\mathrm{M}+\epsilon_2$ where ϵ_1 and ϵ_2 are non-negligible functions in λ. We construct a reduction algorithm $\mathcal{B}$ that acts as the challenger in the randomized verification signature security game and is the attacker in the adaptive IBE security game. We define algorithm $\mathcal{B}$ as follows:

1. The IBE challenger computes $(\mathsf{msk}, \mathsf{pp}) \xleftarrow{\$} \mathsf{Setup}_{\rm IBE}(1^\lambda)$ and sends pp to the IBE attacker $\mathcal{B}$.
2. $\mathcal{B}$ sets $\mathsf{vk} = \mathsf{pp}$ and sends vk as the verification key to adversary $\mathcal{A}$.
3. Attacker $\mathcal{A}$ submits signature queries on a polynomial number of messages m_i. On each message, $\mathcal{B}$ sets $\mathsf{ID}_i = m_i$ and sends a secret key query for identity ID_i to the IBE challenger. $\mathcal{B}$ receives $\mathsf{sk}_{\mathsf{ID}_i}$ and sends $\sigma_i = \mathsf{sk}_{\mathsf{ID}_i}$ to attacker $\mathcal{A}$ as the signature for m_i.
4. Attacker $\mathcal{A}$ submits a forgery (m^*, σ^*) to $\mathcal{B}$ where $m^* \neq m_i$ for any queried m_i.
5. $\mathcal{B}$ sets $\mathsf{ID}^* = m^*$ and chooses $d_0^*, d_1^* \xleftarrow{\$} \mathcal{M}_{\rm IBE}$ uniformly at random such that $d_0^* \neq d_1^*$. $\mathcal{B}$ sends ID^* as the challenge identity and d_0^*, d_1^* as the challenge messages to the IBE challenger. The IBE challenger chooses $b \xleftarrow{\$} \{0,1\}$ and sends the challenge ciphertext $\mathsf{ct}^* \xleftarrow{\$} \mathsf{Enc}_{\rm IBE}(\mathsf{pp}, d_b^*, \mathsf{ID}^*)$ to $\mathcal{B}$.
6. $\mathcal{B}$ runs $\mathsf{Verify}_{\rm RV}(\mathsf{vk}, m^*, \sigma^*; r)$ q times, where $q = O(\lambda \cdot \epsilon_2^{-2})$ and $r \xleftarrow{\$} \mathcal{R}$ is freshly chosen for each run. Note that q is a polynomial when ϵ_2 is non-negligible. Let count be the number of times $\mathsf{Verify}_{\rm RV}$ outputs 1. If $\mathsf{count} < q\,(1/\mathrm{M} + \epsilon_2/2)$, $\mathcal{B}$ aborts and chooses $b' \xleftarrow{\$} \{0,1\}$ at random as its guess. If $\mathsf{count} \geq q\,(1/\mathrm{M} + \epsilon_2/2)$, $\mathcal{B}$ computes $d' = \mathsf{Dec}_{\rm IBE}(\mathsf{pp}, \sigma^*, \mathsf{ct}^*)$. If $d' = d_0^*$, then $\mathcal{B}$ sets $b' = 0$. If $d' = d_1^*$, $\mathcal{B}$ sets $b' = 1$. Otherwise, d' is a non-challenge

message or $\perp$, so $\mathcal{B}$ guesses $b' \xleftarrow{\$} \{0,1\}$ uniformly at random and sends b' to IBE challenger as its guess.

We claim that $\mathcal{B}$ wins the IBE security game with non-negligible probability. To prove our claim, we define the following events:

- Abort: $\mathsf{count} < q\left(\frac{1}{\mathrm{M}} + \frac{\epsilon_2}{2}\right)$.
- Bad: $\mathrm{VerifyProb}(\lambda, \mathsf{vk}, m^*, \sigma^*) < \frac{1}{\mathrm{M}}$
- Mid: $\frac{1}{\mathrm{M}} \leq \mathrm{VerifyProb}(\lambda, \mathsf{vk}, m^*, \sigma^*) < \frac{1}{\mathrm{M}} + \epsilon_2$
- Good: $\mathrm{VerifyProb}(\lambda, \mathsf{vk}, m^*, \sigma^*) \geq \frac{1}{\mathrm{M}} + \epsilon_2$

We say that $\mathcal{B}$ aborts when event Abort occurs and it chooses a random bit $b' \xleftarrow{\$} \{0,1\}$ as its guess. Events $\mathsf{Bad}, \mathsf{Mid}$, and Good quantify the successful verification probability of the forgery (m^*, σ^*) that $\mathcal{A}$ produces. Event Bad signifies that (m^*, σ^*) verifies with probability no greater than $\chi = 1/\mathrm{M}$. Event Good is the case when the forgery produced by $\mathcal{A}$ verifies with non-negligible advantage over $\chi = 1/\mathrm{M}$. Then, event Mid represents the case in between Bad and Good such that $\mathsf{Bad}, \mathsf{Mid}$, and Good are mutually exclusive events.

The intuition behind running $\mathsf{Verify}_{\mathrm{RV}}$ q times independently is to make sure that the reduction does not use the forgery whenever $\mathcal{A}$ provides a forgery that does worse than random guessing (of the encrypted message). This becomes necessary when we try to visualize the IBE message space to be very small. For instance, consider the scenario where $\mathcal{A}$ can win with advantage 1 in the randomized verification signature security game, but instead of always outputting a correct forgery it creates a bad forgery almost half the time. Therefore, if $\mathcal{B}$ directly uses the forgery to decrypt the challenge ciphertext which contains only one bit, then half the time it would be using a correct forgery and decrypting the challenge correctly, while at other times it would be decrypting incorrectly, and thus $\mathcal{B}$'s advantage in a direct reduction would be zero despite $\mathcal{A}$'s non-negligible advantage. Therefore, we want $\mathcal{B}$ to accept the forgery and not abort only if $\mathcal{A}$ produces a forgery such that verification succeeds with non-negligible probability over $1/\mathrm{M}$. We provide a sketch of how these events relate to the probability that $\mathcal{B}$ wins as follows.

$$\begin{aligned}
\Pr[\,\mathcal{B} \text{ wins}\,] &= \Pr[\,b = b'\,] \\
&= \Pr[\,b = b' \mid \mathsf{Abort}\,] \cdot \Pr[\,\mathsf{Abort}\,] + \Pr[\,b = b' \mid \overline{\mathsf{Abort}}\,] \cdot \Pr[\,\overline{\mathsf{Abort}}\,] \\
&\geq 1/2 \cdot \Pr[\,\mathsf{Abort}\,] + \Pr[\,b = b' \mid \overline{\mathsf{Abort}} \wedge \mathsf{Bad}\,] \cdot \Pr[\,\overline{\mathsf{Abort}} \wedge \mathsf{Bad}\,] \\
&\quad + \Pr[\,b = b' \mid \overline{\mathsf{Abort}} \wedge \mathsf{Good}\,] \cdot \Pr[\,\overline{\mathsf{Abort}} \wedge \mathsf{Good}\,] \\
&\quad + \Pr[\,b = b' \mid \overline{\mathsf{Abort}} \wedge \mathsf{Mid}\,] \cdot \Pr[\,\overline{\mathsf{Abort}} \wedge \mathsf{Mid}\,]
\end{aligned}$$

We know that $\Pr[\mathsf{Good}] = \epsilon_1$. Using Chernoff bound, we can show that $\Pr[\ \mathsf{Abort} \mid \mathsf{Good}\ ] \leq \text{negl}(\lambda)$, where $\text{negl}(\cdot)$ is a negligible function. Therefore, $\Pr[\ \overline{\mathsf{Abort}}\ \wedge\ \mathsf{Good}\ ] \geq \epsilon_1 - \text{negl}(\lambda)$. Similarly, we can show that $\Pr[\ \overline{\mathsf{Abort}}\ \wedge\ \mathsf{Bad}\ ] \leq \text{negl}(\lambda)$.

As described before, we want $\mathcal{B}$ to have non-negligible advantage in the IBE security game when $\mathcal{A}$ submits a forgery with high verification probability i.e. in event Good. Also we show that if the forgery submitted by $\mathcal{A}$ is verifies with at least 1/M probability i.e. event Mid, then $\mathcal{B}$ still wins the IBE security game with probability at least 1/2. We consider the following three mutually exclusive events describing a particular run of $\mathsf{Dec}_{\text{IBE}}$ on challenge ciphertext ct^* given that $\mathcal{B}$ does not abort:

1. Win: $\mathsf{Dec}_{\text{IBE}}$ outputs the challenge message d_b^*.
2. Lose: $\mathsf{Dec}_{\text{IBE}}$ outputs the incorrect challenge message d_{1-b}^*.
3. Split: $\mathsf{Dec}_{\text{IBE}}$ outputs a non-challenge message or $\perp$

Note that $\mathcal{B}$ wins with probability 1 in event Win, probability 0 in event Lose, and probability 1/2 in event Split. We use these different cases to analyze the probability the $\mathcal{B}$ wins in the event $\overline{\mathsf{Abort}}$.

Claim 1. $\Pr[\ b = b' \mid \overline{\mathsf{Abort}} \wedge \mathsf{Good}\ ] \geq \dfrac{\epsilon_2}{4} + \dfrac{1}{2}$

Proof. Let $p = \Pr[\ \mathsf{Win} \mid \overline{\mathsf{Abort}} \wedge \mathsf{Good}\ ] \geq 1/\text{M} + \epsilon_2/2$ since in event Good, $\text{VerifyProb}(\lambda, \mathsf{vk}, m^*, \sigma^*) \geq 1/\text{M} + \epsilon_2/2$. Then, $\Pr[\ \mathsf{Split} \mid \overline{\mathsf{Abort}} \wedge \mathsf{Good}\ ] \geq (1-p)\cdot(1 - 1/(\text{M}-1))$ since the challenge messages were chosen uniformly at random from $\mathcal{M}_{\text{IBE}}$.

$$\begin{aligned}
&\Pr[\ b = b' \mid \overline{\mathsf{Abort}} \wedge \mathsf{Good}\ ] \\
&= \Pr[\ b = b' \mid \overline{\mathsf{Abort}} \wedge \mathsf{Good} \wedge \mathsf{Win}\ ] \cdot \Pr[\ \mathsf{Win} \mid \overline{\mathsf{Abort}} \wedge \mathsf{Good}\ ] \\
&\quad + \Pr[\ b = b' \mid \overline{\mathsf{Abort}} \wedge \mathsf{Good} \wedge \mathsf{Lose}\ ] \cdot \Pr[\ \mathsf{Lose} \mid \overline{\mathsf{Abort}} \wedge \mathsf{Good}\ ] \\
&\quad + \Pr[\ b = b' \mid \overline{\mathsf{Abort}} \wedge \mathsf{Good} \wedge \mathsf{Split}\ ] \cdot \Pr[\ \mathsf{Split} \mid \overline{\mathsf{Abort}} \wedge \mathsf{Good}\ ] \\
&\geq 1 \cdot p + 0 + \frac{1}{2} \cdot \left(1 - \frac{1}{\text{M}-1}\right) \cdot (1-p) \\
&\geq \frac{\epsilon_2}{4} + \frac{1}{2}
\end{aligned}$$

Claim 2. $\Pr[\ b = b' \mid \overline{\mathsf{Abort}} \wedge \mathsf{Mid}\ ] \geq \dfrac{1}{2}$

Proof. The proof of the claim is the same as Claim 1 where $p = \Pr[\ \mathsf{Win} \mid \overline{\mathsf{Abort}} \wedge \mathsf{Mid}\ ] \geq \frac{1}{\text{M}}$. Therefore, we get that $\Pr[\ b = b' \mid \overline{\mathsf{Abort}} \wedge \mathsf{Mid}\ ] \geq 1/2$.

Now we can complete our analysis of the winning probability of the reduction algorithm $\mathcal{B}$ in the IBE security game.

$$
\begin{aligned}
&\Pr[\, \mathcal{B} \text{ wins} \,] \\
&\geq \frac{1}{2} \cdot \Pr[\, \mathsf{Abort} \,] + \Pr[\, b = b' \mid \overline{\mathsf{Abort}} \wedge \mathsf{Good} \,] \cdot \Pr[\, \overline{\mathsf{Abort}} \wedge \mathsf{Good} \,] \\
&\quad + \Pr[\, b = b' \mid \overline{\mathsf{Abort}} \wedge \mathsf{Mid} \,] \cdot (\Pr[\, \overline{\mathsf{Abort}} \,] - \Pr[\, \overline{\mathsf{Abort}} \wedge \mathsf{Bad} \,] - \Pr[\, \overline{\mathsf{Abort}} \wedge \mathsf{Good} \,]) \\
&\geq \frac{1}{2} \cdot \Pr[\, \mathsf{Abort} \,] + \left(\frac{\epsilon_2}{4} + \frac{1}{2}\right) \cdot \Pr[\, \overline{\mathsf{Abort}} \wedge \mathsf{Good} \,] \\
&\quad + \frac{1}{2} \cdot (\Pr[\, \overline{\mathsf{Abort}} \,] - \Pr[\, \overline{\mathsf{Abort}} \wedge \mathsf{Bad} \,] - \Pr[\, \overline{\mathsf{Abort}} \wedge \mathsf{Good} \,]) \\
&= \frac{1}{2} + \frac{\epsilon_2}{4} \cdot \Pr[\, \overline{\mathsf{Abort}} \wedge \mathsf{Good} \,] - \frac{1}{2} \cdot \Pr[\, \overline{\mathsf{Abort}} \wedge \mathsf{Bad} \,] \\
&= \frac{1}{2} + \frac{\epsilon_1 \epsilon_2}{4} - \text{negl}(\lambda)
\end{aligned}
$$

Thus, $\mathsf{Adv}_{\mathcal{B}}^{\text{IBE}} = \left|\Pr[\, \mathcal{B} \text{ wins} \,] - \frac{1}{2}\right| = \left|\frac{1}{2} + \frac{\epsilon_1\epsilon_2}{4} - \text{negl}(\lambda) - \frac{1}{2}\right| = \frac{\epsilon_1\epsilon_2}{4} - \text{negl}(\lambda)$, which is non-negligible if ϵ_1 and ϵ_2 are non-negligible.

We conclude that if $\mathcal{E}$ is an adaptively-secure IBE scheme, then our construction is a $\frac{1}{\text{M}}$-secure randomized verification signature scheme.

5 Amplifying Soundness

Ideally we would want from a randomized verification signature scheme that $\chi = 0$, which could be equivalently stated as that the forgery does not get verified with non-negligible probability. However not all constructions could be provably $\chi = 0$ secure, therefore we show a generic transformation from $\chi = 1 - 1/p$ to $\chi = 0$, where p is a polynomial in the security parameter λ.

5.1 Construction

Let $\mathcal{S}' = (\mathsf{Setup}', \mathsf{Sign}', \mathsf{Verify}')$ be a $\chi = 1 - 1/p$ secure randomized verification signature scheme with message space $\mathcal{M}'$ and verification coin space $\mathcal{R}'$. We construct a $\chi = 0$ secure randomized verification signature scheme as follows:

1. $\mathsf{Setup}_{\text{RV}}(1^\lambda)$: It computes $(\mathsf{sk}', \mathsf{vk}') \xleftarrow{\$} \mathsf{Setup}'(1^\lambda)$ and returns $(\mathsf{sk}, \mathsf{vk})$ where $\mathsf{sk} = \mathsf{sk}'$ and $\mathsf{vk} = \mathsf{vk}'$.
2. $\mathsf{Sign}_{\text{RV}}(\mathsf{sk}, m)$: Let $\mathsf{sk} = \mathsf{sk}'$. It computes $\sigma \xleftarrow{\$} \mathsf{Sign}'(\mathsf{sk}', m)$ and outputs σ as the signature.
3. $\mathsf{Verify}_{\text{RV}}(\mathsf{vk}, m, \sigma; r)$: Let $\mathsf{vk} = \mathsf{vk}'$, $n = \Omega(p\lambda)$, and $r = (r_1, \ldots, r_n) \in (\mathcal{R}')^n$. For all $i \in [n]$, it runs $\mathsf{Verify}'(\mathsf{vk}', m, \sigma; r_i)$. It outputs 1 if all n verifications output 1, and 0 otherwise.

5.2 Correctness and Security

Theorem 2. *If* $\mathcal{S}' = (\mathsf{Setup}', \mathsf{Sign}', \mathsf{Verify}')$ *be a* $\chi = (1 - 1/p)$*-EUFCMA secure signature scheme (Definition 3) with message space* $\mathcal{M}'$ *and verification coin* *space $\mathcal{R}'$, then above construction forms a $\chi = 0$-EUFCMA secure signature scheme as per Definition 3 with message space $\mathcal{M}'$ and verification coin space $(\mathcal{R}')^n$, where $n = \Omega(p\lambda)$ and p is a polynomial in λ.*

Proof. It is straightforward to verify the correctness as it follows directly from the correctness of underlying randomized verification signature scheme. The proof of soundness amplification is the standard direct product amplification [12]. For completeness, we sketch the argument below.

Let $\mathcal{R}^* = \left\{r \in \mathcal{R}' : \mathsf{Verify}'(\mathsf{vk}', m^*, \sigma^*; r) = 1\right\}$ denote the set of random coins on which verifier accepts the forgery (m^*, σ^*). Since, $\mathcal{S}'$ is $\chi = 1 - 1/p$ secure, we know that $\text{VerifyProb}(\lambda, \mathsf{vk}', m^*, \sigma^*) \leq 1 - 1/p + \mu'$ with all but negligible probability, where μ' is a negligible function in λ. Therefore, we can write that with all but negligible probability $|\mathcal{R}^*| \leq (1 - 1/p + \mu')\,|\mathcal{R}'|$. After amplification, $\mathsf{Verify}_{\text{RV}}$ runs Verify' on n uniformly random coins from $\mathcal{R}'$, thus set of random coins in $(\mathcal{R}')^n$ on which $\mathsf{Verify}_{\text{RV}}$ accepts the forgery (m^*, σ^*) is $(\mathcal{R}^*)^n$. So, substituting $n = \Omega(p\lambda)$, we can write that with all but negligible probability,

$$\text{VerifyProb}(\lambda, \mathsf{vk}, m^*, \sigma^*) = \frac{|\mathcal{R}^*|^n}{|\mathcal{R}'|^n} = \left(\frac{|\mathcal{R}^*|}{|\mathcal{R}'|}\right)^n \leq \left(1 - \frac{1}{p} + \mu'\right)^n \leq e^{-\frac{n}{p} + \text{negl}} \leq e^{-\Omega(\lambda)}.$$

Therefore, we can conclude that if $\mathcal{S}'$ is a $\chi = (1 - 1/p)$-EUFCMA secure randomized verification signature scheme, then $\mathcal{S}$ is a $\chi = 0$-EUFCMA secure randomized verification signature scheme, where p is a polynomial in λ.

6 Derandomizing Verification in the Random Oracle Model

In this section, we show how to generically transform a signature scheme with randomized verification to a signature scheme with deterministic verification in the Random Oracle Model (ROM). Consider any $\chi = 0$-secure signature scheme with randomized verification.[2] Let ℓ denote the number of random bits used by the verification algorithm. To make the verification algorithm deterministic, we will use a hash function H that maps (message, signature) pairs to ℓ bit strings. The deterministic verification algorithm takes as input a signature σ, message m, computes $H(\sigma, m)$, and uses this as the randomness. For our security proof, the hash function H is modeled as a random oracle.

Let $\mathcal{S}_{\text{RV}} = (\mathsf{Setup}_{\text{RV}}, \mathsf{Sign}_{\text{RV}}, \mathsf{Verify}_{\text{RV}})$ be a $\chi = 0$ secure randomized verification signature scheme with message space $\mathcal{M}(\cdot)$, signature space $\Sigma(\cdot)$ and verification coin space $\mathcal{R}(\cdot)$. We construct a deterministic verification signature scheme with identical message and signature space:

1. $\mathsf{Setup}_{\text{Det}}(1^\lambda)$: It computes $(\mathsf{sk}_{\text{RV}}, \mathsf{vk}_{\text{RV}}) \xleftarrow{\$} \mathsf{Setup}_{\text{RV}}(1^\lambda)$ and returns $(\mathsf{sk}, \mathsf{vk}) = (\mathsf{sk}_{\text{RV}}, \mathsf{vk}_{\text{RV}})$.

[2] In the previous section, we showed how to amplify the security from $\chi = 1 - 1/\mathsf{poly}(\lambda)$ to $\chi = 0$. Therefore, our starting point in the transformation will be a randomized verification signature scheme with $\chi = 0$.

2. $\mathsf{Sign}_{\mathrm{Det}}(\mathsf{sk}, m)$: Let $\mathsf{sk} = \mathsf{sk}_{\mathrm{RV}}$. It computes $\sigma \xleftarrow{\$} \mathsf{Sign}_{\mathrm{RV}}(\mathsf{sk}_{\mathrm{RV}}, m)$ and outputs σ as the signature.
3. $\mathsf{Verify}_{\mathrm{Det}}(\mathsf{vk}, m, \sigma)$: Let $\mathsf{vk} = \mathsf{vk}_{\mathrm{RV}}$ and $r = H(m, \sigma)$. It outputs $\mathsf{Verify}_{\mathrm{RV}}(\mathsf{vk}_{\mathrm{RV}}, m, \sigma; r)$.

Theorem 3. *If $\mathcal{S}_{\mathrm{RV}} = (\mathit{Setup}_{\mathrm{RV}}, \mathit{Sign}_{\mathrm{RV}}, \mathit{Verify}_{\mathrm{RV}})$ be a $\chi = 0$-EUFCMA secure randomized verification signature scheme (Definition 3) with message space $\mathcal{M}(\cdot)$, signature space $\Sigma(\cdot)$ and verification coin space $\mathcal{R}(\cdot)$, then above construction forms a EUFCMA secure deterministic verification signature scheme as per Definition 1 with message space $\mathcal{M}(\cdot)$ and signature space $\Sigma(\cdot)$.*

Proof. It is straightforward to verify the correctness as it follows directly from the correctness of underlying randomized verification signature scheme. For proving security in the random oracle model, we will assume the hash function H is modeled as a random oracle. Let q denote the number of queries made by the adversary to the random oracle. Without loss of generality, we can assume that the forgery (m^*, σ^*) is one of the q queries.

We can directly reduce an attack on our deterministic verification scheme construction to an attack on $\mathcal{S}_{\mathrm{RV}}$. Our reduction algorithm $\mathcal{B}$ will receive a verification key $\mathsf{vk}_{\mathrm{RV}}$ from $\mathcal{S}_{\mathrm{RV}}$ challenger. It forwards $\mathsf{vk}_{\mathrm{RV}}$ to $\mathcal{A}$. $\mathcal{B}$ relays all signature queries made by $\mathcal{A}$ to $\mathcal{S}_{\mathrm{RV}}$ challenger. For answering hash queries $\mathcal{B}$ maintains a lookup table, and for each new query (m, σ) it chooses a uniformly random coin from $\mathcal{R}$. Finally, when $\mathcal{A}$ submits the forgery (m^*, σ^*), $\mathcal{B}$ forwards it to the challenger.

Suppose $\mathcal{A}$ wins with probability ϵ_1, and let $\mathcal{R}^* = \mathcal{R}_{m^*,\sigma^*}$.

$$\begin{aligned}\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ outputs } (m^*, \sigma^*) \text{ and } H(m^*, \sigma^*) \in \mathcal{R}^*] \\ &\leq \Pr[\mathcal{A} \text{ outputs } (m^*, \sigma^*) \text{ and } H(m^*, \sigma^*) \in \mathcal{R}^* \text{ and } |\mathcal{R}^*|/|\mathcal{R}| \leq \operatorname{negl}(\lambda)] \\ &\quad + \Pr[\mathcal{A} \text{ outputs } (m^*, \sigma^*) \text{ and } |\mathcal{R}^*|/|\mathcal{R}| \geq 1/\mathsf{poly}(\lambda)]\end{aligned}$$

Now, note that $\Pr[\mathcal{A} \text{ outputs } (m^*, \sigma^*) \text{ and } H(m^*, \sigma^*) \in \mathcal{R}^* \text{ and } |\mathcal{R}^*|/|\mathcal{R}| \leq \operatorname{negl}(\lambda)] \leq q \cdot \operatorname{negl}(\lambda)$. This is because for each of the random oracle queries, the response is a uniformly random element in $\mathcal{R}$. As a result, using union bound, $\Pr[\exists \text{ RO query } (m_i, \sigma_i) \text{ such that } |\mathcal{R}_{m_i,\sigma_i}|/|\mathcal{R}| \leq \operatorname{negl}(\lambda) \text{ and } H(m_i, \sigma_i) \in \mathcal{R}_{m_i,\sigma_i}] \leq q \cdot \operatorname{negl}(\lambda)$.

Therefore, $\Pr[\mathcal{B} \text{ wins}] = \Pr[\mathcal{A} \text{ outputs } (m^*, \sigma^*) \text{ and } |\mathcal{R}^*|/|\mathcal{R}| \geq 1/\mathsf{poly}(\lambda)] \geq \epsilon_1 - \operatorname{negl}(\lambda)$.

7 Derandomizing Verification in the Standard Model

In this section, we present a transformation from a $\chi = 0$-EUFCMA secure signature scheme with randomized verification to one that has a deterministic verification algorithm, and prove it EUFCMA secure in the standard model. At a high level, the transformed scheme can be described as follows. The verification key consists of n uniformly random strings $r_1, \ldots, r_n$ from $\mathcal{R}$, where n

is bounded by a sufficiently large polynomial in the security parameter. These random strings are then used by the verification algorithm to check if σ is a valid signature for message m. Note that the random strings $r_1, \ldots, r_n$ used during verification are chosen during Setup and kept as part of the verification key. At first sight, it might appear that, since the random strings are fixed during setup, the reduction to the underlying verification scheme will not go through because now the adversary's forgery may depend on the random coins used to verify. However, this is not the case because the underlying randomized verification signature scheme satisfies $\chi = 0$-EUFCMA security, therefore taking a union bound over all possible message-signature pairs we could rule out this case as if n is sufficiently large, then the probability that some bad message-signature pair gets verified using all $r_1, \ldots, r_n$ (independently) is negligible. Below we describe the transformation.

Let $\mathcal{S}_{\mathrm{RV}} = (\mathsf{Setup}_{\mathrm{RV}}, \mathsf{Sign}_{\mathrm{RV}}, \mathsf{Verify}_{\mathrm{RV}})$ be a $\chi = 0$-EUFCMA secure randomized verification signature scheme with message space $\mathcal{M}(\cdot)$, signature space $\Sigma(\cdot)$ and verification coin space $\mathcal{R}(\cdot)$. We construct a deterministic verification signature scheme with identical message and signature space:

1. $\mathsf{Setup}_{\mathrm{Det}}(1^\lambda)$: The setup algorithm first chooses $(\mathsf{sk}_{\mathrm{RV}}, \mathsf{vk}_{\mathrm{RV}}) \xleftarrow{\$} \mathsf{Setup}_{\mathrm{RV}}(1^\lambda)$. Next, it sets $\mathsf{sk} = \mathsf{sk}_{\mathrm{RV}}$, $n = \Omega(\log(|\mathcal{M}| \cdot |\Sigma|))$, and chooses n coins $\{r_1, \ldots, r_n\}$ where $r_i \xleftarrow{\$} \mathcal{R}$ for $i \leq n$. The verification key vk consists of $(\mathsf{vk}_{\mathrm{RV}}, \{r_1, \ldots, r_n\})$.
2. $\mathsf{Sign}_{\mathrm{Det}}(\mathsf{sk}, m)$: Let $\mathsf{sk} = \mathsf{sk}_{\mathrm{RV}}$. The signing algorithm computes $\sigma \xleftarrow{\$} \mathsf{Sign}_{\mathrm{RV}}(\mathsf{sk}_{\mathrm{RV}}, m)$ and outputs σ as the signature.
3. $\mathsf{Verify}_{\mathrm{Det}}(\mathsf{vk}, m, \sigma)$: Let $\mathsf{vk} = (\mathsf{vk}_{\mathrm{RV}}, \{r_1, \ldots, r_n\})$. The verification algorithm checks if $\mathsf{Verify}_{\mathrm{RV}}(\mathsf{vk}_{\mathrm{RV}}, m, \sigma; r_i) = 1$ for all $i \leq n$. If so, it outputs 1, else it outputs 0.

Theorem 4. *Let $\mathcal{S}_{\mathrm{RV}} = (\mathsf{Setup}_{\mathrm{RV}}, \mathsf{Sign}_{\mathrm{RV}}, \mathsf{Verify}_{\mathrm{RV}})$ be a $\chi = 0$-EUFCMA secure randomized verification signature scheme (Definition 3) with message space $\mathcal{M}(\cdot)$, signature space $\Sigma(\cdot)$ and verification coin space $\mathcal{R}(\cdot)$. The above construction forms a EUFCMA secure deterministic verification signature scheme as per Definition 1 with message space $\mathcal{M}(\cdot)$ and signature space $\Sigma(\cdot)$.*

Proof. It is straightforward to verify the correctness as it follows directly from the correctness of underlying randomized verification signature scheme. To formally describe our reduction we distinguish between two types of forgers that an attacker $\mathcal{A}$ can emulate. Let $\mathcal{R}_{m^*,\sigma^*} = \{r \in \mathcal{R} : \mathsf{Verify}_{\mathrm{RV}}(\mathsf{vk}_{\mathrm{RV}}, m^*, \sigma^*; r) = 1\}$ denote the set of random coins on which verifier accepts the forgery (m^*, σ^*). We distinguish between two types of forgers as follows:

Type 1 forger: $\mathcal{A}$ with non-negligible probability, outputs a forgery (m^*, σ^*) such that $|\mathcal{R}_{m^*,\sigma^*}|/|\mathcal{R}|$ is *non-negligible* in λ.

Type 2 forger: $\mathcal{A}$ with non-negligible probability, outputs a forgery (m^*, σ^*) such that $|\mathcal{R}_{m^*,\sigma^*}|/|\mathcal{R}|$ is *negligible* in λ.

Claim 3. *If $\mathcal{A}$ emulates a Type 1 forger, then there exists a reduction algorithm $\mathcal{B}$ that breaks $\chi = 0$-EUFCMA security of randomized verification signature scheme $\mathcal{S}_{\mathrm{RV}}$.*

Proof. In case of type 1 forgery, we can directly reduce an attack on our deterministic verification scheme construction to an attack on $\mathcal{S}_{\mathrm{RV}}$. Our reduction algorithm $\mathcal{B}$ will receive a verification key $\mathsf{vk}_{\mathrm{RV}}$ from $\mathcal{S}_{\mathrm{RV}}$ challenger. It chooses n coins $\{r_1, \ldots, r_n\} \xleftarrow{\$} \mathcal{R}^n$ and forwards $(\mathsf{vk}_{\mathrm{RV}}, \{r_1, \ldots, r_n\})$ to $\mathcal{A}$. $\mathcal{A}$ relays all signature queries made by $\mathcal{A}$ to $\mathcal{S}_{\mathrm{RV}}$ challenger. By assumption, $|\mathcal{R}_{m^*,\sigma^*}|/|\mathcal{R}|$ is *non-negligible* for a type 1 forger, thus $\mathrm{VerifyProb}(\lambda, \mathsf{vk}_{\mathrm{RV}}, m^*, \sigma^*) = |\mathcal{R}_{m^*,\sigma^*}|/|\mathcal{R}|$ is also non-negligible. Hence, if $\mathcal{A}$ emulates a type 1 forger, then it also breaks $\chi = 0$-EUFCMA security of randomized verification signature scheme $\mathcal{S}_{\mathrm{RV}}$.

Claim 4. *If $\mathcal{A}$ emulates a Type 2 forger, then it does not break $\chi = 0$-EUFCMA security of randomized verification signature scheme $\mathcal{S}_{\mathrm{RV}}$ with all but negligible probability.*

Proof. In case of type 2 forgery, we show that if we set $n = \Omega(\ \log(|\mathcal{M}| \cdot |\Sigma|)\ )$, [3] then (with all but negligible probability) there does not exists any (m, σ) pair such that $\{r_1, \ldots, r_n\} \subseteq \mathcal{R}_{m,\sigma}$ and $|\mathcal{R}_{m,\sigma}|/|\mathcal{R}|$ is negligible. Let Bad denote the set of (m, σ) pairs such that $|\mathcal{R}_{m,\sigma}|/|\mathcal{R}|$ is negligible, and $R_{\max} = \max\limits_{(m,\sigma)\in\mathsf{Bad}} |\mathcal{R}_{m,\sigma}|$. The proof follows from union bound over all such (m, σ) pairs in Bad. First, observe that

$$\forall\ (m, \sigma) \in \mathsf{Bad}, \quad \Pr[\mathsf{Verify}_{\mathrm{Det}}(\mathsf{vk}, m, \sigma) = 1] \leq \left(\frac{R_{\max}}{|\mathcal{R}|}\right)^n \leq (\mu')^n,$$

where μ' is a negligible function. Using union bound, we can write that

$$\Pr\left[\bigvee_{(m,\sigma)\in\mathsf{Bad}} \mathsf{Verify}_{\mathrm{Det}}(\mathsf{vk}, m, \sigma) = 1\right] \leq |\mathcal{M}| \cdot |\Sigma| \cdot (\mu')^n.$$

Since we set $n = \Omega(\ \log(|\mathcal{M}| \cdot |\Sigma|)\ )$, the above expression is negligible in λ. Therefore, no type 2 forger can win with non-negligible probability over random choice of n verification coins.

Therefore, reduction algorithm $\mathcal{B}$ always guesses $\mathcal{A}$ to be a Type 1 forger and plays the standard reduction.

References

1. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_13

[3] Setting $n = \Omega(\ \log(|\mathcal{M}| \cdot |\Sigma|)/\log(1/\mu')\ )$ is sufficient where μ' is a negligible function that depends upon the parameters of the randomized verification scheme.

2. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. SIAM J. Comput. **32**(3), 586–615 (2003)
3. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. J. Cryptol. **17**(4), 297–319 (2004)
4. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001). doi:10.1007/3-540-45325-3_32
5. Cui, Y., Fujisaki, E., Hanaoka, G., Imai, H., Zhang, R.: Formal security treatments for IBE-to-signature transformation: relations among security notions. IEICE Trans. **92–A**(1), 53–66 (2009)
6. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1988)
7. Naor, M.: On cryptographic assumptions and challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003). doi:10.1007/978-3-540-45146-4_6
8. Schröder, D., Unruh, D.: Security of blind signatures revisited. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 662–679. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30057-8_39
9. Unruh, D.: Everlasting multi-party computation. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 380–397. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40084-1_22
10. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005). doi:10.1007/11426639_7
11. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03356-8_36
12. Yao, A.C.: Theory and application of trapdoor functions. In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, SFCS 1982, pp. 80–91 (1982). http://dx.doi.org/10.1109/SFCS.1982.95

Side Channel Attack

Trade-Offs for S-Boxes: Cryptographic Properties and Side-Channel Resilience

Claude Carlet[1], Annelie Heuser[2], and Stjepan Picek[1,3,4](✉)

[1] Universities of Paris VIII and Paris XIII, LAGA, UMR 7539, CNRS,
Saint-Denis, France
[2] CNRS/IRISA, Rennes, France
[3] Massachusetts Institute of Technology, CSAIL, Cambridge, USA
[4] Cyber Security Research Group, Delft University of Technology,
Mekelweg 2, Delft, The Netherlands
stjepan@computer.org

Abstract. When discussing how to improve side-channel resilience of a cipher, an obvious direction is to use various masking or hiding countermeasures. However, such schemes come with a cost, e.g. an increase in the area and/or reduction of the speed. When considering lightweight cryptography and various constrained environments, the situation becomes even more difficult due to numerous implementation restrictions. However, some options are possible like using S-boxes that are easier to mask or (more on a fundamental level), using S-boxes that possess higher inherent side-channel resilience. In this paper we investigate what properties should an S-box possess in order to be more resilient against side-channel attacks. Moreover, we find certain connections between those properties and cryptographic properties like nonlinearity and differential uniformity. Finally, to strengthen our theoretical findings, we give an extensive experimental validation of our results.

1 Introduction

When designing a block cipher, one needs to consider many possible cryptanalysis attacks and often give the best trade-off between the security, speed, ease of implementation, etc. Besides the two main directions in the form of linear [1] and differential [2] cryptanalyses, today the most prominent attacks come from the implementation attacks group where side-channel attacks (SCAs) play an important role. To protect against SCA, one common option is to use various countermeasures such as hiding or masking schemes [3] where one well known example is the threshold implementation [4]. However, such countermeasures come with a cost when implementing ciphers. If considering more resource constrained environments, one often does not have enough resources to implement standard ciphers like AES and therefore one needs to use lightweight cryptography. However, even lightweight ciphers can be too resource demanding especially when the cost of countermeasures is added. Therefore, although countermeasures

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 393–414, 2017.
DOI: 10.1007/978-3-319-61204-1_20

represent the way how to go when considering SCA protection, there is no countermeasure (at least at the current state of the research) that offers sufficient protection against any attack while being cheap enough to be implemented in any environment.

In this paper, we consider how to improve SCA resilience of ciphers without imposing any extra cost. This is possible by considering the inherent resilience of ciphers. We particularly concentrate on block ciphers which utilize S-boxes and therefore study the resilience of S-boxes against side-channel attacks.

In the case of SCA concentrating only on 1-bit of the S-box output, a theoretical connection between the side-channel resistance and differential uniformity of S-boxes has been found in [5]. In particular, the authors showed that the higher the side-channel resistance, the smaller the differential resistance. However, as we show, this extension does not straightforwardly hold when considering more complex leakage models as the Hamming weight of the S-box output, which is the most prominent leakage model in side-channel analysis when considering Correlation Power Analysis (CPA) [6]. We therefore investigate S-box parameters which may influence the side-channel resistance while still having good or optimal cryptographic properties. The (almost) preservation of Hamming weight and a small Hamming distance between x and $F(x)$ are two properties each of which could strengthen the resistance to SCA from an intuitive perspective. Our theoretical and empirical findings show that notably in the case when exactly preserving the Hamming weight, the SCA resilience is improved. Moreover, we relax this assumption and investigate in S-boxes that almost preserve the Hamming weight. For our study, we employ the confusion coefficient [7] as a metric for side-channel resistance. Besides the signal-to-noise-ratio and the number of observed measurements, the confusion coefficient is the factor influencing the success rate of CPA and, moreover, it is the only factor that depends on the underlying considered algorithm and thus on the S-box. More precisely our main contributions are:

1. We calculate (resp. we bound above) the confusion coefficient value of a function F in the two scenarios where:
   (a) x and $F(x)$ have the same Hamming weight.
   (b) in average, $F(x)$ has a Hamming weight near that of x.
2. We observe that the S-boxes with no difference between the Hamming weights of their input and output have nonlinearity equal to 0; more generally, the same happens when the Hamming weight of x and the Hamming weight of $F(x)$ have always the same parity. Such functions are of course to be avoided from a cryptanalysis perspective. Furthermore, we show that more generally as well, for every S-box F, denoting by d_{w_H} the number of inputs x for which the Hamming weights of x and $F(x)$ have different parities, F has nonlinearity at most d_{w_H}. This implies that if the number of inputs x such that $w_H(x) \neq w_H(F(x))$ is at most d_{w_H}, the nonlinearity is at most d_{w_H}. We show in Example 2 that this does not make however the S-box necessarily weak. We emphasize that although these observations could be regarded trivial, they have practical consequences.

3. We show the connection between the number of fixed points in a function F and its nonlinearity.
4. We show that S-boxes such that $F(x)$ lies at a small Hamming distance from x (or more generally from an affine function of x) cannot have high nonlinearity although the obtainable values are not too bad for $n = 4, 8$.
5. In the practical part, we confirm our theoretical findings about the connection between (almost) preserving the Hamming weight and the confusion coefficient by investigating several S-boxes.
6. We investigate the relationship between the confusion coefficient of different key guesses and evaluate a number of S-boxes used in today's ciphers to show that their SCA resilience can significantly differ.

2 Preliminaries

2.1 Generalities on S-Boxes

Let n, m be positive integers, i.e., $n, m \in \mathbb{N}^+$. We denote by $\mathbb{F}_2^n$ the n-dimensional vector space over $\mathbb{F}_2$ and by $\mathbb{F}_{2^n}$ the finite field with 2^n elements. The set of all n-tuples of elements in the field $\mathbb{F}_2$ is denoted by $\mathbb{F}_2^n$, where $\mathbb{F}_2$ is the Galois field with two elements. Further, for any set S, we denote $S \backslash \{0\}$ by S^*. The usual inner product of a and b equals $a \cdot b = \bigoplus_{i=1}^{n} a_i b_i$ in F_2^n.

The Hamming weight $w_H(a)$ of a vector a, where $a \in \mathbb{F}_2^n$, is the number of non-zero positions in the vector. An (n, m)-function is any mapping F from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. An (n, m)-function F can be defined as a vector $F = (f_1, \cdots, f_m)$, where the Boolean functions $f_i : \mathbb{F}_2^n \to \mathbb{F}_2$ for $i \in \{1, \cdots, m\}$ are called the coordinate functions of F.

The component functions of an (n, m)-function F are all the linear combinations of the coordinate functions with non all-zero coefficients. Since for every n, there exists a field $\mathbb{F}_{2^n}$ of order 2^n, we can endow the vector space $\mathbb{F}_2^n$ with the structure of that field when convenient. If the vector space $\mathbb{F}_2^n$ is identified with the field $\mathbb{F}_{2^n}$ then we can take $a \cdot b = tr(ab)$ where $tr(x) = x + x^2 + \ldots + x^{2^{n-1}}$ is the trace function from $\mathbb{F}_{2^n}$ to $\mathbb{F}_2$. The addition of elements of the finite field $\mathbb{F}_{2^n}$ is denoted with "+", as usual in mathematics. Since, often, we identify $\mathbb{F}_2^n$ with $\mathbb{F}_{2^n}$ and if there is no ambiguity, we denote the addition of vectors of $\mathbb{F}_2^n$ when $n > 1$ with "+" as well.

An (n, m)-function F is balanced if it takes every value of $\mathbb{F}_2^m$ the same number 2^{n-m} of times.

The Walsh-Hadamard transform of an (n, m)-function F is (see e.g. [8]):

$$W_F(a, v) = \sum_{x \in \mathbb{F}_2^m} (-1)^{v \cdot F(x) + a \cdot x}, \ a, v \in \mathbb{F}_2^m. \tag{1}$$

The *nonlinearity* *nl* of an (n, m)-function F equals the minimum nonlinearity of all its component functions $v \cdot F$, where $v \in \mathbb{F}_2^{m*}$ [8,9]:

$$nl = 2^{n-1} - \frac{1}{2} \max_{\substack{a \in \mathbb{F}_2^n \\ v \in \mathbb{F}_2^{m*}}} |W_F(a, v)|. \tag{2}$$

The nonlinearity of any (n, m) function F is bounded above by the so-called *covering radius bound*:

$$nl \leq 2^{n-1} - 2^{\frac{n}{2}-1}. \tag{3}$$

In the case $m = n$, a better bound exists. The nonlinearity of any (n, n) function F is bounded above by the so-called *Sidelnikov-Chabaud-Vaudenay bound* [10]:

$$nl \leq 2^{n-1} - 2^{\frac{n-1}{2}}. \tag{4}$$

Bound (4) is an equality if and only if F is an Almost Bent (AB) function, by definition of AB functions [8].

Let F be a function from $\mathbb{F}_2^n$ into $\mathbb{F}_2^m$ with $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^m$. We denote:

$$D_F(a, b) = \{x \in \mathbb{F}_2^n : F(x) + F(x + a) = b\}. \tag{5}$$

The entry at the position (a, b) corresponds to the cardinality of the delta difference table $D_F(a, b)$ and is denoted as $\delta(a, b)$. The *differential uniformity* δ_F is then defined as [11]:

$$\delta_F = \max_{a \neq 0, b} \delta(a, b). \tag{6}$$

Functions that have differential uniformity equal to 2 are called the Almost Perfect Nonlinear (APN) functions. Every AB function is also APN, but the converse does not hold in general. AB functions exist only in an odd number of variables, while APN functions also exist for an even number of variables. When discussing the differential uniformity parameter for permutations, the best possible (and known) value is 2 for any odd n and also for $n = 6$. For n even and larger than 6, this is an open question. The differential uniformity value for the inverse function $F(x) = x^{2^n-2}$ equals 4 when n is even and 2 when n is odd.

2.2 Side-Channel Resistance

Side-channel attacks analyze physical leakage that is unintentionally emitted during cryptographic operations in a device (e.g., through the power consumption [12] or electromagnetic emanation [13]). This side-channel leakage is statistically dependent on the intermediate processed values involving the secret key, which makes it possible to retrieve the secret from the measured data. In particular, as the attacker wants to retrieve the secret key, he makes predictions (hypotheses) on a small enumerable chunk (e.g., byte) of an intermediate state using all possible key values.

The side-channel resistance of implementations against Correlation Power Attack (CPA) [6] depends on three factors: the number of measurement traces, the signal-to-noise ratio (SNR) [14], and the confusion coefficient [7]. The relationship between the three factors is linear in case of low SNR [15]. The confusion coefficient measures the discrepancy between the hypothesis of an intermediate state using the correct (secret) key and any hypothesis made with a (wrong) key assumption. Therefore, as one compares possible intermediate processed values, the confusion coefficient depends on the underlying cryptographic algorithm and thus, if the attacker targets an S-box operation, on the side-channel resistance of that S-box. More precisely, let us assume the attacker exploits an intermediate processed value $F(k_c + t)$ during the first round that depends on the secret key $k_c \in \mathbb{F}_2^n$, an n-bit chunk of the plaintext $t \in \mathbb{F}_2^n$, and an S-box function F. Moreover, let us make the commonly accepted assumption that the device is leaking side-channel information as the Hamming weight (see e.g., [14]) of intermediate values with additive noise N:

$$w_H(F(k_c + t)) + N. \tag{7}$$

As the secret key k_c is unknown to the attacker, he computes for each key guess $k_g \in \mathbb{F}_2^n$ a hypothesis about the intermediate state:

$$y_{k_g,t} = y(k_g, t) = w_H(F(k_g + t)) \tag{8}$$

of the deterministic part of the leakage in Eq. (7). Interestingly, these hypotheses are not independent and their discrepancy is characterized by the confusion coefficient. Originally in [7] the confusion coefficient has been introduced for $(n, 1)$ Boolean functions:

$$\kappa(k_c, k_g) = Pr[(y(k_c, T)) \neq (y(k_g, T))], \tag{9}$$

with T being the random variable whose realization is t. In [5], the authors related $\kappa(k_c, k_g)$ in Eq. (9) to δ_F and showed that the higher the side-channel resistance, the smaller the differential resistance (that is, the higher δ_F). In fact, $\kappa(k_c, k_g)$ is represented as

$$\frac{1}{2^n} \sum_{t \in \mathbb{F}_2^n} \left(F(t + k_c) + F(t + k_g)\right), \tag{10}$$

which can then be straightforwardly connected to δ_F for 1-bit models.

In [16] the authors extend $\kappa(k_c, k_g)$ to the general multi-bit case for CPA and thus to (n, m)-functions F. In this paper, we use the definition given in [15] which is a standardized version of confusion coefficient given in [16] and thus a natural extension of Eq. (9):

$$\kappa(k_c, k_g) = \mathbb{E}\Big\{ \Big(\frac{1}{2}(y(k_c, T) - y(k_g, T)) \Big)^2 \Big\}, \tag{11}$$

where y is assumed to be standardized (i.e., $\mathbb{E}(y(\cdot, T)) = 0, Var(y(\cdot, T)) = 1$). More specifically, Eq. (11) enables us to compare confusion coefficients for different functions F. By substituting $y(*)$ with Eq. (8) and denoting $x = t \oplus k_c$ and $a = k_c + k_g$ we can write $\kappa(k_c, k_g)$ as

$$\mathbb{E}\left(\left(\frac{1}{2} \left(\frac{w_H(F(x))}{\sqrt{\frac{m}{4}}} - \frac{w_H(F(x + a))}{\sqrt{\frac{m}{4}}} \right) \right)^2 \right) = \mathbb{E}\left(\left(\frac{w_H(F(x)) - w_H(F(x + a))}{\sqrt{m}} \right)^2 \right). \tag{12}$$

Now, it is easy to see that from Eq. (12) we cannot straightforwardly derive a connection to δ_F for (n, m) functions. More precisely, for $m = 1$ the square is just 4 times the value of $F(t) + F(t + a)$ and then the confusion coefficient equals

$\delta(a,1)$. For $m > 1$ we have the square of the difference between the weights of $F(t)$ and $F(t+a)$ which is not 4 times the weight of $b = F(t) + F(t+a)$ because the $1-0$ and the $0-1$ count with their signs in the sum. So there is no direct connection with δ_F anymore.

As a decisive criterion for comparison between confusion coefficients, the minimum value of $\kappa(k_c, k_g)$ was specified in [15] as it relates to the success rate when the SNR is low. Note that the higher is the minimum of the confusion coefficient, the lower is the side-channel resilience. This comes from the fact that the lower the confusion coefficient the smaller is the (Euclidean) distance between the correct key k_c and a key guess k_g and thus the harder it is for an attacker to distinguish if the leakage is arising due to a computation with k_c or k_g. A detailed discussion on this will be given in Subsect. 5.2. On the other hand, in [17] authors use $var(\kappa(k_c, k_g))$ as a criterion, where smaller values indicate lower side-channel resilience. Our experiments in Sect. 5 show that both metrics coincide with the empirical resilience using simulations.

In the case $\kappa(k_c, k_g) = 0$ or $\kappa(k_c, k_g) = 1$ for any $k_g \neq k_c$, CPA is not able to distinguish between k_c and this key guess k_g and will thus fail to reveal the secret key exclusively even if the number of measurements goes to infinity. More precisely, $\kappa(k_c, k_g) = 0$ means that for a key guess k_g one observes exactly the same intermediate values (see Eq. (8)) as for the correct key k_c. Contrary for $\kappa(k_c, k_g) = 1$ one observes the complementary value (can be seen from Eqs. (9) and (11)), however, as CPA takes the absolute value of correlation (due to hardware related properties [14]) an attacker again cannot distinguish between k_c and k_g in this case. In general, normalized confusion coefficient values close to 0.5 indicate that k_c and k_g can be easily distinguished (see Eq. (9)). We will show in Sect. 3 and empirically confirm in Sect. 5 that in case of preserving w_H there exists an key guess k_g such that $\kappa(k_c, k_g) = 1$.

3 S-Boxes (Almost) Preserving the Hamming Weight

3.1 Relation to the Confusion Coefficient

To obtain, for an (n, m)-function F, a connection between the confusion coefficient parameter and the Hamming weight preservation (i.e., the fact that, for every x, $F(x)$ has the same Hamming weight as x) or, more generally, a limited average Hamming weight modification, we start with Eq. (12). For any function F, we have:

$$\begin{aligned}
&\frac{1}{m}\mathbb{E}\left(\sum_{i=1}^{m} F_i(x) - \sum_{i=1}^{m} F_i(x+a)\right)^2 \\
&= \frac{1}{m}\mathbb{E}\left(\sum_{i=1}^{m}\left(\frac{1}{2} - \frac{1}{2}(-1)^{F_i(x)}\right) - \sum_{i=1}^{m}\left(\frac{1}{2} - \frac{1}{2}(-1)^{F_i(x+a)}\right)\right)^2 \\
&= \frac{1}{4m}\mathbb{E}\left(\sum_{i=1}^{m}\left((-1)^{F_i(x)} - (-1)^{F_i(x+a)}\right)\right)^2
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{4m} \sum_{1 \le i,j \le m} \mathbb{E}\left((-1)^{F_i(x)} - (-1)^{F_i(x+a)} \right) \left((-1)^{F_j(x)} - (-1)^{F_j(x+a)} \right) \\
&= \frac{1}{4m} \sum_{1 \le i,j \le m} \mathbb{E}\Big((-1)^{F_i(x)+F_j(x)} - (-1)^{F_i(x)+F_j(x+a)} - \\
&\quad (-1)^{F_i(x+a)+F_j(x)} + (-1)^{F_i(x+a)+F_j(x+a)} \Big) \\
&= \frac{1}{2m} \sum_{1 \le i,j \le m} \mathbb{E}\left((-1)^{F_i(x)+F_j(x)} - (-1)^{F_i(x)+F_j(x+a)} \right) \\
&= \frac{1}{2m} \mathbb{E}\left(\left(\sum_{i=1}^{m} (-1)^{F_i(x)} \right)^2 - \left(\sum_{i=1}^{m} (-1)^{F_i(x)} \right) \left(\sum_{i=1}^{m} (-1)^{F_i(x+a)} \right) \right). \qquad (13)
\end{aligned}
$$

Lemma 1 addresses the case where F preserves the Hamming weight, whereas the scenario in which F modifies the Hamming weight in a limited way is described in Lemma 2. Note that the first scenario is a particular case of the second.

Lemma 1. *For an (n,n)-function such that, for every x, $F(x)$ has the same Hamming weight as x, the confusion coefficient equals $\frac{w_H(a)}{n}$.*

Proof. If F preserves the Hamming weight, that is, if $w_H(F(x)) = w_H(x)$ for every x (or more generally, if F is the composition of a function preserving the weight by an affine isomorphism on the right), then the confusion coefficient $\kappa(k_c, k_g) = \mathbb{E}\left(\left(\frac{w_H(F(x)) - w_H(F(x+a))}{\sqrt{n}} \right)^2 \right)$, where $a = k_c + k_g$, becomes $\mathbb{E}\left(\left(\frac{w_H(x) - w_H(x+a)}{\sqrt{n}} \right)^2 \right)$, and by applying Eq. (13) (which is valid for every F) to $F = Id$, we obtain:

$$
\begin{aligned}
&\frac{1}{2n} \mathbb{E}\left(\left(\sum_{i=1}^{n} (-1)^{x_i} \right)^2 - \left(\sum_{i=1}^{n} (-1)^{x_i} \right) \left(\sum_{i=1}^{n} (-1)^{x_i + a_i} \right) \right) \qquad (14) \\
&= \frac{1}{2n} \mathbb{E}\left(\sum_{1 \le i,j \le m} (-1)^{x_i + x_j} - \sum_{1 \le i,j \le m} (-1)^{x_i + x_j + a_j} \right).
\end{aligned}
$$

The expectations of all these sums for $i \neq j$ are null (since the character sums of nonzero linear functions are null), and we obtain:

$$
\frac{1}{2n} \mathbb{E}\left(m - \sum_{1 \le i \le m} (-1)^{a_i} \right) = \frac{1}{n} \mathbb{E}\left(w_H(a) \right) = \frac{w_H(a)}{n}. \qquad (15)
$$

Example 1. *For $n = 4$, Lemma 1 gives $\min_{k_c \neq k_g} \kappa(k_c, k_g) = 0.25$ and for $w_H(a) = n$ we have $\kappa(k_c, k_g) = 1$, which means that the CPA distinguisher is not able to distinguish between these two hypotheses k_g and k_c (see Subsect. 2.2). Note that we give a more detailed discussion about the results and their ramifications in Sect. 5.*

Lemma 2. *For an (n,n)-function such that, on average, $F(x)$ has a Hamming weight near that of x, more precisely, where $\sum_x |w_H(F(x)) - w_H(x)| \leq d_{w_H}$, where d_{w_H} is some number, the standardized confusion coefficient is bounded above by $\frac{w_H(a)}{n} + \frac{4d_{w_H}}{2^n}$.*

Proof. If $\mathbb{E}(|w_H(F(x)) - w_H(x)|) \leq \frac{d_{w_H}}{2^n}$, then according to Lemma 1 and its proof the confusion coefficient $\kappa(k_c,k_g) = \mathbb{E}\left(\left(\frac{w_H(F(x))-w_H(F(x+a))}{\sqrt{n}}\right)^2\right)$ is such that

$$\begin{aligned}
\left|\kappa(k_c,k_g) - \frac{w_H(a)}{n}\right| &\leq \mathbb{E}\left(\left|\left(\frac{w_H(F(x)) - w_H(F(x+a))}{\sqrt{n}}\right)^2 - \left(\frac{w_H(x) - w_H(x+a)}{\sqrt{n}}\right)^2\right|\right) \\
&= \mathbb{E}\left(\left|\left(\frac{w_H(F(x)) - w_H(F(x+a))}{\sqrt{n}} - \frac{w_H(x) - w_H(x+a)}{\sqrt{n}}\right)\right.\right. \\
&\qquad \left.\left.\left(\frac{w_H(F(x)) - w_H(F(x+a))}{\sqrt{n}} + \frac{w_H(x) - w_H(x+a)}{\sqrt{n}}\right)\right|\right) \\
&\leq \frac{2}{n}\left(\max_{x\in\mathbb{F}_2^n} w_H(F(x)) + \max_{x\in\mathbb{F}_2^n} w_H(x)\right)\mathbb{E}\left(|w_H(F(x)) - w_H(x)|\right) \\
&= \frac{4d_{w_H}}{2^n}.
\end{aligned}$$

3.2 Relation to Cryptographic Properties

We study the cryptographic consequences of the preservation of the Hamming weight. Again we first cover the specific case were the input and output of an S-box always have the same Hamming weight, and then the second case where the output has on average a Hamming weight close to that of the corresponding input (see Lemma 3).

If for every x, we have $w_H(F(x)) = w_H(x)$ then the sum (mod 2) of all coordinate functions of F equals the sum (mod 2) of all coordinates of x. This means that F has nonlinearity equal to zero since one of its component functions is linear. Of course, the same happens under the much weaker hypothesis that $w_H(F(x))$ and $w_H(x)$ have always the same parity. Therefore, an S-box function preserving the Hamming weight is cryptographically insecure.

However, if $\sum_x |w_H(F(x)) - w_H(x)| \leq d_{w_H}$, then we have $nl \leq d_{w_H}$. Indeed, this is a direct consequence of the following straightforward result, which has however much importance in our context:

Lemma 3. *If the Hamming weight of the Boolean function:*

$$x \mapsto (w_H(F(x)) - w_H(x)) \ [mod\, 2],$$

that is, $\sum_x((w_H(F(x)) - w_H(x)) \ [mod\, 2])$, is at most d_{w_H}, then we have $nl \leq d_{w_H}$.

Indeed, the Hamming distance between the component function $\sum_i F_i \ [mod\, 2])$ and the linear function $\sum_i x_i \ [mod\, 2])$ is then at most d_{w_H}.

Example 2. *For a $(4, 4)$-function F to have nonlinearity equal to 4 (optimal nonlinearity), it means that d_{w_H} must be at least 4. In order to construct functions with such properties, we ran a genetic algorithm as given by Picek et al. [17]. We use the same settings as there: 30 independent runs, population size equal to 50, 3-tournament selection, and mutation probability 0.3 per individual. The objective is the maximization of the following fitness function:*

$$fitness = nl + \Delta_{nl,4}(n \times 2^n - |w_H(F(x)) - w_H(x)|). \tag{16}$$

Here, $\Delta_{nl,4}$ represents the Kronecker delta function that equals 1 when nonlinearity is 4 and 0 otherwise. Notice we subtract the difference of the Hamming weights of the inputs and outputs of an S-box from the summed Hamming weight value for a $(4, 4)$-function since we work with the maximization problem while that value should be minimized. Interestingly, we observed that finding S-boxes with those properties is a relatively easy task and that the obtained S-boxes never have more than 8 fixed points. We give examples of such S-boxes in Table 1, for instance, S_5 where nonlinearity equals 4 and d_{w_H} is 4.

Next, inspired by our empirical results, we investigate whether it is theoretically possible to construct an S-box with even more fixed points while still having the maximal nonlinearity.

Lemma 4. *If an (n, n)-function has k fixed points then the maximal value of $W_F(a, v)$ when $v \neq 0$ is bounded below by $(k - 1)/(1 - 2^{-n})$. If nl is the nonlinearity of an (n, n)-function, then its number k of fixed points is not larger than $2^n - \lceil (2 - 2^{1-n})\, nl \rceil$.*

Proof. The number of fixed points k of an (n, n)-function F equals:

$$k = 2^{-n} \sum_{v \in \mathbb{F}_2^n} W_F(v, v) = 2^{-n} \sum_{x,v \in \mathbb{F}_2^n} (-1)^{v \cdot (x + F(x))}, \tag{17}$$

which follows from Eq. (1) when $a = v$ and the property that $\sum_{v \in \mathbb{F}_2^n} (-1)^{v \cdot a}$ equals 2^n if $a = 0$ and is null otherwise. The value of $W_F(0, 0)$ involved in Eq. (17) equals 2^n. We take it off and obtain:

$$k - 1 = 2^{-n} \sum_{v \in \mathbb{F}_2^{n*}} W_F(v, v). \tag{18}$$

Then the arithmetic mean of $W_F(v, v)$ when $v \neq 0$ equals $(k-1)/(1-2^{-n})$. This implies that $\max_v W_F(v, v)$ is at least $(k - 1)/(1 - 2^{-n})$ and the nonlinearity cannot be larger than $2^{n-1} - (k - 1)/(2 - 2^{1-n})$. The inequality $nl \leq 2^{n-1} - (k - 1)/(2 - 2^{1-n})$ is equivalent to $k \leq 2^n - \lceil (2 - 2^{1-n})\, nl \rceil$.

4 S-Boxes Minimizing the Hamming Distance

4.1 Relation to the Confusion Coefficient

In real world applications, the device may not only leak in the Hamming weight, but also in the Hamming distance, therefore we now extend our study to the case were the leakage arises from the Hamming distance between x and $F(x)$. Again we first study the relation to the confusion coefficient and then give the connection to cryptographic properties.

By the triangular inequality, we have $|w_H(F(x)) - w_H(x)| \leq d_H(x, F(x))$. This implies that $\sum_x |w_H(F(x)) - w_H(x)| \leq \sum_x d_H(x, F(x))$.

Hence, if $\sum_x d_H(x, F(x)) \leq d_{d_H}$, we can use Lemma 2 and deduce that also in this scenario the confusion coefficient is bounded by $\frac{w_H(a)}{n} + \frac{4d_{d_H}}{2^n}$.

4.2 Relation to Cryptographic Properties

From $\sum_x d_H(x, F(x)) \leq d_{d_H}$, up to adding a linear function (which does not change the nonlinearity nor the differential uniformity), considering S-boxes such that, for every x, $F(x)$ lies at a small distance from x corresponds to considering functions which take a too small number of values. We show that such functions have bad nonlinearity and bad differential uniformity.

Lemma 5. *Let F be any (n,m)-function such that $|F(\mathbb{F}_2^n)| \leq D$, then $\delta_F \geq \frac{2^n}{2^m-1}\left(\frac{2^n}{D} - 1\right)$ and $nl \leq 2^{n-1} - \frac{\frac{2^{n+m-1}}{D} - 2^{n-1}}{2^m-1}$.*

Proof. By using the Cauchy-Schwartz inequality, we obtain $\sum_{a \in \mathbb{F}_2^{n*}} |D_a F^{-1}(0)| = \sum_{b \in \mathbb{F}_2^m} |F^{-1}(b)|^2 - 2^n \geq \frac{(\sum_{b \in \mathbb{F}_2^m} |F^{-1}(b)|)^2}{D} - 2^n = \frac{2^{2n}}{D} - 2^n$, and there exists then $a \in \mathbb{F}_2^{m*}$ such that $|D_a F^{-1}(0)| \geq \frac{\frac{2^{2n}}{D} - 2^n}{2^m-1}$. This proves the first assertion.

We have a partition of $\mathbb{F}_2^n$ into at most D parts by the preimages $F^{-1}(b)$, $b \in \mathbb{F}_2^m$, and there exists then $b \in \mathbb{F}_2^m$ such that $|F^{-1}(b)| \geq \frac{2^n}{D}$; for such b, we have $\sum_{x \in \mathbb{F}_2^n, v \in \mathbb{F}_2^m} (-1)^{v \cdot (F(x)+b)} \geq \frac{2^{n+m}}{D}$, which is equivalent to $\sum_{v \in \mathbb{F}_2^m, v \neq 0} (-1)^{v \cdot b} W_F(0,v) \geq \frac{2^{n+m}}{D} - 2^n$, and then there exists $v \neq 0$ such that $|W_F(0,v)| \geq \frac{\frac{2^{n+m}}{D} - 2^n}{2^m-1}$, which implies that $nl \leq 2^{n-1} - \frac{\frac{2^{n+m-1}}{D} - 2^{n-1}}{2^m-1}$. This proves the second assertion.

If D is small with respect to 2^m (so that 2^{n-1} is small with respect to $\frac{2^{n+m-1}}{D}$) and D is small with respect to $2^{n/2}$ (so that $\frac{2^n}{D}$ is large with respect to $2^{n/2}$), the nonlinearity is bad with respect to the covering radius bound $nl \leq 2^{n-1} - 2^{n/2-1}$. More precisely, if $D \leq \frac{2^m}{\lambda}$ with $\lambda > 1$, then $nl \leq 2^{n-1} - \frac{(\lambda-1)2^{n-1}}{2^m-1} < 2^{n-1} - (\lambda - 1)2^{n-m-1}$ and if $(\lambda-1)2^{n-m}$ is significantly larger than $2^{n/2}$, the nonlinearity is bad with respect to the covering radius bound. We have also that if D is small with respect to 2^m then δ_F is large with respect to 2^{n-m} if $m < n$ and with 2 if $m = n$ (which are the smallest possible values of δ_F).

If F is an (n,n)-function and $x + F(x)$ has low weight for every x, say at most t_{d_H}, which is equivalent to saying that $d_H(x, F(x)) \leq t_{d_H}$ for every x, then its number of values is at most $D = \sum_{i=0}^{t_{d_H}} \binom{n}{i}$ and we can apply the result above to $x + F(x)$, which has the same nonlinearity and the same δ_F as F. As far as we know, these observations are new. Note that we also have the possibility of applying Lemma 3 and then we have that nonlinearity is bounded by t_{d_H}.

Remark 3. *Lemma 5 applies to the case when $d_H(x, F(x)) \leq t_{d_H}$ for every x where x equals $t \oplus k_g$. This represents a setting one would encounter when working for instance with software implementations. Now, if we consider a hardware setting (e.g., FPGA), then we are interested in the case $d_H(t, F(t \oplus k_g)) \leq t_{d_H}$ for every key. However, this case leads to the same observation as before but now with up to adding an affine function instead of up to adding a linear function as given in Lemma 5.*

5 Side-Channel Evaluation

5.1 Evaluation of S-Boxes with (Almost) w_H Preservation

As cryptographically non-optimal examples of S-boxes (almost) preserving w_H we consider five different functions F: the identity mapping (S_1), F not Id but preserving w_H (S_2), the identity mapping with an exchange of the images at position $x = 3$ and $x = 12$, i.e., $F(3) = 12$ and $F(12) = 3$, and as $w_H(3) = 2$ and $w_H(12) = 3$ we have $d_{w_H} = 2$ (see Lemma 1) (S_3), $F(x) = 2^n - x$ which gives the complementary Hamming weight (S_4). Finally, we investigate four S-box functions S_5 to S_8 with the smallest possible distance d_{w_H} that equals 4 and maximal possible nonlinearity equal to 4 (see Subsect. 3.2). S-box functions S_7 and S_8 have furthermore optimal differential uniformity (=4). The mappings are given in Table 1.

Table 1. Specifications of functions F, $(x)w_H(x)$

S-box	(0)0	(1)1	(2)1	(3)2	(4)1	(5)2	(6)2	(7)3	(8)1	(9)2	(10)2	(11)3	(12)2	(13)3	(14)3	(15)4
S_1	(0)0	(1)1	(2)1	(3)2	(4)1	(5)2	(6)2	(7)3	(8)1	(9)2	(10)2	(11)3	(12)2	(13)3	(14)3	(15)4
S_2	(0)0	(4)1	(8)1	(10)2	(1)1	(12)2	(3)2	(13)3	(2)1	(6)2	(9)2	(7)3	(5)2	(14)3	(11)3	(15)4
S_3	(0)0	(1)1	(2)1	(12)3	(4)1	(5)2	(6)2	(7)3	(8)1	(9)2	(10)2	(11)3	(3)1	(13)3	(14)3	(15)4
S_4	(15)4	(14)3	(13)3	(12)2	(11)3	(10)2	(9)2	(8)1	(7)3	(6)2	(5)2	(4)1	(3)2	(2)1	(1)1	(0)0
S_5	(0)0	(4)1	(8)1	(10)2	(1)1	(2)1	(9)2	(6)2	(3)2	(5)2	(7)3	(13)3	(12)2	(11)3	(14)3	(15)4
S_6	(0)0	(9)2	(2)1	(13)3	(8)1	(6)2	(5)2	(7)3	(1)1	(4)1	(12)2	(14)3	(3)2	(10)2	(11)3	(15)4
S_7	(0)0	(2)1	(5)2	(13)3	(8)1	(4)1	(6)2	(7)3	(1)1	(12)2	(10)2	(14)3	(3)2	(9)2	(11)3	(15)4
S_8	(0)0	(1)1	(10)2	(7)3	(2)1	(4)1	(5)2	(13)3	(8)1	(12)2	(3)2	(9)2	(6)2	(11)3	(14)3	(15)4

The confusion coefficients are illustrated in Fig. 1. Note that, the distribution of $\kappa(k_c, k_g)$ is independent on the particular choice of k_c (in the case there are no weak keys) and the values for $\kappa(k_c, k_g)$ are only permuted when choosing different value $k_c \in \mathbb{F}_2^n$. For our experiments we choose $k_c = 0$ and furthermore we order $\kappa(k_c, k_g)$ in an increasing order of magnitude for illustrative purpose. The minimum value of $\kappa(k_c, k_g)$ for $k_g \neq k_c$ is highlighted with a red cross as it is one indicator of the side-channel resistance. Moreover, we mark $\kappa(k_c, k_g) = 0$ or $\kappa(k_c, k_g) = 1$ with a red circle which points out that CPA is not able to distinguish between k_c and the marked k_g.

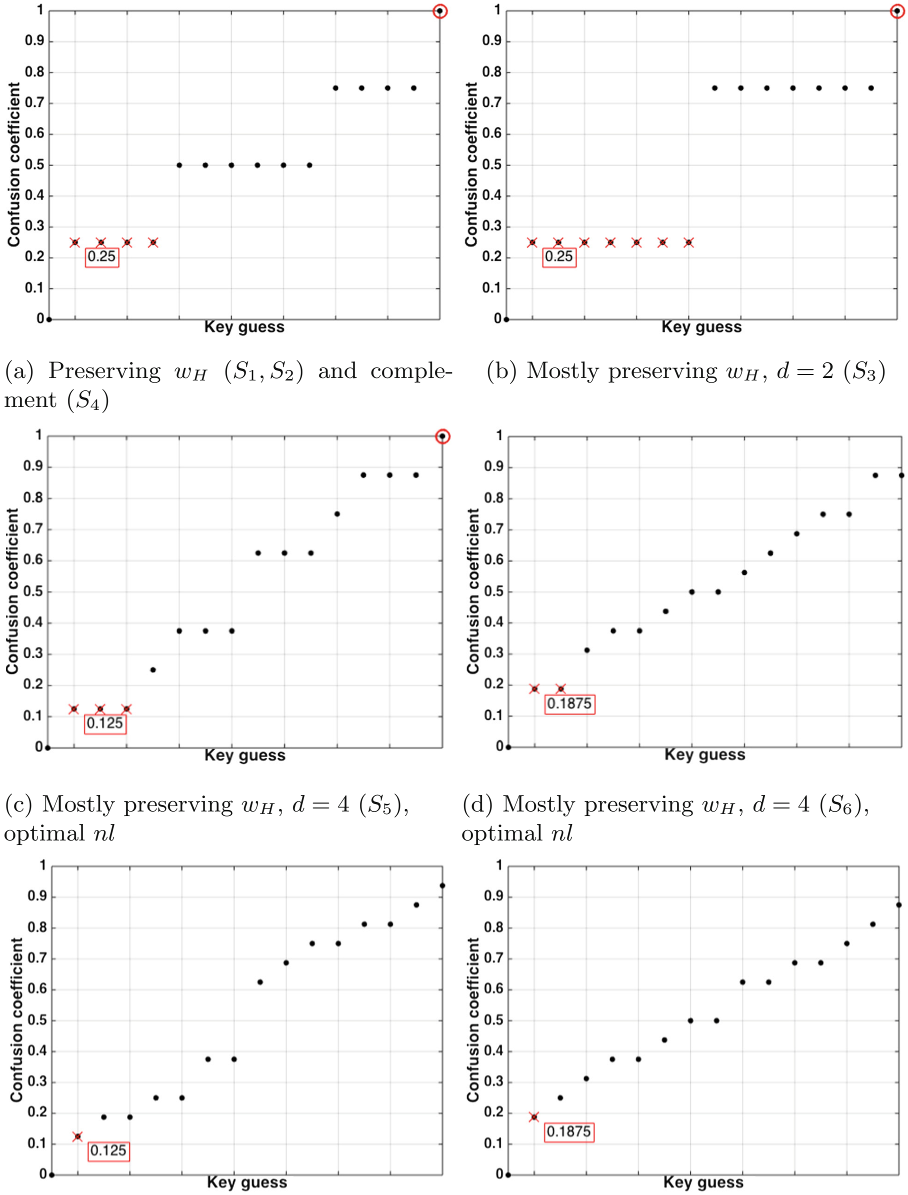
Figure 1a shows that, indeed, k_c is indistinguishable from one key hypothesis k_g if w_H is preserved. Or in other words, even if knowing t and observing $w_H(F(t + k_c)) + N$ with F equal to S_1 the attacker can not exclusively gain information about k_c even if the number of measurements $m \rightarrow \infty$. Moreover, it confirms Lemma 1. Note that in our example $a = k_g$, thus $\kappa(k_c, k_g) = \frac{w_H(k_g)}{4}$. Interestingly, when comparing our results to the study in [5], where the authors investigated $(n, 1)$-functions, we observe that the confusion coefficient takes different values which indeed confirms that the Hamming weight model is not a straightforward extension from 1-bit models. More precisely, in case of linear $(n, 1)$-function the authors observed that the confusion coefficient only takes values from {0,1}, whereas our examples illustrate (as well as our theoretical findings in Sect. 3) that the confusion coefficient is not restricted to only {0,1}, and is equal to 1 for only one particular k_g. Interestingly, for $d_{w_H} = 2$ (see in Fig. 1b) we also have that k_c is indistinguishable for one k_g. Moreover, apart from $\kappa(k_c, k_g) = 1$, only two different values are taken, each 7 times. This means that CPA is not able to distinguish between each of these 7 key guesses and in total only produces three different correlation values. When considering a complementary w_H preservation (e.g. $4 - w_H$) we achieve the same results as for w_H preservation (see also Fig. 1).

Note that, while being illustrative, these first four examples of F are not cryptographically optimal and thus are not suitable in practice. We therefore constructed four S-boxes (S_5 to S_8) with the smallest $d_{w_H}(= 4)$ while having optimal nonlinearity. Note that S_5, S_6 have suboptimal differential uniformity, while S_7, S_8 are cryptographically optimal (i.e. optimal nonlinearity and differential uniformity). Figures 1c to f show the confusion coefficient of S_5 to S_8. We can observe that all S-boxes have a very low minimum confusion coefficient that is even lower than for S_1 to S_4. Even more, as the previously investigated S-boxes, S_5 has $\kappa(k_c, k_g) = 1$. Therefore, we find an S-box with almost Hamming weight preserving for which even with an infinity amount of traces the secret key cannot exclusively be found. As the minimum value of the confusion coefficient of S_5 is low (=0.125) there exists additionally other key hypotheses which are harder to distinguish from the secret key. As a conclusion we can say that indeed exact w_H preserving results in a good side-channel resistance since we have $\kappa(k_c, k_g) = 1$. Moreover, when the w_H is almost preserved we present here S-boxes which have a very low minimum confusion coefficient.

5.2 A Closer Look at the Confusion Coefficient

To understand the exact reason why some (one or more) key guesses result in a smaller confusion coefficient than others and how this is related to F, we concentrate on the connection between k_c, k_g, F, and $\kappa(k_c, k_g)$. Loosely speaking, we are iterating on key guesses influencing the input of F while calculating the confusion coefficient on the measured output of F and being interested in the properties of F. To better address these connections, we split the problem into 2 individual problems.

First, we take a deeper look at the input of F, i.e., $t \oplus k_g$ where $\forall t, k_g \in \mathbb{F}_2^n$ (see Eq. (8)). Clearly, due to the $\oplus$ operation a particular permutation for different key guesses k_g is given. A 2-D representation for $t \oplus k_g$, where k_g is on the horizontal and t on the vertical axis, is given in Fig. 2, where again, for



(a) Preserving w_H (S_1, S_2) and complement (S_4)

(b) Mostly preserving w_H, $d = 2$ (S_3)

(c) Mostly preserving w_H, $d = 4$ (S_5), optimal nl

(d) Mostly preserving w_H, $d = 4$ (S_6), optimal nl

(e) Mostly preserving w_H, $d = 4$ (S_7), optimal nl and δ_F

(f) Mostly preserving w_H, $d = 4$ (S_8), optimal nl and δ_F

Fig. 1. Confusion coefficients

simplicity reasons, $t, k_g \in \mathbb{F}_2^4$. In this figure we furthermore group $t \oplus k_g$ into 4 boxes $(n \times n)$ together, each containing 4×4 values: blue (B_0): $t \oplus k_g \in [0,3]$, yellow (B_1): $t \oplus k_g \in [4,7]$, green (B_2): $t \oplus k_g \in [8,11]$, and red (B_3): $t \oplus k_g \in [12,15]$. Using this color representation we can easily see 4 different permutations $\pi_0, \pi_1, \pi_2, \pi_3$ applied on $(B_0\ B_1\ B_2\ B_3)$. More precisely, when considering a column representation[1] among the key guesses k_g, we have:

- for $k_g \in [0,3]$: no permutation ($\pi_0 = \left(\begin{smallmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{smallmatrix}\right)$),
- for $k_g \in [4,7]$: pairwise swap of elements in each half of matrix ($\pi_1 = \left(\begin{smallmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \end{smallmatrix}\right)$),
- for $k_g \in [8,11]$: additionally reverse ordering of elements ($\pi_2 = \left(\begin{smallmatrix} 0 & 1 & 2 & 3 \\ 2 & 3 & 0 & 1 \end{smallmatrix}\right)$),
- for $k_g \in [12,15]$ additionally a pairwise swap of elements in each half of matrix ($\pi_3 = \left(\begin{smallmatrix} 0 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 \end{smallmatrix}\right)$).

Moreover, as highlighted by the zoom in on each box, within each box (i.e., $B_i, 0 \le i \le 3$) we have the same permutations $\pi_0, \ldots, \pi_3$ on the 4 column entries. Note that the order of permutations is equivalent for each box, or in other words, regardless of the color and position of the box the same permutation is applied. More formally, let $b_{ij} \in [4i, 4i+3]^4$ (for $0 \le i,j \le 3$) denote the columns within B_i, then b_{ij} equals π_j applied on the column vector $(4i\ 4i+1\ 4i+2\ 4i+3)$.

Second, we examine the expression of the confusion coefficient in Eq. (11) itself. Recall from Eq. (8), $y_{k_g,t} = y(k_g,t) = w_H(F(k_g+t))$. Let

$$y_{k_g} = (y(k_g,0), y(k_g,1), \ldots, y(k_g,2^n-1))$$

denote the vector of hypotheses for one key guess k_g over all texts t. Referring to Fig. 2, y_{k_g} relates to one column before its application to F and w_H. The confusion coefficient can be rewritten as

$$\kappa(k_c,k_g) = \frac{1}{4}\left\| y_{k_c} - y_{k_g} \right\|_2^2 \tag{19}$$

with $\|\cdot\|_2$ being the Euclidean norm. Let us recall that we are especially interested in $min_{k_g \neq k_c} \kappa(k_c,k_g)$. Moreover, the elements of $y_{k_c} - y_{k_g}$ are in $[-4,4]$. Now, as Eq. (19) considers not only the difference but its squared values, we may conjecture that the minimum value is most likely reached when the elements of $y_{k_c} - y_{k_g}$ are in $[-1,1]$, which is discussed in more detail and confirmed using several lightweight S-boxes in Appendix A. Roughly speaking, one difference of ±2 is equivalent to 4 changes with ±1 and so on.

Now let us put the observations of both parts together. Our previous findings about the permutations can be straightforwardly applied to the Hamming weight of the output of F. Let us assume w.l.o.g. $k_c = 0$, then for $k_g = 4i + j$ (with $0 \le i,j \le 3$) we have

$$y_{k_g} = \pi_i \left(\pi_j \begin{bmatrix} y_{0,0} \\ y_{0,1} \\ \vdots \\ y_{0,3} \end{bmatrix}^T \pi_j \begin{bmatrix} y_{0,4} \\ y_{0,5} \\ \vdots \\ y_{0,7} \end{bmatrix}^T \pi_j \begin{bmatrix} y_{0,8} \\ y_{0,9} \\ \vdots \\ y_{0,11} \end{bmatrix}^T \pi_j \begin{bmatrix} y_{0,12} \\ y_{0,13} \\ \vdots \\ y_{0,15} \end{bmatrix}^T \right)^T, \tag{20}$$

[1] Note that we also have the same permutations on the row entries, however, we are interested in particular in a column representation as they reflect the key hypotheses.

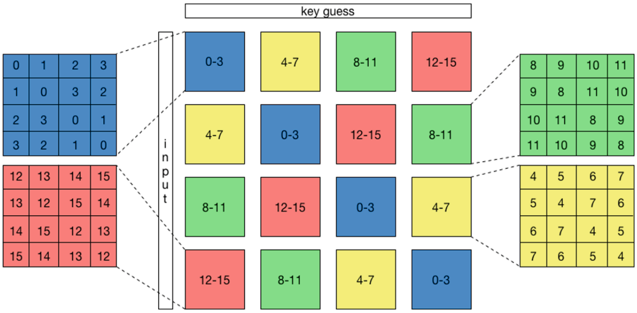


Fig. 2. Illustration of permutations of $t \oplus k_g$ $\forall t, k_g \in \mathbb{F}_2^4$ (input of F) (Color figure online)

with $y_0 = (y_{0,0}, y_{0,1}, \ldots, y_{0,15})$ and $(\cdot)^T$ denoting the transpose. Thus, we are looking for a function F such that the distance

$$\left\| y_0 - \pi_i \left(\pi_j \begin{bmatrix} y_{0,0} \\ y_{0,1} \\ \vdots \\ y_{0,3} \end{bmatrix}^T \pi_j \begin{bmatrix} y_{0,4} \\ y_{0,5} \\ \vdots \\ y_{0,7} \end{bmatrix}^T \pi_j \begin{bmatrix} y_{0,8} \\ y_{0,9} \\ \vdots \\ y_{0,11} \end{bmatrix}^T \pi_j \begin{bmatrix} y_{0,12} \\ y_{0,13} \\ \vdots \\ y_{0,15} \end{bmatrix}^T \right)^T \right\|_2^2 \qquad (21)$$

is as small as possible for any $\pi_i, \pi_j \in \{\pi_0, \pi_1, \pi_2, \pi_3\}$.

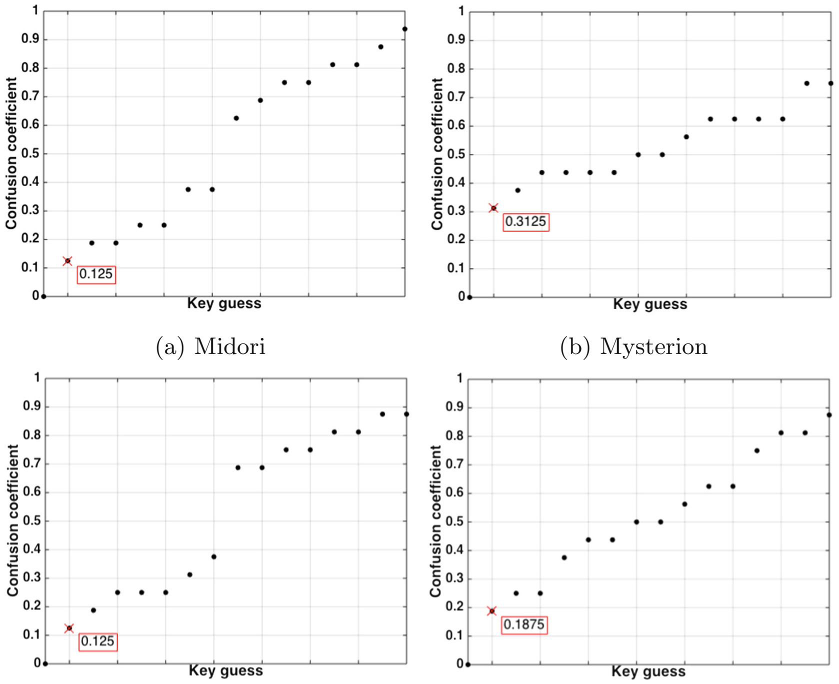
This finding indicates that the order of the Hamming weight of the output of F plays a significant role. To be more precise, the minimum confusion coefficient may depend not only on the distribution of values along the 4 boxes (Example 4), but also on the order within each box (Example 5).

Example 4. *Note that the elements of y_{k_g} follow a binomial distribution due to the application of w_H. Therefore,* 0 *and* 4 *occur once,* 1 *and* 3 *occurs four times, and* 2 *six times. In order to reach a mininum squared Euclidean distance in Eq.* (21) *a natural strategy seems to be to distribute the values broadly among the 4 sets* $[4i, 4i + 3]$ *and to have a small difference between the values in one set. Let us consider the S-box of Midori [18] and Mysterion [19]. From Table 2 one can observe that for Midori we have the following sets: 2,2,3,2 – 3,3,4,3 – 1,2,1,2 – 0,1,1,2. So, the maximal distance between values is 2. Moreover, the first three sets only contain 2 different values and the last has 3. On the contrary, when looking at Mysterion (0,1,2,3 – 2,4,3,2 – 1,3,1,2 – 2,1,3,2), the structure looks less balanced. In particular, the maximal distance is 3 and we have always 3 different values within a set. When comparing the confusion coefficient in Fig. 3 we can observe that Midori has a much smaller minimum confusion coefficient and is thus more SCA resilient.*

Example 5. *Let us consider the S-box of KLEIN [20] and a small modification (S_9) in which we swap* $F(1)$ *with* $F(3)$ *(see Table 2). Note that both functions*

Table 2. Known S-boxes and one modification of KLEIN, $(x)w_H(x)$

S-box	(0)0	(1)1	(2)1	(3)2	(4)1	(5)2	(6)2	(7)3	(8)1	(9)2	(10)2	(11)3	(12)2	(13)3	(14)3	(15)4
Midori	(12)2	(10)2	(13)3	(3)2	(14)3	(11)3	(15)4	(7)3	(8)1	(9)2	(1)1	(5)2	(0)0	(2)1	(4)1	(6)2
Mysterion	(0)0	(1)1	(6)2	(11)3	(10)2	(15)4	(14)3	(3)2	(2)1	(7)3	(4)1	(12)2	(9)2	(8)1	(13)3	(5)2
KLEIN	(7)3	(4)1	(10)2	(9)2	(1)1	(15)4	(11)3	(0)0	(12)2	(3)2	(2)1	(6)2	(8)1	(14)3	(13)3	(5)2
S_9	(7)3	(9)2	(10)2	(4)1	(1)1	(15)4	(11)3	(0)0	(12)2	(3)2	(2)1	(6)2	(8)1	(14)3	(13)3	(5)2



(a) Midori

(b) Mysterion

(c) KLEIN

(d) Modified KLEIN S_9

Fig. 3. Confusion coefficients of Midori, Mysterion, KLEIN, and KLEIN with a small modification

consist of the same values among the sets: 3,1,2,2 – 1,4,3,0 – 2,2,1,2 – 1,3,3,2. For both $\min_{k_g \neq k_c} \kappa(k_c, k_g)$ *is reached for* $k_g = 11$*, thus* $\pi_1 = 2$ *and* $\pi_2 = 2$*. However, as Fig. 3d shows, for KLEIN we have* $\min_{k_g \neq k_c} \kappa(k_c, k_g) = 0.125$*, whereas* $\min_{k_g \neq k_c} \kappa(k_c, k_g) = 0.185$ *for* S_9*, which relates to a squared Euclidean distance (see Eq. (21)) of 8 and 12, respectively.*

Furthermore, in Appendix A we investigate several lightweight S-boxes in terms of minimum confusion coefficient and provide empirical evaluations. Note that, a preliminary study showing the difference of some lightweight S-boxes has been conducted in [21][2]. Our extended results in Appendix A theoretically and empirically confirm [21]. Moreover, the appendix provides details about the minimum Euclidean distance and the permutations π_i, π_j. Additionally, we take a deeper look at the expression of $y_{k_c} - y_{k_g}$ for the key hypothesis k_g that results in the smallest confusion coefficient (i.e., $\arg\min_{k_c \neq k_g} \kappa(k_c, k_g)$). We discover that for S_5 and the S-box proposed in [17], which has optimal properties of the confusion coefficient while holding optimal differential properties, the difference $\|y_{k_c} - y_{k_g}\|_2^2$ has a special particular structure, which is not observed for any other investigated 4-bit S-box.

Concluding, we derived specific criteria influencing the side-channel resistance (in particular in Eq. (21) and our findings in Appendix A) that could be exploited to optimize and find S-boxes in terms of side-channels resistance in future work – especially when adapted for $n > 4$.

6 Conclusions

In this paper, we prove a number of bounds between various cryptographic properties that can be related also with the side-channel resilience of a cipher. Our results confirm some well known intuitions that having an S-box more resilient against SCA will make it potentially more vulnerable against classical cryptanalyses. However, they also show that for the usual sizes of S-boxes, this weakening is moderate and trade-offs are then possible.

Since in this work we concentrated in our practical investigations on the Hamming weight model, in the future we plan to explore possible trade-offs for the Hamming distance model and to extend our (empirical) analysis to larger S-boxes using the theoretical findings in this paper.

Acknowledgments. This work has been supported in part by Croatian Science Foundation under the project IP-2014-09-4882. The parts of this work were done while the third author was affiliated with KU Leuven, Belgium.

A Investigation of Known S-Boxes

We already described properties of the S-boxes of KLEIN, Midori, and Mysterion showing that Midori and KLEIN have both $\min_{k_g \neq k_c} \kappa(k_c, k_g) = 0.125$, whereas for Mysterion it equals 0.3125. Thus, the side-channel resistance of Mysterion is much smaller than that of KLEIN and Midori. Table 3 shows properties of several well-known S-boxes, where π_i and π_j indicate the permutations (see Eq. (21)) for the smallest squared Euclidean distance ($\min \|\cdot\|_2^2$) and thus smallest confusion coefficient ($\min \kappa(k_c, k_g)$).

Note that the squared Euclidean distance should not serve as a new metric as it is in direct relation with the confusion coefficient, but its stated values

[2] Note that, in [22] the authors compared S-boxes regarding another (not normalized) version of the confusion coefficient and derived that their version is not aligned with their empirical results.

Table 3. S-box properties of known ciphers

Name	π_i	π_j	$\min \lVert \cdot \rVert_2^2$	$\min \kappa(k_c, k_g)$	$var(\kappa(k_c, k_g))$	Pw_H
KLEIN [20]	2	2	8	0.125	0.093	6
Midori [18]	0	1	8	0.125	0.096	4
Midori 2 [18]	2	3	16	0.250	0.059	5
Mysterion [19]	2	1	20	0.312	0.034	5
NOEKEON [23]	3	0	12	0.188	0.056	8
Piccolo [24]	1	0	24	0.375	0.028	4
PRESENT [25]	2	1	16	0.25	0.058	3
PRINCE [26]	1	3	12	0.188	0.059	3
RECTANGLE [27]	3	0	16	0.250	0.056	3
SKINNY [28]	1	0	16	0.250	0.037	8
S-box in [17]	0	1	8	0.125	0.104	4

should rather provide information how far y_{k_g} is apart from y_{k_c} in terms of the squared Hamming weight values. Further, as used in [17], we give $var(\kappa(k_c, k_g))$, where the higher the variance, the higher the side-channel resistance. Finally, we specify the Hamming weights preserved (Pw_H). One can observe that Piccolo has the highest minimum value of the confusion coefficient (and the highest minimum squared Euclidean norm) and thus its side-channel resistance is the lowest among the evaluated one. Next, there is Mysterion followed by SKINNY, RECTANGLE, PRESENT, and Midori 2 that all have the same minimum value of the confusion coefficient, but different variances thereof. Then, we have NOEKEON and PRINCE. The lowest minimum confusion coefficient is reached by KLEIN, Midori, and the S-box proposed in [17], which has been found under the constraint of optimal differential properties and the lowest confusion coefficient by using genetic algorithms. Interestingly for the latter one, Fig. 4 illustrates that for one key guess $\kappa(k_c, k_g) = 1$, which we do not observe for any other known S-boxes with optimal differential properties. Moreover, it corresponds to the confusion coefficient of S_5.

Additionally, we take a deeper look at the expression of $y_{k_c} - y_{k_g}$ for the key hypothesis k_g that results in the smallest confusion coefficient and we are interested if the elements in $|y_{k_c} - y_{k_g}|$ are in $[-1, 1]$ (see remark in Subsect. 5.2). Our investigations show that this does not hold for S-boxes with $\kappa(k_c, k_g) \geq 0.25$, but for the ones which are most side-channel resistant. In particular, Midori 2, Mysterion, PRESENT, RECTANGLE, and SKINNY contain two absolute difference of 2 (resulting in a Euclidean distance of 4), whereas Piccolo even has 4 absolute differences of 2. However, we could not observe any absolute difference greater than 2. On the contrary KLEIN, Midori, NOEKEON, PRINCE, and the S-box in [17] only contain absolute differences of one, which is thus equivalent to the Euclidean distance.

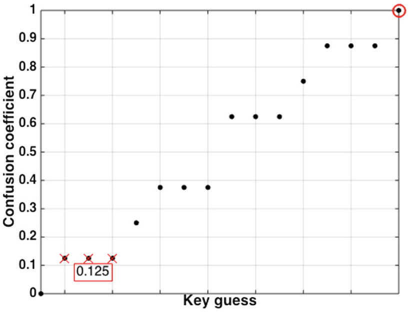


Fig. 4. Confusion coefficient of S-box in [17]

When considering the sum of differences among the 4 sets $[4s, 4s+3]$ for $0 \leq s \leq 3$, we observed interesting distinctions. In particular, let us denote

$$\Delta_s = \left\| \begin{bmatrix} y_{0,4s} \\ y_{0,4s+1} \\ \vdots \\ y_{0,4s+3} \end{bmatrix} - \pi_i \left(\pi_j \begin{bmatrix} y_{0,4s} \\ y_{0,4s+1} \\ \vdots \\ y_{0,4s+3} \end{bmatrix} \right) \right\|_2^2, \tag{22}$$

with π_i and π_j being the permutation resulting in the minimum confusion coefficient.

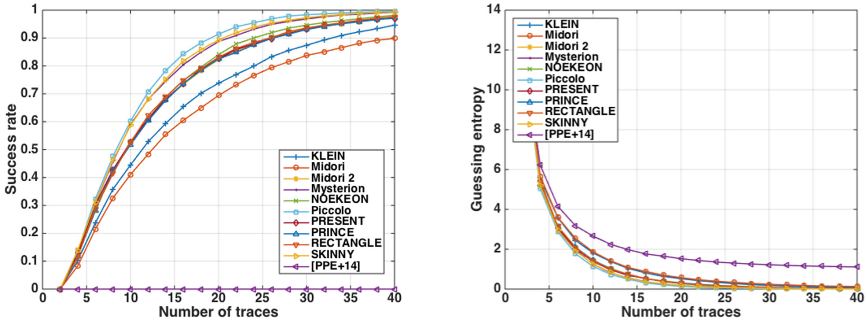
Table 4 highlights that only for the S-box in [17] we have the same difference among all four sets. Note that, in future work this property may additionally help to detect and find S-boxes with better side-channel resistance for $n > 4$.

Table 4. Δ-property (Eq. (22)) of the most resilient known S-boxes

Name	Δ_0	Δ_1	Δ_2	Δ_3
KLEIN	1	3	3	1
Midori	2	2	0	4
NOEKEON	2	4	2	4
PRINCE	4	2	4	2
S-box in [17]	2	2	2	2

Finally, an empirical evaluation of the studied S-boxes is given in Fig. 5. To be reliable we conducted 5 000 independent simulation experiments (SNR = 2) with random secret keys k_c and texts t. Figure 5a shows the first-order success rate (SR), i.e., the empirical probability that the correct secret key is exclusively found. As found due to the properties of the confusion coefficient, the S-box of Piccolo is the weakest, finding the correct key with a SR of 0.9 using

20 measurement traces, whereas KLEIN and Midori require around 35 and 40 traces to reach SR = 0.9. Since the S-box in [17] does not exclusively find the correct key and thus has a SR = 0, we additionally plot the guessing entropy [29] in Fig. 5b which confirms our findings that at least 2 key guesses have to be made.



(a) Success rate

(b) Guessing entropy

Fig. 5. Empirical evaluation of known S-boxes

References

1. Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL cipher. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 81–91. Springer, Heidelberg (1993). doi:10.1007/3-540-47555-9_7
2. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer, Heidelberg (1991). doi:10.1007/3-540-38424-3_1
3. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security). Springer-Verlag New York Inc., Secaucus (2007)
4. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 529–545. Springer, Heidelberg (2006). doi:10.1007/11935308_38
5. Heuser, A., Rioul, O., Guilley, S.: A theoretical study of kolmogorov-smirnov distinguishers. In: Prouff, E. (ed.) COSADE 2014. LNCS, vol. 8622, pp. 9–28. Springer, Cham (2014). doi:10.1007/978-3-319-10175-0_2
6. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28632-5_2
7. Fei, Y., Luo, Q., Ding, A.A.: A statistical model for DPA with novel algorithmic confusion analysis. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 233–250. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33027-8_14

8. Carlet, C.: Vectorial boolean functions for cryptography. In: Crama, Y., Hammer, P.L. (eds.) Boolean Models and Methods in Mathematics, Computer Science, and Engineering, 1st edn, pp. 398–469. Cambridge University Press, New York (2010)
9. Nyberg, K.: On the construction of highly nonlinear permutations. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 92–98. Springer, Heidelberg (1993). doi:10.1007/3-540-47555-9_8
10. Chabaud, F., Vaudenay, S.: Links between differential and linear cryptanalysis. In: Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 356–365. Springer, Heidelberg (1995). doi:10.1007/BFb0053450
11. Nyberg, K.: Perfect nonlinear S-boxes. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 378–386. Springer, Heidelberg (1991). doi:10.1007/3-540-46416-6_32
12. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). doi:10.1007/3-540-48405-1_25
13. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001). doi:10.1007/3-540-44709-1_21
14. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, Heidelberg (2006). ISBN 0-387-30857-1. http://www.dpabook.org/
15. Guilley, S., Heuser, A., Rioul, O.: A key to success. In: Biryukov, A., Goyal, V. (eds.) INDOCRYPT 2015. LNCS, vol. 9462, pp. 270–290. Springer, Cham (2015). doi:10.1007/978-3-319-26617-6_15
16. Thillard, A., Prouff, E., Roche, T.: Success through confidence: evaluating the effectiveness of a side-channel attack. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 21–36. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40349-1_2
17. Picek, S., Papagiannopoulos, K., Ege, B., Batina, L., Jakobovic, D.: Confused by confusion: systematic evaluation of DPA resistance of various S-boxes. In: Meier, W., Mukhopadhyay, D. (eds.) INDOCRYPT 2014. LNCS, vol. 8885, pp. 374–390. Springer, Cham (2014). doi:10.1007/978-3-319-13039-2_22
18. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: a block cipher for low energy (extended version). Cryptology ePrint Archive, Report 2015/1142 (2015). http://eprint.iacr.org/
19. Journault, A., Standaert, F.X., Varici, K.: Improving the security and efficiency of block ciphers based on LS-designs. Codes Crypt. Des. **82**(1–2), 495–509 (2016)
20. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: a new family of lightweight block ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012). doi:10.1007/978-3-642-25286-0_1
21. Heuser, A., Picek, S., Guilley, S., Mentens, N.: Side-channel analysis of lightweight ciphers: does lightweight equal easy? Cryptology ePrint Archive, Report 2017/261 (2017). http://eprint.iacr.org/2017/261
22. Lerman, L., Markowitch, O., Veshchikov, N.: Comparing Sboxes of ciphers from the perspective of side-channel attacks. IACR Cryptology ePrint Archive 2016/993 (2016)
23. Daemen, J., Peeters, M., Assche, G.V., Rijmen, V.: Nessie proposal: the block cipher Noekeon. Nessie submission (2000). http://gro.noekeon.org/

24. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: *Piccolo*: an ultra-lightweight blockcipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 342–357. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23951-9_23
25. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74735-2_31
26. Borghoff, J., et al.: PRINCE – a low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34961-4_14
27. Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. Sci. Chin. Inf. Sci. **58**(12), 1–15 (2015)
28. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. Cryptology ePrint Archive, Report 2016/660 (2016). http://eprint.iacr.org/2016/660
29. Standaert, F., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks (extended version). IACR Cryptology ePrint Archive 2006/139 (2006)

A Practical Chosen Message Power Analysis Approach Against Ciphers with the Key Whitening Layers

Chenyang Tu[1,2], Lingchen Zhang[1,2](✉), Zeyi Liu[1,2,3], Neng Gao[1,2], and Yuan Ma[1,2]

[1] State Key Laboratory of Information Security,
Institute of Information Engineering, CAS, Beijing, China
{tuchenyang,zhanglingchen,liuzeyi,gaoneng,mayuan}@iie.ac.cn
[2] Data Assurance and Communication Security Research Center, CAS, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China

Abstract. The key whitening is a technique intended to enhance the strength of a block cipher. Although some research work involves DPA attacks against the key whitening layer in the compact architecture, there are no literatures dedicated in the influence of the key whitening layers in the loop architecture from the standpoint of DPA. In this paper, we propose a practical chosen message power analysis approach against the loop architecture of ciphers with the key whitening layers, thus proving that the key whitening technique does not enhance the security of ciphers regard to DPA. Our approach follows a reduction strategy: we recover the whitening key in the general cipher with the key whitening layer and reduce other complicated key whitening layers to the general case. In order to further manifest the validity of the new approach, we carry extensive experiments on two ISO standardized ciphers CLEFIA and Camellia implemented in loop architecture on FPGA, and the keys are recovered as expected.

Keywords: DPA · Key whitening · Chosen message · Loop architecture

1 Introduction

Key whitening is a technique intended to enhance the strength of a block cipher by adding key-relevant operations on plaintext and ciphertext without major changes to the algorithm [15,16]. The key whitening layer consists of steps that combine the data with portions of the key before the first round and after the last

C. Tu—This work is supported by the National Natural Science Foundation of China (Nos. 61402470 and 61602476) and National Basic Research Program of China (No. 2013CB338001). Besides, the work is also supported by Youth Innovation Promotion Association, CAS.

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 415–434, 2017.
DOI: 10.1007/978-3-319-61204-1_21

round. The most common operation is XORing or modular adding the whitening key to the plaintext/ciphertext. The key whitening technique is adopted by many block ciphers, such as the ISO standardized Feistel-SP ciphers CLEFIA [2] and Camellia [3], and the lightweight ciphers DESL [15] and PRINCE [16].

Since first proposed by Kocher *et al.* in [1], DPA has proven to be a powerful method of side channel attack against many ciphers. In general, DPA can only deal with a small fraction of the long secret key (*e.g.* several round key bits) through a divide-and-conquer strategy, and its validity is highly dependent on the specific implementation. These traditional cryptographic implementations (*i.e.*, compact architecture [18]) are easily compromised by the conventional DPA, where the hypothesis space of the secret key fraction is only 2^8 or less [17]. For instance, DES, AES and many other ciphers under the compact architecture have shown to be vulnerable to DPA [1,4,7,9,10]. The ciphers with key whitening layers under the same architecture are also easily compromised by DPA, because key whitening layer is implemented independently of the substitution circuit [8].

With the advancement of circuit industry within recent years, a new architecture, *i.e.* loop architecture [18], is proposed, which computes a single round function in one clock cycle. Ciphers under the loop architecture are adopted in the higher computation speed and higher throughput application scenarios. Therefore, the capability of the loop implementation against the side channel attack has attracted researchers' great attention. It has been proved that the conventional DPA needs much more power traces to analyze ciphers under loop architecture with very high computational complexity [13]. In order to deal with the loop architecture efficiently and practically, the adversary usually launches the chosen message DPA [12] instead.

However, it is quite a challenge to launch the DPA methodology against the loop hardware implementations of ciphers with the key whitening layers. The key whitening layer is usually implemented within the first or last round in the loop architecture, which can increase the difficulty of the DPA methodology. In this case, the power consumption of the whitening operation is hard to be recognized from power traces, because the intermediate result of the whitening operation does not appear in registers or on bus as the case in [8]. Following the chosen message DPA [12], the adversary can only get the equivalent key (*i.e.* the value of "the round key add/xor the whitening key" as a unity) on the first/last round, but he would not be capable to directly determine either the whitening key or the round key.

Unless the adversary is able to directly obtain the whitening key or the round key in some way, the adversary has to peel off the first round with the equivalent key and perform DPA with an adaptive manner on the subsequent rounds, which means the adversary must solve one more round to deal with the key whitening layer. Generally, the cost of performing DPA in such way increases 50% to 100% than that without key whitening layer. In order to deal with the key whitening layer without increasing the DPA cost, the core issue is how to reveal either the whitening key or the round key directly in the first/last round. To the best of our knowledge, no research about this issue has been reported in the literature.

In this paper, we propose a practical chosen message DPA approach to recover the whitening key through a reduction strategy. First, by fully exploiting the relationship between the round key and the whitening key, we recover the whitening key in the general cipher with the key whitening layer. Then we successfully reduce other complicated key whitening layers to the general case. As a result, we show that the key whitening technique does not enhance the security of ciphers from the standpoint of DPA.

We take the Feistel-SP ciphers with the key whitening layers as an instance, due to the most comprehensive cases of the key whitening layers in these ciphers (*i.e.*, the key whitening operation on the left branch, on the right branch, and on both branches). According to the relationship in the round function, our approach can launch chosen message DPA to efficiently reveal the whitening key on the left branch. When the whitening key is on the right branch, we are able to reduce the recovery of the whitening key on the right branch to the left branch case through an adaptive chosen message manner. When the whitening keys are on both branches, we can reduce the recovery of the whitening keys to the left branch case and the right branch case respectively. Furthermore, we perform extensive experiments on two ISO standardized ciphers CLEFIA and Camellia with loop FPGA implementations. Experimental results show that all bits of the keys in both ciphers can be recovered as expected.

The remainder of this paper is organized as follows. In Sect. 2, the preliminaries are briefly described. Section 3 illustrates the practical chosen message power analysis method on Feistel-SP ciphers with the key whitening layers. Section 4 elaborates the practical attacks on two loop FPGA implementations of CLEFIA-128 and Camellia-128 in order to prove the effectiveness of our approach. We discuss and summarize the paper in Sect. 5.

2 Preliminaries

2.1 The Compact and Loop Architecture

The hardware implementations of ciphers usually follow two architectures, compact architecture and loop architecture [18], in order to adapt to different application scenarios. In the embedded application scenario, the size, power consumption, and cost of the cryptographic device are tightly restrained. On the other hand, in the higher computation speed and higher throughput application scenario, the performance and efficiency of the cryptographic device are the most important indicators. The compact architecture usually takes several clock cycles to accomplish a single round computation of the cryptographic algorithm, such as reusing the single substitution circuit (*e.g.* Sbox) several times as a subloop instead of using their duplications concurrently. Therefore, the compact architecture, which sacrifices performance to less circuit components, is often applied to the embedded cryptographic device, such as smart cards and wireless sensor nodes.

On the other hand, the loop architecture defines the round function of the cryptographic algorithm as several consecutive operations, which means that a single round is computed in one clock cycle. Compared to the compact architecture, the loop architecture, which has higher throughput or less calculation time, is usually applied to the higher computation speed and higher throughput application scenario, such as the hardware security module in the cloud computing environment, the instant messenger system, the network authentication system and the network routing device. The loop architecture is implemented in ASIC or FPGA chips in the hardware security module, with the advancement of circuit industry within recent years.

2.2 DPA on Key Whitening Layer

The key whitening layers can be compromised by DPA, as long as the adversary is able to locate the power consumption of the whitening operation on power traces. In the compact scenario, the power consumption of the whitening operation is convenient to be observed due to the independent implementation of the key whitening layer. Consequently, the adversary can reveal the whitening key through direct analysis of the power consumption of the whitening operation [8,11]. Unfortunately, the above strategy seems hard to apply in the loop scenario, because the whitening operations are implemented within the first/last round. It leads to much lower signal-to-noise of the slight power consumption which is too difficult to be detected. In this case, the adversary can only get the equivalent key (*i.e.* the value of "the round key adding the whitening key" as a unity) in DPA, but he would not be capable to directly determine either the whitening key or the round key. Therefore, how to reveal the whitening key or the round key in the loop architecture remains an open problem.

2.3 Feistel-SP Structure

The Feistel network is one of the most famous architectures in symmetric cryptography. According to the classifications of [5], the Feistel network has several derivatives, namely, the unbalanced Feistel network, the alternating Feistel network, the numeric Feistel network, and the famous type-1, type-2, and type-3 Feistel networks. Many practical block ciphers utilize the Feistel networks including DES (plain), Skipjack (unbalanced), BEAR/LION (alternating), CAST (type-1), CLEFIA(type-2) and MARS(type-3).

A typical SP type function often consists of three operations, *i.e.*, subkey addition, substitution, and permutation. In the subkey addition, a subkey is XORed to the state. The substitution is applied by Sbox-like non-linear bijection. In the permutation, a linear bijection (generally an MDS multiplication) is performed. Let $S_1, S_2, \cdots, S_t : \{0,1\}^s \longrightarrow \{0,1\}^s$ be non-linear bijections, $P : \{0,1\}^{st} \longrightarrow \{0,1\}^{st}$ be a linear bijection, $k = (k_1, k_2, \cdots, k_t)$ is the round key, then the round function $F : \{0,1\}^{st} \times \{0,1\}^{st} \longrightarrow \{0,1\}^{st}$ of SP type is defined by $F(x,k) = P(S_1(x_1 \oplus k_1), S_2(x_2 \oplus k_2), \cdots, S_t(x_t \oplus k_t))$. The notations s and t represent the size of the non-linear bijection and the number of the non-linear bijections, respectively.

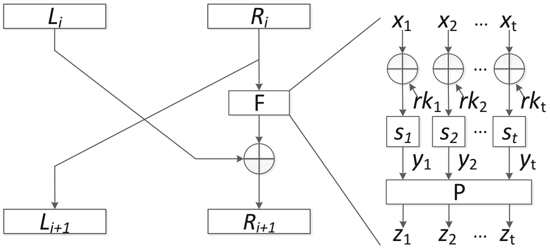


Fig. 1. The typical round function of the Feistel-SP structure

In the Feistel network, the core of the underlying round function is referred to as the F-function. In order to combine the advantages of SPN structures, the Feistel-SP ciphers use well-designed SPN functions as the F-functions. The typical round function of the Feistel-SP structure is shown in Fig. 1.

3 The Design of Our Approach

In this section, we describe the details of our practical chosen message DPA method on the loop architecture of Feistel-SP ciphers with the key whitening layers. Firstly, sharing the similar idea of chosen message DPA [12], we put forward the chosen message DPA which is also suitable for the case of Feistel-SP structure without the key whitening layer. Secondly, we analyze the difficulty to reveal the whitening key in the loop architecture directly. Then we propose an efficient approach to recover the whitening key through a reduction strategy. More precisely, we show how to recover the whitening key on the left branch, and reduce the recovery of the whitening key on the right branch to the left branch case through an adaptive chosen message manner. Combining these two strategies, we manage to perform practical DPA on loop architecture of Feistel-SP structure with the key whitening layer.

3.1 The Chosen Message DPA on the Loop Architecture of Feistel-SP

The typical loop implementation of Feistel-SP is shown in Fig. 1. Let L_i and R_i denote the left and right branches of the i-th round input respectively, and the size of the Sbox is 8-bit. The right branch R_i can be further split into t 8-bit cells, namely, $R_i = x_1||x_2||\cdots||x_t$. Each 8-bit cell x_j is first XORed with the corresponding 8-bit round key rk_j, then all t cells are processed with t parallel Sboxes $S_1, S_2, \cdots, S_t$[1]. Let $y_1||y_2||\cdots||y_t$ denote the output of the Sbox layer, the linear permutation P (normally multiplication with an MDS matrix) updates the state $y_1||y_2||\cdots||y_t$, and $z_1||z_2||\cdots||z_t$ is the output. The right branch of the i-th round output R_{i+1} is then calculated by XORing L_i and the output of P,

[1] The Sboxes can be identical or distinct.

while the left branch L_{i+1} is updated by directly assigning the value of R_i. The above procedures can be described as

$$\begin{aligned} L_{i+1} &= R_i, \\ R_{i+1} &= F(R_i, rk) \oplus L_i \\ &= P(S_1(x_1 \oplus rk_1), S_2(x_2 \oplus rk_2), \cdots, S_t(x_t \oplus rk_t)) \oplus L_i. \end{aligned} \tag{1}$$

In the scenario of loop architecture, Feistel-SP treats the round function as several consecutive operations, thus no intermediate result except R_{i+1} is written into registers in the loop implementations. The power consumption of y_j is much less than R_{i+1} and hard to be observed. Therefore, we are only able to attack at this point when the data R_{i+1} is being written into registers as shown in Fig. 2.

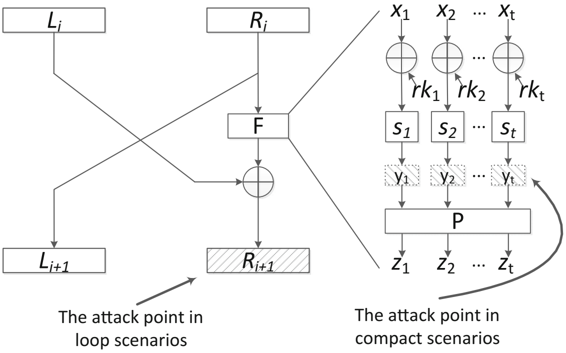


Fig. 2. Different DPA attack points of Feistel-SP

As shown in Fig. 2, regarding a specific cell y_j, the linear permutation P can be represented as follows:

$$\begin{aligned} P(y_1, y_2, \cdots, y_{j-1}, y_j, y_{j+1}, \cdots, y_t) = \ & P(0, 0, \cdots, 0, y_j, 0, \cdots, 0) \oplus \\ & P(y_1, y_2, \cdots, y_{j-1}, 0, y_{j+1}, \cdots, y_t) \\ = \ & WP^j \oplus CP^j, \end{aligned}$$

where the superscript j indicates the function focusing on the cell y_j, and WP^j and CP^j denote the two components of the right half of the equation respectively. The above equation can be rewritten in byte-wise form as follows:

$$P_{[1]}||P_{[2]}||\cdots||P_{[t]} = (WP^j_{[1]}||WP^j_{[2]}||\cdots||WP^j_{[t]}) \oplus (CP^j_{[1]}||CP^j_{[2]}||\cdots||CP^j_{[t]})$$

where the subscript $[k]$ indicates the k-th byte output, and $WP^j_{[k]}$ and $CP^j_{[k]}$ denote the k-th bytes of WP^j and CP^j respectively. If we fix the values of all y_k where $k \neq j$, and keep y_j as a variable, then $WP^j_{[j]}$ can be seen as a function of

y_j and $CP^j_{[j]}$ is a constant. For convenience, we use $P'_{[j]}$ and $mask_{[j]}$ to represent $WP^j_{[j]}$ and $CP^j_{[j]}$ respectively.

Thus, the adversary chooses the specific byte of plaintext message, which corresponds the j-th byte of the target intermediate variable, while fixing other bytes of the plaintext message. The target byte is dependant on two unknown one-byte constant parameters, *i.e.*, the subkey rk_j and the $mask_{[j]}$ generated by P. Therefore, the size of guessed parameters from the whole round key is decreased to a pair of 8-bit values, *i.e.*, the hypothesis space of the secret value falls to 2^{16}, and the input space of the plaintext message is decreased to 2^8, which is suitable for practical DPA. Consequently, by alternately choosing the corresponding plaintext message byte for all possible positions, we can use the DPA attacking model shown in Fig. 3 to launch DPA and recover all t bytes of the round key (for the sake of simplicity, we assume the size of the round key is 32 bits).

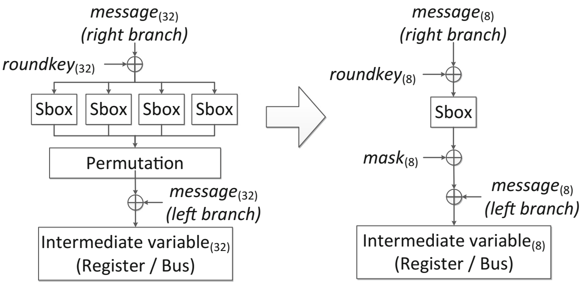


Fig. 3. DPA model of Feistel-SP in loop scenario

3.2 The Difficulty to Reveal the Whitening Key in the Loop Scenario

The whitening keys are generally used before the first round and after the last round. After the key whitening operations, the inputs (outputs) to the first (last) round are covered by the unknown whitening key from the plaintexts (ciphertexts). It seems that such operations would increase the size of unknown parameters, and raise the difficulty to launch a DPA attack. However, since the encryption and decryption of Feistel-SP ciphers follow similar procedures and both the pre-whitening and the post-whitening keys are almost equivalent from the perspective of DPA, we only discuss the encryption procedure in the first round.

Let ML (resp. MR) denote the left (resp. right) message branch, and wkL (resp. wkR) denote the left (resp. right) whitening key. As shown in Fig. 4, the whitening keys can be applied on the left branch, the right branch or both branches, which correspond to three types of whitening operations.

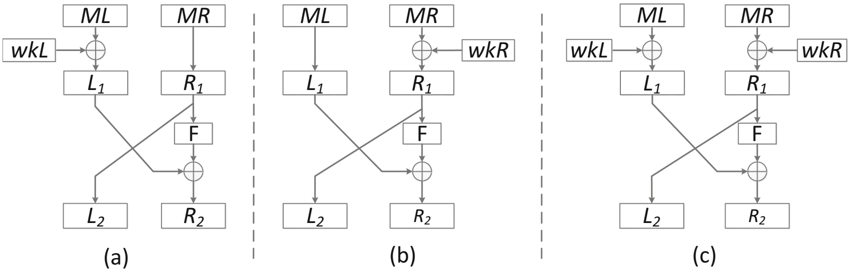


Fig. 4. The whitening key on (a) left branch (b) right branch (c) both branches

There are two main difficulties to apply DPA to reveal the whitening key in loop scenario. The first difficulty is that the power consumption of the whitening operation is hard to detect from power traces similar as Sect. 3.1. In the loop hardware implementation, the whitening operation and the round key addition operation are usually combined as one operation ($i.e.$ $Message \oplus wk \oplus rk$), which is implemented by 3-input XOR gate or LUT. More precisely, there is no standalone whitening operation in the Feistel-SP computing procedure, thus the power consumption of the whitening operation is hard to detect. Therefore, the existing method against the whitening key as mentioned in [8] is not suitable.

Moreover, although we could choose R_{i+1} as the attack point, the whitening key is difficult to be separated from the round key and other intermediate variables by DPA. We assume that there are whitening keys on both branches. Due to the effect of whitening keys wkL and wkR, the DPA attacking model of Feistel-SP with whitening key in loop scenario is shown in Fig. 5(a). For the loop scenario, we use $rk \oplus wkR$ as the equivalent key and $mask \oplus wkL$ as the equivalent mask, thus the model is changed from Fig. 5(a) to (b). However, although we can recover the equivalent key and the equivalent mask by DPA, we are unable to directly determine the values of the whitening keys from the equivalent key and the equivalent mask.

3.3 Recovery of wkL

When we choose R_{i+1} as the attack point, the main drawback for DPA is the difficulty to separate the whitening key from the equivalent key and the equivalent mask. Luckily, we can achieve this goal through a reduction strategy. More precisely, we can efficiently recover the whitening key on the left branch wkL, and reduce the recovery of the whitening key on the right branch to the left branch case.

The recovery of wkL is based on the full exploitation of the complex relationship between the equivalent key and the equivalent mask with a chosen message DPA method. Hereafter we use subscript $[i]$ $(1 \leq i \leq t)$ to indicate the i-th byte of the corresponding variable or the output of one function, and use notation rk_j $(j \geq 1)$ to represent the round key in the j-th round of Feistel-SP. According to

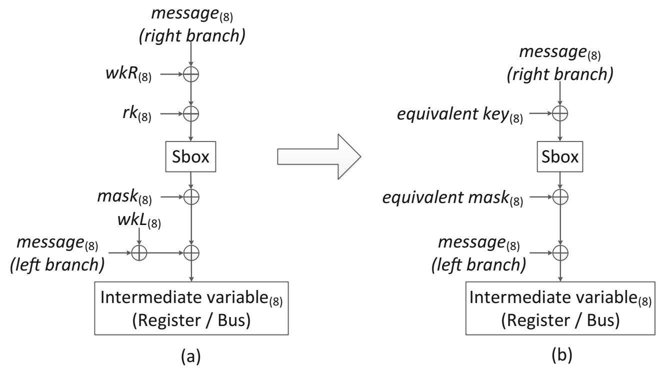


Fig. 5. Chosen message DPA model of Feistel-SP whitening key in the loop scenario

Fig. 4(a), two branches of the first round input $L_1||R_1$ can be described by:

$$L_1||R_1 = (ML \oplus wkL)||MR. \tag{2}$$

Now, we will focus on the attack point R_2 as shown in Fig. 2. Equation 1 can be rewritten as:

$$\begin{aligned} R_2 = &\ F(R_1, rk_1) \oplus L_1 \\ = &\ P(S_1(MR_{[1]} \oplus rk_{1,[1]}), S_2(MR_{[2]} \oplus rk_{1,[2]}), \\ &\ \cdots, S_t(MR_{[t]} \oplus rk_{1,[t]})) \oplus wkL \oplus ML. \end{aligned} \tag{3}$$

where $MR = MR_{[1]}||MR_{[2]}||\cdots||MR_{[t]}$, and $rk_1 = rk_{1,[1]}||rk_{1,[2]}||\cdots||rk_{1,[t]}$.

We focus on the first byte of R_2, and fix all $MR_{[j]}$ $(2 \leq j \leq t)$ to constants, that leads to the constant output of S_j. Thus, Eq. 3 can be rewritten in byte-wise form:

$$\begin{aligned} R_{2,[1]} = &\ F_{[1]}(R_1, rk_1) \oplus L_{1,[1]} \\ = &\ P_{[1]}(S_1(MR_{[1]} \oplus rk_{1,[1]}), S_2(MR_{[2]} \oplus rk_{1,[2]}), \\ &\ \cdots, S_t(MR_{[t]} \oplus rk_{1,[t]})) \oplus wkL_{[1]} \oplus ML_{[1]} \\ = &\ P'_{[1]}(S_1(MR_{[1]} \oplus rk_{1,[1]})) \oplus mask_{1,[1]} \oplus wkL_{[1]} \oplus ML_{[1]} \\ = &\ P'_{[1]}(S_1(MR_{[1]} \oplus rk_{1,[1]})) \oplus MASK_{1,[1]} \oplus ML_{[1]}, \end{aligned} \tag{4}$$

with $ML = ML_{[1]}||ML_{[2]}||\cdots||ML_{[t]}$, $wkL = wkL_{[1]}||wkL_{[2]}||\cdots||wkL_{[t]}$. Moreover, $mask_1 = mask_{1,[1]}||mask_{1,[2]}||\cdots||mask_{1,[t]}$ is the intermediate variable which is generated in the first round. According to Sect. 3.1, $mask_{1,[j]}$ is a byte constant value if all MR_k $(k \neq j)$ are fixed since the round key rk_1 is pre-assigned, and the equivalent mask $MASK_1 = mask_1 \oplus wkL$.

At this time, $R_{2,[1]}$ is highly related to $rk_{1,[1]}$, and $R_{2,[2]}, R_{2,[3]}, \cdots, R_{2,[t]}$ will be treated as noise. Now, we can launch DPA against $R_{2,[1]}$ by enumerating 8-bit $MR_{[1]}$ while fixing other bits of MR and ML. Thus, both 8-bit $rk_{1,[1]}$ and 8-bit

$MASK_{1,[1]}$ are revealed by DPA, where the possible hypotheses space is 2^{16} and the possible input space of random test vector $MR_{[1]}$ is only 2^8. With the same approach, we could analyze $R_{2,[2]}, R_{2,[3]}, \cdots, R_{2,[t]}$ byte by byte, and reveal the values of $rk_{1,[2]}, rk_{1,[3]}, \cdots, rk_{1,[t]}$ and $MASK_{1,[2]}, MASK_{1,[3]}, \cdots, MASK_{1,[t]}$.

According to the complex relationship between rk_1 and $mask_{1,[j]}$, we could iteratively calculate each of $mask_{1,[j]}$ by all bytes of rk_1. More precisely, according to Eq. 4, $mask_{1,[1]}$ is calculated by:

$$\begin{aligned} mask_{1,[1]} = \ & P'_{[1]}(S_1(MR_{[1]} \oplus rk_{1,[1]})) \oplus P_{[1]}(S_1(MR_{[1]} \oplus rk_{1,[1]}), \\ & S_2(MR_{[2]} \oplus rk_{1,[2]}), \cdots, S_t(MR_{[t]} \oplus rk_{1,[t]})). \end{aligned} \tag{5}$$

$mask_{1,[j]}$ $(j \in [2,t])$ is calculated similarly. Then the values of all $wkL_{[k]}$ $(k \in [1,t])$ is iteratively calculated by $wkL_{[k]} = MASK_{1,[k]} \oplus mask_{1,[k]}$. Thus, the whitening key wkL is recovered, and our method is suitable for launching DPA to recover the whitening key on the left branch of Feistel-SP in loop architecture.

3.4 Recovery of wkR

According to Sect. 3.3, we reveal the round key rk_1 and the equivalent mask $MASK_1$ in the first round, and then successfully derive the left branch whitening key wkL from above two parameters. However, due to Figs. 4(b) and 5, the right branch whitening key wkR is hard to be distinguished from the equivalent key, because both wkR and rk_1 have exactly the same effects on the SP type F-function in the first round of Feistel-SP ciphers. More precisely, according to Eq. 4, the first byte of R_2 in this case would be rewritten as:

$$\begin{aligned} R_{2,[1]} &= F_{[1]}(R_1, rk_1) \oplus L_{1,[1]} \\ &= P_{[1]}(S_1(MR_{[1]} \oplus wkR_{[1]} \oplus rk_{1,[1]}), S_2(MR_{[2]} \oplus wkR_{[2]} \oplus rk_{1,[2]}), \\ &\quad \cdots, S_t(MR_{[t]} \oplus wkR_{[t]} \oplus rk_{1,[t]})) \oplus ML_{[1]} \\ &= P'_{[1]}(S_1(MR_{[1]} \oplus wkR_{[1]} \oplus rk_{1,[1]})) \oplus mask_{1,[1]} \oplus ML_{[1]} \\ &= P'_{[1]}(S_1(MR_{[1]} \oplus ek_{1,[1]})) \oplus mask_{1,[1]} \oplus ML_{[1]}. \end{aligned} \tag{6}$$

with $wkR = wkR_{[1]}||wkR_{[2]}||\cdots||wkR_{[t]}$, and the equivalent key $ek_1 = rk_1 \oplus wkR$, and $mask_{1,[1]}$ is a byte constant value if $MR_{[2]}, MR_{[3]}, \cdots, MR_{[t]}$ are fixed since the equivalent key ek_1 is pre-assigned. In this scenario, we can only reveal each byte of ek_1 and $mask_1$ with the approach proposed in Sect. 3.3, and we are unable to split wkR from ek_1.

Therefore, we have to find relations between the case of the whitening key on the right branch and the case of the whitening key on the left branch in order to recover wkR. We focus on the second round of Feistel-SP. As shown in Fig. 6(a), the whitening key wkR is mixed up with $mask_2$ in the second round of Feistel-SP, and we choose R_3 as the attack point in the second round. Equations 3 and 4 can be rewritten as:

$$\begin{aligned} R_3 = & \; F(R_2, rk_2) \oplus L_2 \\ = & \; P(S_1(R_{2,[1]} \oplus rk_{2,[1]}), S_2(R_{2,[2]} \oplus rk_{2,[2]}), \\ & \cdots, S_t(R_{2,[t]} \oplus rk_{2,[t]})) \oplus wkR \oplus MR. \end{aligned} \tag{7}$$

$$\begin{aligned} R_{3,[1]} = & \; F_{[1]}(R_2, rk_2) \oplus L_{2,[1]} \\ = & \; P_{[1]}(S_1(R_{2,[1]} \oplus rk_{2,[1]}), S_2(R_{2,[2]} \oplus rk_{2,[2]}), \\ & \cdots, S_t(R_{2,[t]} \oplus rk_{2,[t]})) \oplus wkR_{[1]} \oplus MR_{[1]} \\ = & \; P'_{[1]}(S_1(R_{2,[1]} \oplus rk_{2,[1]})) \oplus mask_{2,[1]} \oplus wkR_{[1]} \oplus MR_{[1]} \\ = & \; P'_{[1]}(S_1(R_{2,[1]} \oplus rk_{2,[1]})) \oplus MASK_{2,[1]} \oplus MR_{[1]}. \end{aligned} \tag{8}$$

where $mask_{2,[1]}$ is a byte constant value if $R_{2,[2]}, R_{2,[3]}, \cdots, R_{2,[t]}$ are fixed since the round key rk_2 is pre-assigned. Therefore, as shown in Fig. 6(b), if we can control the input of the second round $L_2||R_2$ as the plaintext message $ML||MR$, we will reduce the case of the whitening key on the right branch in the second round to the case of the whitening key on the left branch in the first round.

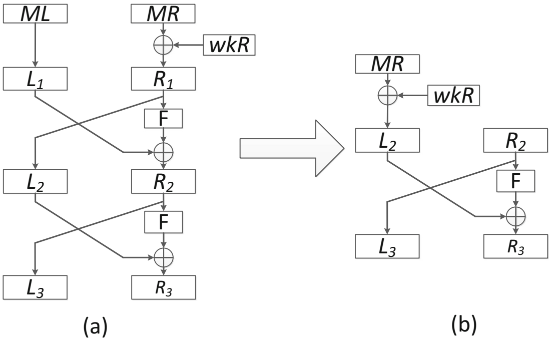


Fig. 6. The reduce from the right branch case to the left branch case

Fortunately, we can control the input of the second round $L_2||R_2$ in an adaptive chosen message manner. Firstly, we reveal ek_1 in the first round. Then based on the values of $L_2||R_2$ which meets the chosen message requirements, we calculate the corresponding plaintext message with ek_1 through the inverse transformation of the Feistel-SP round function. Finally, we could establish the chosen plaintext set which is suitable for the second round and reduce the case of wkR to the case of wkL, and reveal wkR for the second round of Feistel-SP.

On Recovering the Whitening Keys on both Branches. Our approach is also suitable for Feistel-SP ciphers with the key whitening layer on both branches, as shown in Fig. 4(c). The specific procedures of this scenario are as follows:

- **Step 1. Reveal the Whitening Key wkL in the First Round.** The first step of our method is to reveal wkL in the first round. We establish the chosen plaintext set which enumerates all possible values of the target bytes $(MR_{[1]})$, and fixes other bytes to constants. Then, we repeat the DPA attack against R_2 several times to recover all bytes of ek_1 and $MASK_1$. Finally, we calculate $mask_1$ by ek_1 and reveal wkL with the equation $wkL = MASK_1 \oplus mask_1$.
- **Step 2. Reveal the Whitening Key wkR in the Second Round.** The second step of our method is to establish the chosen plaintext set which is used for the second round, through an adaptive chosen message manner. According to Sect. 3.4, it can be done with similar strategy of Step 1 to recover wkR.

4 Applications

We apply these techniques to two typical Feistel-SP ciphers, CLEFIA-128 [2] and Camellia-128 [3], to verify the effectiveness of our method. We implement both ciphers with the loop architecture on a Virtex-5 Xilinx FPGA on SASEBO-GII board. Pearson Correlation Coefficient based Power Analysis (CPA) is applied during the security analysis. The aim is to recover the master keys in CLEFIA-128 and Camellia-128, and the master keys are recovered as expected in both experiments, thus manifesting the correctness of our approach.

4.1 Application to CLEFIA-128

Specification of CLEFIA-128. CLEFI-128 is a type-2 Feistel-SP cipher proposed at FSE 2007 by Shirai *et al.*. It is standardized by ISO [19] as a lightweight cipher. It encrypts a 128-bit plaintext into a 128-bit ciphertext with a 128-bit master key after applying the round function 18 times, as shown in Fig. 7.

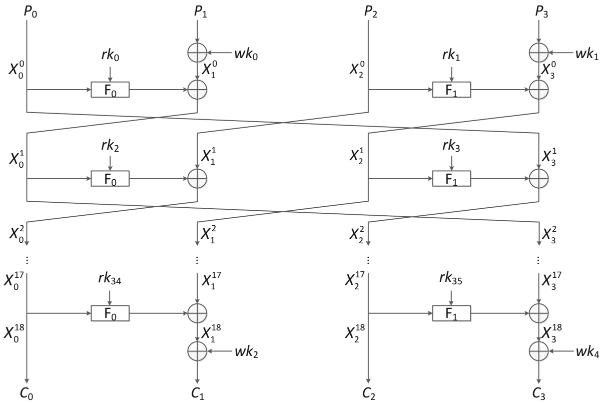


Fig. 7. CLEFIA encryption algorithm for 128-bit key

Let P and K be the 128-bit plaintext and the master key respectively. Thirty-six 32-bit round keys $rk_0, rk_1, \cdots, rk_{35}$ and four 32-bit whitening keys wk_0, wk_1, wk_2, wk_3 are generated from K. These whitening keys are defined as $wk_0||wk_1||wk_2||wk_3 = K$ according to the key schedule.

Let $X_0^i||X_1^i||X_2^i||X_3^i (0 \le i \le 17)$ be an internal input state in each round. The plaintext is loaded into $P_0||P_1||P_2||P_3$. Next, X_1^0 and X_3^0 are updated by the pre-whitening layer, that is $(X_0^0, X_1^0, X_2^0, X_3^0) = (P_0, P_1 \oplus wk_0, P_2, P_3 \oplus wk_1)$. Then, the internal state is updated by the following computation up to the second last round (for $1 \le i \le 17$);

$$X_0^i = X_1^{i-1} \oplus F_0(X_0^{i-1}, rk_{2i-2}), X_1^i = X_2^{i-1},$$
$$X_2^i = X_3^{i-1} \oplus F_1(X_2^{i-1}, rk_{2i-1}), X_3^i = X_0^{i-1}.$$

Two SP type F-functions F_0 and F_1 consist of a 32-bit round key addition, an S-box transformation, and a multiplication by an MDS matrix, as shown in Fig. 8. Four parallel 8-bit Sboxes are applied, followed with an MDS multiplication in each of SP type F-functions. In addition, the MDS matrices for two SP type F-functions are different.

In the last round, $X_0^{18}||X_1^{18}||X_2^{18}||X_3^{18}$ is computed by

$$X_0^{18} = X_0^{17}, X_1^{18} = X_1^{17} \oplus F_0(X_0^{17}, rk_{34}),$$
$$X_2^{18} = X_2^{17}, X_3^{18} = X_3^{17} \oplus F_1(X_2^{17}, rk_{35}).$$

Finally, C_1 and C_3 are updated by the post-whitening layer, *i.e.*, $(C_0, C_1, C_2, C_3) = (X_0^{18}, X_1^{18} \oplus wk_2, X_2^{18}, X_3^{18} \oplus wk_3)$, and $C_0||C_1||C_2||C_3$ is the final ciphertext.

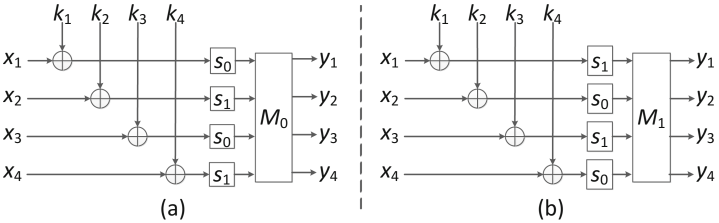


Fig. 8. CLEFIA SP type F-functions for (a) F_0 (b) F_1

Attack of CLEFIA-128. CLEFIA adopts 4-branch Type-2 Feistel network but uses two different diffusion matrices for the diffusion switching mechanism. It has the key whitening layer and only the left branch of the plaintext blocks is XORed with the whitening key. Our aim is to reveal the master keys through the recovery of the whitening keys.

Hereafter we use the notations P_j and C_j for CLEFIA-128 to represent the plaintext and ciphertext in the j^{th} branch, X_j^i to represent the state in the j^{th}

branch immediately after the round operation in round i, X_j^0 to represent the output of the key whitening layer before the first round. We use the notations F_0, F_1, S_0, S_1, M_0, and M_1 to further distinguish the different round functions of CLEFIA-128. The subscript $[n]$ indicates the n^{th} byte of the state.

To reveal the master key K of CLEFIA-128, we focus on the whitening key. As shown in Fig. 7, the four 32-bit whitening keys wk_0, wk_1, wk_2, wk_3 are generated from K. These whitening keys are defined as $wk_0||wk_1||wk_2||wk_3 = K$. Thus, we do not need to calculate the inverse transformation of CLEFIA key schedule to reveal K, if we get the whitening key value. There are two key whitening layers in CLEFIA-128, the pre-whitening before the first round and the post-whitening after the last round. According to the Sect. 3.3, CLEFIA-128 belongs to the case of the whitening key on the left branch. Hence, our attack target is the first round and the last round of CLEFIA-128.

In order to recover both the pre-whitening and the post-whitening keys, the attack should be conducted in both the encryption and the decryption directions respectively. However, since the encryption and decryption of CLEFIA follow similar procedures and F_0 and F_1 are almost equivalent from the perspective of DPA, the attack procedures for recovering the whitening keys are almost identical. Thus we only describe the detailed process to recover wk_0.

The whitening key wk_0 is only XORed with 32-bit plaintext block P_1, and rk_0 is the round key. According to the specification of CLEFIA-128 and Eq. 1, the 32-bit output X_0^1 is described by

$$\begin{aligned} X_0^1 &= F_0(X_0^0, rk_0) \oplus X_1^0 \\ &= M_0(S_0(P_{0,[1]} \oplus rk_{0,[1]}), S_1(P_{0,[2]} \oplus rk_{0,[2]}), \\ &\quad \cdots, S_1(P_{0,[4]} \oplus rk_{0,[4]})) \oplus wk_0 \oplus P_1. \end{aligned} \tag{9}$$

Focusing on the first byte of X_0^1, Eq. 9 could be rewritten as

$$\begin{aligned} X_{0,[1]}^1 &= S_0(P_{0,[1]} \oplus rk_{0,[1]}) \oplus mask_{0,[1]} \oplus wk_{0,[1]} \oplus P_{1,[1]} \\ &= S_0(P_{0,[1]} \oplus rk_{0,[1]}) \oplus MASK_{0,[1]} \oplus P_{1,[1]}. \end{aligned} \tag{10}$$

where $mask_{0,[1]}$ is generated by $S_1(P_{0,[2]} \oplus rk_{0,[2]}), S_0(P_{0,[3]} \oplus rk_{0,[3]})$, and $S_1(P_{0,[4]} \oplus rk_{0,[4]})$, and $MASK_0$ is defined as $MASK_0 = mask_0 \oplus wk_0$.

Therefore, we can reveal the values of rk_0 and $MASK_0$ by chosen message DPA method as mentioned in Sect. 3.3. Then, we calculate $mask_0$ by rk_0 and reveal wk_0 with the equation $wk_0 = MASK_0 \oplus mask_0$. Thus, wk_1, wk_2, and wk_3 can be revealed by similar procedures. After deriving all the whitening keys, the master key K can be easily derived since $K = wk_0||wk_1||wk_2||wk_3$.

We preset the master key K as four consecutive 32-bit words denoted in hex form, `FFEEDDCC-BBAA9988-77665544-33221100`, in our FPGA implementation with the loop architecture. Thus, it is obvious that $wk_0 =$ `FFEEDDCC` and $rk_0 =$ `F3E6CEF9`. We use the attack result of the first Sbox (*i.e.*, $rk_{0,[1]}$ and $MASK_{1,[1]}$) as an example. In Fig. 9(a), there are two max points `F3D4` and `F32B` whose correlation coefficients are 0.0743 and −0.0743 respectively. According to the knowledge of the FPGA platform, the power consumption of the FPGA platform has a negative correlation with the Hamming distance power model. Thus, F32B is the attack result (*i.e.*, $rk_{0,[1]} = \texttt{F3}$ and $MASK_{1,[1]} = \texttt{2B}$), which is revealed within 10000 power traces as shown in Fig. 9(b).

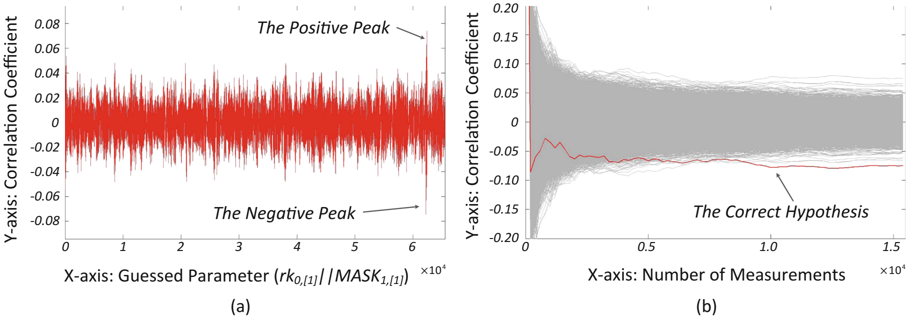


Fig. 9. Correlation traces in different hypothesis and different number of measurements on CLEFIA

We repeat the above procedure 3 times more, and all bytes of rk_0 and $MASK_1$ are revealed as $rk_0 = \texttt{F3E6CEF9}$ and $MASK_1 = \texttt{2BF18258}$. Thus, the value of $mask_1$ is D41F5F94, which is calculated by the round key rk_0. And the wk_0 is FFEEDDCC, calculated by $mask_1 \oplus MASK_1$. The remaining parts of whitening keys are calculated by similar way. Now, we calculate K is FFEEDDCC-BBAA9988-77665544-33221100, which is identical with the preset master key.

In our method, we only attack 2 rounds, *i.e.*, the first round and the last round. Due to the equivalent key of the second round, the conventional DPA must be launched on the first 4 rounds for revealing enough consecutive round keys. Therefore, the cost of our method (*e.g.*, power traces or recovered round key fractions) is only half of the conventional method.

4.2 Application to Camellia-128

Specification of Camellia-128. Camellia-128, proposed at SAC 2000 by Aoki *et al.* [3], was jointly designed by NTT and Mitsubishi Electric Corporation. It is widely acknowledged and recommended by ISO [20], NESSIE [21], and CRYPTREC [22]. It encrypts a 128-bit plaintext into a 128-bit ciphertext with a 128-bit master key after applying the round function 18 times, as shown in Fig. 10.

Let P and K be a 128-bit plaintext and a secret key, respectively. Eighteen 64-bit round keys $rk_1, rk_2, \cdots, rk_{18}$ and four 64-bit whitening keys wk_1, wk_2, wk_3, wk_4 are generated from K. Let L_r and R_r $(0 \leq r \leq 18)$ be left and right 64-bit of the internal state in each round. According to the key schedule, the pre-whitening keys are defined as $wk_1 || wk_2 = K$. We omit the descriptions of the FL and FL^{-1} layers after the 6^{th} and 12^{th} rounds, since they have no impacts on our work.

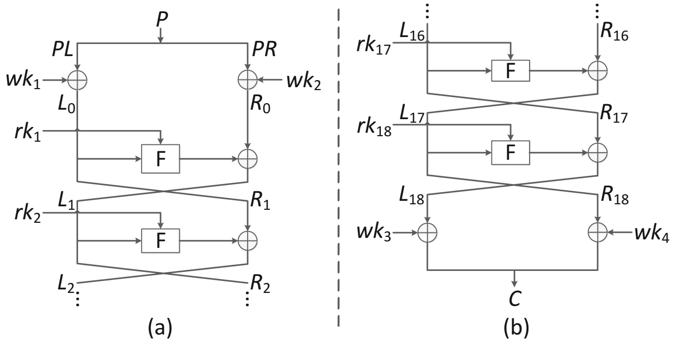


Fig. 10. Camellia encryption algorithm for (a) First two rounds (b) Last two rounds

The round function consists of a 64-bit subkey addition, Sbox transformation, and a diffusion layer called P-layer, as shown in Fig. 11. Eight parallel 8-bit Sboxes are applied, followed with the P-layer which operates on a 64-bit value $(z_1||z_2||\cdots||z_8)$. The corresponding output $(z'_1||z'_2||\cdots||z'_8)$ is computed as follows.

$$z'_1 = z[1,3,4,6,7,8], z'_2 = z[1,2,4,5,7,8],$$
$$z'_3 = z[1,2,3,5,6,8], z'_4 = z[2,3,4,5,6,7],$$
$$z'_5 = z[1,2,6,7,8], z'_6 = z[2,3,5,7,8],$$
$$z'_7 = z[3,4,5,6,8], z'_8 = z[1,4,5,6,7].$$

Here, $z[s,t,u,\cdots]$ means $z_s \oplus z_t \oplus z_u \oplus \cdots$

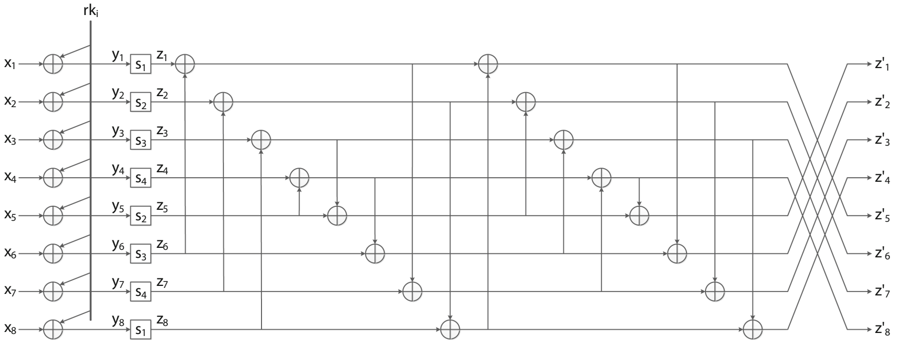


Fig. 11. Camellia SP type F-function

Attack of Camellia-128. Camellia is a Feistel-SP cipher but its P-layer does not satisfy the maximum branch number. It has the key whitening layer and the whole plaintext is XORed with the whitening key. Our aim is to reveal the master keys through the recovery of the whitening keys.

We use the notations L_i and R_i for Camellia-128 to represent the state immediately after the round operation in the i^{th} round, especially L_0 and R_0 to represent the output of the whitening layer before the first round, respectively. We use the notations PL and PR to differentiate the left and right of plaintext, and S_1, S_2, S_3, S_4, and M to further distinguish the different Sboxes and diffusion operation of Camellia-128. The subscript $[n]$ indicates the n^{th} byte of the state.

Camellia-128 belongs to the case of whitening keys on both branches. To reveal the master key K of Camellia-128, we focus on two 64-bit pre-whitening key wk_1 and wk_2, which are defined as $wk_1||wk_2 = K$. The two whitening keys are XORed with two plaintext blocks PL and PR respectively, before the first round. Hence, our attack target is the first two rounds of Camellia-128. According to the attack procedure in the first round of Camellia-128 which is similar to that of CLEFIA-128, thus we only describe the detailed process to recover wk_1 in the second round.

Now, we show how to recover the whitening keys wk_1 by attacking the round function F of the second round. The whitening keys wk_1 and wk_2 are parallel XORed with two 64-bit plaintext blocks PL and PR respectively, and rk_1 and rk_2 is the round keys as shown in Fig. 10. The equivalent key ek_1 and the whitening key wk_2 are recovered in the first round by the attack process. Therefore, we focus on the second round.

According to the specification of Camellia-128 and Eq. 1, the 64-bit output L_2 is described by

$$\begin{aligned} L_2 = \ & F(L_1, rk_2) \oplus R_1 \\ = \ & M(S_1(L_{1,[1]} \oplus rk_{2,[1]}), S_2(L_{1,[2]} \oplus rk_{2,[2]}), \\ & \cdots, S_1(L_{1,[8]} \oplus rk_{1,[8]})) \oplus wk_1 \oplus PL. \end{aligned} \tag{11}$$

Focusing on the first byte of L_2, Eq. 11 could be rewritten as

$$\begin{aligned} L_{2,[1]} &= S_1(L_{1,[1]} \oplus rk_{2,[1]}) \oplus mask_{2,[1]} \oplus wk_{1,[1]} \oplus PL_{[1]} \\ &= S_1(L_{1,[1]} \oplus rk_{2,[1]}) \oplus MASK_{2,[1]} \oplus PL_{[1]}. \end{aligned} \tag{12}$$

where
$mask_{2,[1]}$ is generated by $S_2(L_{1,[2]} \oplus rk_{2,[2]}), S_3(L_{1,[3]} \oplus rk_{2,[3]}), \cdots, S_1(L_{1,[8]}$ $\oplus rk_{2,[8]})$ and $MASK_2$ is defined as $MASK_2 = mask_2 \oplus wk_1$.

Before launch attack in the second round, we should calculate the corresponding plaintext message with ek_1 and wk_2 through the inverse transformation of Camellia round function, according to the value of L_1 which meets the chosen message requirements. Fortunately, we find that there is an one-to-one mapping relationship between PR and L_1. Thus, we can control L_2 by enumerating PR, and reveal the value wk_1 by the method as mentioned in Sect. 3.4. After deriving all the whitening keys, the master key K can be easily derived since $K = wk_1||wk_2$.

We preset the master key K as four consecutive 32-bit words denoted in hex form, `01234567-89ABCDEF-FEDCBA98-76543210`, in our FPGA implementation with the loop architecture. Thus, it is obvious that $wk_1 =$ `01234567-89ABCDEF`.

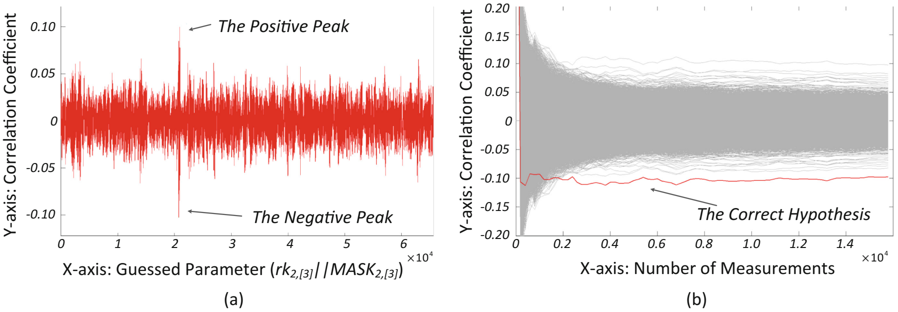


Fig. 12. Correlation traces in different (a) hypothesis and (b) number of measurements on Camellia

We use the attack result of the third Sbox (*i.e.*, $rk_{2,[3]}$ and $MASK_{2,[3]}$) as an example. In Fig. 12(a), there are two max points `40B8` and `4047` whose correlation coefficients are 0.107 and −0.107 respectively. According to the negative correlation between the power consumption of the FPGA platform and the Hamming distance power model, `4047` is the attack result (*i.e.*, $rk_{2,[3]} =$ `40` and $MASK_{2,[3]} =$ `47`), which is revealed within 4000 power traces as shown in Fig. 12(b).

We repeat the above procedure 7 times more, and all bytes of rk_2, $MASK_2$, $mask_2$ and are revealed as similar as the case of CLEFIA-128. Thus, the wk_1 is `01234567-89ABCDEF`, calculated by $mask_2 \oplus MASK_2$. Now, we calculate K is `01234567-89ABCDEF-FEDCBA98-76543210`, which is identical with the preset master key.

In our method, we only attack the first two rounds. Due to the equivalent keys of the first two rounds, the conventional DPA must be launched on the first four rounds for revealing enough consecutive round keys. Therefore, the cost of our method is only half of the conventional method.

5 Conclusion and Discussion

In this paper, we propose a practical chosen message DPA approach to recover the whitening key through a reduction strategy, and show that the key whitening technique does not enhance the security of ciphers from the standpoint of DPA. Then we apply our method to CLEFIA-128 and Camellia-128, and the master keys are recovered as expected. Following the results presented in this work, several problems which are worth further investigations:

- **Optimizations.** The first natural question emerges is whether our method can be further optimized. One possible direction of improvement is taking advantage of more powerful DPA method, such as the adaptive strategy in [6,14] and the multiple CPA in [12], to discriminate the correct hypothesis from the key candidates in a more efficient way. Other potential optimizations are also possible directions for future research.

- **Countermeasures.** Next to optimality, another important question is to determine the countermeasures against such attack. Our method is suitable for the unprotected loop hardware implementations. Several common countermeasures on the compact architecture [17], can be considered to apply to the loop architecture in order to resist our approach, while the resource consumption will have a corresponding increase. Moreover, the countermeasures based on the mask methodology should be used with caution on the loop implementations of ciphers, because the mask countermeasures usually lead to slow computation speed. Considering the limitation of high computation speed and high throughput in the application scenario, the loop implementations of ciphers must keep the high performance, when the countermeasures against such attack are applied on these implementations. Thus, a trade-off between performance and security should be considered by the vendor.

References

1. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). doi:10.1007/3-540-48405-1_25
2. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher CLEFIA (extended abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74619-5_12
3. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: *Camellia*: a 128-bit block cipher suitable for multiple platforms — design and analysis. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 41–54. Springer, Heidelberg (2001). doi:10.1007/3-540-44983-3_4
4. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28632-5_2
5. Hoang, V.T., Rogaway, P.: On generalized Feistel networks. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 613–630. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14623-7_33
6. Kopf, B., Basin, D.A.: An information-theoretic model for adaptive side-channel attacks. In: Ning, P., De Capitani Vimercati di, S., Syverson, P.F. (eds.) Proceedings of the 14th ACM Conference on Computer and Communications Security, ACM-CCS 2007, pp. 286–296. ACM (2007)
7. Kim, Y., Ahn, J., Choi, H.: Power and electromagnetic analysis attack on a smart card implementation of CLEFIA. In: International Conference on Security and Management, SAM 2013 (2013)
8. Akkar, M.-L., Bevan, R., Dischamp, P., Moyart, D.: Power analysis, what is now possible. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 489–502. Springer, Heidelberg (2000). doi:10.1007/3-540-44448-3_38
9. Lu, Y., O'Neill, M.P., McCanny, J.V.: Differential power analysis resistance of Camellia and countermeasure strategy on FPGAs. In: International Conference on Field-Programmable Technology, pp. 183–189 (2009)
10. Xiao, L., Heys, H.: A simple power analysis attack against the key schedule of the Camellia block cipher. Inf. Process. Lett. **95**, 409–412 (2005). Elsevier

11. Bayrak, A.G., Regazzoni, F., Novo, D., Ienne, P.: Sleuth: automated verification of software power analysis countermeasures. In: Bertoni, G., Coron, J.-S. (eds.) CHES 2013. LNCS, vol. 8086, pp. 293–310. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40349-1_17
12. Moradi, A., Schneider, T.: Improved side-channel analysis attacks on xilinx bitstream encryption of 5, 6, and 7 series. In: Standaert, F.-X., Oswald, E. (eds.) COSADE 2016. LNCS, vol. 9689, pp. 71–87. Springer, Cham (2016). doi:10.1007/978-3-319-43283-0_5
13. Moradi, A., Kasper, M., Paar, C.: Black-box side-channel attacks highlight the importance of countermeasures. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 1–18. Springer, Heidelberg (2012). doi:10.1007/978-3-642-27954-6_1
14. Veyrat-Charvillon, N., Standaert, F.-X.: Adaptive chosen-message side-channel attacks. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 186–199. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13708-2_12
15. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New lightweight DES variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196–210. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74619-5_13
16. Borghoff, J., et al.: PRINCE – a low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34961-4_14
17. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, New York (2007). ISBN: 978-0-387-30857-9
18. Rodríguez-Henríquez, F., Saqib, N.A., Díaz-Pèrez, A., Koc, Ç.K.: Cryptographic Algorithms on Reconfigurable Hardware (Signals and Communication Technology). Springer, Heidelberg (2006)
19. ISO/IEC 29192–2:2011. Information technology - Security techniques - Lightweight cryptography-Part 2: Block ciphers (2011)
20. ISO/IEC 18033–3:2010. Information technology - Security techniques - Encryption Algorithms-Part 3: Block ciphers (2010)
21. New European Schemes for Signatures, Integrity, and Encryption (NESSIE). NESSIE Project Announces Final Selection of Crypto Algorithms (2003)
22. Cryptography Research and Evaluation Committees (CRYPTREC). e-Government recommended ciphers list (2003)

Side-Channel Attacks Meet Secure Network Protocols

Alex Biryukov, Daniel Dinu(✉), and Yann Le Corre

SnT, University of Luxembourg,
6, Avenue de la Fonte, 4364 Esch-sur-Alzette, Luxembourg
{alex.biryukov,dumitru-daniel.dinu,yann.lecorre}@uni.lu

Abstract. Side-channel attacks are powerful tools for breaking systems that implement cryptographic algorithms. The Advanced Encryption Standard (AES) is widely used to secure data, including the communication within various network protocols. Major cryptographic libraries such as OpenSSL or ARM mbed TLS include at least one implementation of the AES. In this paper, we show that most implementations of the AES present in popular open-source cryptographic libraries are vulnerable to side-channel attacks, even in a network protocol scenario when the attacker has limited control of the input. We present an algorithm for symbolic processing of the AES state for any input configuration where several input bytes are variable and known, while the rest are fixed and unknown as is the case in most secure network protocols. Then, we classify all possible inputs into 25 independent evaluation cases depending on the number of bytes controlled by attacker and the number of rounds that must be attacked to recover the master key. Finally, we describe an optimal algorithm that can be used to recover the master key using Correlation Power Analysis (CPA) attacks. Our experimental results raise awareness of the insecurity of unprotected implementations of the AES used in network protocol stacks.

Keywords: Side-channel attack · Secure network protocol · CPA · AES

1 Introduction

Side-channel attacks use observations made during the execution of an implementation of a cryptographic algorithm to recover secret information. From the multitude of side-channel attacks, Correlation Power Analysis (CPA) [5] stands out as a very efficient and reliable technique. Its success is augmented by the minimally invasive methods employed for the acquisition of the side-channel information. Some of the most frequently used sources of side-channel leakage are the power consumption or the electromagnetic (EM) emissions of a device under attack.

Nowadays, the AES [23] is the most popular symmetric cryptographic algorithm in use. It is widely deployed to secure data in transit or at rest. Various network protocols rely on the AES in different modes of operation to provide

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 435–454, 2017.
DOI: 10.1007/978-3-319-61204-1_22

security services such as confidentiality and authenticity. The usage spectrum of the AES stretches from powerful servers and personal computers to resource constrained devices such as wireless sensor nodes. While the security of the algorithm and its implementations have been placed under scrutiny since it became the symmetric cryptographic standard, with a few notable exceptions, most of the previous work focused on the AES itself and less on the usage of the AES in complex systems.

By far, most of the experimental results reported in the side-channel literature are for implementations of the AES. They usually assume the attacker has full control of the AES input. This is not the case in a real world communication protocol, when often a major part of the input is fixed and only few bytes are variable. Moreover, sometimes the attacker cannot control these variable bytes and she has to passively observe executions of the targeted algorithm without being able to trigger encryptions of her own free will. With the notable exceptions of [16,24], the security of communication scenarios based on the AES against side-channel attacks has not been thoroughly analyzed so far. Thus, in this paper we analyze for the first time how much control of the AES input does an attacker need to recover the secret key of the cipher by performing a side-channel attack against a communication protocol.

Numerous standards for communication in the Internet of Things (IoT) such as IEEE 802.15.4 [15] and LoRaWAN [21] use the AES to encrypt and authenticate the Medium Access Control (MAC) layer frames. The 802.15.4 standard uses a variant of the AES-CCM [9,34], while LoRaWAN uses AES-CMAC [31]. The same CCM mode is used with the AES to encrypt the IPsec Encapsulating Security Payload (ESP) [14]. According to [29] the security architecture of IEEE 802.15.4 relies on four categories of security suites: none, AES-CTR, AES-CBC-MAC, and AES-CCM. A typical input for the AES-CTR and AES-CCM modes used in the IEEE 802.15.4 protocol is shown in Fig. 1. In this particular example, an attacker can manipulate up to 12 bytes of the input (`Source Address` and `Frame Counter`), while the other input bytes (`Flags`, `Key Counter` and `Block Counter`) are fixed. The attack on IEEE 802.15.4 wireless sensor nodes described in [24] assumes the control of only four input bytes (`Frame Counter`), while the remaining input bytes are constant. Thus the following question arises: *How many input bytes should an attacker change in the injected messages in order to fully recover the master key without triggering any network protection mechanism?*

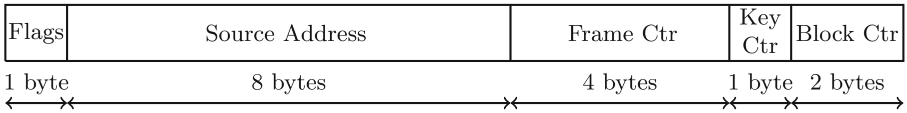


Fig. 1. The first input block for the AES-CTR and AES-CCM modes used in IEEE 802.15.4.

While numerous network protocols use the AES to secure the communication between end nodes, major cryptographic libraries such as OpenSSL [25] and ARM mbed TLS [2] do not have a side-channel protected implementation of the AES for devices that do not support the AES-NI [13] instruction set as is the case with most IoT devices. Therefore, an elaborate analysis of the security of the unprotected implementations of the AES used in communication protocols is necessary. Only such a careful analysis can assess the impact of side-channel attacks on the security of real world systems using unprotected implementations of the AES.

In this work, we chose to focus on CPA attacks thanks to their efficiency and reliability. We opted for a non-invasive measurement setup and hence we selected the EM emissions of the target processor as source of side-channel leakage. The target is an ARM Cortex-M3 processor mounted on a STM32 Nucleo [32] board from STMicroelectronics. These processors are widely used for low-power applications and meet the requirements for use in the IoT.

The IoT will be a security nightmare if the whole ecosystem is not designed with security in mind. While many communication protocols for the IoT are in formative stages, the threat model of the IoT is less understood despite it is widely accepted that its attack surface is large. Although we focus on a particular side-channel attack (i.e. power/EM), other side-channel attacks such as timing, fault, cache or data remanence attacks might pose a similar or even a higher threat for the security of the IoT ecosystem. Attacks that do not exploit side-channel information, such as those used to compromise Internet-connected computers, should not be neglected since they have certain advantages over side-channel attacks. Thus, our work adds another piece to the security puzzle of the IoT by showing the need for side-channel countermeasures to prevent a somehow overlooked threat.

Research Contributions. This paper performs for the first time a thorough analysis of all possible attack scenarios against software implementations of the AES used to secure various communication protocols. Firstly, we present an algorithm for symbolic processing of a given input state of the AES. The algorithm outputs the number of rounds and the bytes that must be attacked to recover the secret key. Then, using this algorithm we perform a classification of all possible inputs depending on the number of rounds that must be attacked in order to recover the master key. The result is a set of 25 independent evaluation cases. Secondly, we describe an optimal algorithm that uses the above-mentioned symbolic representation to recover the master key of the AES using CPA attacks. The algorithm explores all possible combinations of input key bytes and discards the invalid key candidates, thus yielding only the correct master key if enough power traces with a good signal-to-noise ratio are provided. Afterwards, we evaluate the results of the attack algorithm in each of the 25 evaluation cases identified in the classification step using traces from an ARM Cortex-M3 processor.

Our results show that popular implementations of the AES found in well-known and widely used cryptographic libraries can be broken using CPA attacks.

The only requirement is that a part of the AES input is known and variable, while the rest is constant, which is a common scenario in communication protocols. Knowledge of the AES implementation strategy improves the attack results, but it is not crucial. All software tools presented in this paper are in the public domain[1] to support reproducibility of results and to maximize reusability.

2 Preliminaries

2.1 Description of the AES

We give a brief description of the AES [23] to recall relevant aspects of the algorithm and to introduce the notation used in this paper. For more details on the AES algorithm, we refer the reader to the official specifications.

The AES standard uses the 128-bit block length version of the Rijndael cipher [8] with three different key lengths: 128, 192, and 256 bits. The round function is applied to the 4×4 byte state matrix 10, 12, or 14 times depending on the key length. It comprises four transformations: `SubBytes`, `ShiftRows`, `MixColumns`, and `AddRoundKey`. The final round function does not include the `MixColumns` transformation.

Let $s_{i,j}$ be the state byte located at row i and column j $(0 \leq i, j \leq 3)$, k_l the corresponding round key byte $(l = 16 \cdot r + i + 4 \cdot j)$ and r the round number. After application of the `AddRoundKey` transformation, each byte of the state becomes $s'_{i,j} = s_{i,j} \oplus k_l$, where the "$\oplus$" symbol denotes bitwise exclusive or of two 8-bit values. The non-linear `SubBytes` operation transforms each byte of the state using an 8-bit S-box S as follows: $s'_{i,j} = S[s_{i,j}]$. The `ShiftRows` transformation performs a rotation of row i by i bytes to the left. In the `MixColumns` transformation, a polynomial multiplication over $GF(2^8)$ is applied to each column of the state matrix. The symbol "$\bullet$" is used for multiplication of two numbers in $GF(2^8)$, while $\{01\}$, $\{02\}$, and $\{03\}$ are 8-bit vectors representing elements from $GF(2^8)$.

The key schedule expands the master key into the 16-byte round keys. The round constant array `Rcon` contains the powers of $\{02\}$ in $GF(2^8)$ as described in the specifications.

2.2 Correlation Power Analysis

Correlation Power Analysis (CPA) [5] is a side-channel attack in which the attacker correlates the power model of a sensitive intermediate value of the target cryptographic algorithm with the measured power consumption or electromagnetic emission (EM) of the device running the target algorithm. Then, she chooses the key hypothesis that gives the maximum correlation coefficient as the most likely key. Compared to classical Differential Power Analysis (DPA) [17] attacks, CPA attacks have several advantages in terms of efficiency, robustness and number of experiments, but are more resource demanding. Agrawal *et al.* [1]

[1] https://github.com/cryptolu/aes-cpa.

introduced the electromagnetic emissions of a target device as a source of leakage for side-channel attacks.

A CPA attack can be split into two phases: acquisition and attack. In the acquisition phase, the attacker observes and records the leakage of the target device (power consumption or electromagnetic emission) for different inputs. While the acquisition of power consumption traces requires insertion of a resistor into the circuitry of the target device to measure the voltage across it, the observation of electromagnetic emission is non-invasive; it only requires an electromagnetic probe placed in the vicinity of the leaking spot. In the attack phase, the attacker correlates these observations with the modeled power consumption of the selection function to recover the secret key. A selection function combines a known input with the secret material to be recovered.

In this work we focus on the electromagnetic emissions of an ARM Cortex-M3 processor clocked at 8 MHz running various software implementations of the AES. The acquisition was performed from a spot above the chip using a Langer RF-K 7-4 H-field probe. The signal was amplified by 30dB and fed into a Teledine LeCroy WaveRunner 8254M-MS oscilloscope sampling at 500 MS/s. For more details on the measurement setup we refer the reader to the full version of this paper.

2.3 Attacking Temporary Key Bytes

To attack the AES in counter mode, Jaffe introduced a technique that propagates a DPA attack to later rounds. It can be used when just few key bytes of the AES input are known and variable, while the others are fixed (constant) and unknown [16]. Next we briefly describe how the unknown fixed bytes can be incorporated into a round key byte to recover a temporary key byte. Then, using these temporary key bytes the attack can be carried into later rounds until enough round key bytes are recovered to reverse the key schedule.

Using a CPA attack an adversary can recover only those key bytes that are XORed with variable and known state bytes in the `AddRoundKey` transformation. The gist of Jaffe's technique is that an attacker can still recover a temporary key byte when an input byte of the `AddRoundKey` transformation is the result of the `MixColumns` transformation applied to at least one known and variable input byte while the other input bytes are unknown and constant.

To better illustrate how this technique works, let us consider the first state byte $s'_{0,0}$ after performing the first round function:

$$s'_{0,0} = (\{02\} \bullet s_{0,0}) \oplus (\{03\} \bullet s_{1,1}) \oplus (\{01\} \bullet s_{2,2}) \oplus (\{01\} \bullet s_{3,3}) \oplus k_{16}$$

Suppose now that the input bytes $s_{0,0}$ and $s_{1,1}$ are known and variable (key bytes k_0 and k_5 were successfully recovered using a side-channel attack on the `SubBytes` transformation of the first round), while the other input bytes ($s_{2,2}$ and $s_{3,3}$) are unknown, but fixed. Thus $s'_{0,0}$ can be written as $(\{02\} \bullet s_{0,0}) \oplus (\{03\} \bullet s_{1,1}) \oplus k'_{16}$, where the constant part is included in the temporary key k'_{16} that will be recovered by attacking the `SubBytes` transformation of the second round; $k'_{16} = (\{01\} \bullet s_{2,2}) \oplus (\{01\} \bullet s_{3,3}) \oplus k_{16}$. The temporary key k'_{16} enables the computation of four state bytes in the following round. In this way, the attack is carried into the next rounds until all state bytes are known; consequently, the real key bytes can be recovered.

The technique works similarly when three input bytes are known and variable. Though, when only one input byte is known and variable, the attacker will recover the same two equally likely key candidates for two bytes of the same column of the cipher state. For example, when only $s_{3,3}$ is known and variable while the other input bytes are unknown and fixed, then $s'_{0,0} = (\{01\} \bullet s_{3,3}) \oplus k'_{16}$ and $s'_{1,0} = (\{01\} \bullet s_{3,3}) \oplus k'_{17}$. Thus attacking either of the two, an attacker will get two equally likely key bytes (k'_{16} and k'_{17}). If the state bytes are not processed in order by the `SubBytes` transformation, the attacker will not know which key byte corresponds to $s'_{0,0}$ and which key byte corresponds to $s'_{1,0}$.

2.4 Software Implementations of the AES

There are various ways to implement the AES in software depending on the execution time, code size and RAM consumption requirements. Other factors that influence the implementation strategy are the cipher mode of operation and the number of plaintext blocks to be encrypted. Schwabe and Stoffelen [30] identified four different strategies to implement the AES in software: traditional, T-tables, vector permute, and bit slicing. In this paper, we consider the following two implementation approaches for the AES that are relevant for a secure communication protocol:

- The *straightforward implementation* (**S-box** strategy) performs the four round transformations as described above. The substitution layer is implemented using a 256-byte lookup table based on S-box S. This implementation approach is suitable for 8-bit architectures.
- The *table based implementation* (**T-table** strategy) uses four lookup tables (T_0, T_1, T_2, and T_3) of 1024 bytes each to perform the `SubBytes`, `ShiftRows`, and `MixColumns` operations at the cost of 16 table lookups, 16 masks and 16 XORs per round, except for the final round. A low memory alternative uses just one T-table, but performs 12 additional rotations per round. This strategy was initially described by the designers of Rijndael [8]. It leads to very fast implementations on 32-bit platforms.

We did not analyze bit-sliced or vector permute implementations because such implementations are uncommon in cryptographic libraries due to the following limitations. The bit-sliced implementations process at least two blocks in parallel and thus they can be applied only to non-feedback modes of operation. The vector permute implementations require support for vector permute instructions, but most of the resource constrained microcontrollers for the IoT do not support such instructions.

An analysis of the existing AES implementations used by different open source cryptographic libraries is given in Table 1. The default implementations of the AES for platforms that do not support the AES-NI [13] instructions in popular cryptographic libraries such as OpenSSL [12,25] or mbed TLS [2,11] use the

Table 1. A summary of the existing AES implementations used by open source cryptographic libraries written in C/C++. All the T-table implementations are vulnerable to the attack described in this paper.

Library	Language	Version	Last update	AES-NI	T-table
Botan [27]	C++	2.1.0	Apr 2017	✓	✓
cryptlib [6]	C	3.4.3	Feb 2017	✓	✓
Crypto++ [7]	C++	5.6.5	Oct 2016	✓	✓
Libgcrypt [18]	C	1.7.6	Jan 2017	✓	✓
libtomcrypt [10]	C	1.17	Apr 2017	✗	✓
libsodium [19]	C	1.0.12	Mar 2017	✓	✗
mbed TLS [2]	C	2.4.2	Mar 2017	✓	✓
Nettle [22]	C	3.3	Oct 2016	✓	✓
OpenSSL [25]	C	1.1.0e	Feb 2017	✓	✓
wolfCrypt [35]	C	3.10.2	Feb 2017	✓	✓

T-table approach. Except for libsodium [19], all other cryptographic libraries analyzed have an implementation of the AES based on the T-table strategy. Moreover, these implementations are not protected against side-channel attacks such as DPA or cache attacks. It is well know that unprotected implementations of cryptographic algorithms are an easy target for DPA attacks. Recently, researchers from Rambus Cryptography Research Division have shown that even an unprotected software implementation based on AES-NI instructions can be attacked with DPA [28]. The T-table implementations of the AES are vulnerable to various cache attacks as shown in [20,26]. Although the unprotected T-table implementations are vulnerable to side channel attacks, nine out of the ten libraries considered in Table 1 have such an implementation of the AES.

3 Quantifying the Leakage

Biryukov *et al.* [4] introduced the correlation coefficient difference metric to analyze the leakage of different selection functions in the context of CPA. The correlation coefficient difference δ gives the difference between the correlation coefficient of the correct key and the correlation coefficient of the most likely key guess, where the most likely key is different from the correct key.

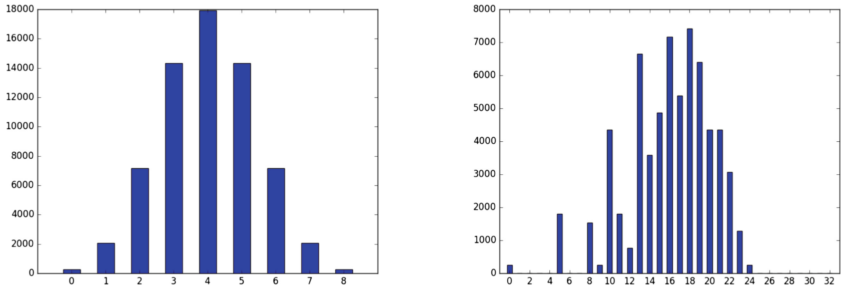
We use the correlation coefficient difference to quantify the leakages of two selection functions: φ_1 based on the AES S-box and φ_2 based on the AES T-table. The two selection functions are defined below:

$$\varphi_1 : \mathbb{F}_2^8 \mapsto \mathbb{F}_2^8, \quad \varphi_1(x \oplus k) = S(x \oplus k)$$

$$\varphi_2 : \mathbb{F}_2^8 \mapsto \mathbb{F}_2^{32}, \quad \varphi_2(x \oplus k) = T(x \oplus k)$$

Table 2. Correlation coefficient difference δ between the correlation of the correct key and the correlation of the most likely key [4], for different Hamming weights of the correct key; $\bar{\delta}$ and $\mathsf{SE}_{\bar{\delta}}$ are the mean and the standard error for a 95% confidence interval, respectively. The leakages are acquired from an ARM Cortex-M3 processor.

	Correct key									$\bar{\delta}$	$\mathsf{SE}_{\bar{\delta}}$
	`0x00`	`0x01`	`0x03`	`0x07`	`0x0F`	`0x1F`	`0x3F`	`0x7F`	`0xFF`		
φ_1	0.146	0.126	0.108	0.156	0.126	0.960	0.153	0.140	0.084	0.126	0.020
φ_2	0.104	0.072	0.143	0.074	0.070	0.126	0.078	0.044	0.028	0.082	0.028



(a) S-box

(b) T-table

Fig. 2. Distribution of the Hamming weight of the output of the AES (a) S-box and (b) T-table for all possible input combinations.

When using simulated leakages, the values of the correlation coefficient difference are 0.813 and 0.7 for φ_1 and φ_2, respectively. These values are the same regardless of the correct key used. In the simulated environment, the leakages of the two selection functions are very high and the difference between them is about 14% of the first one. On the other hand, the mean correlation coefficient difference $\bar{\delta}$ for different values of the correct key using leakages acquired from an ARM Cortex-M3 processor is given in Table 2. The measurements were performed at a sampling rate of 500 MS/s using assembly implementations of the analyzed selection functions. Increasing the sampling rate to 1 GS/s does not significantly improve the results. The mean correlation coefficient difference $\bar{\delta}$ is positive for both selection functions, which means they leak enough information about the secret key such that an attacker can recover the key byte using only one key guess. In practice, the selection function based on the AES S-box leaks about 50% more than the selection function based on the AES T-table. This can be explained by the distribution of the Hamming weight of the two selection functions for all possible input combinations (See Fig. 2).

The reader can easily observe in Fig. 2a that the distribution of values in the case of the AES S-box follows a binomial distribution. On the other hand, the distribution of values in the case of the AES T-table shown Fig. 2b does not resemble a binomial distribution. Moreover, there are 14 out of 32 possible output values that never occur (i.e. 1, 2, 3, 4, 6, 7, 25, 26, 27, 28, 29, 30, 31, and 32). These differences between the distribution of the Hamming weights of the output of the two selection functions φ_1 and φ_2 explain why the leakage of φ_1 is greater than the leakage of φ_2 as quantified using the correlation coefficient difference. This means that a CPA attack against an implementation based on the T-table strategy requires more effort (i.e. power traces) compared to a CPA attack against an implementation based on the S-box strategy.

4 Generating the Evaluation Cases

In this section we describe the algorithm for symbolic processing of a given initial state to determine the number of rounds required to recover the master key of the AES. We used this algorithm to explore all possible attack cases and to choose the relevant evaluation cases for our scenario. The algorithm relies on the following symbolic representation of a byte situated at row i and column j of the AES state at the start of round r:

$$s^r_{i,j} = \begin{cases} 0, & \text{the corresponding key byte can not be recovered} \\ 1, & \text{the corresponding key byte can be recovered} \\ -n, & n \text{ temporary key bytes can be recovered} \end{cases}$$

Thus, the byte $s^r_{i,j}$ is variable if its symbolic representation is different from 0 and fixed (constant) when its symbolic representation is 0. Due to the `MixColumns` transformation, each column of the state at round $r+1$ can be expressed as a function of four bytes of the state at round r. At the start of round $r+1$ each byte of the state is updated using the following rules:

- if the number of variable input bytes is 0, then the symbolic representation of the output byte is set to 0;
- if the number of variable input bytes is 1, then the symbolic representation of the output byte is updated as follows:
  - if the variable input byte is multiplied by $\{01\}$ in the `MixColumns` transformation, then the symbolic representation of the output byte is set to -2^{p+1}, where p is the number of independent input pairs. A new pair is added to the output byte;
  - else, the symbolic representation of the output byte is set to -2^p;
- if the number of variable input bytes is 2 or 3, then the symbolic representation of the output byte is set to -1;
- if the number of variable input bytes is 4, then the symbolic representation of the output byte is set to 1.

Besides updating the symbolic representation of the state, the algorithm keeps a list of key pairs for each byte of the state and carries this list into the next round. The algorithm stops when the symbolic representation of all bytes in a round is 1. It outputs the symbolic representation of the state and the associated key pairs. These can be used to compute the number of rounds

	Round 1				Round 2				Round 3				Round 4			
State	1	0	0	0	-1	0	0	0	-1	-2	-4	-2	1	1	1	1
	0	0	0	0	-2	0	0	0	-2	-2	-2	-2	1	1	1	1
	0	0	0	0	-2	0	0	0	-2	-1	-2	-4	1	1	1	1
	0	0	0	0	-1	0	0	0	-1	-1	-4	-4	1	1	1	1
Pairs	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	S_3	S_4	S_1	S_6	S_6	S_7	S_7
	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	S_1	$\emptyset$	$\emptyset$	$\emptyset$	S_2	S_3	S_1	S_1	S_6	S_6	S_7	S_7
	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	S_1	$\emptyset$	$\emptyset$	$\emptyset$	S_2	$\emptyset$	S_1	S_5	S_6	S_6	S_7	S_7
	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	S_4	S_5	S_6	S_6	S_7	S_7

$$S_1 = \{1\}; \quad S_2 = \{2\}; \quad S_3 = \{3\}; \quad S_4 = S_1 \cup \{4\} = \{1,4\};$$
$$S_5 = S_1 \cup \{5\} = \{1,5\}; \quad S_6 = S_3 \cup S_5 = \{1,3,5\}; \quad S_7 = S_2 \cup S_4 = \{1,2,4\}$$

Fig. 3. Symbolic processing of an initial state.

required to recover the master key and the number of possible master keys. The pseudocode for the algorithm is given in the full version of this paper.

Figure 3 gives a graphical representation of how the algorithm works when only the first byte of the initial state is variable and known, while the other bytes are fixed and unknown. By attacking the result of the SubByte transformation applied to the first byte of the state in the first round, the key byte k_0 is recovered. This recovered key byte allows a carry of the attack into the second round where four key bytes $(k'_{16}, k'_{17}, k'_{18}, k'_{19})$ can be recovered by attacking the result of the SubBytes transformation. Because the attacker cannot distinguish between k'_{17} and k'_{18}, a new pair $S_1 = \{1\}$ is added to the corresponding state bytes. Then, the attacker targets the third round, where she can recover temporary key bytes for all state bytes. The pair S_1 from previous round affects all bytes of the third and fourth column of the state and thus the corresponding pairs are

Table 3. Possible attack outcomes for different number of bytes (**Bytes**) controlled by attacker. **Rnds** is the number of rounds that have to be attacked in order to recover the master key. **Prop. (%)** is the proportion of a given evaluation case with respect to all possible input configurations for a fixed number of bytes controlled by attacker.

Bytes	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
min(Rnds)	4	4	4	3	3	3	3	3	3	3	3	3	3	3	3	1
Prop. (%)	100	100	100	14.1	35.2	55.9	72.7	84.7	92.3	96.7	98.9	99.8	100	100	100	100
max(Rnds)	4	4	4	4	4	4	4	4	4	4	4	4	3	3	3	1
Prop. (%)	100	100	100	85.9	64.8	44.1	27.3	15.3	7.7	3.3	1.1	0.2	100	100	100	100
Trade-off	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗

updated accordingly. In addition, new pairs are added when the attacker can not distinguish between key candidates as shown in Fig. 3. In the fourth round, the attacker is able to recover all round key bytes. Then, having all the round key bytes of the fourth round, she can reverse the AES key schedule to get the master key.

The attacker has to build 2^p possible round keys, where p is the number of independent pairs associated with the state bytes of the last attacked round. For the example in Fig. 3, the number of possible keys is 2^5 because $card(S) = card(S_6 \cup S_7) = card(\{1, 2, 3, 4, 5\}) = 5$. Thus, in addition to the number of rounds to attack, the algorithm for symbolic processing of an initial state gives the number of possible master keys to be recovered by an attacker. Though, the

Table 4. All evaluation cases with an example of a possible initial state for each evaluation case. **Bytes** gives the number of bytes controlled by attacker; **Rounds** gives the number of rounds that have to be attacked to recover the master key.

Case	Bytes	Rounds	Possible initial state
0	1	4	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
1	2	4	[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
2	3	4	[1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
3	4	3	[1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4	4	4	[1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
5	5	3	[1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
6	5	4	[1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
7	6	3	[1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
8	6	4	[1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
9	7	3	[1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
10	7	4	[1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]
11	8	3	[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
12	8	4	[1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0]
13	9	3	[1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]
14	9	4	[1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0]
15	10	3	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]
16	10	4	[1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0]
17	11	3	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
18	11	4	[1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0]
19	12	3	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
20	12	4	[1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1]
21	13	3	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]
22	14	3	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0]
23	15	3	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]
24	16	1	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

attacker does not have to check all 2^p candidates to see which one is the correct one since she can discard the wrong candidates based on the difference between the correlation coefficients of the first two key candidates as we will show in Sect. 5.

Using the algorithm for symbolic processing of an initial state we evaluated all possible input combinations. More precisely, we considered all configurations of the initial state when the attacker controls i bytes of the input for $i \in [1, 16]$. When the attacker controls i bytes, there are $\binom{16}{i}$ possible input configurations. This results in $2^{16} - 1$ possible configurations of the initial state in total. Then, we classified these inputs into equivalence classes (evaluation cases) depending on the number of rounds that must be attacked in order to recover the master key. The results are summarized in Table 3. When the attacker controls between four and eleven bytes of the input, a trade-off between the input configuration and the number of rounds to be attacked is possible. When this is the case, the proportion of possible input configurations shows which evaluation case is more likely to appear if the initial state is chosen at random. Thus, when the attacker controls only four or five bytes of the input, it is crucial to carefully choose an input configuration from the limited set of possible input configurations that minimize the number of rounds to be attacked.

We give an example of a possible initial state for each of the 25 distinct evaluation cases identified after processing all possible input combinations in Table 4. Any possible input configuration for the AES encryption falls into one of these evaluation cases depending on the number of bytes controlled by attacker and the number of rounds that must be attacked in order to recover the master key.

5 The Attack

The attack we present in this section uses the symbolic representation of the AES state (described in Sect. 4) in conjunction with CPA attacks to recover individual bytes of the AES round keys. After executing Algorithm 1, the attacker has all round key bytes of round R. Thus, she is able to recover the master key of the cipher by reversing the key schedule.

The algorithm follows the symbolic representation of the state to infer which key bytes must be attacked and how many key candidates it should yield for each attacked key byte. By tracking the pairs associated with the recovered key bytes, the algorithm is able to discard all impossible round keys, thus saving computational resources. Indeed, the algorithm uses an optimal number of CPA attacks to recover the master key.

Initially, the set of known pairs is empty and all possible keys are considered valid. The algorithm keeps track of 2^p possible keys, where p is the total number of independent pairs in the symbolic representation of the state at round R.

The main loop of the algorithm runs through all rounds that must be attacked. At each round, the key bytes corresponding to variable state bytes are attacked to recover one or more temporary key bytes or a round key byte. Depending on the pairs associated with the byte to be attacked, there are three possible cases:

Algorithm 1. The attack

```
Require: $state$                                   ▷ Initial state: 0 – fixed byte, 1 – variable byte
Require: $\lambda = (plaintexts, traces)$          ▷ Recorded in the acquisition phase
$state, pairs = \textsc{Process}(state)$           ▷ Symbolic processing (Sect. 4)
$known\_pairs = \emptyset, \quad mapped\_pairs = \emptyset$
$keys[2^p] = \emptyset, \quad valid\_keys[2^p] = True$   ▷ $p$ is the number of independent pairs
for $r = 1$ to $R$ do                              ▷ $R$ is the number of rounds to be attacked
    for $i = 0$ to $15$ do
        if $state[r][i] \neq 0$ then
            if $pairs[r][i] == \emptyset$ then                                  ▷ No pair
                $keys[0, \cdots, 2^p - 1][\mathrm{r}][i] = \mathsf{CPA}(\lambda, keys[0], r, i)$
            else if $pairs[r][i] \subseteq known\_pairs$ then                  ▷ Known pair(s)
                if $i \notin mapped\_pairs[pairs[r][i]]$ then
                    $mask = 0, \quad temp\_keys = \emptyset, \quad \alpha_{max} = -1$
                    for $pair \in pairs[r][i]$ do
                        $mask = mask \vee 2^{pair-1}$
                    end for
                    for $j \in [0, 2^p - 1]$ do
                        if $valid\_keys[j]$ and $temp\_keys[j \wedge mask] == \emptyset$ then
                            $temp\_keys[j \wedge mask], \alpha = \mathsf{CPA}(\lambda, keys[j], r, i)$
                            if $\alpha > \alpha_{max}$ then
                                $\alpha_{max} = \alpha$
                            end if
                        end if
                        $valid\_keys[j][r][i] = temp\_keys[j \wedge mask]$
                    end for
                    for $j \in [0, 2^p - 1]$ do
                        if $abs(state[r][i]) == 1$ and $\alpha + \beta < \alpha_{max}$ then
                            $valid\_keys[j] = False$
                        end if
                    end for
                end if
            else                                                                ▷ New pair
                $mask = 2^{pairs[r][i] - new\_pair}, \quad k_1 = k_2 = \emptyset$
                for $j \in [0, 2^p - 1]$ do
                    if $k_1[j \wedge mask] == \emptyset$ then
                        $k_1[j \wedge mask], k_2[j \wedge mask] = \mathsf{CPA}(\lambda, keys[j], r, i)$
                    end if
                    if $j \wedge 2^{new\_pair-1}$ then
                        $keys[j][r][i] = k_1[j \wedge mask], \quad keys[j][r][i'] = k_2[j \wedge mask]$
                    else
                        $keys[j][r][i'] = k_2[j \wedge mask], \quad keys[j][r][i'] = k_1[j \wedge mask]$
                    end if
                end for
                $known\_pairs = known\_pairs \cup new\_pair$
                Add $(i, i')$ to $mapped\_pairs[new\_pair]$
            end if
        end if
    end for
end for
return $keys[i]$, where $valid\_keys[i] == True$ for $i \in [0, 2^p - 1]$
```

- **No pair**: If the symbolic representation does not have a pair associated with the byte of the state to be used for the attack, then the algorithm will recover a single key byte which is distributed to all possible keys.
- **New pair**: If one of the pairs associated with the byte under attack is not present in the set of known pairs, then the algorithm will recover 2^u possible values for the corresponding key byte, where u is the number of known independent pairs associated with the byte under attack. The number of known pairs determines the number of CPA attacks to be performed. Using a mask based on the existing pairs and a mask for the new pair, the algorithm correctly distributes the recovered key byte values to all possible keys. The new pair is added to the set of known pairs and the two indexes of the state affected by the recovered temporary keys are mapped to this new pair. This mapping prevents the computation of the same temporary keys twice.
- **Known pairs(s)**: In the case where the t independent pairs associated with the key byte to be attacked are known but not mapped to the current state byte, the algorithm performs 2^t CPA attacks. Then, it distributes the attack results (the recovered key and the difference between the correlation coefficients of the first two most likely key candidates α) to the corresponding bytes of all possible keys. Afterwards, the possible keys for which the value of α is less than the maximum observed value α_{max} minus a threshold β are marked as invalid. In this way, only the combination of keys yielding the highest correlation peak is selected. At this moment, the input pairs are solved in the sense that the algorithm can uniquely assign each of the two temporary keys of a pair to the corresponding state bytes. As a consequence, the algorithm will not further process the possible keys marked as invalid. Thus, this optimization improves the algorithm efficiency by reducing the number of performed CPA attacks.

Finally, the algorithm returns all possible keys which are marked as valid. If the threshold β tends to zero, the algorithm will return only one possible key. When the quality of the side-channel acquisition is good (i.e. high signal-to-noise ratio) and there are enough power traces, the algorithm yields the correct key.

5.1 Optimality

We prove that our algorithm uses the minimum number of CPA attacks possible to recover the master key and thus is optimal. Hence, the lower bounds provided in Table 5 are optimal.

Theorem 1. *Algorithm 1 performs an optimal number of CPA attacks to recover the 16-byte master key of the AES.*

Proof. The only way an attacker can recover the 16-byte master key of the AES is to recover all key bytes of a round r and then to reverse the key schedule. Since the function that derives the round keys of round i from the round keys of round $i-1$ is bijective, knowledge of all round key bytes of a round r leads to the knowledge of the master key.

Let us assume that Algorithm 1 uses n individual CPA attacks for a given initial state and it is not optimal. Thus, there exists at least an algorithm that is able to recover the master key using only m CPA attacks, with $m < n$. We show next that such an algorithm does not exist. If there exists an algorithm that uses less CPA attacks than Algorithm 1, then this algorithm attacks at least one key byte less. But if it does so, then the attack can not be carried into later rounds any more because the state byte corresponding to the unrecovered key yields unknown and variable state bytes after `MixColumns` transformation. These bytes can not be recovered using a CPA attack and thus the attack fails. As a consequence, there is no algorithm that uses less CPA attacks than Algorithm 1. □

5.2 Choosing the Best Attack Strategy

For up to seven bytes controlled by attacker, our attack algorithm (Algorithm 1) is more efficient than the classic attack algorithm where all possible key bytes are attacked to recover the master key. The gain varies between 15% and 68% of the number of CPA attacks required by the classic attack. When an attacker has control of more than seven input bytes, our algorithm performs the same number of CPA attacks as the classic attack. At the same time, our algorithm gives a unique master key, provided that there are available enough traces with a high signal-to-noise ratio. This is not the case for a classic attack unless an additional mechanism to discard invalid keys, as the one in Algorithm 1, is employed.

An attacker willing to reduce the duration of the offline phase of the attack (without increasing the number of rounds that must be attacked) can use the results in Table 5 in corroboration with the data in Table 3 to adjust the attack accordingly. More precisely, if an attacker is able to control up to n bytes of the AES input, she can choose to control m ($m \leq n$) bytes of the input because m variable bytes minimize the number of CPA attacks required to recover the master key. This decision has to be made before performing the side-channel acquisition since it influences the chosen inputs. Another argument in favor of

Table 5. The number of individual CPA attacks required to recover the master key for different number of bytes (**Bytes**) controlled by attacker; **min(Rnds)/max(Rnds)** and **Bytes** precisely identify the evaluation case. **Classic attack** does not use the optimizations introduced in **Algorithm 1** to discard the invalid keys. **Gain** gives the number of CPA attacks saved by an attacker using **Algorithm 1** over an attacker using **Classic attack**.

Bytes		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
min(Rnds)	Clasic attack	150	104	188	80	66	52	46	40	41	42	43	44	45	46	47	16
	Algorithm 1	48	42	48	38	38	38	39	40	41	42	43	44	45	46	47	16
	Gain	102	62	140	42	28	14	7	0	0	0	0	0	0	0	0	0
max(Rnds)	Classic attack	150	104	188	110	72	50	51	52	53	54	55	56	45	46	47	16
	Algorithm 1	48	42	48	48	49	50	51	52	53	54	55	56	45	46	47	16
	Gain	102	62	140	62	23	0	0	0	0	0	0	0	0	0	0	0

using less variable input bytes is that the attack is much more difficult to detect if the injected packets have fewer variable bytes and mimic the appearance of a normal network traffic. For example, when $n = 12$, an attacker can choose $m = 4, 5, \text{or } 6$ to reduce the complexity of the offline attack from 44 to 38 individual CPA attacks, while still attacking just three rounds. The result is an overall improvement of the attack efficiency by 14% over the classic attack.

An even better decision can be made with the help of experimental results for different configurations of the input from a similar target to the one to be attacked in addition to the results presented so far. For this reason, in the next section we determine experimentally the number of traces required to recover the master key for each evaluation case using EM leakages from an ARM Cortex-M3 processor.

6 Results

For the experimental evaluation, we considered two unprotected implementations of the AES written in ANSI C. The first implementation uses the S-box implementation strategy, while the second one uses the T-table implementation strategy. For each of the 25 evaluation cases we measured up to 2000 EM traces. The acquisition took about 90 min for an evaluation case. The samples were split into files corresponding to the AES round number. Then, we mounted the attack presented in Algorithm 1 using an increasing number of traces in the interval $[100, 2000]$ with a step of 100 traces until the guessing entropy converged to zero.

For each implementation we considered two selection functions based on the AES S-box and T-table, respectively. The minimum number of traces for which the guessing entropy becomes zero and remains stable is pictorially shown in Fig. 4 for each evaluation case. All attacks recovered the full 16-byte master key with less than 1600 EM traces. In general, the master key was recovered with fewer traces when the selection function perfectly matched the implementation strategy. Though, our results show that full key recovery is possible even when the selection function does not perfectly match the attacked implementation. The attacks on the S-box implementation using the T-table selection function needed 204 more traces on average to recover the master key compared to the attacks on the same implementation using the S-box selection function. Similarly, using the S-box selection function instead of the T-table selection function to attack the implementation based on the T-table strategy required 354 more traces on average. For details on the exact number of traces required to recover the master key for each evaluation case and attack scenario we refer the reader to the full version of this paper.

Countermeasures. Our experimental results show that side-channel countermeasures such as masking must be employed in order to protect the AES implementations based on lookup tables (S-box and T-table implementation strategies) even in a communication protocol scenario, when the adversary has

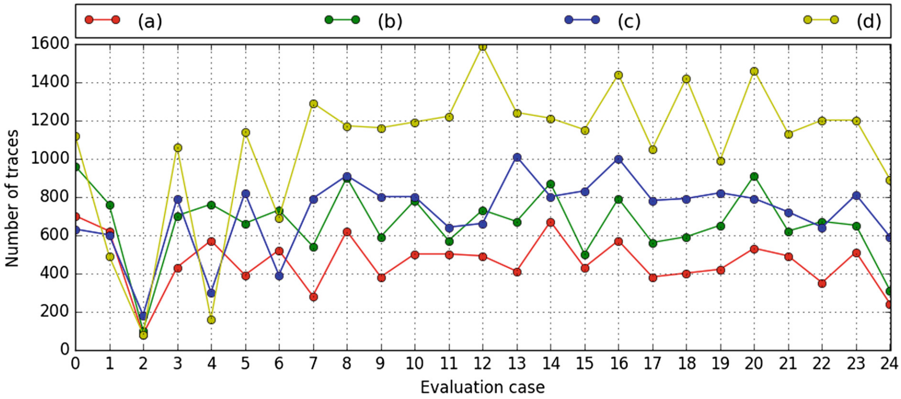


Fig. 4. The number of EM traces required to fully recover the master key. Scenarios: **(a)** S-box implementation, S-box selection function; **(b)** S-box implementation, T-table selection function; **(c)** T-table implementation, T-table selection function; **(d)** T-table implementation, S-box selection function.

a limited control of the input. Masking non-linear lookup tables is a challenging task since it adds a considerable penalty on execution time and memory usage [33].

Although not present in many cryptographic libraries due to their limitations (i.e. can not be used in a feedback mode of operation such as CCM), the bitsliced implementations have a lower CPA leakage than implementations using lookup tables [4], but they are still vulnerable to DPA attacks [3].

A lightweight primitive (block cipher or authenticated encryption), particularly one designed for efficient masking, is a good replacement for the AES-CCM when considering side-channel protection.

Other countermeasures, such as a key refreshing mechanism, can support a defense in depth approach. However, any additional countermeasure affects the overall efficiency of an IoT protocol and consequently the most effective ones (i.e. masking) must have priority given the resource constraints.

7 Conclusions

In this paper, we presented an extensive security analysis of AES software implementations against CPA attacks in the context of network protocols. In this scenario the attacker has control of several input bytes, while the remaining input bytes are fixed. To asses the resilience of AES implementations to all possible input combinations, we presented an algorithm for symbolic processing of the cipher state. Then, we classified all possible inputs into 25 independent evaluation cases depending on the number of input bytes controlled by attacker and the number of rounds that must be attacked to recover the master key. Finally, we described an optimal algorithm that recovers the master key by mounting the minimum number of CPA attacks possible. It makes clever decisions based on the set of key pairs that affects the key byte under attack and the correlation coefficient of possible key candidates to discard impossible keys.

We showed that unprotected implementations of the AES based on the S-box and T-table strategies can be broken even when the attacker controls only one input byte of the cipher with less than 1600 electromagnetic traces acquired from a 32-bit ARM Cortex-M3 processor in about one hour. Knowledge of the implementation strategy does not significantly improve the attack outcome, nor does it reduce the attack complexity. Thus, unprotected implementations of the AES should not be used to secure the communication between end devices in network protocols. Care must be taken when using implementations of the AES from popular open-source cryptographic libraries since most of them are not protected against side-channel attacks.

Acknowledgements. We would like to thank the ACNS 2017 reviewers for their valuable feedback and Johann Großschädl for proofreading the final version of this paper. The work of Daniel Dinu is supported by the CORE project ACRYPT (ID C12-15-4009992) funded by the Fonds National de la Recherche (FNR) Luxembourg.

References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM side-channel(s). In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003). doi:10.1007/3-540-36400-5_4
2. ARM. mbed TLS. https://tls.mbed.org/. Accessed Apr 2017
3. Balasch, J., Gierlichs, B., Reparaz, O., Verbauwhede, I.: DPA, bitslicing and masking at 1 GHz. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 599–619. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48324-4_30
4. Biryukov, A., Dinu, D., Großschädl, J.: Correlation power analysis of lightweight block ciphers: from theory to practice. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 537–557. Springer, Cham (2016). doi:10.1007/978-3-319-39555-5_29
5. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28632-5_2
6. cryptlib. The cryptlib Security Software Development Toolkit. http://www.cryptlib.com/. Accessed Apr 2017
7. Crypto++. Crypto++: a free C++ class library of cryptographic schemes. https://www.cryptopp.com/. Accessed Apr 2017
8. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
9. Dworkin, M.J.: Recommendation for block cipher modes of operation: the CCM mode for authentication and confidentiality. NIST Special Publication 800-38C (2007)
10. GitHub. libtomcrypt: a fairly comprehensive, modular and portable cryptographic toolkit. https://github.com/libtom/libtomcrypt. Accessed Apr 2017
11. GitHub. mbed TLS - An open source, portable, easy to use, readable and flexible SSL library. https://github.com/ARMmbed/mbedtls/blob/development/library/aes.c. Accessed Apr 2017

12. GitHub. OpenSSL - TLS/SSL and crypto library. https://github.com/openssl/openssl/blob/master/crypto/aes/aes_core.c. Accessed Apr 2017
13. Hofemeier, G., Chesebrough, R.: Introduction to intel AES-NI and intel secure key instructions. Technical report. https://software.intel.com/sites/default/files/m/d/4/1/d/8/Introduction_to_Intel_Secure_Key_Instructions.pdf. Accessed Apr 2017
14. Housley, R.: Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP). RFC 4309, December 2005. https://tools.ietf.org/html/rfc4309
15. IEEE. IEEE Standard for Low-Rate Wireless Networks. https://standards.ieee.org/about/get/802/802.15.html
16. Jaffe, J.: A first-order DPA attack against AES in counter mode with unknown initial counter. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 1–13. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74735-2_1
17. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). doi:10.1007/3-540-48405-1_25
18. Libgcrypt. Libgcrypt: a general purpose cryptographic library based on the code from GnuPG. https://www.gnu.org/software/libgcrypt/. Accessed Apr 2017
19. libsodium. The Sodium crypto library (libsodium). https://download.libsodium.org/doc/. Accessed Apr 2017
20. Lipp, M., Gruss, D., Spreitzer, R., Maurice, C., Mangard, S.: Armageddon: cache attacks on mobile devices. In: Holz, T., Savage, S. (eds.) 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, 10–12 August 2016, pp. 549–564. USENIX Association (2016)
21. LoRa Alliance. Wide Area Networks for IoT. https://www.lora-alliance.org/. Accessed Apr 2017
22. Nettle. Nettle - a low-level cryptographic library. http://www.lysator.liu.se/nisse/nettle/. Accessed Apr 2017
23. NIST. Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197 (2001)
24. O'Flynn, C., Chen, Z.: Power Analysis Attacks Against IEEE 802.15.4 Nodes. In: Standaert, F.-X., Oswald, E. (eds.) COSADE 2016. LNCS, vol. 9689, pp. 55–70. Springer, Cham (2016). doi:10.1007/978-3-319-43283-0_4
25. OpenSSL. Cryptography and SSL/TLS Toolkit. https://www.openssl.org/. Accessed Apr 2017
26. Osvik, D.A., Shamir, A., Tromer, E.: Cache attacks and countermeasures: the case of AES. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 1–20. Springer, Heidelberg (2006). doi:10.1007/11605805_1
27. Randombit. mbed TLS. https://botan.randombit.net/. Accessed Apr 2017
28. Saab, S., Rohatgi, P., Hampel, C.: Side-channel protections for cryptographic instruction set extensions. Cryptology ePrint Archive, Report 2016/700 (2016). http://eprint.iacr.org/2016/700
29. Sastry, N., Wagner, D.: Security considerations for IEEE 802.15.4 networks. In: Jakobsson, M., Perrig, A. (eds.) Proceedings of the 2004 ACM Workshop on Wireless Security, Philadelphia, PA, USA, 1 October 2004, pp. 32–42. ACM (2004)
30. Schwabe, P., Stoffelen, K.: All the AES you need on Cortex-M3 and M4. In: Selected Areas in Cryptography-SAC (2016)
31. Song, J., Poovendran, R., Lee, J., Iwata, T.: The AES-CMAC algorithm. RFC 4493, June 2006. https://tools.ietf.org/html/rfc4493
32. STMicroelectronics. STM32 MCU Nucleo. http://www.st.com/en/evaluation-tools/stm32-mcu-nucleo.html. Accessed Apr 2017

33. Vadnala, P.K.: Time-memory trade-offs for side-channel resistant implementations of block ciphers. In: Handschuh, H. (ed.) CT-RSA 2017. LNCS, vol. 10159, pp. 115–130. Springer, Cham (2017). doi:10.1007/978-3-319-52153-4_7
34. Whiting, D., Housley, R., and N. Ferguson. Counter with CBC-MAC (CCM). RFC 3610, September 2003. https://tools.ietf.org/html/rfc3610
35. wolfSSL. wolfCrypt Embedded Crypto Engine. https://www.wolfssl.com/wolfSSL/Products-wolfcrypt.html. Accessed Apr 2017

Cryptographic Protocol

Lattice-Based DAPS and Generalizations: Self-enforcement in Signature Schemes

Dan Boneh(✉), Sam Kim, and Valeria Nikolaenko

Stanford University, Stanford, USA
dabo@cs.stanford.edu

Abstract. Double authentication preventing signatures (DAPS) is a mechanism, due to Poettering and Stebila, for protecting certificate authorities (CAs) from coercion. We construct the first lattice-based DAPS signatures, thereby providing the first post-quantum DAPS system. We go further and generalize DAPS to a more general mechanism we call *predicate* authentication preventing signatures (PAPS). Here, for a given k-ary predicate ϕ, a PAPS system for ϕ is regular signature scheme. However, if the signer ever signs k messages $m_1, \ldots, m_k$ such that $\phi(m_1, \ldots, m_k)$ is true then these k signatures reveal the signer's secret key. This self-enforcement mechanism incentivizes the signer to never sign conflicting messages, namely messages that satisfy the predicate ϕ. The k conflicting messages can be signed at different times and the signatures may be generated independently of one another. We further generalize to the case when the signatures are generated by multiple signers. We motivate these primitives, give precise definitions, and provide several constructions. These primitives are challenging to construct and give rise to many new elegant open research questions.

1 Introduction

Suppose a web site such as facebook.com buys certificates for its domain from a certificate authority (CA) called XYZ. These certificates enable web browsers to establish a (one-sided) authenticated session with facebook.com. Sometime later, a law enforcement agency or a nation state that has jurisdiction over the CA compels XYZ to secretly issue a fresh certificate for facebook.com. The CA has no choice but to comply. The agency can then use this issued certificate in a man-in-the-middle attack on facebook.com. Web users have no way to detect that this is happening and that their traffic is being intercepted. We emphasize that the rogue certificate is issued by the same CA from which facebook.com normally buys its certificates. The only user-side signal is that a previously unseen public-key is being served in a facebook.com certificate, but this happens frequently under normal operation at a large site and would not generally look suspicious.

Some technologies, such as certificate transparency (CT) [LLK15] as well as CONIKS [MBB+15], are designed to detect situations where a CA such as XYZ issues a fake certificate for a domain. These technologies empower an origin domain, facebook.com in this case, to detect that a fake certificate was issued for its domain.

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 457–477, 2017.
DOI: 10.1007/978-3-319-61204-1_23

Poettering and Stebila [PS14] proposed a very different defense against the scenario described above. Their idea, called double-authentication-preventing signatures, or DAPS for short, is as follows: suppose XYZ signs all its certificates using a signature scheme where the signing algorithm uses the secret signing key sk to sign a pair $(\mathsf{subj}, \mathsf{payload})$. Here subj is the domain-name to which the certificate is issued and $\mathsf{payload}$ is all other fields in the certificate. The resulting signature σ can be verified as a standard digital signature. The key property of DAPS is the following: suppose XYZ publishes two valid signatures σ_1 and σ_2 *for the same* subj *but for different payloads*, say one on $(\mathsf{subj}, \mathsf{payload}_1)$ and another on $(\mathsf{subj}, \mathsf{payload}_2)$. Then these two signatures enable anyone to expose XYZ's secret signing key sk. The point is that XYZ can argue that it should not be forced to issue the rogue certificate for facebook.com because that would expose its signing key thereby causing massive collateral damage to *all* of XYZ's customers. Whether this argument is effective remains to be seen, but the idea itself is interesting and, as we show below, leads to interesting and challenging cryptographic questions.

How to Use DAPS. There are many practical holes in the basic DAPS proposal described above that prevent it from being used as is, but with a bit of thought they can be addressed. However, our goal here is not to argue that DAPS will be deployed in practice, but rather to motivate this as an interesting cryptographic question. Towards this goal, we examine broader applications of DAPS as well as an elegant generalization.

Other Applications for DAPS. Beyond certificates, DAPS can be a useful "self-enforcement" security mechanism. For example, suppose Eve owns a certain patent and wants to sell the rights to the patent. Bob wants to buy the patent from Eve, but he is worried that Eve will sell the patent to multiple people. Using DAPS, Eve can use the patent number as the subject and use "owned by Bob" as the payload. If she tries to sell the same patent to two different people she will end up signing two pairs of messages with the same subject, but different payload. The resulting two signatures can be combined to expose Eve's private key. If this private key is of high value to Eve then this self-enforcement mechanism will prevent her from double-selling the same patent to two people. This way, Bob has some confidence in the exclusivity of the deal with Eve.

PAPS: DAPS for General Predicates. The previous paragraph motivates a more general elegant primitive which we call *predicate-authentication-preventing signatures*, or PAPS for short. Let $\mathcal{M}$ be a message space and let $\phi : \mathcal{M}^k \to \{0,1\}$ be a predicate. A PAPS scheme for ϕ lets the signer sign any message $m \in \mathcal{M}$, just as in a regular signature scheme. However, if over the life of the secret key, the signer signs messages $m_1, \ldots, m_k \in \mathcal{M}$ such that $\phi(m_1, \ldots, m_k) = 1$ then these k signatures can be combined to expose the signer's secret signing key. This secret key extraction should work no matter how the k signatures are generated: as long as all k signatures are valid, it should be possible to extract the secret key. For security, as long as the predicate ϕ is never satisfied, the signature scheme should be existentially unforgeable under a chosen message attack, as for regular signatures.

Notice that DAPS is a special case of PAPS: the key space $\mathcal{M}$ is a set of pairs $\mathcal{M} = \mathcal{S} \times \mathcal{P}$ and the predicate ϕ_{DAPS} is simply the 2-ary predicate:

$$\phi_{\text{DAPS}}\big(\ (x_1, y_1),\ (x_2, y_2)\ \big) = 1 \quad \Leftrightarrow \quad x_1 = x_2 \text{ and } y_1 \neq y_2.$$

More general predicates come up naturally. For example, suppose a web site owns k machines and it wants to generate a different key-pair for each machine, necessitating a different certificate for each machine. The analogue of DAPS is a k-way DAPS where the message space $\mathcal{M}$ is again $\mathcal{M} = \mathcal{S} \times \mathcal{P}$, but now the predicate ϕ is the $(k+1)$-ary predicate

$$\phi\big(\ (x_1, y_1), \dots, (x_{k+1}, y_{k+1})\ \big) = 1 \quad \Leftrightarrow \quad \begin{Bmatrix} x_1 = \cdots = x_{k+1} \text{ and} \\ y_1, \dots, y_{k+1} \text{ are all distinct.} \end{Bmatrix}$$

This lets the site use a different certificate for each of its k machines, but if another certificate is issued then the CA's secret key is exposed. We give a construction for this predicate in Sect. 5 as well as for several other predicates.

Proving security of a PAPS construction is non trivial. For example, suppose the message space is $\mathcal{M} = \mathbb{F}_p$ for some prime p. Consider the 3-ary predicate ϕ defined as $\phi(x, y, z) = 1$ if and only if $x + y + z = 0$. That is, the secret key should leak if the signer ever signs three messages whose sum is zero. However, if the signer never signs three messages satisfying this condition, the signature scheme should be existentially unforgeable. To prove existential unforgeability, the simulator must interact with the adversary, answering all the adversary's adaptive signature queries, and using the adversary's existential forgery to solve a challenge problem. The problem is that, because the adversary's first two queries can be for arbitrary messages, the simulator must be prepared to provide a signature for *all* messages $m \in \mathcal{M}$. In particular, the simulator will know the signature on three messages x, y, z satisfying $x + y + z = 0$. But then the simulator can extract the secret signing key, and can produce any forgery by itself, meaning that the adversary is not helping the simulator. Nevertheless, in Sect. 5 we are able to prove security for several generalized predicates, though not for the 3-way summation predicate.

A Further Generalization: Multi-signer PAPS. We can further generalize the notion of PAPS to the setting of k signers where each signer has its own signing/public key-pair. As before, let ϕ be a k-ary predicate $\phi : \mathcal{M}^k \to \{0, 1\}$. Suppose that for $i = 1, \dots, k$ signer number i signs message $m_i \in \mathcal{M}$. Then, if $\phi(m_1, \dots, m_j) = 1$, then these k signatures (along with the k messages and k public keys) can be used to expose some secret s chosen at setup time.

Multi-signer PAPS come up naturally when considering certificates. Suppose that the agency, instead of asking XYZ to issue the rogue certificate for facebook.com, it asks a different CA to provide a certificate for facebook.com. We are now in a 2-signer scenario. If the predicate ϕ_{DAPS} can be made to work on the two signatures, despite them being from different CAs, then going to a different CA will not help the agency. We define multi-signer PAPS in Sect. 6 where we also give several constructions.

1.1 Contributions

In this work we build a lattice-based DAPS construction based on Short Integer Solutions (SIS) problem and the Learning with Errors (LWE) problem. Our construction builds upon the structure of the fully homomorphic signature scheme of Gorbunov et al. [GVW15]. In their construction, a signature consists of a preimage of a specially formed target matrix of a lattice trapdoor function. To make key leakage a feature rather than a form of insecurity, we carefully hash the messages to derive the target matrix such that two different message of the same subject leads to two matrices for which two preimage matrices leak a trapdoor. As in [PS14], we prove security in the random oracle model.

Also, as we discuss above, we extend DAPS to a more general primitive that we call predicate authentication preventing signatures (PAPS). In this setting, signatures of any messages that satisfy a certain predicate defined on these messages leak a signer's secret key. To motivate the notion, we show that for certain simple, but useful predicates, PAPS can already be constructed from DAPS.

Finally, we further extend PAPS to a multi-authority settings where signatures from different signers can also leak some shared private information. We give formal definitions in this setting and show that our lattice DAPS construction can be extended to this setting as well using the property that two short preimages of a specifically formed target matrix of lattice trapdoor functions can be merged to give a trapdoor of an extended lattice.

1.2 Related Work

Previous Works on DAPS. The notion of double-authentication-preventing signatures was introduced by Poettering and Stabila [PS14]. They provide a construction based on extractable trapdoor functions that can be constructed using the group of quadratic residues modulo a Blum integer. Subsequently, Bellare, Poettering, and Stebila [BPS15] gave a generic construction based on trapdoor identification schemes where the private randomness committed by the prover can be extracted using a trapdoor. We note that although lattice-based identification schemes have appeared in the literature [Lyu08,Lyu12], the construction from [PS14] does not directly give a lattice-based DAPS construction since lattice trapdoors are randomized with multiple preimages. Constructing DAPS from lattice-based assumptions is an interesting and important goal since they provide hardness even against quantum computers, a setting for which the previous two works do not provide security.

Delegating Restricted Signing Keys. A number of works in the literature have focused on schemes that allow an authority to delegate signing keys with some restricted functionalities (with function privacy). These include attribute-based signatures [MPR11], functional signatures [BGI14] and policy-based signature [BF14]. In these schemes, a signer is restricted to sign only certain messages that satisfy a predicate requirement. One difference between these notions and DAPS/PAPS is that in the former, the restriction is done by a central authority to restrict other signers, while in the latter, the authority restricts itself as a self-enforcement mechanism. Another major difference is that in the former, the restriction is determined with respect to each individual message while in the latter, the restriction is determined by all of the past messages that the signer signs, which is what makes DAPS and PAPS an interesting theoretical notion.

Ring/Group Signatures. A similar notion of double-signing preventing mechanism exists in the setting of group signatures [TX03] and ring signatures [RST01, BKM06] called Revocable-iff-Linked (RiffL) signatures [ALSY06]. In this setting, a signer can sign on behalf of a group; however, if it signs twice or more, then the identity of the signer is leaked. As in the discussion of the previous paragraph, one difference in the DAPS setting compared to RiffL is that DAPS is a self-enforcing mechanism, which means that the linkability is not enforced by another trusted authority of the system, but by itself. However, the more fundamental difference is that in DAPS, the act of double-signing immediately gives away the signing key or private data rather than simply leaking the information that it double signed. As was discussed in [PS14], there are instances where simply leaking the fact that a CA double signed may not be enough of a penalty (i.e. the 2011 Comodo incident[1]) and DAPS is designed to cope with these type of situations.

2 Preliminaries

Basic Notation. For an integer N, we write $[N]$ to denote the set $\{1, ..., N\}$. We use bold lowercase letters (e.g., $\mathbf{x}, \mathbf{w}$) to denote vectors and bold uppercase letters (e.g., $\mathbf{A}, \mathbf{G}$) to denote matrices. For a matrix $\mathbf{A}$, we use $\mathbf{A}^T$ to denote the trasnpose of $\mathbf{A}$ and for a vector $\mathbf{x}$, we use $\|\mathbf{x}\|$ to denote its Euclidean norm. In general, we write λ for the security parameters. We say a function $\epsilon(\lambda)$ is negligible in λ, if $\epsilon(\lambda) = o(1/\lambda^c)$ for every $c \in \mathbb{N}$, and we write $\mathsf{negl}(\lambda)$ to denote a negligible function in λ. We say that an event occurs with *negligible probability* if the probabilty of the event is $\mathsf{negl}(\lambda)$, and an event occurs with *overwhelming probability* if its complement occurs with negligible probability.

Entropy and Statistical Distance. The *statistical distance* between two random variables X and Y over a finite domain Ω is defined as

$$\mathsf{SD}(X, Y) = \frac{1}{2} \sum_{\omega \in \Omega} \left| \Pr[X = \omega] - \Pr[Y = \omega] \right|.$$

We say that two distribution ensembles $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are *statistically indistinguishable*, denoted $\overset{\text{stat}}{\approx}$, if it holds that $\mathsf{SD}(X_\lambda, Y_\lambda)$ is negligible in λ. The *min-entropy* of a random variable X, denoted $\mathbf{H}_\infty(X|Y)$, is defined as

$$\mathbf{H}_\infty(X|Y) \overset{\text{def}}{=} -\log \left(\mathop{\mathbf{E}}_{y \leftarrow Y} \left[\max_x \Pr[X = x | Y = y] \right] \right)$$

[1] https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html.

The optimal probability of an unbounded adversary guessing X given the correlated value Y is $2^{-\mathbf{H}_\infty(X|Y)}$.

2.1 Circular Security

In this section we briefly recall the notion of circular security. A public key encryption (PKE) consists of three algorithms $\Pi_{\mathsf{pke}} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ where $\mathsf{PKE.KeyGen}$ takes in a unary representation of the security parameter λ and outputs a public and secret key pair $(\mathsf{pk}, \mathsf{sk})$. The encryption algorithm takes in a public key pk and a message m and generates a ciphertext ct. The decryption algorithm takes in a secret key sk and a ciphertext ct and outputs a message m. For correctness, we require that for all $\lambda \in \mathbb{N}$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, we have that $\mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{PKE.KeyGen}(\mathsf{pk}, m)) = m$ with overwhelming probability.

Definition 1 (Circular Security [CL01, BRS02]**).** *A public-key encryption scheme Π_{pke} is* circular secure *if for all efficient adversaries $\mathcal{A}$, there is a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Adv}^{\mathsf{circ}}_{\Pi_{\mathsf{pke}},\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left|\Pr[\mathsf{Expt}^{(0)}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{(1)}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where for each $b \in \{0, 1\}$, and $\lambda \in \mathbb{N}$, the experient $\mathsf{Expt}^{(b)}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ is defined as follows:

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$.
2. $\mathsf{ct}_0 \leftarrow \mathsf{PKE.Encrypt}(\mathsf{pk}, 0^{|\mathsf{sk}|})$.
3. $\mathsf{ct}_1 \leftarrow \mathsf{PKE.Encrypt}(\mathsf{pk}, \mathsf{sk})$.
4. $b' \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{ct}_b)$.
5. *Output* $b' \in \{0, 1\}$.

Circular security assumption on lattice-based cryptosystems have been used extensively throughout the literature [Gen09, BV11, BV14, BGV12, GSW13] with some positive results showing that some common forms of lattice-based encryption schemes can be shown to be circular secure [ACPS09, ASP12].

2.2 Broadcast Encryption

A broadcast encryption scheme BE [FN93] consists of a tuple of algorithms $\Pi_{\mathsf{be}} = (\mathsf{BE.KeyGen}, \mathsf{BE.Encrypt}, \mathsf{BE.Decrypt})$ defined as follows:

1. $\mathsf{BE.KeyGen}(1^\lambda, N) \rightarrow (\{\mathsf{sk}_i\}_{i\in[N]}, \mathsf{pk})$: On input the security parameter λ and a positive integer $N \in \mathbb{N}$, the key generation algorithm outputs a set of secret keys $\{\mathsf{sk}_i\}_{i\in[N]}$ and a public key pk.
2. $\mathsf{BE.Encrypt}(\mathsf{pk}, \mathsf{msg}, T) \rightarrow \mathsf{ct}_T$: On input a public key pk, a message m, and a set of intended recipients $T \subseteq [N]$, the encryption algorithm outputs a ciphertext ct_T.
3. $\mathsf{BE.Decrypt}(\mathsf{sk}_i, \mathsf{ct}) \rightarrow m'$: On input a secret key sk_i and a ciphertext ct, the decryption algorithm outputs a message m'.

Correctness. For correctness we require that for all $\lambda \in \mathbb{N}$, $N \in \mathbb{N}$, $T \subseteq [N]$, $(\{\mathsf{sk}_i\}_{i \in [N]}, \mathsf{pk}) \leftarrow \mathsf{BE.KeyGen}(1^\lambda, N)$, we have $\mathsf{BE.Decrypt}(\mathsf{sk}_i, \mathsf{BE.Encrypt}(\mathsf{pk}, \mathsf{ct}, T)) = m$ for all $i \in T$.

Security. For broadcast encryption scheme, we define the following security notion.

Definition 2. *A broadcast encryption scheme Π_{be} is secure if for all efficient adversaries $\mathcal{A}$, there is a negligible function* $\mathsf{negl}(\lambda)$ *such that*

$$\mathsf{Adv}_{\Pi_{\mathsf{be}},\mathcal{A}}(\lambda) \stackrel{\text{def}}{=} \left|\Pr[\mathsf{Expt}^{(0)}_{\mathsf{BE},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{(1)}_{\mathsf{BE},\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda),$$

where for each $b \in \{0,1\}$, and $\lambda \in \mathbb{N}$, the experiment $\mathsf{Expt}^{(b)}_{\mathsf{BE},\mathcal{A}}$ is defined as follows:

- $(\{\mathsf{sk}_i\}_{i \in [N]}, \mathsf{pk}) \leftarrow \mathsf{BE.KeyGen}(1^\lambda, N)$.
- $(T, m_0, m_1) \leftarrow \mathcal{A}_0(\mathsf{pk})$.
- $\mathsf{ct}_b \leftarrow \mathsf{BE.Encrypt}(\mathsf{pk}, m_b)$.
- $b' \leftarrow \mathcal{A}_1(\{\mathsf{sk}_i\}_{i \in [N] \setminus T}, \mathsf{ct}_b)$.
- *Output $b' \in \{0,1\}$.*

A number of constructions for broadcast encryption schemes have been proposed in the literature [FN93, BGW05, BWZ14] with short ciphertext where the length of the ciphertext scales sublinearly in the number of users in the system. For linear length ciphertext, a broadcast encryption scheme can be constructed generically from a regular public key encryption scheme by concatenating the N instances of the public key encryption schemes.

2.3 Background on Lattices

In this section, we describe some of the results and notations for lattice-based cryptography that are used throughout the paper.

Lattice and Gaussians. Let n, q, m be positive integers. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we let $\Lambda_q^\perp(\mathbf{A})$ denote the lattice $\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{0} \mod q\}$. More generally, for $\mathbf{u} \in \mathbb{Z}_q^n$, we let $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ denote the shifted lattice $\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{u} \mod q\}$.

Regev [Reg09] defined a natural distribution on $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ called a *discrete Gaussian* parameterized by a scalar $s > 0$. We use $\mathcal{D}_s(\Lambda_q^{\mathbf{u}}(\mathbf{A}))$ to denote this distribution. For a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $s > \widetilde{O}(\sqrt{n})$, a vector $\mathbf{x}$ sampled from $\mathcal{D}_s(\Lambda_q^{\mathbf{u}}(\mathbf{A}))$ has Euclidean norm less than $s\sqrt{m}$ with overwhelming probability. For a matrix $\mathbf{U} = (\mathbf{u}_1 | \dots | \mathbf{u}_k) \in \mathbb{Z}_q^{n \times k}$, we let $\mathcal{D}_s(\Lambda_q^{\mathbf{U}}(\mathbf{A}))$ be a distribution on matrices in $\mathbb{Z}^{m \times k}$ where the i-th column is sampled from $\mathcal{D}_s(\Lambda_q^{\mathbf{u}_i}(\mathbf{A}))$ for $i = 1, ..., k$.

The SIS Problem. Let n, m, q, β be positive integers. In the $\text{SIS}(n, m, q, \beta)$ problem, the adversary is given a uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and its goal is to find a vector $\mathbf{u} \in \mathbb{Z}_q^m$ with $\mathbf{u} \neq \mathbf{0}$ and $\|\mathbf{u}\| \leq \beta$ such that $\mathbf{A} \cdot \mathbf{u} = \mathbf{0}$.

The SIS problem is known to be as hard as certain worst-case lattice problems. In particular, for any $m = \mathsf{poly}(n)$, any $\beta > 0$, and any sufficiently large $q \geq \beta \cdot \mathsf{poly}(n)$, solving $\text{SIS}(n, m, q, \beta)$ is at least as hard as approximating certain worst-case lattice problems such as the Shortest Vector Problem (GapSVP) and the Short Independent Vectors Problem (SIVP) on n-dimensional lattices to within $\beta \cdot \mathsf{poly}(n)$ factor [Ajt96, Mic04, MR07, MP13]. The hardness of SIS is also implied by the LWE problem.

The LWE Problem. Let n, m, q be positive integers and χ a noise distribution over $\mathbb{Z}_q$. In the $\text{LWE}(n, m, q, \chi)$ problem, the adversary's goal is to distinguish between the two distributions:

$$(\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{e}) \qquad \text{and} \qquad (\mathbf{A}, \mathbf{u})$$

where $\mathbf{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{u} \overset{\$}{\leftarrow} \mathbb{Z}_q^m$ are uniformly sampled.

We say that a noise distribution χ is B-bounded if its support is in $[-B, B]$. For any fixed $d > 0$, and sufficiently large q, taking χ as a certain q/n^d-bounded distribution, the $\text{LWE}(n, m, q, \chi)$ problem is as hard as approximating certain worst-case lattice problems such as GapSVP and SIVP on n-dimensional lattices to within $\mathsf{poly}(n)$ factor [Reg09, Pei09, ACPS09, MM11, MP12, BLP+13].

Matrix Norms. For a matrix $\mathbf{R} \in \mathbb{Z}^{k \times m}$, we define the matrix norms:

- $\|\mathbf{R}\|$ denotes the ℓ_2 length of the longest column of $\mathbf{R}$.
- $\|\mathbf{R}\|_2$ is the operator norm of $\mathbf{R}$ defined as $\|\mathbf{R}\|_2 = \sup_{\|\mathbf{x}\|=1} \|\mathbf{R}\mathbf{x}\|$.

Note that always $\|\mathbf{R}\| \leq \|\mathbf{R}\|_2 \leq \sqrt{k}\|\mathbf{R}\|$ and that $\|\mathbf{R} \cdot \mathbf{S}\|_2 \leq \|\mathbf{R}\|_2 \cdot \|\mathbf{S}\|_2$.

Lattice Trapdoors. Here, we review the known results about lattice trapdoors which make the SIS and LWE problems easy to solve with knowledge such trapdoor. For this work, it is convenient to work with the notion of "gadget" based trapdoors as formalized in [MP12]. In such setting, there is a structured public gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times n\ell}$ for $\ell = \lceil \log q \rceil$. A trapdoor for a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is an integer matrix $\mathbf{R} \in \mathbb{Z}_q^{n \times n\ell}$ such that $\mathbf{AR} = \mathbf{HG}$ for some invertible matrix $\mathbf{H} \in \mathbb{Z}_q^{n \times n}$. The quality of a trapdoor is measured by the operator norm of $\mathbf{R}$ where the smaller norm $\|\mathbf{R}\|_2$ means higher quality. We will often use the symbol $\mathbf{T_A}$ to denote the trapdoor matrix $\mathbf{R}$.

Since the exact constructions and algorithm details of such trapdoors are not needed for this work, we abstract out these details and summarize the relevant results in the lemma below.

Lemma 1 **([Ajt96, GPV08, AP11, MP12]).** *There exist polynomial time algorithms* $\mathsf{TrapGen}, \mathsf{SamPre}, \mathsf{Sam}, \mathsf{Invert}$ *such that the following holds. Given positive integers $n \geq 1, q \geq 2$, there exists $m^* = O(n \log q)$ such that for $k = \mathsf{poly}(n)$, we have:*

- $\mathsf{TrapGen}(1^n, 1^m, q) \to (\mathbf{A}, \mathbf{T_A})$*: A randomized algorithm that when* $m \geq m^*$*, outputs a full-rank matrix* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ *and a trapdoor* $\mathbf{T_A} \in \mathbb{Z}^{m\times n\ell}$ *such that* $\mathbf{A}$ *is statistically close to uniform and* $\|\mathbf{R}\|_2 = O(\sqrt{m})$*.*
- $\mathsf{SamPre}(\mathbf{A}, \mathbf{T_A}, \mathbf{V}, s) \to \mathbf{U}$*: A randomized algorithm that on input* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$*, a trapdoor* $\mathbf{T_A}$ *of* $\mathbf{A}$*, a matrix* $\mathbf{V} \in \mathbb{Z}_q^{m\times k}$*, and* $s = \widetilde{O}(\|\mathbf{T_A}\|_2)$*, outputs a random sample* $\mathbf{U} \in \mathbb{Z}^{m\times k}$ *from the distribution* $\mathcal{D}_s(\Lambda_q^{\mathbf{V}}(\mathbf{A}))$*.*
- $\mathsf{Sam}(1^m, 1^k, q, s) \to \mathbf{U}$*: A randomized algorithm that samples a matrix* $\mathbf{U} \in \mathbb{Z}^{m\times k}$ *such that each of its column is sampled from* $\mathcal{D}_{\mathbb{Z}^m,s}$*. We have that for* $s \geq \omega(\sqrt{\log m})$ *the matrix* $\mathbf{V} = \mathbf{A}\cdot\mathbf{U}$ *is statistically close to a uniform matrix in* $\mathbb{Z}_q^{n\times k}$ *and furthermore, the distribution of* $\mathbf{U}$ *given* $\mathbf{V}$ *is* $\mathcal{D}_s(\Lambda_q^{\mathbf{V}}(\mathbf{A}))$*.*
- $\mathsf{Invert}(\mathbf{A}, \mathbf{T_A}, \mathbf{b}) \to \mathbf{s}$*: A deterministic algorithm that on input* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$*, a trapdoor* $\mathbf{T_A}$ *of* $\mathbf{A}$*, and an LWE vector* $\mathbf{b} = \mathbf{A}^T\mathbf{s} + \mathbf{e}$ *for* $\|\mathbf{e}\| \leq q/O(\|\mathbf{T_A}\|_2)$*, outputs the unique secret vector* $\mathbf{s}$*.*

To simplify the notation, thoughout the paper, we will always assume that the gadget matrix $\mathbf{G}$ has the same width m as the matrix $\mathbf{A}$ output by the algorithm TrapGen.

2.4 FRD Encoding

In this section, we review an encoding function $H : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$ that maps vectors in $\mathbb{Z}_q^n$ to invertible matrices in $\mathbb{Z}_q^{n\times n}$ with the property that for any two distinct vectors $\mathbf{u}$ and $\mathbf{v}$, the difference between the outputs $H(\mathbf{u})$ and $H(\mathbf{v})$ is never singular, i.e., $\det(H(\mathbf{u}) - H(\mathbf{v})) \neq 0$.

Definition 3. *Let* q *be a prime and* n *a positive integer. We say that an efficiently computable function* $H : \mathbb{Z}_q^n \to \mathbb{Z}_q^{n\times n}$ *is a* full-rank difference *(FRD) encoding scheme if for all distinct* $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_q^n$*, the matrix* $H(\mathbf{u}) - H(\mathbf{v}) \in \mathbb{Z}_q^{n\times n}$ *is full rank.*

This notion was formalized in [ABB10] and an injective encoding function satisfying the definition was explicitly constructed by generating an additive subgroup $\mathcal{G}_{\mathrm{FRD}}$ of full-rank matrices by embedding ring multiplications into matrices (similar techniques were used in [CD09,PR06,LM06]). We do not explicitly provide the construction here and mainly use the result as a black box throughout this work.

3 Predicate-Authentication-Preventing Signatures

In this section, we formally define the notion of *predicate authentication preventing signatures* (PAPS) which generalizes the notion of *double authentication preventing signatures* (DAPS) that was introduced in [PS14].

3.1 PAPS Framework

The syntax for predicate-authentication-preventing signatures largely coincides with the syntax for standard signature schemes with an additional extraction algorithm that can extract some private information of the signer (i.e. the signing key) given the signatures of messages that satisfy a particular predicate.

Definition 4 (PAPS). *A* predicate-authentication-preventing signature *(PA PS) on a corresponding message space $\mathcal{M}$, and a predicate $f : \mathcal{M}^k \rightarrow \{0,1\}$ is a tuple of efficient algorithms $\Pi_{\mathsf{paps}} = (\mathsf{PAPS.KeyGen}, \mathsf{PAPS.Sign}, \mathsf{PAPS.Verify}, \mathsf{PAPS.Extract})$ defined as follows:*

- $\mathsf{PAPS.KeyGen}(1^\lambda) \rightarrow (\mathsf{sk}, \mathsf{vk})$*: On input a security parameter 1^λ, the key generation algorithm* $\mathsf{PAPS.KeyGen}$ *outputs a signing key* sk *and a verification key* vk*.*
- $\mathsf{PAPS.Sign}(\mathsf{sk}, \mathsf{msg}) \rightarrow \sigma$*: On input a signing key* sk *and a message* $\mathsf{msg} \in \mathcal{M}$*, the signing algorithm* $\mathsf{PAPS.Sign}$ *outputs a signature σ.*
- $\mathsf{PAPS.Verify}(\mathsf{vk}, \mathsf{msg}, \sigma)$*: On input a verification key* vk*, a message* $\mathsf{msg} \in \mathcal{M}$*, and a signature σ, the verification algorithm* $\mathsf{PAPS.Verify}$ *accepts/rejects.*
- $\mathsf{PAPS.Extract}(\mathsf{vk}, \{(\mathsf{msg}_i, \sigma_i)\}_{i \in [k]}) \rightarrow \mathsf{sk}'$*: On input a verification key* vk*, and a set of message and signature pairs, the extraction algorithm* $\mathsf{PAPS.Extract}$ *outputs the secret key* sk'*.*

Correctness. We say that a PAPS scheme is correct if, for all $\lambda \in \mathbb{N}$, $\mathsf{msg} \in \mathcal{M}$, and $\sigma \leftarrow \mathsf{PAPS.Sign}(\mathsf{sk}, \mathsf{msg})$, we have that $\mathsf{PAPS.Verify}(\mathsf{vk}, \mathsf{msg}, \sigma) = 1$.

3.2 Extraction

As in the case of [PS14], we can consider two notions of key extractability depending on whether the signer generates its keys at setup honestly or adversarially. We call the scenario for which the keys are always generated honestly as the *trusted setup model* and the scenario for which the keys can potentially be generated adversarially as the *untrusted setup model.* Before defining these two notions formally, we first define the notion of a *compromising set of signatures.*

Definition 5 (Compromising set of signatures). *Let $f : \mathcal{M}^k \rightarrow \{0,1\}$ be a predicate defined on k messages. Then, for a fixed verification key* vk*, a set of k message/signature pairs $\{(\mathsf{msg}_i, \sigma_i)\}_{i \in [k]}$ is f-*compromising *if each signature σ_i is a valid signature of* msg_i *and the k messages satisfy the predicate f; that is, if* $\mathsf{PAPS.Verify}(\mathsf{vk}, \mathsf{msg}_i, \sigma_i) = 1$ *for all $i = 1, ..., k$ and* $f(\mathsf{msg}_1, ..., \mathsf{msg}_k) = 1$.

We now define formally the two notions of key extractability.

Definition 6 (Extractability in trusted setup). *Fix a predicate $f : \mathcal{M}^k \rightarrow \{0,1\}$. We say that a* PAPS *scheme on a message space $\mathcal{M}$ is f-*extractable *in the* trusted setup model *if for all $\lambda \in \mathbb{N}$,* $\mathsf{secret} \in \mathcal{S}$*, for all key pairs* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{PAPS.KeyGen}(1^\lambda)$*, and for all compromising set of signatures $S = \{(\mathsf{msg}_i, \sigma_i)\}_{i \in [k]}$, we have that* $\mathsf{PAPS.Extract}(\mathsf{vk}, S) = \mathsf{sk}$ *with overwhelming probability.*

For the untrusted setup model, instead of running the honest key generation algorithm, we allow an adversary to generate the keys along with a set of compromising set of signatures and require that the extraction algorithm succeeds on recovering the signing key from these set of signatures. Formally, we define extractability in the untrusted setup as follows.

Definition 7 (Extractability in untrusted setup). *Fix a predicate* $f : \mathcal{M}^k \to \{0,1\}$*. We say that a* PAPS *scheme on a message space* $\mathcal{M}$ *is* f-extractable *if for all efficient adversary* $\mathcal{A}$*, we have that*

$$\Pr\left(\begin{matrix} (\mathsf{vk}, S = \{(\mathsf{msg}_i, \sigma_i)\}) \leftarrow \mathcal{A}(1^\lambda) \\ \mathsf{sk}' \leftarrow \mathsf{PAPS.Extract}(\mathsf{vk}, S) \end{matrix} : \begin{matrix} S \ f\text{-}compromising \\ \wedge \ \mathsf{sk}' = \mathsf{sk} \end{matrix}\right) = 1 - \mathsf{negl}(\lambda)$$

Double-Authentication-Preventing Signatures. The notion of double-authentication-preventing signatures is a special case of PAPS. Specifically, in DAPS, the data to be signed $\mathcal{M}_{\mathsf{DAPS}}$ is split into two parts: a *subject* and a *payload*. Then, we consider the following 2-ary predicate in this message space

$$F_{\mathsf{DAPS}}((\mathsf{subj}_0, \mathsf{payload}_0), (\mathsf{subj}_1, \mathsf{payload}_1)) = \begin{cases} 1 \text{ if } \mathsf{subj}_0 = \mathsf{subj}_1, \mathsf{payload}_0 \neq \mathsf{payload}_1 \\ 0 \text{ otherwise.} \end{cases}$$

The predicate is designed specifically for the certificate authority setting where a CA that signs two different messages pertaining to the same subject is penalized by leaking the CA's signing key. In this work, we will mainly focus on constructing PAPS for this particular predicate function.

Remark. An alternative formulation of extractability is allowing some private information of the signer to be extracted instead of the signing key. In this case, the key generation algorithm can take in some secret information to be leaked by a compromising set of signatures and generate the keys accordingly. This formulation generalizes the notion above where extraction always leaks the signing key and can be more befitting for certain applications (see Sect. 7).

3.3 Unforgeability

The security game for the unforgeability notion for PAPS is similar to the standard unforgeability notion for digital signatures where the adversary has access to a signing oracle and wins if it forges a new signature. However, since PAPS is designed precisely to leak the signing key on a compromising set of signatures, we require that the adversary's queries to the signing oracle is limited to non-compromising sets of signatures.

Definition 8 (Unforgeability). *An* f*-extractable* PAPS *scheme* $\Pi_{\mathsf{paps}} = (\mathsf{PAPS.KeyGen}, \mathsf{PAPS.Sign}, \mathsf{PAPS.Verify}, \mathsf{PAPS.Extract})$ *is unforgeable if for all efficient adversary* $\mathcal{A}$*, we have that*

$$\mathsf{Adv}^{\mathsf{uf}}_{\Pi_{\mathsf{paps}},\mathcal{A}}(\lambda) \overset{\text{def}}{=} \Pr[\mathsf{Expt}_{\Pi_{\mathsf{paps}},\mathcal{A},\mathsf{uf}} = 1] \leq \mathsf{negl}(\lambda)$$

where the experiment $\mathsf{Expt}_{\Pi_{\mathsf{paps}},\mathcal{A},\mathsf{uf}}$ *is defined as follows:*

1. $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{PAPS.KeyGen}(1^\lambda)$*.*
2. $(\mathsf{msg}^*, \sigma^*) \leftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{Sign}}(\cdot)}(\mathsf{vk})$*.*
3. $\mathcal{A}$ *wins if* $\mathsf{Verify}^*(\mathsf{msg}^*, \sigma^*)$ *accepts.*

where the signing oracle $\mathcal{O}_{\mathsf{Sign}}(\cdot)$ *and* $\mathsf{Verify}^*(\cdot,\cdot)$ *are defined as follows:*

- *Oracle* $\mathcal{O}_{\mathsf{Sign}}(\cdot)$ *maintains a list* SignedList *of all the previous valid queries made by* $\mathcal{A}$*. For a query* msg*, that the adversary makes,* $\mathcal{O}_{\mathsf{Sign}}(\cdot)$ *checks whether there exists a compromising set of signatures in* SignedList $\cup$ $\{\mathsf{msg}\}$*. If this is the case, then* $\mathcal{O}_{\mathsf{Sign}}(\cdot)$ *outputs* $\perp$*. Otherwise, it outputs* $\mathsf{PAPS.Sign}(\mathsf{sk}, \mathsf{msg})$*.*
- *Verifier* $\mathsf{Verify}^*(\mathsf{msg}, \sigma)$ *accepts if* $\mathsf{msg} \notin$ SignedList *and* $\mathsf{PAPS.Verify}(\mathsf{vk}, \mathsf{msg}, \sigma)$ *accepts.*

4 DAPS from Lattices

In this section, we describe our DAPS construction. For clarity of exposition, we first describe a variant of the dual-Regev encryption scheme [GPV08] which we will use in our construction as a blackbox. This way, we can abstract out the details of the encryption component and present our DAPS construction in a simpler and more intuitive way. After presenting our DAPS construction, we prove its extractable properties and also show that its security can be based on the security of the encryption scheme and the SIS hardness assumption.

4.1 Trapdoor Dual-Regev Encryption

In this section, we describe a simple variant of the dual-Regev encryption scheme [GPV08] that we will use as a blackbox for our DAPS construction. We present the encryption scheme here mainly for clarity of exposition and the only difference between the encryption scheme presented here and the original dual-Regev encryption scheme is the use of full trapdoors as the secret key of the scheme as opposed to a short preimage vector as is the case in [GPV08].

We use n as the security parameter λ. Let m, q be the trapdoor parameters dependent on n (Lemma 1). Let χ be a B-bounded noise distribution where $B = O(q/m)$. We construct a public key encryption scheme $\Pi_{\mathsf{pke}} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ as follows:

- $\mathsf{PKE.KeyGen}(1^n)$: On input the security parameters 1^n, the key generation algorithm generates trapdoor $(\mathbf{A}, \mathbf{T_A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. It sets the public key $\mathsf{pk} = \mathbf{A}$ and $\mathsf{sk} = \mathbf{T_A}$.
- $\mathsf{PKE.Encrypt}(\mathsf{pk}, m)$: On input the public key pk and a message $m \in \{0, 1\}$, the encryption algorithm samples uniformly random vectors $\mathbf{d}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, and an error vector from the noise distribution $\mathbf{e} \leftarrow \chi^{m+1}$. It computes the vector
$$\mathbf{b} = [\mathbf{A} \mid \mathbf{d}]^T \mathbf{s} + \mathbf{e} + [\mathbf{0} \mid \lceil q/2 \rceil \cdot m].$$
It outputs the ciphertext $\mathsf{ct} = (\mathbf{d}, \mathbf{b}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{m+1}$.

- $\mathsf{PKE.Decrypt}(\mathsf{sk}, \mathsf{ct})$: On input the secret key sk and a ciphertext $\mathsf{ct} = (\mathbf{d}, \mathbf{b})$, the decryption algorithm parses $\mathbf{b} = (\mathbf{b}_0, b_1)$. It then computes $\mathbf{s} \leftarrow \mathsf{Invert}(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{b})$ and $m' = b_1 - \mathbf{d}^T\mathbf{s}$. It outputs $m \in \{0, 1\}$ such that m' is close to $\lceil q/2 \rceil \cdot m$.

Correctness. Let $\mathbf{b} = (\mathbf{b}_0 = \mathbf{A}^T\mathbf{s} + \mathbf{e}_0, b_1 = \mathbf{d}^T\mathbf{s} + e_1)$. The $\mathsf{TrapGen}$ algorithm outputs a trapdoor $\mathbf{T}_\mathbf{A}$ such that $\|\mathbf{T}_\mathbf{A}\|_2 = O(\sqrt{m})$. This means that for $B = O(q/\|\mathbf{T}_\mathbf{A}\|\sqrt{m}) = O(q/m)$, the algorithm $\mathsf{Invert}(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{b})$ for $\mathbf{b} = \mathbf{A}^T\mathbf{s} + \mathbf{e}$ correctly outputs the secret vector $\mathbf{s}$. Then, $b_1 - \mathbf{d}^T\mathbf{s} = (\mathbf{d}^T\mathbf{s} + e_1 + \lceil q/2 \rceil \cdot m) - \mathbf{d}^T\mathbf{s} = \lceil q/2 \rceil \cdot m + \cdot e_1$ which is correctly decoded for given bound on the error distribution χ.

Remark. We note that the correctness still holds with a different trapdoor $\mathbf{T}_\mathbf{A}{}'$ for which $\mathbf{T}_\mathbf{A} \neq \mathbf{T}_\mathbf{A}{}'$ as long as $\mathbf{T}_\mathbf{A}{}'$ is of sufficient quality. In fact, we can flexibly adjust the parameter B for the noise distribution χ of the scheme to allow for correct decryption by a slightly lower quality trapdoor. For instance, if we take the parameter to be $B = O(q/m^{3/2})$, then a trapdoor $\mathbf{T}_\mathbf{A}{}'$ of quality $\|\mathbf{T}_\mathbf{A}{}'\|_2 \leq O(m)$ can correctly decrypt the ciphertext. This property will be used for the extractability of our DAPS construction.

Security. The security reduction easily follows from the security proof of the original dual-Regev encryption scheme and we do not reproduce it here.

Theorem 1. *The PKE scheme above is IND-CPA secure assuming that the $LWE(n, m, q, \chi)$ problem is hard.*

For our DAPS construction, we will also assume that the dual-Regev PKE scheme above is circular secure.

4.2 DAPS Construction

We present our construction for DAPS from lattices. Fix a security parameter $n \in \mathbb{N}$ and let m, q, s be the corresponding trapdoor parameters. We use a hash function $\mathsf{H}_{\mathsf{msg}} : \{0, 1\}^* \times \{0, 1\}^* \rightarrow \mathbb{Z}_q^{n \times n}$ that maps arbitrary messages to a full rank matrix in $\mathcal{G}_{\mathrm{FRD}}$ and a hash function $\mathsf{H}_{\mathsf{subj}} : \{0, 1\}^* \rightarrow \mathbb{Z}_q^{n \times m}$ that maps a subject to an SIS matrix for the scheme. We also use the dual-Regev public key encryption scheme $\mathit{\Pi}_{\mathsf{pke}} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Encrypt}, \mathsf{PKE.Decrypt})$ as decribed in Sect. 4.1 with a B-bounded noise distribution where $B = O(q/m^{3/2})$. We construct $\mathit{\Pi}_{\mathsf{daps}} = (\mathsf{DAPS.KeyGen}, \mathsf{DAPS.Sign}, \mathsf{DAPS.Verify}, \mathsf{DAPS.Extract})$ as follows:

- $\mathsf{DAPS.KeyGen}(1^n)$: On the security parameter 1^n, the key generation algorithm first runs $(\mathbf{A}, \mathbf{T}_\mathbf{A}) \leftarrow \mathsf{PKE.KeyGen}(1^n)$. It then encrypts $\mathsf{ct} \leftarrow \mathsf{PKE.Encrypt}(\mathbf{A}, \mathbf{T}_\mathbf{A})$ and set $\mathsf{sk} = \mathbf{T}_\mathbf{A}$ and $\mathsf{vk} = (\mathbf{A}, \mathsf{ct})$.

- $\mathsf{DAPS.Sign}(\mathsf{sk}, \mathsf{subj}, \mathsf{payload})$: On input the signing key sk, a subject subj, and a message $\mathsf{payload}$, the signing algorithm hashes the message $\mathbf{H}_{\mathsf{msg}} \leftarrow \mathsf{H}_{\mathsf{msg}}(\mathsf{subj}, \mathsf{payload})$ and also hashes the subject $\mathbf{B}_{\mathsf{subj}} \leftarrow \mathsf{H}_{\mathsf{subj}}(\mathsf{subj})$. Then, it computes

$$\mathbf{U} \leftarrow \mathsf{SamPre}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, \mathbf{B}_{\mathsf{subj}} + \mathbf{H}_{\mathsf{msg}} \cdot \mathbf{G}, s)$$

  where $\mathbf{G}$ is the publicly known gadget matrix. Finally, it outputs $\mathbf{U}$ as the signature.
- $\mathsf{DAPS.Verify}(\mathsf{vk}, \mathsf{subj}, \mathsf{payload}, \sigma)$: On input the verification key vk, a subject subj, a message $\mathsf{payload}$, and a signature $\sigma = \mathbf{U}$, the algorithm hashes the message $\mathbf{H}_{\mathsf{msg}} \leftarrow \mathsf{H}_{\mathsf{msg}}(\mathsf{subj}, \mathsf{payload})$ and also derives the matrix $\mathbf{B}_{\mathsf{subj}} \leftarrow \mathsf{H}_{\mathsf{subj}}(\mathsf{subj})$. Then, the algorithm verifies that

$$\mathbf{A} \cdot \mathbf{U} = \mathbf{B}_{\mathsf{subj}} + \mathbf{H}_{\mathsf{msg}} \cdot \mathbf{G}$$

  and that $\|\mathbf{U}\| \leq s \cdot \sqrt{m}$.
- $\mathsf{DAPS.Extract}(\mathsf{vk}, (\mathsf{subj}_1, \mathsf{payload}_1, \sigma_1), (\mathsf{subj}_2, \mathsf{payload}_2, \sigma_2))$: On input a verification key $\mathsf{vk} = (\mathbf{A}, \mathsf{ct})$, and two subject/message pairs $(\mathsf{subj}_1, \mathsf{payload}_1)$, $(\mathsf{subj}_2, \mathsf{payload}_2)$ and their signatures $\sigma_1 = \mathbf{U}_1, \sigma_2 = \mathbf{U}_2$, the extraction algorithm runs $\mathsf{sk}' \leftarrow \mathsf{PKE.Decrypt}(\mathbf{U}_1 - \mathbf{U}_2)$ and outputs sk'.

Signing Correctness. The correctness follows easily from the correctness of the trapdoor algorithm SamPre (Lemma 1) and the tail bounds of discrete Gaussian.

Security and Extrability. We now state the security and extractability of the construction above.

Theorem 2. *The* DAPS *construction above is unforgeable assuming the hardness of* $\mathrm{SIS}(n, m, q, \beta)$ *for* $\beta = O(s)$ *and circular security of* Π_{pke} *modeling the hash functions* $\mathsf{H}_{\mathsf{msg}}$ *and* $\mathsf{H}_{\mathsf{subj}}$ *as random oracles.*

Theorem 3. *Assuming* $\mathsf{H}_{\mathsf{msg}}$ *as a collision-resistant hash function, the* DAPS *construction above is* F_{DAPS}*-extractable.*

5 Extensions of DAPS to Other Predicates

In this work, we provide sample PAPS constructions for a number of simple predicates. Due to space limitations, we describe and prove our constructions in the full version.

6 Multi-authority Setting

For many practical situations, there is not a single authority signer, but multiple authorities who sign messages. For these type of situations, it is useful to extend the PAPS framework to the multi-authority setting where any compromising set of signatures by the different authorities reveals some private information of the signers.

We note that in this scenario, allowing a compromising set of signatures to reveal a secret key of a signer is not well-formulated in that any signer can compute a compromising set of signatures itself using its own signing key and extract another signer's secret key. Therefore, for the multi-authority setting, we let the extraction algorithm to reveal some predefined private data and define an additional algorithm PAPS.CommGen that takes in this private information that we denote by secret of a subset of the signers and generates a commitment comm pertaining to secret. For correctness, we require that a compromising set of signatures from these subset of signers along with the commitment comm allow anyone to reveal the private information secret.

Definition 9 (Multi-authority PAPS). *A* multi-authority predicate-authentication-preventing signature *(MAPAPS) on a corresponding message space $\mathcal{M}$, secret space $\mathcal{S}$, and a predicate $f : \mathcal{M}^k \rightarrow \{0,1\}$ is a tuple of efficient algorithms consisting of the* PAPS *algorithms $\Pi_{\mathsf{paps}} = (\mathsf{PAPS.KeyGen}, \mathsf{PAPS.Sign}, \mathsf{PAPS.Verify}, \mathsf{PAPS.Extract})$ with two additional algorithms* (PAPS.CommGen, PAPS.CommExtract) *defined as follows:*

- $\mathsf{PAPS.CommGen}(\{\mathsf{vk}_i\}, \mathsf{secret}) \rightarrow \mathsf{comm}_T$*: On input a set of verification keys $T = \{\mathsf{vk}_i\}$ and some private data $\mathsf{secret} \in \mathcal{S}$, the commitment generation algorithm generates a public commitment comm_T.*
- $\mathsf{PAPS.CommExtract}(\mathsf{comm}_T, \{(\mathsf{msg}_i, \sigma_i)\}_{i\in[k]}) \rightarrow \mathsf{secret}'$*: On input a public commitment comm_T, and a set of message and signature pairs, the extraction algorithm outputs private date secret'.*

Informally speaking, the PAPS.CommGen takes in a set of verification keys of the signers and generates a hiding commitment of some private data that belongs to these signers. On a compromising set of signatures produced by any of these signers allows anyone to extract this private information using the PAPS.CommExtract algorithm.

Correctness. As in the regular PAPS setting, we require that the algorithms PAPS.KeyGen, PAPS.Sign, PAPS.Verify satisfies the signing correctness requirement as in Sect. 3.

6.1 Extractability

In addition to the extractability property of PAPS.Extract in the single authority setting as in Sect. 3, we require an additional extractability in the multi-authority case where it is required that PAPS.CommExtract extract private information from the public commitment comm.

Definition 10. *Let $f : \mathcal{M}^k \rightarrow \{0,1\}$ be a predicate defined on k messages and $T = \{\mathsf{vk}_i\}$ be a set of verification keys. Then, a set of k message/signature pairs*

$\{(\mathsf{msg}_j, \sigma_j)\}_{j\in[k]}$ *is* f-compromising *with respect to* T *if each signature* σ_j *is a valid signature of* msg_j *by some verification key* $\mathsf{vk}_i \in T$*, and the* k *messages satisfy the predicate* f*; that is, if for all* $j \in [k]$*,* $\mathsf{PAPS.Verify}(\mathsf{vk}_i, \mathsf{msg}_j, \sigma_j) = 1$ *for some* $\mathsf{vk}_i \in T$ *and* $f(\mathsf{msg}_1, ..., \mathsf{msg}_k) = 1$*.*

We define the standard extractability condition as follows.

Definition 11. *Fix a predicate* $f : \mathcal{M}^k \to \{0, 1\}$*. We say that a* MAPAPS *scheme on a message space* $\mathcal{M}$ *is* f*-commitment-extractable if for all* $\lambda \in N$*,* $\mathsf{secret} \in \mathcal{S}$*, a set of verification keys* $T = \{\mathsf{vk}_i\}$*,* $\mathsf{comm}_T \leftarrow \mathsf{PAPS.CommGen}(T, \mathsf{secret})$ *and for all compromising set of signatures* $S = \{(\mathsf{msg}_j, \sigma_j)\}_{j\in[k]}$ *with respect to* T*, we have that* $\mathsf{PAPS.CommExtract}(\mathsf{comm}_T, S) = \mathsf{secret}$ *with overwhelming probability.*

6.2 Security

Commitment Privacy. To prevent the extractability notion above from being satisfied trivially, we require a privacy requirement on the secret data of PAPS. Specifically, we require that an adversary with access limited to non-compromising sets of signatures do not learn information about the secret data. Let N be the number of authorities in the system.

Definition 12 (Privacy). *An* f*-extractable* PAPS *scheme* $\Pi_{\mathsf{paps}} = (\mathsf{PAPS.KeyGen}, \mathsf{PAPS.CommGen}, \mathsf{PAPS.Sign}, \mathsf{PAPS.Verify}, \mathsf{PAPS.CommExtract})$ *is data-hiding if for any efficient adversary* $\mathcal{A}$*, we have that*

$$\mathsf{Adv}^{\mathsf{dh}}_{\Pi_{\mathsf{paps}},\mathcal{A}}(\lambda) \overset{\text{def}}{=} \left|\Pr[\mathsf{Expt}^{(0)}_{\Pi_{\mathsf{paps}},\mathcal{A},\mathsf{dh}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{(1)}_{\Pi_{\mathsf{paps}},\mathcal{A},\mathsf{dh}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Expt}^{(b)}_{\Pi_{\mathsf{paps}},\mathcal{A},\mathsf{dh}}$ *is defined as follows:*

1. $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{PAPS.KeyGen}(1^\lambda)$ *for* $i = 1, ..., N$*.*
2. $(T, \mathsf{secret}_0, \mathsf{secret}_1) \leftarrow \mathcal{A}_0(\{\mathsf{vk}\}_{i\in[N]})$*.*
3. $\mathsf{comm}_b \leftarrow \mathsf{PAPS.CommGen}(\{\mathsf{vk}_i\}_{t\in T}, \mathsf{secret}_b)$*.*
4. *Output* $\mathcal{A}_1^{\mathcal{O}_{\mathsf{Sign}}(\cdot,\cdot)}(\{\mathsf{sk}_i\}_{i\in[N]\setminus T}, \mathsf{comm}_b)$*.*

where the signing oracle $\mathcal{O}_{\mathsf{Sign}}(\cdot, \cdot)$ *is defined as follows:*

– *Oracle* $\mathcal{O}_{\mathsf{Sign}}(\cdot, \cdot)$ *maintains a list* SignedList *of all the previous valid queries made by* $\mathcal{A}$*. For a query* msg *and an index* $i \in T$*, that the adversary makes* $\mathcal{O}_{\mathsf{Sign}}(\cdot, \cdot)$ *checks whether there exists a compromising set of signatures with respect to* T *in* SignedList $\cup \{\mathsf{msg}\}$*. If this is the case, the* $\mathcal{O}_{\mathsf{Sign}}(\cdot, \cdot)$ *outputs* $\perp$*. Otherwise, it outputs* $\mathsf{PAPS.Sign}(\mathsf{sk}, \mathsf{msg})$*.*

7 Multi-authority DAPS

In this section we describe our construction of DAPS in the setting of multi-authority. We describe the scheme for the DAPS predicate, but the extensions in Sect. 5 translate directly to the multi authority setting since it uses the DAPS predicate as a black box. As in Sect. 4, we first describe a broadcast encryption scheme from the variant of the dual-Regev encryption scheme and then present our MADAPS construction using the broadcast encryption scheme. We prove that our scheme satisfies both the extractability and security requirements.

7.1 Broadcast Encryption

In this section, we present the broadcast encryption scheme. For clarity, we slightly alter the syntax of broadcast encryption where the global public key pk is divided into a number of public keys pk_i for each user in the system and the encryption algorithm takes in a subset of these public keys. Fix a security parameter n and let m, q, and χ be the corresponding parameter as in Sect. 4.1. We construct $\Pi_{\mathsf{be}} = (\mathsf{BE.KeyGen}, \mathsf{BE.Encrypt}, \mathsf{BE.Decrypt})$ as follows:

- $\mathsf{BE.KeyGen}(1^n, N)$: On input the security parameter 1^n, the key generation algorithm generates trapdoors $(\mathbf{A}_i, \mathbf{T}_{\mathbf{A}i}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ for $i = 1, ..., N$. It outputs $\{(\mathbf{A}_i, \mathbf{T}_{\mathbf{A}i})\}_{i\in[N]}$.
- $\mathsf{BE.Encrypt}(\{\mathsf{pk}_i\}_{i\in T}, m)$: On input a set of public keys $\{\mathsf{pk}_i\}_{i\in T} = \{\mathbf{A}_i\}_{i\in T}$ and a message $m \in \{0, 1\}$, the encryption algorithm first samples uniformly random vectors $\mathbf{d}, \mathbf{s} \overset{\$}{\leftarrow} \mathbb{Z}_q^n$ and error vector $\mathbf{e}_0 \leftarrow \chi^m$ and then computes

  $$\mathsf{ct}_0 = \mathbf{d}^T\mathbf{s} + \mathbf{e}_0 + \lceil q/2 \rceil \cdot m$$

  Then for $t \in T$, it generates a fresh error vector from the noise distribution $\mathbf{e}_t \leftarrow \chi^m$ and computes

  $$\mathsf{ct}_t = \mathbf{A}_t^T\mathbf{s} + \mathbf{e}_t$$

  It outputs $\mathsf{ct}_T = (\mathsf{ct}_0, \{\mathsf{ct}_t\}_{t\in T})$.
- $\mathsf{BE.Decrypt}(\mathsf{sk}_i, \mathsf{ct}_T)$: On input the secret key sk_i for $i \in T$ and a ciphertext $\mathsf{ct}_T = (\mathsf{ct}_0, \{\mathsf{ct}_t\}_{t\in T})$, the decryption algorithm computes $\mathbf{s} \leftarrow \mathsf{Invert}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, \mathsf{ct}_i)$. It then computes $m' = \mathsf{ct}_0 - \mathbf{d}^T\mathbf{s}$ and output $m \in \{0, 1\}$ such that m' is closest to $\lceil q/2 \rceil \cdot m$.

We note that each ciphertext component $(\mathsf{ct}_0, \mathsf{ct}_i)$ makes up a regular ciphertext of the dual-Regev encryption scheme. Therefore, the correctness of the BE scheme above follows directly from the correctness of the dual-Regev scheme. Also, the construction of the scheme above is a generic concatenation of the regular public key encryption scheme and therefore, the security follows directly from the security of the public key encryption scheme.

As in the case of the dual-Regev public key encryption scheme, one can correctly decrypt the ciphertext even with a different trapdoor for the public matrices $\mathbf{A}_i$ as long as it is of sufficient quality. Furthermore, any trapdoor of a concatenated matrix $[\mathbf{A}_i \mid \mathbf{A}_j]$ is sufficient to recover the encryption randomness used in the encryption and therefore decrypt a ciphertext intended for users i and j. More formally, there exists a decryption algorithm as follows:

- $\mathsf{BE.Decrypt}'(\mathsf{sk}_{i,j}, \mathsf{ct}_T)$: On input a trapdoor $\mathsf{sk}_{i,j} = \mathbf{T}_{[\mathbf{A}_i|\mathbf{A}_j]}$ for $i, j \in T$, and a ciphertext $\mathsf{ct}_T = (\mathsf{ct}_0, \{\mathsf{ct}_t\}_{t \in T})$, the decryption algorithm computes $\mathbf{s} \leftarrow \mathsf{Invert}([\mathbf{A}_i|\mathbf{A}_j], \mathbf{T}_\mathbf{A}, [\mathsf{ct}_i|\mathsf{ct}_j])$. It then computes $m' = \mathsf{ct}_0 - \mathbf{d}^T\mathbf{s}$ and outputs $m \in \{0,1\}$ such that m' is closest to $\lceil q/2 \rceil \cdot m$.

7.2 Multi-authority DAPS Construction

In this section, we extend the DAPS construction from Sect. 4.2 to the multi-authority setting (MADAPS). In addition to the algorithms in Π_{daps}, we define two additional algorithms DAPS.CommGen and DAPS.CommExtract as defined in Sect. 6. Let $\Pi_{\mathsf{be}} = (\mathsf{BE.KeyGen}, \mathsf{BE.Encrypt}, \mathsf{BE.Decrypt})$ be the broadcast encryption scheme as defined above with B-bounded noise distribution χ where $B = O(q/m^{3/2})$. Fix a security parameter $n \in \mathbb{N}$ and let m, q, s be the corresponding trapdoor parameters. We define the algorithms DAPS.CommGen and DAPS.CommExtract as follows:

- $\mathsf{DAPS.CommGen}(\{\mathsf{vk}_i\}, \mathsf{secret})$: On input a set of verification keys $T = \{\mathsf{vk}_i\} = \{\mathbf{A}_i\}$ and some private data $\mathsf{secret} \in \mathcal{S}$, the commitment generation algorithm computes $\mathsf{ct}_T \leftarrow \mathsf{BE.Encrypt}(T, \mathsf{secret})$ and sets $\mathsf{comm} = \mathsf{ct}_T$.
- $\mathsf{DAPS.CommExtract}(\mathsf{comm}_T, (\mathsf{subj}_0, \mathsf{payload}_0, \sigma_0), (\mathsf{subj}_1, \mathsf{payload}_1, \sigma_1))$: On input the parameters comm_T and a pair of compromising pair of signatures $\sigma_0 = \mathbf{U}_i$ and $\sigma_1 = \mathbf{U}_j$ corresponding to vk_i and vk_j respectively, the extraction algorithm concatenates the keys $\tilde{\mathbf{U}} = \begin{bmatrix} \mathbf{U}_i \\ -\mathbf{U}_j \end{bmatrix}$. It runs $\mathsf{secret}' \leftarrow \mathsf{BE.Decrypt}'(\tilde{\mathbf{U}}, \mathsf{comm}_T)$ and outputs the private data secret'.

Extractability and Privacy. We now state the extractability and privacy properties of the MADAPS construction above. We provide the proofs of the following theorems in the full version.

Theorem 4. *Assuming* $\mathsf{H}_{\mathsf{msg}}$ *is a collision-resistant hash function, the* MADAPS *construction above is* F_{DAPS}*-extractable.*

Theorem 5. *The* MADAPS *construction above is data-hiding assuming that the broadcast encryption scheme* Π_{be} *is secure.*

Acknowledgements. This work is supported by NSF, DARPA, the Simons foundation, and a grant from ONR. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA.

References

[ABB10] Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_28

[ACPS09] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03356-8_35

[Ajt96] Ajtai, M.: Generating hard instances of lattice problems. In: STOC (1996)

[ALSY06] Au, M.H., Liu, J.K., Susilo, W., Yuen, T.H.: Constant-size ID-based linkable and revocable-iff-linked ring signature. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 364–378. Springer, Heidelberg (2006). doi:10.1007/11941378_26

[AP11] Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. Theory Comput. Syst. **48**(3), 535–553 (2011)

[ASP12] Alperin-Sheriff, J., Peikert, C.: Circular and KDM security for identity-based encryption. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 334–352. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30057-8_20

[BF14] Bellare, M., Fuchsbauer, G.: Policy-based signatures. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 520–537. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54631-0_30

[BGI14] Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54631-0_29

[BGV12] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: ITCS. ACM (2012)

[BGW05] Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005). doi:10.1007/11535218_16

[BKM06] Bender, A., Katz, J., Morselli, R.: Ring signatures: stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 60–79. Springer, Heidelberg (2006). doi:10.1007/11681878_4

[BLP+13] Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: STOC. ACM (2013)

[BPS15] Bellare, M., Poettering, B., Stebila, D.: Double-authentication-preventing signatures and trapdoor identification. Cryptology ePrint Archive, Report 2015/1157 (2015). http://eprint.iacr.org/

[BRS02] Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003). doi:10.1007/3-540-36492-7_6

[BV11] Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22792-9_29

[BV14] Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. SIAM J. Comput. **43**(2), 831–871 (2014)

[BWZ14] Boneh, D., Waters, B., Zhandry, M.: Low overhead broadcast encryption from multilinear maps. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 206–223. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44371-2_12

[CD09] Cramer, R., Damgård, I.: On the amortized complexity of zero-knowledge protocols. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 177–191. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03356-8_11

[CL01] Camenisch, J., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001). doi:10.1007/3-540-44987-6_7

[FN93] Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994). doi:10.1007/3-540-48329-2_40

[Gen09] Gentry, C.: Fully homomorphic encryption using ideal lattices. STOC **9**, 169–178 (2009)

[GPV08] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC. ACM (2008)

[GSW13] Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_5

[GVW15] Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: STOC. ACM (2015)

[LLK15] Laurie, B., Langley, A., Kasper, E.: Certificate Transparency. RFC 6962, (October 2015)

[LM06] Lyubashevsky, V., Micciancio, D.: Generalized compact knapsacks are collision resistant. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 144–155. Springer, Heidelberg (2006). doi:10.1007/11787006_13

[Lyu08] Lyubashevsky, V.: Lattice-based identification schemes secure under active attacks. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 162–179. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78440-1_10

[Lyu12] Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29011-4_43

[MBB+15] Melara, M.S., Blankstein, A., Bonneau, J., Felten, E.W., Freedman, M.J.: CONIKS: bringing key transparency to end users. USENIX Secur. **15**, 383–398 (2015)

[Mic04] Micciancio, D.: Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor. SIAM J. Comput. **34**(1), 118–169 (2004)

[MM11] Micciancio, D., Mol, P.: Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 465–484. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22792-9_26

[MP12] Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012.

LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29011-4_41

[MP13] Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 21–39. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_2

[MPR11] Maji, H.K., Prabhakaran, M., Rosulek, M.: Attribute-based signatures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 376–392. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19074-2_24

[MR07] Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. SIAM J. Comput. **37**(1), 267–302 (2007)

[Pei09] Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: STOC. ACM (2009)

[PR06] Peikert, C., Rosen, A.: Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 145–166. Springer, Heidelberg (2006). doi:10.1007/11681878_8

[PS14] Poettering, B., Stebila, D.: Double-authentication-preventing signatures. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 436–453. Springer, Cham (2014). doi:10.1007/978-3-319-11203-9_25

[Reg09] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM (JACM) **56**(6), 34 (2009)

[RST01] Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001). doi:10.1007/3-540-45682-1_32

[TX03] Tsudik, G., Xu, S.: Accumulating composites and improved group signing. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 269–286. Springer, Heidelberg (2003). doi:10.1007/978-3-540-40061-5_16

Forward-Secure Searchable Encryption on Labeled Bipartite Graphs

Russell W.F. Lai(✉) and Sherman S.M. Chow

Department of Information Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{russell,sherman}@ie.cuhk.edu.hk

Abstract. Forward privacy is a trending security notion of dynamic searchable symmetric encryption (DSSE). It guarantees the privacy of newly added data against the server who has knowledge of previous queries. The notion was very recently formalized by Bost (CCS '16) independently, yet the definition given is imprecise to capture how forward secure a scheme is. We further the study of forward privacy by proposing a generalized definition parametrized by a set of updates and restrictions on them. We then construct two forward private DSSE schemes over labeled bipartite graphs, as a generalization of those supporting keyword search over text files. The first is a generic construction from any DSSE, and the other is a concrete construction from scratch. For the latter, we designed a novel data structure called cascaded triangles, in which traversals can be performed in parallel while updates only affect the local regions around the updated nodes. Besides neighbor queries, our schemes support flexible edge additions and intelligent node deletions: The server can delete all edges connected to a given node, without having the client specify all the edges.

1 Introduction

In searchable symmetric encryption (SSE), an encrypted database can be queried with minimal leakage of information about the plaintext database to the hosting server. The client is additionally allowed to update the encrypted database in dynamic SSE (DSSE) without reencrypting from scratch. Since its introduction [16], many SSE schemes with different trade-offs between efficiency, security, and query expressiveness have been proposed [1]. Most earlier schemes were not dynamic. The first sublinear dynamic SSE scheme was proposed by Kamara *et al.* [11], but the query and update operations are inherently sequential. Some later schemes [8,10,17] feature parallelizable algorithms for queries and updates. Parallelism made DSSE an attractive solution for outsourcing data to cloud platform which fully leverages the multiprocessors.

Sherman Chow is supported in part by General Research Fund (Grant No. 14201914) and the Early Career Award from Research Grants Council, Hong Kong; and Huawei Innovation Research Program (HIRP) 2015 (Project No. YB2015110147).

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 478–497, 2017.
DOI: 10.1007/978-3-319-61204-1_24

1.1 Security and Forward Privacy of SSE Schemes

Ideally, the knowledge of an encrypted database, together with a sequence of adaptively issued queries and updates, should not reveal any information about the plaintext database and the query results to the server. Although this can be achieved theoretically through techniques involving obfuscation [4] or oblivious RAM [7], the resulting solutions are not particularly efficient. Typically, a practical DSSE scheme tolerates the leakage of search and access patterns during queries, and some internal structure of the encrypted database during updates. Formally, the security is parametrized by a set of leakage functions describing these leakages. While some leakages seem to pose no harm, some have been exploited in attacks [9,18].

Forward privacy, advocated by Stefanov *et al.* [17], requires that newly added data remains private against the server, who has knowledge about previous queries. The property is arguably essential to all DSSE schemes, for otherwise, the ability to update in DSSE is somewhat useless as future data are less protected. Indeed, one of the recent attacks by Zhang *et al.* [18] exploits the leakage during updates in non-forward-private schemes.

Only a limited number of solutions [2,7,15,17] in the literature claimed to have forward privacy. The notion is not well understood in the earlier works [7, 15,17], and is only formally defined recently by Bost [2]. However, we argue that this definition cannot precisely describe in what sense a DSSE scheme is forward private. (See discussion in Sect. 3.2 for details.)

1.2 Our Formulation

We consider DSSE over labeled bipartite graphs, where nodes can be partitioned into two disjoint subsets X and Y, such that edges never connect two nodes from the same partition, and each edge is labeled with data from the set W. A neighbor query on node $x \in X$ (or $y \in Y$), returns a sequence of $(x, y, w) \in X \times Y \times W$ tuples if (x, y) is an edge on the graph labeled with w.

This abstract setting captures typical DSSE queries such as keyword searches over files (considering X and Y as the sets of keywords and files respectively), and labeled subgraphs queries over general graphs (considering X and Y as sets of nodes with outgoing and incoming edges respectively, and W as the set of edge labels). To generalize, we also consider neighbor queries over the entire bipartite graphs, *i.e.*, both X and Y, which enables interesting bi-directional searching applications. Bi-directional search is useful to efficiently support update in DSSE [11] (since deleting a file in a DSSE supporting keyword searches implicitly requires finding all keywords the file contains). It also opens possibilities of interesting new queries such as related keyword search (which first searches for documents containing the queried keyword, then collects other keywords which are also contained in many of the matching documents).

1.3 Update Functionalities of DSSE Schemes

While the query functionality of SSE for keyword search is somewhat standard, the supported update types have large variations. Updates include additions and deletions, and can be edge-based or node-based. Schemes supporting only node additions are reasonable for some data type: *e.g.*, x as keywords and y as text files. Yet, edge updates allow fine-grained modification of existing data. In particular, schemes supporting edge additions are superior to those supporting only node additions, as the latter can be simulated by the former.

The benefits of supporting only edge deletions are however questionable, as they require the client to know about the edge to be deleted. It is unrealistic for the motivating application of SSE for keyword search: The client needs to know all keywords of a given file to completely remove the file from the server. It is desirable for a DSSE scheme to support node deletions: upon provided a node y from the client, the server can intelligently remove all edges connecting y.

To the best of our knowledge, most existing schemes only support either edge-based updates or node-based updates[1]. Supporting edge additions and node deletions simultaneously, while confining leakage, poses some technical challenges.

1.4 SSE as a Data Structure Problem

With edge additions and node deletions in mind, it is not an easy task to devise a parallel and dynamic (let alone forward private) SSE scheme. Intuitively, for data structures supporting parallel traversal, maintaining the traversal efficiency after an update often requires some global adjustment of the data structure. Consider a balanced binary search tree. A series of deletions can degenerate the tree into a linked list which requires sequential access; and balancing the tree may require the rotation of multiple tree nodes. (That may explain why the first parallel DSSE [10] utilizing a red-black tree which only stores all the files in the leaf level, resulting in an efficiency loss when compared with storing some of them in internal nodes.) In the context of DSSE, delegating the maintenance work to the server often implies excessive leakage of the internal data structure.

A notable approach for (non-dynamic) SSE schemes is the invert-index used by Curtmola *et al.* [5,6], in which the encrypted database consists of an index mapping hashed keywords to sets of files containing the keywords. This inverted-index allows the server to search in time linear in the number of matching files, which is optimal. Many subsequent works follow this framework (explicitly or implicitly), which utilize some data structure to represent the sets of data, pointed by the (hashed) queries in the index. The efficiency of queries and updates correspond to the efficiency of traversing and updating the sets respectively. On the other hand, the leakage of the internal structure of the encrypted database during updates corresponds to the amount of information required or changed to update the data structure storing the sets. Most efforts for designing (D)SSE schemes is dedicated to choosing or designing this data structure.

[1] A few exceptions include Lai-Chow [13] and a modified version of the one by Kamara *et al.* [11]. However, these schemes leak substantial information during updates.

1.5 Our Results

This work furthers the study of forward privacy of DSSE schemes over labeled bipartite graphs. We present three technical results. First, we give another formal, generalized definition of forward privacy. Specifically, our definition is parametrized by a set of updates and a restriction function on these updates. It generalizes the only existing one by Bost [2], by increasing the number of classes of leakage functions allowed, yet making each class more specific. Our definition still captures the essence of forward privacy even though the leakage in different classes might vary substantially. Since different existing SSE schemes implicitly assumed different flavors of forward privacy, we believe that our generalized, parameterized definition of forward privacy is of particular interest.

Second, we propose a simple generic construction of forward private DSSE from any DSSE, which preserves the efficiency of the base scheme. The forward privacy obtained is for edge additions, such that the addition of an edge (x, y) does not leak both x and y, hence hiding the edge. This generic transformation provides insights of what constitutes forward privacy in DSSE. Since the result applies on any DSSE, again we believe it is of independent interest.

Lastly, we construct a DSSE scheme from scratch which achieves a stronger forward privacy for edge additions, such that the addition of an edge (x, y) does not leak either x or y. Our construction utilizes a specially crafted data structure named cascaded triangles[2], which supports parallel queries and updates, and has the property that adding or deleting data only affects a constant amount of existing data. Thanks to cascaded triangles, our construction features minimal leakage, and optimal query and update complexity in terms of both computation and communication up to a constant factor.

Both of our constructions support flexible edge additions and intelligent node deletions: The server can delete all edges connected to a given node, without having the client specify all the edges. It is one of a few in the literature(See footnote 1).

2 Definitions

We present the necessary definitions for data representation and DSSE. For more detailed explanations, we refer the readers to the full version.

2.1 Data Representation

Let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{W}$ be sets, where $\mathcal{X}$ and $\mathcal{Y}$ are disjoint, *i.e.*, $\mathcal{X} \cap \mathcal{Y} = \phi$. We denote by $\mathcal{G} = \mathcal{G}(\mathcal{X}, \mathcal{Y}, \mathcal{W})$ a set of labeled bipartite graphs specified by these sets. For a labeled bipartite graph $G \in \mathcal{G}$, its edges are labeled with $w \in W \subseteq \mathcal{W}$, and

[2] While the design of cascaded triangles is original, we do not rule out the possibility that there are similar data structures outside the literature of SSE. To the best of our knowledge, we are unaware of any common similar data structure. There are false relatives such as fractional cascading which solves totally different problems.

are running across $X \subseteq \mathcal{X}$ and $Y \subseteq \mathcal{Y}$. Each edge can be uniquely represented by the tuple $(x, y, w) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{W}$. The neighbor query function Qry maps a node $q = x$ (or $q = y$) and a graph G to a set of all edges connecting node x, denoted by $(x, *, *)$ (or a set of all edges connecting node y, denoted by $(*, y, *)$). Similarly, we use $(x, y, *)$ to denote the set of edges in the form of $(x, y, \cdot)$, which should be a singleton. The update function Udt maps an update u and a graph G to a new graph G'. The update $u = (\mathsf{Op}, \cdot, \cdot, \cdot)$ where $\mathsf{Op} = \mathsf{Add}$ or $\mathsf{Op} = \mathsf{Del}$ takes one of the following forms:

1. Edge Addition: $u = (\mathsf{Add}, x, y, w)$ adds the edge (x, y, w) to G.
2. Node Deletion: $u = (\mathsf{Del}, q)$ deletes the set of edges $(q, *, *)$ (or $(*, q, *)$) from G. Since $\mathcal{X}$ and $\mathcal{Y}$ are disjoint, there is no ambiguity.

2.2 Dynamic Searchable Symmetric Encryption (DSSE)

We present a definition of DSSE for the labeled bipartite graphs defined above, and briefly describe its security against adaptive chosen query attack (CQA2).

Definition 1. *A dynamic symmetric searchable encryption (DSSE) scheme for the space of labeled bipartite graphs specified by $\mathcal{G}$ is a tuple of algorithms and interactive protocols* $\mathsf{DSSE}.(\mathsf{Setup}, \mathsf{Qry}_e, \mathsf{Udt}_e)$ *such that:*

- $(K, \mathsf{EDB}) \leftarrow \mathsf{Setup}(1^\lambda)$*: In the setup algorithm, the user inputs the security parameter λ. It outputs a secret key K, and an (initially empty) encrypted database* EDB *to be outsourced to the server. Alternatively, one can define a setup algorithm which takes as input the security parameter λ, and a graph G. In this case, the algorithm outputs a key K and an encrypted database* EDB *encrypting G.*
- $((K', R), (\mathsf{EDB}', R) \leftarrow \mathsf{Qry}_e((K, q), \mathsf{EDB})$*: In the query protocol, the user inputs a secret key K and a query $q \in \mathcal{X} \cup \mathcal{Y}$. The server inputs the encrypted database* EDB*. The user outputs a possibly updated key K', while the server outputs a possibly updated encrypted database* EDB'*. Both the user and the server output a sequence of responses R. For non-interactive schemes, the user first runs* $(K', \tau_q) \leftarrow \mathsf{QryTkn}(K, q)$ *to generate a query token τ_q. The server then runs* $(\mathsf{EDB}', R) \leftarrow \mathsf{Qry}_e(\tau_q, \mathsf{EDB})$ *and outputs the query results R.*
- $(K', \mathsf{EDB}') \leftarrow \mathsf{Udt}_e((K, u), \mathsf{EDB})$*: In the update protocol, the user inputs a secret key K and an update $u \in \{\mathsf{Add}, \mathsf{Del}\} \times (\mathcal{X} \cup \mathcal{Y})$. The server inputs the encrypted database* EDB*. The user outputs a possibly updated key K'. The server outputs an updated encrypted database* EDB'*. For non-interactive schemes, the user first runs* $(K', \tau_u) \leftarrow \mathsf{UdtTkn}(K, u)$ *to generate an update token τ_u. The server then runs* $\mathsf{EDB}' \leftarrow \mathsf{Udt}_e(\tau_u, \mathsf{EDB})$ *and outputs the updated database* EDB'*.*

A DSSE *scheme for the space $\mathcal{G}$ is said to be correct if, for all $\lambda \in \mathbb{N}$, all K and* EDB *output by* $\mathsf{Setup}(1^\lambda)$*, and all sequences of queries and updates, the responses to the plaintext queries equal those to the corresponding encrypted queries.*

Let $\mathcal{L}_e$, $\mathcal{L}_q$, and $\mathcal{L}_u$ be stateful leakage algorithms. A DSSE scheme is said to be ($\mathcal{L}_e$, $\mathcal{L}_q$, $\mathcal{L}_u$)-secure against adaptive dynamic chosen-query attacks (CQA2), if there exists a simulator which, when given the leakages specified by the leakage algorithms, indistinguishably simulates the encrypted database, the query results, and the updates. The formal definition can be found in the full version.

3 Forward Privacy in DSSE

Intuitively, forward privacy ensures that newly added data remains hidden to the server who might have learned some secrets during previous queries, until it must be revealed by a later query. To formalize, instead of tinkering with the CQA2-security definition of DSSE, we define forward privacy based on the property of the leakage function $\mathcal{L}_u$. This is more convenient since the information leaked by $\mathcal{L}_u$ is sufficient for the simulator in the CQA2-security definition to simulate the updates. Similar to the semantic security of encryption schemes, we require that the leakages $\mathcal{L}_u$ on a pair of updates are indistinguishable, capturing the idea that not even a single bit about the update is leaked to the server. Note that the definition does not limit the types of the update u. Indeed, we can consider forward privacy for not only additions, but also deletions. In layman terms, suppose that the server learns during the query protocol about the association of q with some data, which is then deleted by the client. The server should not notice that the data are deleted until q is queried again: By that time the server can compare the query results and discover the deletion.

3.1 Our Definition

We first give a general definition of forward privacy parametrized by a set of updates and a restriction function, then discuss useful ways to parameterize it.

Definition 2 (Forward Privacy). *Let* DSSE *be a* $(\mathcal{L}_e, \mathcal{L}_q, \mathcal{L}_u)$*-CQA2 secure DSSE scheme for labeled bipartite graphs specified by* $\mathcal{G}$*. Let* $\mathcal{U}$ *be a set of updates and restriction* $p : \mathcal{U}^2 \to \{0, 1\}$ *be a predicate function. We say that* DSSE *is:* $(\mathcal{U}, p)$*-forward private, if for any* $u_b \in \mathcal{U}$ *where* $b \in \{0, 1\}$ *such that* $p(u_0, u_1) = 1$*, and any PPT distinguisher* $\mathcal{D}$*, it holds that*

$$|\Pr[\mathcal{D}(\mathcal{L}_u(u_0)) = 1] - \Pr[\mathcal{D}(\mathcal{L}_u(u_1)) = 1]| \leq \mathsf{negl}(\lambda).$$

Table 1 lists some useful combinations of $\mathcal{U}$ and p, denoted by $\mathcal{U}_i$ and p_i respectively for $i \in [6]$. One may also consider a set $\mathcal{U}$ which is a union of some of the (disjoint) $\mathcal{U}_i$'s, and a restriction p which is a composition of the corresponding p_i's: For $p_i : \mathcal{U}_i^2 \to \{0, 1\}$, define $p = p_i + p_j : (\mathcal{U}_i \cup \mathcal{U}_j)^2 \to \{0, 1\}$ such that $(p_i + p_j)(u) = 1$ if $u \in \mathcal{U}_i$ and $p_i(u) = 1$, or $u \in \mathcal{U}_j$ and $p_j(u) = 1$.

Note that there might exist schemes which are $(\mathcal{U}_i, p_i)$- and $(\mathcal{U}_j, p_j)$-forward private but not $(\mathcal{U}_i \cup \mathcal{U}_j, p_i + p_j)$-forward private. For example, if $\mathcal{U}_i$ and $\mathcal{U}_j$ are sets of additions and deletions respectively, while the distinguisher cannot tell which addition or deletion is chosen, it can separate additions from deletions.

Table 1. Useful combinations of $\mathcal{U}$ and p as parameters for forward privacy. (See Sect. 2.1 and the full version for details about the update operations.)

i	Type	Sets of updates $\mathcal{U}_i$	Updates $u_b \in \mathcal{U}_i$ $b \in \{0,1\}$	Possible conditions such that $p_i = 1$ (If there are no restrictions, $p_i \equiv 1$)
1	Edge	$\{(\mathsf{Add}, x, y, w)\}$	$(\mathsf{Add}, x_b, y_b, w_b)$	$x_0 = x_1$ or $y_0 = y_1$
2	Node	$\{(\mathsf{Add}, \mathbf{x}, y, \mathbf{w})\}$	$(\mathsf{Add}, \mathbf{x}_b, y_b, \mathbf{w}_b)$	$y_0 = y_1$
3	Node	$\{(\mathsf{Add}, x, \mathbf{y}, \mathbf{w})\}$	$(\mathsf{Add}, x_b, \mathbf{y}_b, \mathbf{w}_b)$	$x_0 = x_1$
4	Edge	$\{(\mathsf{Del}, x, y)\}$	(Del, x_b, y_b)	$x_0 = x_1$ or $y_0 = y_1$
5	Node	$\{(\mathsf{Del}, x)\}$	(Del, x_b)	$\lvert(x_0, *, *)\rvert = \lvert(x_1, *, *)\rvert$
6	Node	$\{(\mathsf{Del}, y)\}$	(Del, y_b)	$\lvert(*, y_0, *)\rvert = \lvert(*, y_1, *)\rvert$

3.2 Bost's Definition

The forward privacy definition of Bost [2] requires that the leakage function of $u = (\mathsf{Op}, x, y, w)$ can be written as a function of the operation Op, the node y, and $|(*, y, *)|$, *i.e.*, the number of edges connected to y, where Op can be addition or deletion. We argue that the range of leakage functions which satisfy this requirement is so wide, such that the definition does not precisely describe in what sense a DSSE scheme is forward secure. At one extreme, consider a scheme of which the leakage function is given by $\mathcal{L}_u(\mathsf{Op}, x, y, w) = (\mathsf{Op}, y, |(*, y, *)|)$. If $\mathcal{L}_u$ accurately (not overly) captures the leakage, then adding or removing edges to or from a node y leaks the identity of the node y itself. Such scheme is vulnerable to frequency attacks: The attacker keeps a table mapping each y to the number of times y is updated. With the aid of external information, it can possibly extract information about y, such as its importance. Similar attacks have been demonstrated using search patterns [14]. At another extreme, Bost's construction [2] leaks nothing during updates, *i.e.*, $\mathcal{L}_u(\mathsf{Op}, x, y, w) = \phi$. On the other hand, Bost's definition is also not general enough. There are other types of leakage functions, *e.g.*, $\mathcal{L}_u(\mathsf{Op}, x, y, w) = (\mathsf{Op}, x, |(x, *, *)|)$, which intuitively capture forward privacy, but are not covered by this definition. In contrast, our definition of $(\mathcal{U}, p)$-forward privacy classified different types of update in a more fine-grained manner.

In the perspective of our model, Bost's definition can be regarded as special cases of $(\mathcal{U}_1, p_1)$ - (achieved by our generic construction in Sect. 4) and $(\mathcal{U}_4, p_4)$-forward privacy. Intuitively, the restrictions p_1 and p_4 mean that an update $u = (\mathsf{Op}, x, y, w)$ is protected by hiding one end of the connection between x and y. Therefore, similar to the above, it might be the case that the updates u_0 and u_1, where $u_b = (\mathsf{Add}, x_b, y, w)$ for $b \in \{0, 1\}$, are linkable as they correspond to the same node y, making the scheme vulnerable to the same frequency attack. On the other hand, $(\mathcal{U}_1, 1)$ - (achieved by our concrete construction in Sect. 5) and $(\mathcal{U}_4, 1)$-forward privacy completely hides the relation (x, y), making the addition and deletion of an edge (x, y, w) oblivious respectively (since w can be hidden simply by symmetric key encryption). Whether or not the stronger forward privacy is needed depends on the specific application scenarios.

3.3 Forward Privacy for Deletions

In the rest of this work, we focus on $\mathcal{U} = \mathcal{U}_1 = \{(\mathsf{Add}, x, y, w)\}$, with and without restrictions, *i.e.*, $p = p_1$ and $p \equiv 1$, respectively. In other words, we do not consider forward privacy for deletions. To argue for this design decision, we observe that while the scheme of Bost [2] performs "lazy edge deletion" (adding "deleted" edges rather than actually deleting), ours perform actual node deletion. The former increases the size of the encrypted database (by one edge), hence it is possible to make (edge) additions and deletions indistinguishable. The latter allows immediate space reclamation. This makes edge additions (which usually increase the size of the encrypted database) and node deletions easily distinguishable. Furthermore, since actual deletions of different nodes may result in a shrink of the database size in varying degrees, they are also easily distinguishable. In effect, we trade "forward privacy for (lazy) deletions" for efficiency. If forward privacy for deletions is a concern and lazy deletions are acceptable, using a similar technique of maintaining another instance of encrypted database for lazy deletions [2], schemes which are forward private for edge additions can be generically transformed to provide forward privacy for both edge additions and node deletions (simultaneously): To delete a node $y \in Y$, the client adds an edge connecting y to a special "deleted" node in X. We leave the details to the full version of this paper. In this sense, forward privacy for additions is a key property. We also believe that it is sufficient for practical applications by itself.

4 Forward Privacy from Any DSSE

In this section, we will show that a DSSE scheme with $(\mathcal{U}_1, p_1)$-forward privacy can be constructed from any DSSE scheme $\mathcal{E}$, where $\mathcal{U}_1$ and p_1 are defined in Table 1. For simplicity of our description below, we assume the base scheme to be non-interactive, so that the resulting scheme is also non-interactive[3]. Our transformation can be easily adapted to interactive schemes.

Our construction is inspired by that of Rizomiliotis and Gritzalis [15] which uses fresh keys for newly added data. The main idea is to locally maintain a table γ of pseudorandom function (PRF) keys K_x and counters c_x for each query $q = x$, so that adding an edge (x, y, w) is translated to adding another edge $(F(K_x, c_x), y, w)$. Our scheme also adopts the technique of Hahn and Kerschbaum [8], who observe that when the set $(x, *, *)$ is leaked upon querying on x, there is no need to protect the set by encryption any longer. To speed up subsequent queries, the server should thus transfer the set encrypted in the scheme to a plaintext bipartite graph $\hat{G}$. We assume an efficient data structure for representing the graph $\hat{G}$, so that neighbor queries, edge additions, and node deletions in $\hat{G}$ are parallelizable and have time complexity linear in the number of affected nodes only. There might be many ways to construct such a data structure. Cascaded triangles introduced in Sect. 5 is one example.

[3] Candidate base schemes include [13] and a modified version of [11].

4.1 Our Construction

Let $\mathcal{E}$ be a DSSE scheme for $\mathcal{G} = \mathcal{G}(\{0,1\}^\lambda, \mathcal{Y}, \mathcal{W})$, and $F : \{0,1\}^\lambda \times \mathcal{X} \to \{0,1\}^\lambda$ be a pseudorandom function (PRF). We construct a DSSE scheme for $\mathcal{G}' = \mathcal{G}'(\mathcal{X}, \mathcal{Y}, \mathcal{W})$. The resulting scheme supports queries over $\mathcal{X}$, assuming the base scheme supports queries over $\{0,1\}^\lambda$. Figures 1 and 2 formally describe the construction.

The setup algorithm initializes the base scheme $\mathcal{E}$, which yields a secret key $\tilde{K}$ and encrypted database $\tilde{\mathsf{EDB}}$. It also initializes an empty dictionary γ and an empty bipartite graph $\hat{G} \in \mathcal{G}'$. The new secret key K consists of $\tilde{K}$ and γ, while $\tilde{\mathsf{EDB}}$ and $\hat{G}$ are outsourced to the server. The dictionary γ maps a query $q = x \in \mathcal{X}$ to a PRF key K_x and a counter c_x.

To perform an update $u = (\mathsf{Add}, x, y, w)$, the client increments $c_x \leftarrow c_x + 1$, and transforms the update into $\tilde{u} = (\mathsf{Add}, F(K_x, c_x), y, w)$ of the base scheme. To perform an update $u = (\mathsf{Del}, x)$, the client removes the x-th row of γ, and sends to the server the update tokens for $(\mathsf{Del}, F(K_x, i))$ for $i \in [c_x]$. The update $u = (\mathsf{Del}, y)$ is processed as in the base scheme.

Finally, to query $q = x \in \mathcal{X}$, the client removes the x-th row of γ, and sends to the server the query tokens of $F(K_x, i)$ for $i \in [c_x]$. Given these tokens, the server retrieves the intended response R. Additionally, the client also sends the update tokens for $(\mathsf{Del}, F(K_x, i))$ for $i \in [c_x]$. The server uses these tokens to collect and remove the set of edges $R = (x, *, *)$ from the base scheme, and merge R to the plaintext graph $\hat{G}$.

The correctness follows directly from that of the base scheme $\mathcal{E}$.

```
$(K, \mathsf{EDB}) \leftarrow \mathsf{Setup}(1^\lambda)$
    $(\tilde{K}, \tilde{\mathsf{EDB}}) \leftarrow \mathcal{E}.\mathsf{Setup}(1^\lambda)$
    $\gamma = \phi, \hat{G} = \phi$
    return $K = (\tilde{K}, \gamma), \mathsf{EDB} = (\tilde{\mathsf{EDB}}, \hat{G})$
```

```
$(K', \tau_q) \leftarrow \mathsf{QryTkn}(K, q)$
if $q = x \in \mathcal{X} \ \wedge \ \gamma[x] \neq \bot$ then
    $(K_x, c_x) \leftarrow \gamma[x]$
    $\gamma[x] \leftarrow \bot$
    for $i = 1, \ldots, c_x$ do
        $\tilde{\tau}_i \leftarrow \mathcal{E}.\mathsf{QryTkn}(\tilde{K}, F(K_x, i))$
        $\tilde{\tau}_i^- \leftarrow \mathcal{E}.\mathsf{UdtTkn}(\tilde{K}, (\mathsf{Del}, F(K_x, i)))$
    endfor
    $\tau_q \leftarrow (x, \{\tilde{\tau}_i, \tilde{\tau}_i^-\}_{i=1}^{c_x})$
else
    $(q = x \in \mathcal{X})\ ?\ \tau_q \leftarrow x \ :\ \tau_q \leftarrow \bot$
endif
return $(K, \tau_q)$
```

```
$(\mathsf{EDB}, R) \leftarrow \mathsf{Qry}_e(\tau_q, \mathsf{EDB})$
Parse $\tau_q$ as $(x, \{\tilde{\tau}_i, \tilde{\tau}_i^-\}_{i=1}^{cx})$
$R \leftarrow \mathsf{Qry}(x, \hat{G})$
for $i = 1, \ldots, c_x$ do
    $\tilde{R} \leftarrow \mathcal{E}.\mathsf{Qry}_e(\tilde{\tau}_i, \tilde{\mathsf{EDB}})$
    $\tilde{\mathsf{EDB}} \leftarrow \mathcal{E}.\mathsf{Udt}_e(\tilde{\tau}_i^-, \tilde{\mathsf{EDB}})$
    $R \leftarrow R \cup \tilde{R}$
endfor
foreach $(\tilde{x}, y, w) \in R$ do
    $R \leftarrow R \setminus \{(\tilde{x}, y, w)\} \cup \{(x, y, w)\}$
    $\hat{G} \leftarrow \mathsf{Udt}((\mathsf{Add}, x, y, w), \hat{G})$
endfor
return $(\mathsf{EDB}, R)$
```

```
$\mathsf{EDB}' \leftarrow \mathsf{Udt}_e(\tau_u^+, \mathsf{EDB}),\ u = (\mathsf{Add}, \ldots)$
$\tilde{\mathsf{EDB}} \leftarrow \mathcal{E}.\mathsf{Udt}(\tilde{\tau}_u^+, \tilde{\mathsf{EDB}})$
return $\mathsf{EDB}$
```

Fig. 1. Algorithms of generic construction for forward privacy

```
$(K', \tau_u^+) \leftarrow \mathsf{UdtTkn}(K, u = (\mathsf{Add}, \ldots))$

parse $u$ as $(\mathsf{Add}, x, y, w)$
if $\gamma[x] = \perp$ then
  $K_x \leftarrow \{0,1\}^\lambda, c_x \leftarrow 1$
else
  $(K_x, c_x) \leftarrow \gamma[x], c_x \leftarrow c_x + 1$
endif
$\gamma[x] \leftarrow (K_x, c_x)$
$\tau_u^+ \leftarrow \mathcal{E}.\mathsf{UdtTkn}(\tilde{K}, (\mathsf{Add}, F(K_x, c_x), y, w))$
return $(K, \tau_u^+)$
```

```
$\mathsf{EDB}' \leftarrow \mathsf{Udt}_e(\tau_u^-, \mathsf{EDB}),\ u = (\mathsf{Del}, \cdot)$

if $\tau_u^- = (x \in \mathcal{X}, \{\tilde{\tau}_i^-\}_{i=1}^{c_x})$ then
  $\hat{G} \leftarrow \mathsf{Udt}((\mathsf{Del}, x), \hat{G})$
  for $i = 1, \ldots, c_x$ do
    $\widetilde{\mathsf{EDB}} \leftarrow \mathcal{E}.\mathsf{Udt}_e(\tau_i^-, \widetilde{\mathsf{EDB}})$
  endfor
elseif $\tau_u^- = (y \in \mathcal{Y}, \tilde{\tau}^-)$ then
  $\hat{G} \leftarrow \mathsf{Udt}((\mathsf{Del}, y), \hat{G})$
  $\widetilde{\mathsf{EDB}} \leftarrow \mathcal{E}.\mathsf{Udt}_e(\tau^-, \widetilde{\mathsf{EDB}})$
endif
return $\mathsf{EDB}$
```

```
$(K', \tau_u^-) \leftarrow \mathsf{UdtTkn}(K, u = (\mathsf{Del}, \cdot))$

if $u = (\mathsf{Del}, x)$ then
  if $\gamma[x] \neq \perp$ then
    $(K_x, c_x) \leftarrow \gamma[x]$
    $\gamma[x] \leftarrow \perp$
    for $i = 1, \ldots, c_x$ do
      $r_i \leftarrow F(K_x, i)$
      $\tilde{\tau}_i^- \leftarrow \mathcal{E}.\mathsf{UdtTkn}(\tilde{K}, (\mathsf{Del}, r_i))$
    endfor
    $\tau_u^- \leftarrow (x, \{\tilde{\tau}_i^-\}_{i=1}^{c_x})$
  else
    $\tau_u^- \leftarrow x$
  endif
elseif $u = (\mathsf{Del}, y)$ then
  $\tilde{\tau}^- \leftarrow \mathcal{E}.\mathsf{UdtTkn}(\tilde{K}, (\mathsf{Del}, y))$
  $\tau_u^- \leftarrow (y, \tilde{\tau}^-)$
else
  $\tau_u^- \leftarrow \perp$
endif
return $\tau_u^-$
```

Fig. 2. Algorithms of generic construction for forward privacy (cont.)

4.2 Analysis

Efficiency. Our generic transformation almost preserves the efficiency of the underlying DSSE scheme. For most algorithms, the preservation is apparent. We highlight the slightly more complicated cases, namely, the query on x and the update (Del, x). In the former, c_x queries on $F(K_x, i)$ for $i \in [c_x]$ are executed, while in both cases c_x deletions $(\mathsf{Del}, F(K_x, i))$ for $i \in [c_x]$ are required. We analyze their efficiency assuming the following operations of the underlying DSSE scheme each takes constant time: the computation of the query token for each $F(K_x, i)$; the server computation for querying on each $F(K_x, i)$ (since by the pseudorandomness of F, the query should only return a single edge); the computation of each delete token; and the server computation for deleting each set $(F(K_x, i), *, *)$ (since each set is actually a singleton). Overall, the resulting scheme incurs $O(c_x)$ computation and communication costs for both the client and the server where c_x is the number of newly matched data item. These are extra costs on top of the costs for retrieving the previously matched data (in plaintext), *i.e.*, constant computation cost of the client, sublinear computation cost of the server, and sublinear communication cost of both. Since c_x is reset to zero whenever x is queried, the amortized extra costs of both the client and the server are low.

Security. Let $\mathcal{E}$ be an $(\tilde{\mathcal{L}_e}, \tilde{\mathcal{L}_q}, \tilde{\mathcal{L}_u})$-CQA2 secure DSSE scheme for labeled bipartite graphs specified by $\mathcal{G}$; and $F : \{0,1\}^\lambda \times \mathbb{N} \rightarrow \{0,1\}^\lambda$ be a pseudorandom function. Our construction is CQA2-secure and forward private with respect to the following leakage functions. The proof can be found in the full version.

- $\mathcal{L}_e(\mathcal{G}') = \tilde{\mathcal{L}_e}(\mathcal{G})$,
- $\mathcal{L}_u(\mathsf{Add}, x, y, w) = (\tilde{x}, \tilde{\mathcal{L}_u}(\mathsf{Add}, \tilde{x}, y, w))$ for dummy node $\tilde{x} \leftarrow \{0,1\}^\lambda$,
- $\mathcal{L}_u(\mathsf{Del}, x) = (x, \{\tilde{\mathcal{L}_u}(\mathsf{Del}, \tilde{x}_i)\}_{i=1}^{c_x})$,
- $\mathcal{L}_u(\mathsf{Del}, y) = (y, \tilde{\mathcal{L}_u}(\mathsf{Del}, y))$,
- $\mathcal{L}_q(x) = (x, \mathsf{AP}_t(x), \{\tilde{\mathcal{L}_q}(\tilde{x}_i), \tilde{\mathcal{L}_u}(\mathsf{Del}, \tilde{x}_i)\}_{i=1}^{c_x})$, for dummy nodes $\tilde{x}_i$ defined by $\mathcal{L}_u(\mathsf{Add}, x, y, \cdot)$ for updates after the previous query on x.

Theorem 1. *Assume that $\mathcal{E}$ is $(\tilde{\mathcal{L}_e}, \tilde{\mathcal{L}_q}, \tilde{\mathcal{L}_u})$-CQA2 secure, and F is pseudorandom, then the above construction is $(\mathcal{L}_e, \mathcal{L}_q, \mathcal{L}_u)$-CQA2 secure. Furthermore, let p'_1 be a function such that $p'_1(u_0, u_1) = 1$ if and only if $y_0 = y_1$ and $w_0 = w_1$*[4]*. Then the construction is $(\mathcal{U}_1, p'_1)$-forward private, where $\mathcal{U}_1$ is defined in Table 1.*

5 Forward Privacy from Scratch

We next construct (interactive) DSSE which achieves forward privacy directly. First, a new data structure, named *cascaded triangles*, is designed to represent labeled bipartite graphs which supports neighbor queries, edge additions, and node deletions efficiently. We then transform it into its encrypted version.

The construction of cascaded triangles is motivated by the following. Since the neighbor queries and node deletions require traversing the sets $(x, *, *)$ and $(*, y, *)$, their data structure representations are critical for the efficiency, and later the security of the resulting DSSE scheme. In particular, they determine whether the desired operations can be executed in parallel, and how much information has to be leaked to the server for performing such operations. For example, linked list [11] traversal and updates are inherently sequential, yet updating only has a local effect (on the previous and next nodes). However, random binary search tree [12] exhibits parallel traversal and updates, yet updating affects (or leaks) the subtree rooted from the altered node. Thus, cascaded triangles is designed to support parallel traversal and local updates simultaneously.

5.1 Warm Up: Plaintext Cascaded Triangles

Overview. Our goal is to store the bipartite graph G so that neighbor queries and deletions over X or Y can be executed in sublinear time. We can do so by pre-computing the set of edges connected to each x (and y), and storing the set by a data structure which allows efficient traversal. For any x, consider

[4] We can drop the restriction $w_0 = w_1$ by simply encrypting w during additions.

the set of edges connecting x. We pack this set into multiple perfect binary trees, called triangles, by first forming the largest triangle possible, subtracting the edges which are already packed, then continuing to form the next largest triangle. The resulting triangles thus have strictly decreasing (cascading) heights, except for the last two which may have equal heights. This invariant is to be maintained in any later updates. To add an edge connecting x, we check if the two shortest triangles have the same height. If so, we add a new node representing the new edge on top of the two triangles, merging them into one larger triangle. Otherwise, the new node is added as a new triangle of height 1. To delete an edge, we delete the node representing this edge by replacing it with the root of the shortest triangle, splitting the latter into two smaller triangles. We can see that the invariant is still maintained after each addition and each deletion. Finally, to traverse the data structure, one may use any (parallel) tree traversal algorithms.

Setup. Concretely, cascaded triangles consists of dictionaries γ, δ, and η. We can think of γ as local states stored at the client side, while δ and η are outsourced to the server. The dictionary η is the one to store the actual data. It maps an address addr to a tuple (a, b), where $b = (\mathsf{chd}_0, \mathsf{chd}_1)$ specifies the addresses of the left and right child respectively. b is maintained so that nodes in η form perfect binary trees (triangles). This means either both addresses $(\mathsf{chd}_0, \mathsf{chd}_1)$ are empty ($\perp$) or both are valid addresses occupied in η. To store an edge (x, y, w) into η, the edge is copied twice into $a^{\triangle} = a^{\triangledown} = (x, y, w)$. The tuples $(a^{\triangle}, b^{\triangle})$ and $(a^{\triangledown}, b^{\triangledown})$ are stored at random addresses $\mathsf{addr}^{\triangle}$ and $\mathsf{addr}^{\triangledown}$ in η respectively. The addresses $\mathsf{addr}^{\triangle}$ and $\mathsf{addr}^{\triangledown}$ are said to be *duals* of each other, and are registered in δ, *i.e.*, $\delta[\mathsf{addr}^{\triangle}] = \mathsf{addr}^{\triangledown}$ and $\delta[\mathsf{addr}^{\triangledown}] = \mathsf{addr}^{\triangle}$.

Globally, we describe how the nodes in η are connected to each other via the addresses stored in b. We collect all $\eta[\mathsf{addr}] = (a, b)$ corresponding to the edges in $(q, *, *)$ (or $(*, q, *)$). Let $n_q = |(q, *, *)|$ (or $|(*, q, *)|$). We pack these tuples into triangles of cascading heights $h_1 \leq h_2 < h_3 < \ldots < h_k$, where $k \leq \lceil \lg n_q \rceil + 1$. Note the possible equality between h_1 and h_2 but not the others. The ordering of the nodes is implicitly determined by the update algorithms, but does not matter here. Using the procedures described in the overview, given the size n_q, the heights $h_1, \ldots, h_k$ are uniquely determined. It is possible to represent the heights compactly by a trinary string $h \in \{0, 1, 2\}^{\lceil \lg n_q \rceil}$, such that the i-th trit (trinary digit) is set to t, if there are t triangles of height i. Due to the constraints on the heights, only the least significant non-zero trit can be set to 2. Finally, the addresses of the roots and the heights of these triangles are stored in $\gamma[q] = (\mathsf{addr}_1, \ldots, \mathsf{addr}_k, h)$.

Queries and Traversal. Traversing the sets $(q, *, *)$ and $(*, q, *)$ are straightforward with the above structure: First, retrieve the roots of the triangles from $\gamma[q] = (\mathsf{addr}_1, \ldots, \mathsf{addr}_k, h)$. Then, use parallel tree-traversal algorithms to traverse the trees from the root starting at each of these addresses. Notice that the neighbor query function Qry on x and y are supported by traversing the sets $\gamma[x] = (x, *, *)$ and $\gamma[y] = (*, y, *)$ respectively.

Add. To add a new edge (x, y, w), we first retrieve $\gamma[x] = (\mathsf{addr}_1^{\vartriangle}, \ldots, \mathsf{addr}_k^{\vartriangle}, h^{\vartriangle})$ and check whether the triangles rooted at $\mathsf{addr}_1^{\vartriangle}$ and $\mathsf{addr}_2^{\vartriangle}$ have the same height.

To do so, we take a detour to describe the $+1$ *operation* in $h + 1$. Recall that only the least significant non-zero trit, say the i-th trit, in h can be set to 2. The operation $h + 1$ adds 1 to the i-th trit (instead of the least significant trit as in normal addition), which sets the i-th trit to 0 and carries 1 to the $(i+1)$-trit. Denote this event by $\mathsf{Carry}(h+1) = 1$. Otherwise, $h+1$ simply adds 1 to the least significant trit as in normal addition, denoted by $\mathsf{Carry}(h+1) = 0$. Later, for deletion, we would need the -1 *operation* which is the "reverse" of $+1$. Concretely, $h-1$ subtracts 1 from the least significant non-zero trit, say the i-th trit, and set the $(i-1)$-th trit to 2 if $i > 1$.

With the above procedures, checking the heights of the first two triangles can be done by simply checking whether the least significant non-zero trit in $h^{\vartriangle}$ equals 2, or equivalently whether $\mathsf{Carry}(h^{\vartriangle} + 1) = 1$. If that is the case, we add $(a^{\vartriangle}, b^{\vartriangle})$ where $a^{\vartriangle} = (x, y, w)$ and $b^{\vartriangle} = (\mathsf{addr}_1^{\vartriangle}, \mathsf{addr}_2^{\vartriangle})$ to a random address $\mathsf{addr}^{\vartriangle}$ in η. We then update $\gamma[x] \leftarrow (\mathsf{addr}^{\vartriangle}, \mathsf{addr}_3^{\vartriangle}, \ldots, \mathsf{addr}_k^{\vartriangle}, h^{\vartriangle} + 1)$, where the two addresses $\mathsf{addr}_1^{\vartriangle}$ and $\mathsf{addr}_2^{\vartriangle}$ are replaced by the new $\mathsf{addr}^{\vartriangle}$. Otherwise, we add $(a^{\vartriangle}, b^{\vartriangle})$ where $a^{\vartriangle} = (x, y, w)$ and $b^{\vartriangle} = (\perp, \perp)$ to a random address $\mathsf{addr}^{\vartriangle}$ in η, and update $\gamma[x] \leftarrow (\mathsf{addr}^{\vartriangle}, \mathsf{addr}_1^{\vartriangle}, \ldots, \mathsf{addr}_k^{\vartriangle}, h^{\vartriangle} + 1)$. The difference is highlighted in red.

Similarly, we retrieve $(\mathsf{addr}_1^{\triangledown}, \ldots, \mathsf{addr}_k^{\triangledown}, h^{\triangledown}) \leftarrow \gamma[y]$ and check whether $\mathsf{Carry}(h^{\triangledown} + 1) = 1$. If so, we add $((x, y, w), (\mathsf{addr}_1^{\triangledown}, \mathsf{addr}_2^{\triangledown}))$ to a random address $\mathsf{addr}^{\triangledown}$ in η, and update $\gamma[y] \leftarrow (\mathsf{addr}^{\triangledown}, \mathsf{addr}_3^{\triangledown}, \ldots, \mathsf{addr}_k^{\triangledown}, h^{\triangledown} + 1)$. Otherwise, we add $((x, y, w), (\perp, \perp))$ to $\mathsf{addr}^{\triangledown}$, and update $\gamma[y] \leftarrow (\mathsf{addr}^{\triangledown}, \mathsf{addr}_1^{\triangledown}, \ldots, \mathsf{addr}_k^{\triangledown}, h^{\triangledown} + 1)$.

Delete. To delete node x, or equivalently the set of edges $(x, *, *)$, we first traverse the set $(x, *, *)$ using the above traversal algorithm. We delete all the traversed nodes in η as well as the row $\gamma[x]$. It remains to delete the dual nodes of the traversed nodes. To do so, for each traversed address $\mathsf{addr}^{\vartriangle}$ with $a = (x, y, w)$, look up $\delta[\mathsf{addr}^{\vartriangle}] = \mathsf{addr}^{\triangledown}$ and $\gamma[y] = (\mathsf{addr}_1^{\triangledown}, \ldots, \mathsf{addr}_k^{\triangledown}, h^{\triangledown})$. We wish to replace the content of $\eta[\mathsf{addr}^{\triangledown}] = (a^{\triangledown}, b^{\triangledown})$ located in the middle of some triangle by the content of $\eta[\mathsf{addr}_1^{\triangledown}]$, the root of the smallest triangle, which splits the smallest triangle into two smaller ones. In this way, the heights of the resulting triangles still satisfy the required constraints.

Concretely, we perform the following steps. (1) Look up $\delta[\mathsf{addr}_1^{\triangledown}] = \mathsf{addr}_1^{\vartriangle}$. (2) Delete $\delta[\mathsf{addr}^{\vartriangle}]$ and $\delta[\mathsf{addr}_1^{\triangledown}]$. (3) Update $\delta[\mathsf{addr}_1^{\vartriangle}] \leftarrow \mathsf{addr}^{\triangledown}$ and $\delta[\mathsf{addr}^{\triangledown}] \leftarrow \mathsf{addr}_1^{\vartriangle}$. (4) Look up $\eta[\mathsf{addr}_1^{\triangledown}] = (a_1^{\triangledown}, b_1^{\triangledown})$, where $b_1^{\triangledown} = (\mathsf{addr}_0^{\blacktriangledown}, \mathsf{addr}_1^{\blacktriangledown})$. (5) Update $\eta[\mathsf{addr}^{\triangledown}] \leftarrow (a_1^{\triangledown}, b_1^{\triangledown})$ and delete $\eta[\mathsf{addr}_1^{\triangledown}]$. (6) Update $\gamma[y] = (\mathsf{addr}_0^{\blacktriangledown}, \mathsf{addr}_1^{\blacktriangledown}, \mathsf{addr}_2^{\triangledown}, \ldots, \mathsf{addr}_k^{\triangledown}, h^{\triangledown} - 1)$, where $h^{\triangledown} - 1$ is the reverse of $h^{\triangledown} + 1$. We omit the deletion of $(*, y, *)$ which is similar to the above.

Efficiency. The storage cost of cascaded triangles is $O(|X| + |Y| + |G|) = O(|G|)$. The complexity of querying (or deleting) x and y are $O(|(x, *, *)|)$ and $O(|(*, y, *)|)$ respectively. Addition of an edge can be computed in constant time.

5.2 Our Construction: Encrypted Cascaded Triangles

We now transform the plaintext cascaded triangles into its encrypted version. Recall that the goal of the client is to encrypt a labeled bipartite graph G into an encrypted database EDB which still supports neighbor queries, edge additions, and node deletions. To do so, the client represents G using cascaded triangles, stores γ locally, and outsources δ and the encrypted η to the server. The encryption should be *non-committing*, such that there exists a simulator which can simulate the ciphertexts for new data without any leakage. When some data is to be returned upon queries or is deleted, the simulator is given enough leakage so that it can "explain" the dummy ciphertexts. Furthermore, when the sets $(x, *, *)$ and $(*, y, *)$ are leaked upon querying x and y respectively, there is no need to protect the sets by encryption any longer [8]. To speed up subsequent queries, the server should transfer the set from the encrypted η to a plaintext labeled bipartite graph $\hat{G}$. Thus, we can conceptually split G into two disjoint subgraphs $G = \tilde{G} \cup \hat{G}$, where $\tilde{G}$ is encrypted and $\hat{G}$ is in plaintext.

Encrypted cascaded triangles are similar to the plaintext counterparts. We highlight the differences and omit the identical parts.

Setup and Overview. Let $\mathsf{NCE}.(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric-key non-committing encryption scheme[5]. The correctness of our scheme will follow directly from that of NCE. For each edge (x, y, w) in the encrypted subgraph $\tilde{G}$, $\eta[\mathsf{addr}^{\triangle}]$ stores the tuple $(c_a^{\triangle}, c_b^{\triangle})$, where $c_a^{\triangle}$ and $c_b^{\triangle}$ are non-committing ciphertexts of $a^{\triangle} = (x, y, w)$ and $b^{\triangle} = (\mathsf{chd}_0, \mathsf{chd}_1)$ under the keys $\mathsf{ak}^{\triangle}$ and $\mathsf{bk}^{\triangle}$ respectively. The keys $\mathsf{ak}^{\triangle}$ and $\mathsf{bk}^{\triangle}$ are independently generated for each x. Similarly, $\eta[\mathsf{addr}^{\triangledown}]$ stores the tuple $(c_a^{\triangledown}, c_b^{\triangledown})$ encrypted under $\mathsf{ak}^{\triangledown}$ and $\mathsf{bk}^{\triangledown}$ respectively. The keys $\mathsf{ak}^{\triangledown}$ and $\mathsf{bk}^{\triangledown}$ are independently generated for each y. For each x, $\gamma[x]$ additionally stores secret keys $\mathsf{ak}^{\triangle}$ and $\mathsf{bk}^{\triangle}$ associated to x. Similarly, for each y, $\gamma[y]$ additionally stores secret keys $\mathsf{ak}^{\triangledown}$ and $\mathsf{bk}^{\triangledown}$ associated to y. Formally, the setup protocol in Fig. 3 shows the initialization of these dictionaries.

Queries. Queries are similar to the plaintext case. Apart from the addresses, the client also sends ak and bk retrieved from $\gamma[q]$ to the server. Using bk, the server decrypts all $b = (\mathsf{chd}_0, \mathsf{chd}_1)$ and traverses the sub-trees. Using ak, it decrypts all $a = (x, y, w)$ which are returned as query results. The server also returns the previous query results $\hat{R}$ stored in $\hat{G}$.

[5] For example, a ciphertext for message m with randomness r can be computed as $c = (r, \mathsf{PRF}(K, r) \oplus m)$, where PRF, modeling a random oracle, is a pseudorandom function with secret key K. In practice, one may substitute PRF with an HMAC.

```
$(K, \mathsf{EDB}) \leftarrow \mathsf{Setup}(1^\lambda, |\mathcal{X}|, |\mathcal{Y}|, |\mathcal{W}|)$
$\gamma = \phi, \delta = \phi, \eta = \phi, \hat{G} = \phi$
return $K = \gamma, \mathsf{EDB} = (\delta, \eta, \hat{G})$

$\mathsf{Trav}(\mathsf{EDB}, \{\mathsf{addr}_j\}_{j=1}^k, \mathsf{ak}, \mathsf{bk})$
if $k > 1$ then
  $R = \phi, D = \phi$
  for $j = 1, \ldots, k$ do
    $(R', D') \leftarrow \mathsf{Trav}(\mathsf{EDB}, \mathsf{addr}_j, \mathsf{ak}, \mathsf{bk})$
    $R = R \cup R', D = D \cup D'$
  endfor
else
  $(c_a, c_b) \leftarrow \eta[\mathsf{addr}_1], \eta[\mathsf{addr}_1] \leftarrow \perp$
  $\overline{\mathsf{addr}}_1 \leftarrow \delta[\mathsf{addr}_1], \delta[\mathsf{addr}_1] \leftarrow \perp$
  if $\mathsf{ak} \neq \perp$ then

$\mathsf{Trav}(\mathsf{EDB}, \{\mathsf{addr}_j\}_{j=1}^k, \mathsf{ak}, \mathsf{bk})$ (cont.)
    $(x, y, w) \leftarrow \mathsf{NCE.Dec}(\mathsf{ak}, c_a)$
  else
    $(x, y, w) \leftarrow c_a$
  endif
  $(\mathsf{chd}_0, \mathsf{chd}_1) \leftarrow \mathsf{NCE.Dec}(\mathsf{bk}, c_b)$
  if $\mathsf{chd}_0 \neq \perp$ $(\vee\ \mathsf{chd}_1 \neq \perp)$ then
    $(R', D') \leftarrow \mathsf{Trav}(\mathsf{EDB}, \mathsf{chd}_0, \mathsf{ak}, \mathsf{bk})$
    $(R'', D'') \leftarrow \mathsf{Trav}(\mathsf{EDB}, \mathsf{chd}_1, \mathsf{ak}, \mathsf{bk})$
    $R = R' \cup R'' \cup \{(x, y, w)\}$
    $D = D' \cup D'' \cup \{\overline{\mathsf{addr}}_1\}$
  else
    $R = \{(x, y, w)\}, D = \{\overline{\mathsf{addr}}_1\}$
  endif
endif
return $(R, D)$
```

Fig. 3. Setup and traverse algorithm of encrypted cascaded triangles

```
$((K', R), (\mathsf{EDB}', R)) \leftarrow \mathsf{Qry}_e((K, q), \mathsf{EDB})$

Client                                                                    Server
$R = \phi$                                   $\xrightarrow{q}$             $\hat{R} \leftarrow \mathsf{Qry}(q, \hat{G})$
if $\gamma[q] \neq \perp$ then               $\xleftarrow{\hat{R}}$
  $(\mathsf{ak}, \mathsf{bk}, \{\mathsf{addr}_j\}_{j=1}^k, h) \leftarrow \gamma[q]$   $\xrightarrow{\mathsf{ak}, \mathsf{bk}, \{\mathsf{addr}_j\}_{j=1}^k}$   $(R, D) \leftarrow \mathsf{Trav}(\mathsf{EDB}, \{\mathsf{addr}_j\}_{j=1}^k, \mathsf{ak}, \mathsf{bk})$
  $\gamma[q] \leftarrow \perp$               $\xleftarrow{R}$             parse $D$ as $(\overline{\mathsf{addr}}_j)_j$
  if $q = x \in \mathcal{X}$ then
    parse $R$ as $\{(x, z_j, w_j)\}_j$
  else // $q = y \in \mathcal{Y}$
    parse $R$ as $\{(z_j, y, w_j)\}_j$
  endif
              $(K, \mathsf{EDB}) \leftarrow \mathsf{DelDual}((K, \{z_j\}_j, (\mathsf{EDB}, \{\overline{\mathsf{addr}}_j\}_j))$
endif
$R \leftarrow R \cup \hat{R}$                                              $R \leftarrow R \cup \hat{R}$
                                                                          foreach $(x, y, w) \in \hat{R}$ do
                                                                            $\hat{G} \leftarrow \mathsf{Udt}((\mathsf{Add}, x, y, w), \hat{G})$
                                                                          endfor
return $(K, R)$                                                           return $(\mathsf{EDB}, R)$
```

Fig. 4. Query protocol of encrypted cascaded triangles

```
$(K', \mathsf{EDB}') \leftarrow \mathsf{DelDual}((K, \{q_j\}_j), (\mathsf{EDB}, \{\overline{\mathsf{addr}}_j\}_j))$

Client                                                                                              Server
for $j = 1, \ldots, n$ do
    $(\mathsf{ak}_j, \mathsf{bk}_j, \{\mathsf{addr}_{j,i}\}_{i=1}^{\ell_j}, h_j) \leftarrow \gamma[q_j]$          $\xrightarrow{\mathsf{addr}_j := \mathsf{addr}_{j,1}}$          $\mathsf{addr}_j^\blacklozenge \leftarrow \delta(\mathsf{addr}_j)$
    $(\mathsf{chd}_0, \mathsf{chd}_1) \leftarrow \mathsf{NCE.Dec}(\mathsf{bk}_j, \eta[\mathsf{addr}_j].c_b)$          $\xleftarrow{\eta[\mathsf{addr}_j].c_b}$          $\delta(\mathsf{addr}_j) \leftarrow \perp$
    if $\mathsf{chd}_0 \neq \perp$ $(\vee\ \mathsf{chd}_1 \neq \perp)$ then                                  $\delta(\mathsf{addr}_j^\blacklozenge) \leftarrow \overline{\mathsf{addr}}_j$
        $\mathsf{addr}_{j,0} \leftarrow \mathsf{chd}_0, \mathsf{addr}_{j,1} \leftarrow \mathsf{chd}_1$                    $\delta(\overline{\mathsf{addr}}_j) \leftarrow \mathsf{addr}_j^\blacklozenge$
        $\gamma[q_j] \leftarrow (\mathsf{ak}_j, \mathsf{bk}_j, \{\mathsf{addr}_{j,i}\}_{i=0}^{\ell_j}), h_j - 1)$          $\eta[\overline{\mathsf{addr}}_j].c_a = \eta[\mathsf{addr}_j].c_a$
    else                                                                                            $\eta[\mathsf{addr}_j] \leftarrow \perp$
        $\gamma[q_j] \leftarrow (\mathsf{ak}_j, \mathsf{bk}_j, \{\mathsf{addr}_{j,i}\}_{i=2}^{\ell_j}), h_j - 1)$
    endif
endfor
```

Fig. 5. Dual deletion protocol of encrypted cascaded triangles

As mentioned before, for the efficiency of subsequent queries, the server should remove revealed entries from η and add them to the plaintext subgraph $\hat{G}$. Thus, the client and the server cooperates to delete the set $(q, *, *)$ or $(*, q, *)$ from η. This conceptually removes the set from the encrypted subgraph $\tilde{G}$. Finally, the server adds the set to $\hat{G}$. Formally, the query protocol is shown in Fig. 4, which utilizes the subroutine Trav and subprotocol $\mathsf{DelDual}$ shown in Figs. 3 and 5 respectively.

Add. Instead of sending $(a^\triangle, b^\triangle)$ in the clear, the client sends their ciphertexts $(c_a^\triangle, c_b^\triangle)$, encrypted under $\mathsf{ak}^\triangle$ and $\mathsf{bk}^\triangle$ retrieved from $\gamma[x]$, to the server respectively. In the case where $\gamma[x] = \perp$, the client generates new secret keys $\mathsf{ak}^\triangle$ and $\mathsf{bk}^\triangle$ using the key generation algorithm of the non-committing encryption scheme. Sending $(a^\triangledown, b^\triangledown)$ requires a similar treatment. Figure 6 formally describes the addition protocol.

Delete. Deletion of $(x, *, *)$ is almost identical to querying x, except that the client does not send out $\mathsf{ak}^\triangle$. Instead, the server returns all $c_a^\triangle$ so that the client can decrypt them locally. After obtaining all the edges (x, y, w), the client and the server cooperate to delete $(x, *, *)$ from η as in the query algorithm. This conceptually removes $(x, *, *)$ from the encrypted subgraph $\tilde{G}$. Finally, the server also removes $(x, *, *)$ from the plaintext subgraph $\hat{G}$. Deletion of $(*, y, *)$ is done similarly. Figure 7 shows the deletion protocol, which also utilizes the subroutines Trav and $\mathsf{DelDual}$ in Figs. 3 and 5 respectively.

```
$(\cdot, \mathsf{EDB}') \leftarrow \mathsf{Udt}_e(\cdot, \mathsf{EDB})$
Receive $(\mathsf{addr}^{\triangle}, \mathsf{addr}^{\triangledown}, c_a^{\triangle}, c_b^{\triangle}, c_a^{\triangledown}, c_b^{\triangledown})$
$\delta[\mathsf{addr}^{\triangle}] \leftarrow \mathsf{addr}^{\triangledown}$, $\delta[\mathsf{addr}^{\triangledown}] \leftarrow \mathsf{addr}^{\triangle}$
$\eta[\mathsf{addr}^{\triangle}] \leftarrow (c_a^{\triangle}, c_b^{\triangle})$, $\eta[\mathsf{addr}^{\triangledown}] \leftarrow (c_a^{\triangledown}, c_b^{\triangledown})$
return EDB

$(K', \cdot) \leftarrow \mathsf{Udt}_e((K, u = (\mathsf{Add}, x, y, w)), \cdot)$
$\mathsf{addr}^{\triangle}, \mathsf{addr}^{\triangledown} \leftarrow \{0,1\}^*$
for $(q, \Diamond) \in \{(x, \triangle), (y, \triangledown)\}$ do
  if $\gamma[q] = \perp$ then
    $\mathsf{ak}, \mathsf{bk} \leftarrow \mathsf{NCE.KGen}(1^{\lambda})$
    $\mathsf{addr}_1, \mathsf{addr}_2 \leftarrow \perp$
    $\gamma[q] \leftarrow (\mathsf{ak}, \mathsf{bk}, \{\mathsf{addr}\}, 1)$
  else
    $(\mathsf{ak}, \mathsf{bk}, \{\mathsf{addr}_j\}_{j=1}^{k}, h) \leftarrow \gamma[q]$
    if $\mathsf{Carry}(h+1)$ then
      $\gamma[q] \leftarrow (\mathsf{ak}, \mathsf{bk}, \{\mathsf{addr}, \mathsf{addr}_j\}_{j=3}^{k}, h+1)$
    else
      $\gamma[q] \leftarrow (\mathsf{ak}, \mathsf{bk}, \{\mathsf{addr}, \mathsf{addr}_j\}_{j=1}^{k}, h+1)$
    endif
  endif
  $c_a^{\Diamond} \leftarrow \mathsf{NCE.Enc}(\mathsf{ak}, (x, y, w))$
  $c_b^{\Diamond} \leftarrow \mathsf{NCE.Enc}(\mathsf{bk}, (\mathsf{addr}_1, \mathsf{addr}_2))$
endfor
Send $(\mathsf{addr}^{\triangle}, \mathsf{addr}^{\triangledown}, c_a^{\triangle}, c_b^{\triangle}, c_a^{\triangledown}, c_b^{\triangledown})$
return $K$
```

Fig. 6. Addition protocol of encrypted cascaded triangles

```
$(K', \mathsf{EDB}') \leftarrow \mathsf{Udt}_e((K, u = (\mathsf{Del}, q)), \mathsf{EDB})$

if $\gamma[q] \neq \perp$ then                $\xrightarrow{q}$                      $\hat{G} \leftarrow \mathsf{Udt}((\mathsf{Del}, q), \hat{G})$
  $(\mathsf{ak}, \mathsf{bk}, \{\mathsf{addr}_i\}_i, h) \leftarrow \gamma[q]$   $\xrightarrow{\mathsf{bk}, \{\mathsf{addr}_i\}_i}$   $(R, D) \leftarrow \mathsf{Trav}(\mathsf{EDB}, \{\mathsf{addr}_i\}_i, \perp, \mathsf{bk})$
  $\gamma[q] \leftarrow \perp$                  $\xleftarrow{R}$                       parse $D$ as $(\overline{\mathsf{addr}}_j)_j$
  if $q \in \mathcal{X}$ then
    $\{(q, z_j, w_j)\}_j \leftarrow \mathsf{NCE.Dec}(\mathsf{ak}, R)$
  else // $q \in \mathcal{Y}$
    $\{(z_j, q, w_j)\}_j \leftarrow \mathsf{NCE.Dec}(\mathsf{ak}, R)$
  endif
            $(K, \mathsf{EDB}) \leftarrow \mathsf{DelDual}((K, \{z_j\}_j), (\mathsf{EDB}, \{\overline{\mathsf{addr}}_j\}_j))$
endif
return $K$                                                      return EDB
```

Fig. 7. Deletion protocol of encrypted cascaded triangles

5.3 Analysis

Storage Cost. For each x, if $n_x = |(x, *, *) \cap \tilde{G}| > 0$, the client stores two λ-bit keys of non-committing encryption, one $\lceil \lg n_x \rceil$-trit string, and at most $\lceil \lg n_x \rceil + 1$ λ-bit addresses. Similar storage is required for each y. In the extreme case where the client adds all possible data and never queries or deletes, the storage cost is $O(\mathsf{poly} \cdot (|\mathcal{X}| \lg |\mathcal{Y}| + |\mathcal{Y}| \lg |\mathcal{X}|))$. However, querying and deleting x (or y) removes

$(x, *, *)$ (or $(*, y, *)$) from $\tilde{G}$, which waives the client local storage for x (or y). Thus, the storage of a reasonable client would be much smaller.

The storage cost of the server is linear in the number of edges (x, y, w) added to the server, which is optimal. Furthermore, if an edge (x, y, w) is revealed due to a previous query, it is stored in plaintext instead of ciphertext, where the former is much better in terms of locality [3].

Computation and Communication Cost. Both the client and the server perform essentially no work during setup: They just initialize empty dictionaries.

During a query on q, the client first looks up its dictionary $\gamma[q]$, which consists of two λ-bit keys of NCE, one $\lceil \lg n_q \rceil$-trit string, and at most $(\lceil \lg n_q \rceil + 1)$ λ-bit addresses. The query q, the keys, and the addresses are sent to the server. The server traverses the set $(x, *, *)$ if $q = x \in \mathcal{X}$, or the set $(*, y, *)$ if $q = y \in \mathcal{Y}$. In the former, the server needs to perform $O(|(x, *, *) \cap \tilde{G}|)$ decryption, and execute $\mathsf{Qry}(q, \hat{G})$ which takes time $|(x, *, *) \cap \hat{G}|$. It then returns the query result of size $|(x, *, *)| = |(x, *, *) \cap \tilde{G}| + |(x, *, *) \cap \hat{G}|$ to the client. The client looks up and sends $|(x, *, *) \cap \tilde{G}|$ addresses to the server, which performs the same amount of I/O tasks, and returns $O(|(x, *, *) \cap \tilde{G}|)$ ciphertexts to the client. The client finalizes by performing $O(|(x, *, *) \cap \tilde{G}|)$ decryption and the same amount of I/O tasks. Overall, both computation and communication complexities for both the client and the server are in the order of $O(\mathsf{poly} \cdot |(x, *, *)|)$, which is optimal for the server. In contrast to the presentation in the formal construction, the number of rounds can be compressed into 4.

Since deletion is almost identical to querying except for local decryption, their overall computation and communication complexities are identical.

For addition, the client performs a constant amount of I/O tasks to update γ, while sending 2 λ-bit addresses and 4 ciphertexts to the server. The server simply writes the ciphertexts to the specified addresses. Therefore, the overall computation, round, and communication complexities for both the client and the server are constant, which is again optimal.

Note that our scheme supports batch operations (querying, addition, and deletion) straightforwardly. In such case, the computation and communication complexities increase linearly while the round complexity remains unchanged.

Security. In the full version, we prove that our scheme is secure against adaptive chosen query attack with very minimal leakage. In particular, our scheme achieves $(\mathcal{U}_1, 1)$-forward privacy, where $\mathcal{U}_1$ is defined in Table 1. We begin by defining the leakage functions. For setup, $\mathcal{L}_u$ only leaks the sizes of the spaces, *i.e.*, $|\mathcal{X}|$, $|\mathcal{Y}|$, and $|\mathcal{W}|$. For addition, $\mathcal{L}_u$ only leaks the update type and the time $\mathsf{Time}(x, y)$ of the addition as the new addresses are truly random and the ciphertexts can be simulated by the simulator of the non-committing encryption scheme. For deletion of $(x, *, *)$, $\mathcal{L}_u$ leaks the update type, x, and the time $\mathsf{Time}(x, y)$ when each of the edges $(x, y, w) \in (x, *, *)$ is added. It also leaks, for each y such that $(x, y, w) \in (x, *, *)$, the time $\mathsf{Time}(*, y)$ when the last edge in $(*, y, *)$ is added. Leakage for deleting $(*, y, *)$ is defined similarly. Finally, for queries on q, $\mathcal{L}_q$ leaks all information leaked by $\mathcal{L}_u$ upon deletion of $(x, *, *)$

if $q = x \in \mathcal{X}$, or $(*, y, *)$ if $q = y \in \mathcal{Y}$, and the access patterns $\mathsf{AP}_t(q)$ of q, assuming it is sorted by the time each response is added. Formally, we define:

- $\mathcal{L}_e(\mathcal{G}) = (|\mathcal{X}|, |\mathcal{Y}|, |\mathcal{W}|)$
- $\mathcal{L}_u(\mathsf{Add}, x, y, w) = (\mathsf{Add}, \mathsf{Time}(x, y))$
- $\mathcal{L}_u(\mathsf{Del}, x) = (\mathsf{Del}, x, \{(\mathsf{Time}(x, y), \mathsf{Time}(*, y)) : (x, y, \cdot) \in (x, *, *)\})$
- $\mathcal{L}_u(\mathsf{Del}, y) = (\mathsf{Del}, y, \{(\mathsf{Time}(x, y), \mathsf{Time}(x, *)) : (x, y, \cdot) \in (*, y, *)\})$
- $\mathcal{L}_q(x) = (x, \mathsf{AP}_t(x), \{(\mathsf{Time}(x, y), \mathsf{Time}(*, y)) : (x, y, \cdot) \in (x, *, *)\})$
- $\mathcal{L}_q(y) = (y, \mathsf{AP}_t(y), \{(\mathsf{Time}(x, y), \mathsf{Time}(x, *)) : (x, y, \cdot) \in (*, y, *)\})$

The simulation is sketched as follows. The simulator first initializes empty dictionaries δ and η, and the empty plaintext graph $\hat{G}$. It maintains a table T mapping time t to time-address tuples $((t_0, \mathsf{addr}), (t_1, \mathsf{addr}^\triangledown)$.

For addition at time t, it samples random addresses addr and $\mathsf{addr}^\triangledown$, and registers them in δ. It sets $T[t] \leftarrow ((t, \mathsf{addr}), (t, \mathsf{addr}^\triangledown))$. It simulates the ciphertexts in $\eta[\mathsf{addr}]$ and $\eta[\mathsf{addr}^\triangledown]$ using the simulator of NCE.

For deletion of $(x, *, *)$, it is given x, which allows it to delete $(x, *, *)$ from $\hat{G}$. It is also given the time $\mathsf{Time}(x, y)$ when each of the edges $(x, y, w) \in (x, *, *)$ is added, and for each y such that $(x, y, w) \in (x, *, *)$ the time $\mathsf{Time}(*, y)$ when the last edge in $(*, y, *)$ is added. It recalls from the table T all addr and $\mathsf{addr}^\triangledown$ pairs which are created at time $\mathsf{Time}(x, y)$ and $\mathsf{Time}(*, y)$. With the knowledge of these addresses, the simulator can maintain δ and η as in the real scheme. It must also output simulated bk and explain the ciphertexts c_b encrypting the addresses. To achieve this, the simulator passes the ciphertexts and the corresponding addresses to the simulator of the non-committing encryption scheme, where the latter outputs the simulated bk. Deletion of $(*, y, *)$ is simulated similarly.

Queries are simulated almost identically as in the simulation of deletions, except that the simulator must now also output simulated ak and explain the ciphertexts c_a encrypting the query result. Similar to the above, this can be done by calling the simulator of the non-committing encryption scheme.

Theorem 2. *Assume that* $\mathsf{NCE}.(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *is a symmetric-key non-committing encryption scheme with message space* $\{0, 1\}^{\max(\lg|\mathcal{X}| + \lg|\mathcal{Y}| + \lg|\mathcal{W}|, 2\lambda)}$*, the above construction is* $(\mathcal{L}_e, \mathcal{L}_q, \mathcal{L}_u)$*-CQA2 secure. Furthermore, the above construction is* $(\mathcal{U}_1, 1)$*-forward private, where* $\mathcal{U}_1$ *is defined in Table 1.*

References

1. Bösch, C., Hartel, P., Jonker, W., Peter, A.: A survey of provably secure searchable encryption. ACM Comput. Surv. **47**(2), 18:1–18:51 (2014)
2. Bost, R.: $\sum o\varphi o\varsigma$: forward secure searchable encryption. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016, pp. 1143–1154 (2016)
3. Cash, D., Tessaro, S.: The locality of searchable symmetric encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 351–368. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_20

4. Chen, Y.-C., Chow, S.S.M., Chung, K.-M., Lai, R.W.F., Lin, W.-K., Zhou, H.-S.: Cryptography for parallel RAM from indistinguishability obfuscation. In: Sudan, M. (ed.) ITCS 2016, Cambridge, MA, USA, 14–16 January 2016, pp. 179–190. ACM (2016)
5. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Juels, A., Wright, R.N., Sabrina De Capitani di Vimercati, (eds.) ACM CCS 2006, Alexandria, Virginia, USA, 30 October–3 November 2006, pp. 79–88. ACM Press (2006)
6. Curtmola, R., Garay, J.A., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: Improved definitions and efficient constructions. J. Comput. Secur. **19**(5), 895–934 (2011)
7. Garg, S., Mohassel, P., Papamanthou, C.: TWORAM: efficient oblivious RAM in two rounds with applications to searchable encryption. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 563–592. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53015-3_20
8. Hahn, F., Kerschbaum, F.: Searchable encryption with secure and efficient updates. In: Ahn, G.-J., Yung, M., Li, N. (eds.) ACM CCS 2014, Scottsdale, AZ, USA, 3–7 November 2014, pp. 310–320. ACM Press (2014)
9. Islam, S.M., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In: NDSS 2012, San Diego, CA, USA, 5–8 February 2012. The Internet Society (2012)
10. Kamara, S., Papamanthou, C.: Parallel and dynamic searchable symmetric encryption. In: Sadeghi, A.-R. (ed.) FC 2013. LNCS, vol. 7859, pp. 258–274. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39884-1_22
11. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012, Raleigh, NC, USA, 16–18 October 2012, pp. 965–976. ACM Press (2012)
12. Lai, R.W.F., Chow, S.S.M.: Structured encryption with non-interactive updates and parallel traversal. In: 35th IEEE International Conference on Distributed Computing Systems, ICDCS 2015, Columbus, OH, USA, 29 June–2 July 2015, pp. 776–777 (2015)
13. Lai, R.W.F., Chow, S.S.M.: Parallel and dynamic structured encryption. In: SECURECOMM 2016 (2016, to appear)
14. Liu, C., Zhu, L., Wang, M., Tan, Y.: Search pattern leakage in searchable encryption: attacks and new construction. Inf. Sci. **265**, 176–188 (2014)
15. Rizomiliotis, P., Gritzalis, S.: ORAM based forward privacy preserving dynamic searchable symmetric encryption schemes. In: Proceedings of the 2015 ACM Workshop on Cloud Computing Security Workshop, CCSW 2015, Denver, Colorado, USA, 16 October 2015, pp. 65–76 (2015)
16. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, Oakland, CA, USA, pp. 44–55. IEEE Computer Society Press, May 2000
17. Stefanov, E., Papamanthou, C., Shi, E.: Practical dynamic searchable encryption with small leakage. In: NDSS 2014, San Diego, CA, USA, 23–26 February 2014. The Internet Society (2014)
18. Zhang, Y., Katz, J., Papamanthou, C.: All your queries are belong to us: the power of file-injection attacks on searchable encryption. In: 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, 10–12 August 2016, pp. 707–720 (2016)

Bounds in Various Generalized Settings of the Discrete Logarithm Problem

Jason H.M. Ying(✉) and Noboru Kunihiro

The University of Tokyo, Tokyo, Japan
jason_ying@it.k.u-tokyo.ac.jp

Abstract. This paper examines the generic hardness of the generalized multiple discrete logarithm problem, where the solver has to solve k out of n instances for various settings of the discrete logarithm problem. For generic k and n, we introduce two techniques to establish the lower bounds for this computational complexity. One method can be shown to achieve asymptotically tight bounds for small inputs in the classical setting. The other method achieves bounds for larger inputs as well as being able to adapt for applications in other discrete logarithm settings. In the latter, we obtain the generalized lower bounds by applying partitions of n and furthermore show that our chosen method of partition achieves the best bounds. This work can be regarded as a generalization and extension on the hardness of the multiple discrete logarithm problem analyzed by Yun (EUROCRYPT '15). Some explicit bounds for various n with respect to k are also computed.

Keywords: Discrete logarithm · Generalized multiple discrete logarithm · Chebyshev's inequality · Optimization · Gaussian elimination

1 Introduction

Discrete logarithms arise in many aspects of cryptography. The hardness of the discrete logarithm problem is central in many cryptographic schemes; for instance in signatures, key exchange protocols and encryption schemes. In fact, many variants of the discrete logarithm problem have evolved over the years. Some of these include the Bilinear Diffie-Hellman Exponent Problem [6], the Bilinear Diffie-Hellman Inversion Problem [4], the Weak Diffie-Hellman Problem [14] and the Strong Diffie-Hellman Problem [5]. We first describe the classical discrete logarithm problem and its generalizations. Formal descriptions of the statements are as follow.

Let G be a cyclic group such that $|G| = p$ where p a prime and denote g to be a generator of G so that $G = \langle g \rangle$.

The discrete logarithm problem (DLP) is defined as follows: Given G, p and any h selected uniformly at random from G, find $x \in \mathbb{Z}_p$ satisfying $g^x = h$.

The k-Multiple Discrete Logarithm (k-MDL) is defined as follows:

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 498–517, 2017.
DOI: 10.1007/978-3-319-61204-1_25

Definition 1 (MDL). *Given G, p and k elements $h_1, h_2, \ldots, h_k$ selected uniformly at random from G, find non-negative integers $x_1, x_2, \ldots, x_k$ satisfying $g^{x_i} = h_i \; \forall \; 1 \leq i \leq k$.*

In particular when $k = 1$, the 1-MDL is equivalent to DLP.

For $k \leq n$, we define the (k, n)-Generalized Multiple Discrete Logarithm ((k, n)-GMDL) as follows:

Definition 2 (GMDL). *Given G, p and n elements $h_1, h_2, \ldots, h_n$ selected uniformly at random from G, find k pairs (i, x_i) satisfying $g^{x_i} = h_i$ where $i \in S$ and where S is a k-subset of $\{1, \ldots, n\}$.*

As the definition suggests, (k, n)-GMDL can be viewed as a generalization of k-MDL. In particular when $n = k$, the (k, k)-GMDL is equivalent to the k-MDL.

Cryptographic constructions based on DLP are applied extensively. For instance, an early application of the DLP in cryptography came in the form of the Diffie-Hellman key exchange protocol [8] for which the security is dependent on the hardness of the DLP. Among some of the others include the ElGamal encryption and signature schemes [9] as well as Schnorr's signature scheme and identification protocol [16]. The multiple discrete logarithm problem mainly arises from elliptic curve cryptography. NIST recommended a small set of fixed (or standard) curves for use in cryptographic schemes [1] to eliminate the computational cost of generating random secure elliptic curves. The implications for the security of standard elliptic curves over random elliptic curves were analysed based on the efficiency of solving multiple discrete logarithm problems [11].

In a generic group, no special properties which are exhibited by any specific groups or their elements are assumed. Algorithms for a generic group are termed as generic algorithms. There are a number of results pertaining to generic algorithms for DLP and k-MDL. Shoup showed that any generic algorithm for solving the DLP must perform $\Omega(\sqrt{p})$ group operations [17]. There are a few methods for computing discrete logarithm in approximately $\sqrt{p}$ operations. For example, Shanks Baby-Step-Giant-Step method computes the DLP in $\tilde{O}(\sqrt{p})$ operations. One other method is the Pollard's Rho Algorithm which can be achieved in $O(\sqrt{p})$ operations [15]. Since then, further practical improvements to the Pollard's Rho Algorithm have been proposed in [3,7,18] but the computational complexity remains the same. There exist index calculus methods which solve the DLP in subexponential time. However, such index calculus methods are not relevant in our context since they are not applicable for a generic group.

An extension of Pollard's Rho algorithm was proposed in [13] which solves k-MDL in $O(\sqrt{kp})$ group operations if $k \leq O(p^{1/4})$. It was subsequently shown in [10] that $O(\sqrt{kp})$ can in fact be achieved without the imposed condition on k. The former's method of finding discrete logarithm is sequential in the sense that they are found one after another. However in the latter's method, all the discrete logarithms can only be obtained towards the end. Finally, it was presented in [19] that any generic algorithm solving k-MDL must require at least $\Omega(\sqrt{kp})$ group operations if $k = o(p)$.

Our Contributions. In the context of our work, suppose an adversary has knowledge or access to many instances of the discrete logarithm problem either from a generic underlying algebraic group or from a standard curve recommended by NIST. Our work investigates how difficult it is for such an adversary to solve subcollections of those instances. One of our result outcomes in this work shows that an adversary gaining access to additional instances of the DLP provides no advantage in solving some subcollection of them when k is small and for corresponding small classes of n. Our techniques are also applicable to other standard non-NIST based curves. For instance, the results in this work are relevant to Curve25519 [2] which has garnered considerable interest in recent years. Furthermore, we also establish formal lower bounds for the generic hardness of solving the GMDL problem for larger k values. As a corollary, these results provide the lower bounds of solving the GMDL problem for the full possible range of inputs k. Part of this work can be viewed as a generalization of the results in [19].

More specifically, we introduce two techniques to solve such generalized multiple discrete logarithm problems. The first we refer to as the matrix method which is also shown to achieve asymptotically tight bounds when the inputs are small. From this, we obtain the result that the GMDL problem is as hard as the MDL problem for $k = o\left(\frac{p^{1/3}}{\log^{2/3} p}\right)$ and $kn^2 = o\left(\frac{p}{\log^2 p}\right)$. This strictly improves the result of [13] where the equivalence is achieved for a smaller range of inputs satisfying $k = o(p^{\frac{1}{4}})$ and $k^2n^2 = o(p)$. The second technique is referred as the block method which can be applied for larger inputs. We also show that the block partitioning in this method is optimized. Moreover, when n is relatively small with respect to k, the bounds that are obtained in this way are also asymptotically tight. Furthermore, we demonstrate that the block method can be adapted and applied to generalized versions of other discrete logarithm settings introduced in [12] to also obtain generic hardness bounds for such problems. For instance, part of this work also shows that solving one out of n instances of the Discrete Logarithm Problem with Auxiliary Inputs is as hard as solving a single given instance when n is not too large. In addition, we also explain why the matrix method cannot be extended to solve these problems.

2 Preliminaries

For generic groups of large prime order p, denote T_k, $T_{k,n}$ to be the expected workload (in group operations) of an optimal algorithm solving the k-MDL problem and (k, n)-GMDL problem respectively.

Lemma 1 and Corollary 1 for a special case $k = 1$ are attributed to [13].

Lemma 1. $T_1 \leq T_{1,n} + 2n \log_2 p$

Proof. Given an arbitrary $h \in G = \langle g \rangle$, obtaining x such that $g^x = h$ can be achieved in time T_1. For all i, $1 \leq i \leq n$, select integers r_i uniformly at random from the set $\{0, \ldots, p-1\}$ and define $h_i := g^{r_i}h = g^{x+r_i}$. All the h_i are random since all the r_i and h are random. Apply a generic algorithm with inputs of $(h_1, h_2, \ldots, h_n)$ that solves the $(1, n)$-GMDL problem in time $T_{1,n}$. The resulting algorithm outputs (j, y) such that $h_j = g^y$, $1 \leq j \leq n$. Therefore, $x \equiv y - r_j \bmod p$, thus solving the 1-MDL problem within $T_{1,n} + 2n \log_2 p$ group multiplications. □

Corollary 1. *For all* $n = o\left(\frac{\sqrt{p}}{\log p}\right)$, $T_{1,n} = \Omega(\sqrt{p})$.

Proof. Since $T_1 = \Omega(\sqrt{p})$ [17], when $n = o\left(\frac{\sqrt{p}}{\log p}\right)$, it follows directly from Lemma 1 that $T_{1,n} = \Omega(\sqrt{p})$. □

It was also obtained in [13] that the GMDL problem is as hard as the MDL problem if $kn \ll \sqrt{p}$. Since $k \leq n$, this equivalence is valid for $k = o(p^{\frac{1}{4}})$ and $k^2 n^2 = o(p)$.

3 Generalized Bounds of $T_{k,n}$ for Small k

The first method we introduce is to obtain an improved lower bound of $T_{k,n}$ for small k. We refer to this as the Matrix technique.

We seek to obtain an upper bound of T_k based on $T_{k,n}$. Given $g^{x_i} = h_i$ $\forall$ $1 \leq i \leq k$, T_k represents the time to solve all such x_i. For all $1 \leq i \leq n$, denote y_i by the following[1]:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{k-1} \\ 1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \ldots & \alpha_n^{k-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix}$$

Next, multiply each g^{y_i} with a corresponding random element g^{r_i} where $0 \leq r_i \leq p-1$. By considering these randomized $g^{y_i + r_i}$ as inputs to a (k, n)-GMDL solver, this solver outputs solutions to k out of n of such discrete logarithms. These solutions are of the form $y_i + r_i$. As such, a total of k values of y_i can be obtained by simply subtracting from their corresponding r_i. We claim that any k collections of y_i is sufficient to recover all of $x_1, x_2, \ldots, x_k$. Indeed, it suffices to show that any k-by-k submatrices of

$$V = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{k-1} \\ 1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \ldots & \alpha_n^{k-1} \end{pmatrix}$$

has non zero determinant. This can be satisfied by simply letting $\alpha_i = i$ since V is a Vandermonde matrix.

[1] This representation of y_i came about from a suggestion by Phong Q. Nguyen.

In this case, recovering $x_1, x_2, \ldots, x_k$ from k number of y_i requires solving a k-by-k system of linear equations. This can be achieved in $O(k^3)$ arithmetic operations using Gaussian elimination. Crucially, this does not involve any group operations. On the other hand, group operations are incurred from the computations of all $g^{y_i} = g^{x_1+\alpha_i x_2+\cdots+\alpha_i^{k-1}x_k}$. Since $\alpha_i = i$, this process requires the computations of each $(g^{x_j})^{i^{j-1}}$ $\forall\ 1 \leq i \leq n,\ 1 \leq j \leq k$. Denote $a_{i,j} = (g^{x_j})^{i^{j-1}}$. By noting that $a_{i+1,j} = a_{i,j}^{(\frac{i+1}{i})^{j-1}}$, it can be concluded that computing $a_{i+1,j}$ given $a_{i,j}$ requires at most $2(j-1)\log_2 \frac{i+1}{i}$ group multiplications. Moreover, $a_{1,j} = g^{x_j}$ is already known. Hence the total number of groups multiplications required to compute all $(g^{x_j})^{i^{j-1}}$ is at most

$$\sum_{j=1}^{k}\sum_{i=1}^{n-1} 2(j-1)\log_2\left(\frac{i+1}{i}\right) = k(k-1)\log_2 n.$$

Furthermore, each addition in the exponent of $g^{x_1+\alpha_i x_2+\cdots+\alpha_i^{k-1}x_k+r_i}$ constitutes a group multiplication. Therefore, kn group multiplications are necessary in this step. Thus, the total number of group multiplications required to compute all of $g^{y_i+r_i}$ is at most $kn + k(k-1)\log_2 n$. Since $k \leq n < p$, the above expression can be bounded from above by $2kn\log_2 p$ and so it follows that

$$T_k \leq T_{k,n} + 2kn\log_2 p$$

Since $T_k = \Omega(\sqrt{kp})$ [19], $T_{k,n}$ is asymptotically as large as T_k if $nk\log p \ll \sqrt{kp}$. Hence, $T_{k,n} = \Omega(\sqrt{kp})$ if $k = o\left(\frac{p^{1/3}}{\log^{2/3} p}\right)$ and $n\sqrt{k} = o\left(\frac{\sqrt{p}}{\log p}\right)$. Moreover, this bound is asymptotically tight since there exists an algorithm which solves k-MDL in $O(\sqrt{kp})$.

4 Generalized Bounds of $T_{k,n}$ for Larger k

The matrix technique has been shown to provide asymptotically tight bounds required to solve k out of n of the multiple discrete logarithm in the classical setting when the inputs are small. One main limitation of this technique is that it is only applicable to the classical DLP and cannot be extended for other variants or other settings of the DLP. This will be shown in further details in the subsequent sections. Moreover, in light of the fact that the bound of T_k can be achieved for large inputs extending to $o(p)$, the matrix method is not sufficient to obtain analogous bounds for larger such k values. In this section, we address these issues by introducing the block method to evaluate lower bounds of $T_{k,n}$ for general k, including large k values. Moreover, the block technique can also be applied to other variants or other settings in the MDLP. This will be also described in the later sections.

Proposition 1. *Suppose that $n \geq k^2$. Then,*

$$T_k \leq kT_{k,n} + 2nH_k\log_2 p$$

where H_k denotes the k^{th} harmonic number, $H_k = \sum_{i=1}^{k}\frac{1}{i}$.

Proof. Given arbitraries $h_1, h_2, \ldots, h_k \in G = \langle g \rangle$, obtaining $x_1, x_2, \ldots, x_k$ such that $g^{x_i} = h_i \ \forall\ 1 \leq i \leq k$ can be achieved in time T_k. Consider n elements partitioned into k blocks each of size approximately s_k, where $s_k = \frac{n}{k}$. Each block is labelled i where i ranges from 1 to k. For each i, $1 \leq i \leq k$, select about s_k integers $r_{i,j}$ uniformly at random from $\mathbb{Z}_p$ and define $h_{i,j} := g^{r_{i,j}} h_i = g^{x_i + r_{i,j}}$. This generates n of $h_{i,j}$ which when applied to a generic algorithm for the (k, n)-GMDL problem, outputs k pseudo solutions. Computing each $h_{i,j}$ requires at most $2 \log_2 p$ group multiplications. Since each block is about size $\frac{n}{k} \geq k$, these k pseudo solutions might be derived from the same block. In which case, the algorithm outputs k of $((i', j),\ y_{i',j})$ such that $h_{i',j} = g^{y_{i',j}}$ for some i'. As a result, one can obtain $x_{i'} \equiv y_{i',j} - r_{i',j} \bmod p$ but derive no other information of other values of x_i. This invokes at most $T_{k,n} + 2n \log_2 p$ group operations. Figure 1 illustrates an overview of the first phase.

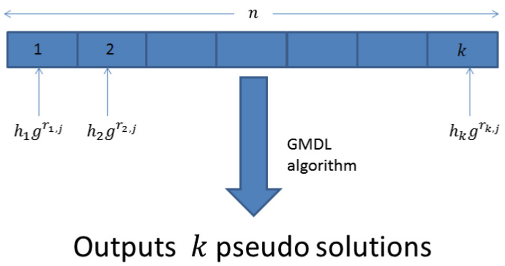


Fig. 1. Overview of the first phase

The second phase proceeds as follows. Since 1 out of k discrete logarithms has been obtained, discard the block for which that determined discrete logarithm is contained previously. For each of $i \in \{1, \ldots, k\} \setminus \{i'\}$, select about $\frac{s_k}{k-1}$ integers $r_{i,j}$ uniformly at random from $\mathbb{Z}_p$ and define $h_{i,j} := g^{r_{i,j}} h_i = g^{x_i + r_{i,j}}$. Incorporate these new values of $h_{i,j}$ into the remaining $k-1$ blocks. Hence, each of the remaining $k-1$ unsolved discrete logarithms are contained separately in $k-1$ blocks each of size approximately $\frac{n}{k-1}$. This generates n of $h_{i,j}$ which when applied to a generic algorithm for the (k, n)-GMDL problem, outputs k pseudo solutions. Since each block is about size $\frac{n}{k-1} \geq k$, these k pseudo solutions might once again be derived from the same block. In which case, the algorithm outputs k of $((i'', j), y_{i'',j})$ such that $h_{i'',j} = g^{y_{i'',j}}$ for some i''. As a result, one can obtain $x_{i''} \equiv y_{i'',j} - r_{i'',j} \bmod p$ but derive no other information for the other remaining values of x_i. This second phase incurs at most $T_{k,n} + 2s_k \log_2 p$ group operations.

The third phase executes in a similar manner to the second phase as follows. For each of $i \in \{1, \ldots, k\} \setminus \{i', i''\}$, select about $\frac{s_{k-1}}{k-2}$ integers $r_{i,j}$ uniformly at random from $\mathbb{Z}_p$ and define $h_{i,j} := g^{r_{i,j}} h_i = g^{x_i + r_{i,j}}$. Incorporate these new values of $h_{i,j}$ into the remaining $k-2$ blocks. Hence, each of the remaining $k-2$ unsolved discrete logarithms are contained separately in $k-2$ blocks each of size approximately $\frac{n}{k-2}$. This generates n of $h_{i,j}$ which when applied to a generic algorithm for the (k,n)-GMDL problem, outputs k pseudo solutions. Since each block is about size $\frac{n}{k-2} \geq k$, these k pseudo solutions might be derived from the same block. In which case, the algorithm outputs k of $((i^{(3)}, j),\ y_{i^{(3)},j})$ such that $h_{i^{(3)},j} = g^{y_{i^{(3)},j}}$ for some $i^{(3)}$. As a result, one can obtain $x_{i^{(3)}} \equiv y_{i^{(3)},j} - r_{i^{(3)},j} \mod p$ but derive no other information for the other remaining values of x_i. This second phase incurs at most $T_{k,n} + 2s_{k-1}\log_2 p$ group operations.

During each phase of the process, a generic algorithm for the (k,n)-GMDL problem can never guarantee outputs deriving from different blocks since $n \geq k^2$ implies $\frac{n}{k-i+1} \geq k$, $\forall 1 \leq i \leq k$. In general for $i \geq 2$, the maximum number of group operations required in the i^{th} phase is given by $T_{k,n} + 2s_{k+2-i}\log_2 p$. The process terminates when all k discrete logarithms have been obtained. Since each phase outputs exactly 1 out of k discrete logarithms, all k discrete logarithms can be determined after k phases. Therefore, the following inequality can be obtained.

$$T_k \leq T_{k,n} + 2n\log_2 p + \sum_{i=2}^{k}(T_{k,n} + 2s_{k+2-i}\log_2 p)$$

Since $s_k = \frac{n}{k}$,

$$\sum_{i=2}^{k}(T_{k,n} + 2s_{k+2-i}\log_2 p) = (k-1)T_{k,n} + (2\log_2 p)\sum_{i=2}^{k} s_i = (k-1)T_{k,n} + (2\log_2 p)\sum_{i=2}^{k}\frac{n}{i}.$$

Hence, it follows that

$$T_k \leq kT_{k,n} + 2n(1 + \sum_{i=2}^{k}\frac{1}{i})\log_2 p = kT_{k,n} + 2nH_k\log_2 p.$$

This completes the proof. □

Remark 1. When $k = 1$, Proposition 1 corresponds to Lemma 1.

By regarding $k = k(p)$ as a function of p such that $\lim_{p\to+\infty} k(p) = +\infty$, the asymptotic bounds of $T_{k,n}$ can be obtained.

Theorem 1. *Suppose $k^3\log^2 k = o\left(\frac{p}{\log^2 p}\right)$, then $T_{k,n} = \Omega\left(\sqrt{\frac{p}{k}}\right)$ for all n satisfying $n = k^2 + \Omega(1)$ and $n = o\left(\frac{\sqrt{kp}}{(\log k)(\log p)}\right)$.*

Proof. Clearly $k^3\log^2 k = o\left(\frac{p}{\log^2 p}\right)$ implies that $k = o(p)$. Hence from [19], $T_k = \Omega(\sqrt{kp})$. It follows from Proposition 1 that if $nH_k\log_2 p \ll T_k$, then $T_{k,n} = \Omega(\frac{1}{k}\sqrt{kp}) = \Omega\left(\sqrt{\frac{p}{k}}\right)$. Moreover, since

$$\lim_{k\to+\infty}(H_k - \log k) = \lim_{k\to+\infty}[(\sum_{i=1}^{k}\frac{1}{i}) - \log k] = \gamma$$

where γ is the Euler-Mascheroni constant, the condition $nH_k \log_2 p \ll T_k$ implies that $n = o\left(\frac{\sqrt{kp}}{(\log k)(\log p)}\right)$. The lower bound $n = k^2 + \Omega(1)$ is obtained by noting that n has to be of size at least k^2 from the condition of Proposition 1. Finally, since the lower bound for n cannot be asymptotically greater than its upper bound, $k(p)$ has to satisfy $k^3 \log^2 k = o\left(\frac{p}{\log^2 p}\right)$. This completes the proof of Theorem 1. □

Remark 2. Although Theorem 1 holds for a wide asymptotic range of n as given, the $T_{k,n}$ bound becomes sharper as n approaches $\frac{\sqrt{kp}}{(\log k)(\log p)}$. In essence, Theorem 1 does not yield interesting bounds but is a prelude to the more essential Theorem 2 which requires Proposition 1 and is hence included.

Proposition 2. *Suppose that $k < n < k^2$. Then,*

$$T_k \leq (r + \frac{n}{k})T_{k,n} + 2rk \log_2 p + 2nH_{\lceil \frac{n}{k} \rceil} \log_2 p$$

where $r = \left\lceil \frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})} \right\rceil$.

Proof. The proof comprises two main phases. The former consists of the initializing phase followed by subsequent subphases. It utilizes the extended pigeon hole principle to obtain more than one solution during each of the initial subphases. The latter phase takes place after some point where the number of remaining unknown discrete logarithms is small enough such that each subphase can only recover one discrete logarithm. After this point, the method of determining all other discrete logarithms essentially mirrors that of the method described in the proof of Proposition 1. The formal proof and details are given as follows.

The initializing phase proceeds as follows. Given arbitraries $h_1, h_2, \ldots, h_k \in G = \langle g \rangle$, obtaining $x_1, x_2, \ldots, x_k$ such that $g^{x_i} = h_i \ \forall \ 1 \leq i \leq k$ can be achieved in time T_k. Consider n elements partitioned into k blocks each of size approximately s_k, where $s_k = \frac{n}{k}$. Each block is labelled i where i ranges from 1 to k. For each i, $1 \leq i \leq k$, select about s_k integers $r_{i,j}$ uniformly at random from $\mathbb{Z}_p$ and define $h_{i,j} := g^{r_{i,j}} h_i = g^{x_i + r_{i,j}}$. This generates n of $h_{i,j}$ which when applied to a generic algorithm for the (k, n)-GMDL problem, outputs k pseudo solutions. Computing each $h_{i,j}$ requires at most $2 \log_2 p$ group multiplications. Since each block is about size $\frac{n}{k} < k$, by the extended pigeon hole principle, at least $\frac{k^2}{n}$ out of these k solutions must be derived from distinct blocks. In other words, at least $\frac{k^2}{n}$ correspond to distinct i values and as a result, $\frac{k^2}{n}$ discrete logarithms out of k discrete logarithms can be obtained during this initializing phase. This invokes at most $T_{k,n} + 2n \log_2 p$ group operations.

The first subphase proceeds as follows. Since $\frac{k^2}{n}$ out of k discrete logarithms have been obtained, discard all the blocks for which those determined discrete logarithms are contained previously. Thus, about $k - \frac{k^2}{n} = \frac{k(n-k)}{n}$ blocks remain, each of size approximately $\frac{n}{k}$. For each of the remaining blocks i, select about k integers $r_{i,j}$ uniformly distributed across each i and uniformly at random from

$\mathbb{Z}_p$. Define $h_{i,j} := g^{r_{i,j}} h_i = g^{x_i + r_{i,j}}$. Incorporate these new values of $h_{i,j}$ into the remaining $\frac{k(n-k)}{n}$ blocks. Hence, each of the remaining $\frac{k(n-k)}{n}$ unsolved discrete logarithms are contained separately in $\frac{k(n-k)}{n}$ blocks each of size approximately $\frac{n^2}{k(n-k)}$. This generates n of $h_{i,j}$ which when applied to a generic algorithm for the (k,n)-GMDL problem, outputs k pseudo solutions. This incurs a maximum of $T_{k,n} + 2k \log_2 p$ group operations.

If $k \leq \frac{n^2}{k(n-k)}$, these k pseudo solutions might be derived from the same block and hence phase two begins in which the method described in Proposition 1 can then be applied on this new set of blocks.

If $k > \frac{n^2}{k(n-k)}$, the extended pigeon hole principle ensures that at least $\frac{k^2(n-k)}{n^2}$ of the k pseudo solutions correspond to distinct i values and a result, about $\frac{k^2(n-k)}{n^2}$ discrete logarithms can be obtained in the first subphase.

Subsequent subphases are similar to the first subphase. In general for the r^{th} subphase, since about $\frac{k^2(n-k)^{r-1}}{n^r}$ solutions will have been obtained in the $(r-1)^{th}$ subphase, discard all the blocks for which those determined discrete logarithms are contained previously. By a simple induction, it can be shown that the number of remaining blocks in the r^{th} subphase is about $\frac{k(n-k)^r}{n^r}$. The induction proceeds as follows. The base case has already been verified in the first subphase. Suppose the result holds for $r = m-1$ for some $m \geq 2$. By the inductive hypothesis, the number of blocks remaining in the $(m-1)^{th}$ subphase is about $\frac{k(n-k)^{m-1}}{n^{m-1}}$. During the m^{th} subphase, since about $\frac{k^2(n-k)^{m-1}}{n^m}$ solutions have already been obtained previously and are thus discarded, the number of remaining blocks is given by

$$\frac{k(n-k)^{m-1}}{n^{m-1}} - \frac{k^2(n-k)^{m-1}}{n^m} = \frac{k(n-k)^m}{n^m}.$$

This completes the induction. For each of the remaining blocks i, select about k integers $r_{i,j}$ uniformly distributed across each i and uniformly at random from $\mathbb{Z}_p$. Define $h_{i,j} := g^{r_{i,j}} h_i = g^{x_i + r_{i,j}}$. Incorporate these new values of $h_{i,j}$ into the remaining $\frac{k(n-k)^r}{n^r}$ blocks. Hence, each of the remaining $\frac{k(n-k)^r}{n^r}$ unsolved discrete logarithms are contained separately in $\frac{k(n-k)^r}{n^r}$ blocks each of size approximately $\frac{n^{r+1}}{k(n-k)^r}$. This generates n of $h_{i,j}$ which when applied to a generic algorithm for the (k,n)-GMDL problem, outputs k pseudo solutions. Each of the r^{th} subphase requires at most $T_{k,n} + 2k \log_2 p$ group operations.

When $k \leq \frac{n^{r+1}}{k(n-k)^r}$, the k outputs can only guarantee one solution. Hence, as soon as r satisfies the above inequality, the first main phase terminates at the end of the r^{th} subphase and the second main phase commences. That is

$$k \leq \frac{n^{r+1}}{k(n-k)^r} \implies r = \left\lceil \frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})} \right\rceil.$$

At the beginning of second main phase, there are a total of about $\frac{k(n-k)^r}{n^r} - 1$ unresolved discrete logarithms. The rest of the procedure follows starting from the second phase of Proposition 1 until the end. Hence, it can immediately be derived from the proof of Proposition 1 that the number of group operations required to solve them all is at most $2[(\frac{k(n-k)^r}{n^r}-1)T_{k,n} + nH_{\lceil \frac{k(n-k)^r}{n^r} \rceil} - n] \log_2 p$. Since

$$k \leq \frac{n^{r+1}}{k(n-k)^r} \implies \frac{k(n-k)^r}{n^r} \leq \frac{n}{k},$$

the maximum number of group operations required in the second main phase is given by $(\frac{n}{k}-1)T_{k,n} + 2(nH_{\lceil \frac{n}{k} \rceil} - n) \log_2 p$. The number of group operations required during the first main phase is the sum of the number required for the initializing phase and the number required for all the subphases. Therefore, the maximum number of group operations required in the first main phase is given by

$$T_{k,n} + 2n \log_2 p + r(T_{k,n} + 2k \log_2 p) = (r+1)T_{k,n} + 2n \log_2 p + 2rk \log_2 p.$$

Thus the maximum number of group operations required to execute both the first main phase and the second main phase is given by

$$(r+1)T_{k,n} + 2(n+rk) \log_2 p + (\frac{n}{k} - 1)T_{k,n} + 2(nH_{\lceil \frac{n}{k} \rceil} - n) \log_2 p.$$

Hence, $T_k \leq (r + \frac{n}{k})T_{k,n} + 2rk \log_2 p + 2nH_{\lceil \frac{n}{k} \rceil} \log_2 p$. □

Remark 3. One other approach is to replace the (k,n)-GMDL solver with the $(1,n)$-GMDL solver during the second main phase. Both yield identical asymptotic results given in Sect. 7 as the computational bottleneck arises from the first main phase.

By regarding $k = k(p)$ as a function of p such that $\lim_{p \to +\infty} k(p) = +\infty$, the asymptotic bounds of $T_{k,n}$ can be obtained.

Theorem 2. *Suppose k, n satisfy the following conditions:*

1. $k = o(p)$ *2.* $n = k^2 - \Omega(1)$

3. $\frac{n}{\sqrt{k}} \log(\frac{n}{k}) = o\left(\frac{\sqrt{p}}{\log p}\right)$ *4.* $\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})} \sqrt{k} = o\left(\frac{\sqrt{p}}{\log p}\right)$

Then,

$$T_{k,n} = \Omega\left(\frac{\sqrt{k}}{\frac{n}{k}+r}\sqrt{p}\right)$$

where $r = r(k,n) \geq 1$ is any function of k and n satisfying $r(k,n) = \Omega\left(\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})}\right)$.

Proof. Condition 1 is necessary to utilize results in [19] for a lower bound of T_k. Condition 2 is required in order to apply the result of Proposition 2. Conditions 3 and 4 can be obtained by requiring $nH_{\lceil \frac{n}{k} \rceil} \log p \ll T_k$ and $rk \log p \ll T_k$ respectively and noting that $T_k = \Omega(\sqrt{kp})$. Hence from Proposition 2 and under these conditions, $T_{k,n} = \Omega\left(\frac{\sqrt{k}}{\frac{n}{k}+r}\sqrt{p}\right)$. □

It should be mentioned that $r(k,n)$ can be taken to be any function satisfying $r(k,n) = \Omega\left(\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})}\right)$. However, it is clear that $T_{k,n}$ achieves sharper bounds for asymptotically smaller choices of r. One other point of note is that when $n = k$ where $k = o(p)$, all the 4 conditions are satisfied and $r(k,n)$ can be taken to be 1. In this case, $T_k = T_{k,k} = \Omega(\sqrt{kp})$. which indeed corresponds to the bound obtained in [19].

5 Optimizing the Partition of n

From the methods described in the proofs of Propositions 1 and 2, n is partitioned into blocks of approximately equal size at each phase. There are many ways to perform such partitions of n. The running time of each phase is partially determined by the number of uniformly randomly chosen $r_{i,j}$, which invariably depends on the partition of n. In this section, we show that the method of partition described in the proofs of the earlier Propositions minimizes the expected number of chosen $r_{i,j}$ required and hence results in the fastest running time among all other possible partitions. We first consider the case where a (k,n)-GMDL solver output solutions derived from the same block so only one discrete logarithm can be determined at each phase. We follow this up by considering general scenarios where a (k,n)-GMDL solver outputs solutions derived from multiple blocks.

5.1 Pseudo Solutions Deriving from the Same Block

Denote s_i to be the size of block i, $k \leq s_i$, $1 \leq i \leq k$. So that $\sum_{i=1}^{k} s_i = n$. Let $p_{i,k}$ be the conditional probability that the k output solutions derive from block i given that the k output solutions derive from the same block. Then, $p_{i,k}$ can be expressed by the following.

$$p_{i,k} = \frac{\binom{s_i}{k}}{\sum_{j=1}^{k} \binom{s_j}{k}}$$

Suppose a solution is derived from block i. Upon discarding block i, s_i of $r_{i,j}$ have to be randomly chosen to fill the remaining blocks so that they sum back up to n. Let $E_k^{(1)}$ be the expected number of randomly chosen $r_{i,j}$ required. Then, $E_k^{(1)}$ can be expressed by the following.

$$E_k^{(1)} = \sum_{i=1}^{k} p_{i,k} s_i = \frac{\sum_{i=1}^{k} \binom{s_i}{k} s_i}{\sum_{i=1}^{k} \binom{s_i}{k}}$$

Our objective is therefore to minimize $E_k^{(1)}$ given $\sum_{i=1}^k s_i = n$. We expand the admissible values of s_i to the set of positive real numbers so that $s_i \in \mathbb{R}^+$. In this way, $\binom{s_i}{k}$ is defined as $\binom{s_i}{k} = \frac{s_i(s_i-1)\ldots(s_i-k+1)}{k!}$. We prove the following result.

Theorem 3. *Given that* $\sum_{i=1}^k s_i = n$,

$$\frac{\sum_{i=1}^k \binom{s_i}{k} s_i}{\sum_{i=1}^k \binom{s_i}{k}} \geq \frac{n}{k}.$$

Proof. Without loss of generality, assume that $s_1 \leq s_2 \leq \cdots \leq s_k$. Thus, $\binom{s_1}{k} \leq \binom{s_2}{k} \leq \cdots \leq \binom{s_i}{k}$. By Chebyshev's sum inequality,

$$k \sum_{i=1}^k \binom{s_i}{k} s_i \geq \left(\sum_{i=1}^k s_i \right) \left(\sum_{i=1}^k \binom{s_i}{k} \right).$$

The result follows by replacing $\sum_{i=1}^k s_i$ with n in the above inequality. □

Hence, $E_k^{(1)} \geq \frac{n}{k}$ and it is straightforward to verify that equality holds if $s_1 = s_2 = \cdots = s_k$. Therefore, the method of partitioning n into blocks of equal sizes at each phase as described in the proof of the Proposition 1 indeed minimizes the running time.

5.2 Pseudo Solutions Deriving from Multiple Blocks

Denote s_i to be the size of block i, $1 \leq i \leq k$. so that $\sum_{i=1}^k s_i = n$. Let $p_{i_1,i_2,\ldots,i_m,k}$ be the conditional probability that the k output solutions derive from blocks $i_1, i_2, \ldots, i_m$ given that the k output solutions derive from m distinct blocks. Then, $p_{i_1,i_2,\ldots,i_m,k}$ satisfies the following.

$$p_{i_1,i_2,\ldots,i_m,k} \sum_{\{i_1\ldots i_m\}\subseteq\{1,\ldots,k\}} \sum_{k_1+\cdots+k_m=k} \binom{s_{i_1}}{k_1} \cdots \binom{s_{i_m}}{k_m} = \sum_{k_1+\cdots+k_m=k} \binom{s_{i_1}}{k_1} \cdots \binom{s_{i_m}}{k_m}$$

A more concise representation can be expressed as follows.

$$p_{i_1,i_2,\ldots,i_m,k} \sum_{\{i_1\ldots i_m\}\subseteq\{1,\ldots,k\}} \sum_{k_1+\cdots+k_m=k} \prod_{t=1}^m \binom{s_{i_t}}{k_t} = \sum_{k_1+\cdots+k_m=k} \prod_{t=1}^m \binom{s_{i_t}}{k_t} \quad (1)$$

We can further simplify the above expression by the following lemma.

Lemma 2.

$$\sum_{k_1+\cdots+k_m=k} \prod_{t=1}^m \binom{s_{i_t}}{k_t} = \binom{s_{i_1} + \cdots + s_{i_m}}{k}$$

Proof. For brevity, denote $s = s_{i_1} + \cdots + s_{i_m}$.

Consider the polynomial $(1+x)^s$. By the binomial theorem, $(1+x)^s = \sum_{r=0}^{s} \binom{s_{i_1}+\cdots+s_{i_m}}{r} x^r$. On the other hand,

$$(1+x)^s = (1+x)^{s_{i_1}} \dots (1+x)^{s_{i_m}} = \prod_{t=1}^{m} \sum_{r_t=0}^{s_{i_t}} \binom{s_{i_t}}{r_t} x^{r_t}.$$

In this instance, the coefficient of x^r is the sum of all products of binomial coefficients of the form $\binom{s_{i_t}}{r_t}$ where the r_t sum to r. Therefore,

$$\prod_{t=1}^{m} \sum_{r_t=0}^{s_{i_t}} \binom{s_{i_t}}{r_t} x^{r_t} = \sum_{r=0}^{s} \sum_{r_1+\dots r_m=r} \prod_{t=1}^{m} \binom{s_{i_t}}{r_t} x^r.$$

Hence,

$$\sum_{r=0}^{s} \binom{s_{i_1}+\cdots+s_{i_m}}{r} x^r = \sum_{r=0}^{s} \sum_{r_1+\dots r_m=r} \prod_{t=1}^{m} \binom{s_{i_t}}{r_t} x^r.$$

The result follows by equating the coefficients of x^r on both sides of the above equation. □

Corollary 2.

$$p_{i_1,i_2,\dots,i_m,k} \sum_{\{i_1\dots i_m\}\subseteq\{1,\dots,k\}} \binom{s_{i_1}+\cdots+s_{i_m}}{k} = \binom{s_{i_1}+\cdots+s_{i_m}}{k}$$

Proof. Follows directly from Eq. (1) and Lemma 2. □

Suppose m solutions are derived from blocks $i_1,\dots,i_m$. Upon discarding blocks $i_1,\dots,i_m$, $s_{i_1}+\cdots+s_{i_m}$ of $r_{i,j}$ have to be randomly chosen to fill the remaining blocks so that they sum back up to n. Denote $E_k^{(m)}$ to be the expected number of randomly chosen $r_{i,j}$ required. Then, a generalized form of $E_k^{(m)}$ can be expressed as follows.

$$E_k^{(m)} = \sum_{\{i_1\dots i_m\}\subseteq\{1,\dots,k\}} p_{i_1,i_2,\dots,i_m,k}(s_{i_1}+\cdots+s_{i_m})$$

From the result of Corollary 2, this implies that $E_k^{(m)}$ satisfies

$$E_k^{(m)} \sum_{\{i_1\dots i_m\}\subseteq\{1,\dots,k\}} \binom{s_{i_1}+\cdots+s_{i_m}}{k} = \sum_{\{i_1\dots i_m\}\subseteq\{1,\dots,k\}} \binom{s_{i_1}+\cdots+s_{i_m}}{k}(s_{i_1}+\cdots+s_{i_m}).$$

Once again, we seek to maximize $E_k^{(m)}$ given $\sum_{i=1}^{k} s_i = n$. As before, we expand the admissible values of s_i to the set of positive real numbers so that $s_i \in \mathbb{R}^+$. In this way, $\binom{s_i}{k}$ is defined as $\binom{s_i}{k} = \frac{s_i(s_i-1)\dots(s_i-k+1)}{k!}$. We prove the following result.

Theorem 4. *Given that* $\sum_{i=1}^{k} s_i = n$,

$$\sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} \binom{s_{i_1} + \cdots + s_{i_m}}{k} (s_{i_1} + \cdots + s_{i_m}) \geq \frac{mn}{k} \sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} \binom{s_{i_1} + \cdots + s_{i_m}}{k}.$$

Proof. By Chebyshev's sum inequality,

$$\binom{k}{m} \sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} \binom{s_{i_1} + \cdots + s_{i_m}}{k} (s_{i_1} + \cdots + s_{i_m})$$

$$\geq \left(\sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} \binom{s_{i_1} + \cdots + s_{i_m}}{k} \right) \left(\sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} (s_{i_1} + \cdots + s_{i_m}) \right).$$

Since $\sum_{i=1}^{k} s_i = n$,

$$\sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} (s_{i_1} + \cdots + s_{i_m}) = \binom{k-1}{m-1} n.$$

Hence, we obtain

$$\binom{k}{m} \sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} \binom{s_{i_1} + \cdots + s_{i_m}}{k} (s_{i_1} + \cdots + s_{i_m})$$

$$\geq \binom{k-1}{m-1} n \sum_{\{i_1 \dots i_m\} \subseteq \{1,\dots,k\}} \binom{s_{i_1} + \cdots + s_{i_m}}{k}.$$

By elementary algebraic operations, it is straightforward to verify that $\frac{\binom{k-1}{m-1}}{\binom{k}{m}} = \frac{m}{k}$ from which the result follows. □

Hence, $E_k^{(m)} \geq \frac{mn}{k}$ and it is straightforward to verify that equality holds if $s_1 = s_2 = \cdots = s_k$. Therefore, the method of partitioning n into blocks of equal sizes at each phase as described in the proof of the Proposition 2 indeed minimizes the running time.

6 Applications in Other MDLP Settings

We demonstrate how the block method can be adapted to obtain bounds in other generalized multiple discrete logarithm settings. We consider applications to the $(e_1, \dots, e_d)$-Multiple Discrete Logarithm Problem with Auxiliary Inputs (MDLPwAI) as well as the $\mathbb{F}_p$-Multiple Discrete Logarithm Problem in the Exponent ($\mathbb{F}_p$-MDLPX). Let $G = \langle g \rangle$ be a cyclic group of large prime order p. Their formal definitions are as follow.

Definition 3 (MDLPwAI). *Given G, g, p, e_i and $g^{{x_i}^{e_1}}, g^{{x_i}^{e_2}}, \dots g^{{x_i}^{e_d}}$ $\forall$ $1 \leq i \leq k$, find non-negative integers x_1, x_2, $\dots$, $x_k \in \mathbb{Z}_p$.*

Definition 4 ($\mathbb{F}_\mathbf{p}$-MDLPX). *Let $\chi \in \mathbb{F}_p$ be an element of multiplicative order N. Given G, g, p, χ and $g^{\chi^{x_i}}$ $\forall$ $1 \leq i \leq k$, find non-negative integers x_1, x_2, $\dots$, $x_k \in \mathbb{Z}_N$.*

The computational complexity of MDLPwAI was analysed in [12]. In the same paper, the authors introduced the $\mathbb{F}_p$-MDLPX and also analysed its complexity.

Here, we define the Generalized Multiple Discrete Logarithm Problem (GMDLPwAI) with Auxiliary Inputs and the Generalized $\mathbb{F}_p$-Multiple Discrete Logarithm Problem in the Exponent ($\mathbb{F}_p$-GMDLPX) to be solving k out of n instances of the MDLPwAI and $\mathbb{F}_p$-MDLPX respectively. We provide the formal definitions below.

Definition 5 (GMDLPwAI). *Given G, g, p, e_i and $g^{{x_i}^{e_1}}, g^{{x_i}^{e_2}}, \dots g^{{x_i}^{e_d}}$ $\forall$ $1 \leq i \leq n$, find k pairs (i, x_i), $x_i \in \mathbb{Z}_p$, where $i \in S$ such that S is a k-subset of $\{1, \dots, n\}$.*

Definition 6 ($\mathbb{F}_\mathbf{p}$-GMDLPX). *Let $\chi \in \mathbb{F}_p$ be an element of multiplicative order N. Given G, g, p, χ and $g^{\chi^{x_i}}$ $\forall$ $1 \leq i \leq n$, find k pairs (i, x_i), $x_i \in \mathbb{Z}_N$, where $i \in S$ such that S is a k-subset of $\{1, \dots, n\}$.*

6.1 Block Based GMDLPwAI

The block method can be adapted to obtain bounds for the GMDLPwAI by randomizing the input elements in the following way. Given $g^{{x_i}^{e_1}}$, select random integers $r_{i,j} \in \mathbb{Z}_p^*$ and compute values of $(g^{{x_i}^{e_1}})^{{r_{i,j}}^{e_1}} = g^{(r_{i,j}x_i)^{e_1}}$ as inputs into the GMDLPwAI solver. For each $r_{i,j}$, reduce $r_{i,j}^{e_1}$ modulo p and then $g^{(r_{i,j}x_i)^{e_1}}$ can be computed within $2\log_2 p$ group operations. Repeat this procedure for all $e_2, \dots, e_d$. We show how the x_i can be recovered. For instance, suppose the solver outputs solution (l, y) corresponding to some particular input $g^{(r_{i,j}x_i)^{e_1}}$. In which case, x_i can thus be obtained by solving $r_{i,j}x_i \equiv y \bmod p$. Such congruence equations are efficiently solvable since $\gcd(r_{i,j}, p) = 1$.

Let T'_k and $T'_{k,n}$ denote the time taken in group operations for an optimal algorithm to solve the MDLPwAI and GMDLPwAI problems respectively.

Suppose $k < n < k^2$. Then by adapting the block technique applied to the GMDL problem before, it can be shown that

$$T'_k \leq (r + \frac{n}{k})T'_{k,n} + 2d(rk + nH_{\lceil \frac{n}{k} \rceil}) \log_2 p$$

where $r = \left\lceil \frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})} \right\rceil$.

It has been conjectured in [12] that $T'_k = \Omega(\sqrt{kp/d})$ for values of $e_i = i$. Assuming this conjecture, we can conclude from our results that for all polynomially bounded inputs with $d = O(p^{1/3-\epsilon})$, $\epsilon > 0$, $T'_{k,n}$ is bounded by

$$T'_{k,n} = \Omega\left(\frac{\sqrt{k}}{\frac{n}{k}+r}\sqrt{\frac{p}{d}}\right)$$

where $r = r(k,n) \geq 1$ is any function of k and n satisfying $r(k,n) = \Omega\left(\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})}\right)$.

When $k = 1$, the above bound is not applicable since $n \geq k^2$ in this situation. Nevertheless, we show how an unconditional bound for $T'_{1,n}$ can still be obtained in this specific case without the assumption of any conjecture. Similar to the generalized case, randomize input elements of the form $g^{(r_{i,j}x_i)^{e_1}}$. One of the x_i can then be computed by solving $r_{i,j}x_i \equiv y \bmod p$ where y is a given known output. The process terminates here since one of the x_i has already been obtained. Hence, we have the following inequality: $T'_1 \leq T'_{1,n} + 2nd\log_2 p$. From the results of [5], $T'_1 = \Omega(\sqrt{p/d})$. It follows that for $n = o\left(\frac{\sqrt{p}}{d^{3/2}\log p}\right)$,

$$T'_{1,n} = \Omega(\sqrt{p/d}).$$

6.2 Matrix Based GMDLPwAI

In this section, our objective is to find values of e_i for which the matrix method can applied to solve the GMDLPwAI. We recall from Sect. 3 that the validity of the method for the k-MDL problem necessitates $g^{x_1+x_2+\cdots+x_k}$ to be efficiently computable from given values of $g^{x_1}, g^{x_2}, \ldots, g^{x_k}$. Moreover, the GMDLPwAI can be viewed as a generalization of the k-MDL problem (i.e. the GMDLPwAI reduces to the k-MDL problem when $d = 1$ and $e_1 = 1$). In a similar vein, it is required that $g^{f_i(x_1+x_2+\cdots+x_k)}$ to be efficiently computable, from the given known values of the GMDLPwAI. In the instance of GMDLPwAI, $f_i(x) = x^{e_i}$. In that regard, suppose $g^{f_{i0}(x_1+x_2+\cdots+x_k)} = g^{a_1 f_{i1}(x_1)} g^{a_2 f_{i2}(x_2)} \ldots g^{a_k f_{ik}(x_k)}\ \forall\ x_i$ and for some integer constants a_i so that it can be efficiently computed. We prove the following result.

Theorem 5. *Let $f_i(x) = x^{e_i}$, where $e_i \in \mathbb{Z}$ are not necessarily distinct. If for some integer constants a_i such that*

$$f_0(x_1 + x_2 + \cdots + x_k) \equiv a_1 f_1(x_1) + a_2 f_2(x_2) + \ldots a_k f_k(x_k) \mod p$$

for all odd primes p and for all $x_1, x_2, \ldots, x_k \in \mathbb{Z}_p$, then the only solutions are of the form $f_i = x^{c_i(p-1)+1}$ for some integer constants c_i.

Proof. For each $1 \leq i \leq k$, substitute $x_i = 1$ and $x_{i'} = 0$ for $i \neq i'$. We obtain $f_0(1) = a_i f_i(1)\ \forall\ i$. Hence, $a_i = 1\ \forall\ i$. Upon establishing that all a_i values have to be 1, we proceed as follows.

Let $x_2 = x_3 = \cdots = x_k = 0$. This implies that $f_0(x_1) \equiv f_1(x_1) \bmod p$ for all x_1. Similarly, let $x_1 = x_3 = \cdots = x_k = 0$. This implies that $f_0(x_2) \equiv f_2(x_2) \bmod p$ for all x_2. Continuing in this fashion, it can be deduced that $f_0(x) \equiv f_1(x) \equiv \dots f_k(x) \bmod p \;\forall\; x$. Next, let $x_3 = x_4 = \cdots = x_k = 0$. This implies $f_0(x_1+x_2) \equiv f_0(x_1)+f_0(x_2) \bmod p$. We claim that $f_0(x) \equiv xf_0(1) \bmod p \;\forall\; x$. It clear that the result holds true for $x = 0, 1$. By applying an inductive argument,

$$f_0(x+1) \equiv f_0(x) + f_0(1) \equiv xf_0(1) + f_0(1) \equiv (x+1)f_0(1) \mod p$$

and the claim follows. Hence, p divides $x^{e_0} - x$ for all $x \in \mathbb{Z}_p$. Since $(\mathbb{Z}/p\mathbb{Z})^\times \cong C_{p-1}$, there exists a generator of the cyclic group $x_0 \in \mathbb{Z}_p$ such that if p divides $x_0^{e_0} - x_0$, then $e_0 \equiv 1 \bmod p-1$. Moreover, since we have earlier established that $f_0(x) \equiv f_1(x) \equiv \dots f_k(x) \bmod p$, it can be concluded that $e_i \equiv 1 \bmod p-1$ for all i. Hence, $f_i = x^{c_i(p-1)+1}$ and it is straightforward to verify that these are indeed solutions to the original congruence equation. □

From the result of Theorem 5, if e_i is of the form $e_i = c_i(p-1)+1$, then $g^{f_i(x_1+x_2+\cdots+x_k)}$ be can efficiently computed. However, $g^{x^{c_i(p-1)+1}} = g^x$ so such e_i values reduces to the classical multiple discrete logarithm problem. Therefore, the matrix method is not applicable to solve the GMDLPwAI.

6.3 Block Based $\mathbb{F}_p$-GMDLPX

The block method can be also adapted to obtain bounds for the $\mathbb{F}_p$-GMDLPX by randomizing the input elements with the computations $(g^{\chi^{x_i}})^{\chi^{r_{i,j}}} = g^{\chi^{x_i+r_{i,j}}}$ where $r_{i,j} \in \mathbb{Z}_p$ are selected randomly. For each $r_{i,j}$, reduce $\chi^{r_{i,j}}$ modulo p and then $g^{\chi^{x_i+r_{i,j}}}$ can be computed within $2\log_2 p$ group operations.

Suppose the solver outputs solution (l, y) corresponding to some particular input $g^{\chi^{x_i+r_{i,j}}}$. In which case, x_i can thus be obtained by solving $r_{i,j} + x_i \equiv y \bmod p$. The analysis and obtained bounds in this case is similar to the classical GMDL problem which was already discussed in Sect. 5 so we will omit the details here.

Let T''_k and $T''_{k,n}$ denote the time taken in group operations for an optimal algorithm to solve the $\mathbb{F}_p$-MDLPX and $\mathbb{F}_p$-GMDLPX problems respectively. It has been shown in [12] that T''_k can be achieved in $O(\sqrt{kN})$. If this is optimal, then our results show that

$$T''_{k,n} = \Omega\left(\frac{\sqrt{k}}{\frac{n}{k}+r}\sqrt{N}\right)$$

where $r = r(k,n) \geq 1$ is any function of k and n satisfying $r(k,n) = \Omega\left(\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})}\right)$, subject to the conditions given in Theorem 2.

6.4 Matrix Based $\mathbb{F}_p$-GMDLPX

The matrix method does not apply here since there is no known efficient method to compute $g^{\chi^{x_i+x_j}}$ given $g^{\chi^{x_i}}$ and $g^{\chi^{x_j}}$ if the Diffie-Hellman assumption holds.

7 Some Explicit Bounds of $T_{k,n}$

The conditions imposed in Theorem 2 might initially seem restrictive but in fact they are satisfied by large classes of k and n. In this section, we present some interesting explicit bounds of $T_{k,n}$ by varying n relative to k. For the remainder of this section, k can be taken to be any function satisfying $k = O(p^{1-\epsilon})$, for some $0 < \epsilon < 1$ and c is a constant. The proofs of Proposition 3 and 4 are straightforward applications of Theorem 2.

Proposition 3.

$$T_{k,k+c} = \Omega(\sqrt{kp})$$

where c is a positive constant.

Proposition 4.

$$T_{k,ck} = \Omega(\frac{\sqrt{k}}{\log k}\sqrt{p})$$

where c is a constant, $c > 1$.

Proposition 5.

$$T_{k,k\log^c k} = \Omega(\frac{\sqrt{k}}{\log^{1+c} k}\sqrt{p})$$

where c is a positive constant.

Proof.

$$\log\frac{n}{n-k} = \log\frac{\log^c k}{\log^c k - 1}.$$

Next, consider the function $\log\frac{\log^c k}{\log^c k-1}$, for $k > e$ where e is the base of the natural logarithm. For brevity, denote $x = \log^c k$ so $x > 1$. By the Maclaurin series expansion, for $x > 1$,

$$\log\frac{x}{x-1} = -\log(1-\frac{1}{x}) = \sum_{i=1}^{\infty}\frac{1}{ix^i} > \frac{1}{x}.$$

In particular, this proves that $\log\frac{\log^c k}{\log^c k-1} > \frac{1}{\log^c k}$ when $k > e$. Hence for $k > e$,

$$\frac{\log(\frac{k^2}{n})}{\log(\frac{n}{n-k})} < (\log^c k)\log(\frac{k^2}{n}) = (\log^c k)\log(\frac{k}{\log^c k}) < \log^{1+c} k.$$

Thus $r(k,n)$ can be taken to be $\log^{1+c} k$ when $n = k\log^c k$. From Theorem 2, we obtain

$$T_{k,k\log^c k} = \Omega(\frac{\sqrt{k}}{\log^c k + \log^{1+c} k}\sqrt{p}) = \Omega(\frac{\sqrt{k}}{\log^{1+c} k}\sqrt{p}).$$

□

Table 1 provides a summary of the results for the lower bounds of $T_{k,n}$ with different n relative to k.

Table 1. Some bounds of $T_{k,n}$

k	n	r	$T_{k,n}$
$o\left(\frac{p^{1/3}}{\log^{2/3} p}\right)$	$\{n : n\sqrt{k} = o\left(\frac{\sqrt{p}}{\log p}\right)\}$	N.A	$\Omega(\sqrt{kp})$
$\Omega\left(\frac{p^{1/3}}{\log^{2/3} p}\right)$	$k + c$, $c \geq 0$	constant	$\Omega(\sqrt{kp})$
$\Omega\left(\frac{p^{1/3}}{\log^{2/3} p}\right)$	ck, $c > 1$	$\log k$	$\Omega(\frac{\sqrt{k}}{\log k}\sqrt{p})$
$\Omega\left(\frac{p^{1/3}}{\log^{2/3} p}\right)$	$k \log^c k$, $c > 0$	$\log^{1+c} k$	$\Omega(\frac{\sqrt{k}}{\log^{1+c} k}\sqrt{p})$

8 Conclusion

In this paper, we established rigorous bounds for the generic hardness of the generalized multiple discrete logarithm problem which can be regarded a generalization of the multiple discrete logarithm problem. Some explicit bounds are also computed using both the matrix method and block method. Many instances of $T_{k,n}$ are shown to be in fact asymptotically optimal. The overall best bounds obtained here require the union of results from both of these techniques. Furthermore, we show that the block method can also be adapted to handle generalizations arising in other discrete logarithm problems. We similarly obtain bounds for these generalizations. For instance, a consequence of our result highlights that solving an instance of the MDLPwAI problem is as hard as solving the DLPwAI problem under certain conditions. We also demonstrated why the matrix method is not applicable to these and other variants of DLP.

Acknowledgement. The authors wish to thank Phong Nguyen for valuable discussions and all anonymous reviewers for their helpful comments. This research was partially supported by JST CREST Grant Number JPMJCR14D6, Japan and JSPS KAKENHI Grant Number 16H02780.

References

1. Digital signature standard (DSS). NIST (National Institute of Standards and Technology) FIPS, 186–4 (2013)
2. Bernstein, D.J.: Curve25519: new Diffie-Hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006). doi:10.1007/11745853_14
3. Bernstein, D.J., Lange, T., Schwabe, P.: On the correct use of the negation map in the Pollard Rho method. In: Public Key Cryptography, pp. 128–146 (2011)
4. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24676-3_14
5. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24676-3_4

6. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005). doi:10.1007/11535218_16
7. Cheon, J.H., Hong, J., Kim, M.: Accelerating Pollard's Rho algorithm on finite fields. J. Cryptol. **25**(2), 195–242 (2012)
8. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
9. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithm. IEEE Trans. Inf. Theory **31**(4), 469–472 (1985)
10. Fouque, P.-A., Joux, A., Mavromati, C.: Multi-user collisions: applications to discrete logarithm, even-Mansour and PRINCE. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 420–438. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45611-8_22
11. Hitchcock, Y., Montague, P., Carter, G., Dawson, E.: The efficiency of solving multiple discrete logarithm problems and the implications for the security of fixed elliptic curves. Int. J. Inf. Secur. **3**(2), 86–98 (2004)
12. Kim, T.: Multiple discrete logarithm problems with auxiliary inputs. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 174–188. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48797-6_8
13. Kuhn, F., Struik, R.: Random walks revisited: extensions of Pollard's Rho algorithm for computing multiple discrete logarithms. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 212–229. Springer, Heidelberg (2001). doi:10.1007/3-540-45537-X_17
14. Mitsunari, S., Sakai, R., Kasahara, M.: A new traitor tracing. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **85**(2), 481–484 (2002)
15. Pollard, J.: Monte Carlo methods for index computations mod p. Math. Comput. **32**(143), 918–924 (1978)
16. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 239–252. Springer, New York (1990). doi:10.1007/0-387-34805-0_22
17. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_18
18. Teske, E.: On random walks for Pollard's Rho method. Math. Comput. **70**, 809–825 (2000)
19. Yun, A.: Generic hardness of the multiple discrete logarithm problem. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 817–836. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_27

An Enhanced Binary Characteristic Set Algorithm and Its Applications to Algebraic Cryptanalysis

Sze Ling Yeo[1], Zhen Li[1](✉), Khoongming Khoo[2], and Yu Bin Low[2]

[1] Infocomm Security Department, Institute for Infocomm Research,
1 Fusionopolis Way, #21-01 Connexis, Singapore 138632, Singapore
slyeo@i2r.a-star.edu.sg, lizh0019@gmail.com

[2] DSO National Laboratories, 20 Science Park Drive, Singapore 118230, Singapore
{kkhoongm,lyubin}@dso.org.sg

Abstract. Efficient methods to solve boolean polynomial systems underly the effectiveness of algebraic attacks on cryptographic ciphers and the security of multi-variate cryptosystems. Amongst various polynomial solving algorithms, the binary characteristic set algorithm was recently proposed to solve boolean polynomial systems including those arising from ciphers. In this paper, we propose some novel techniques to enhance the existing characteristic set solver. Specifically, we incorporate the ElimLin procedure and apply basic statistical learning techniques to improve the performance of the characteristic set algorithm. Our experiments show that our enhanced solver EBCSA performs better than existing algebraic methods on some ciphers, including CANFIL and PRESENT ciphers. We also perform the first algebraic cryptanalysis on the PRINCE cipher and an algebraic attack on Toyocrypt in a more practical/realistic setting as compared to previous attacks.

Keywords: Characteristic set algorithm · Algebraic cryptanalysis · ElimLin · Statistical learning

1 Introduction

1.1 Background

In the search for possible cryptographic algorithms in the post-quantum era, multi-variate cryptography has emerged as one of the potential candidates. This branch of cryptography comprises schemes based on the difficulty to solve multivariate polynomials over finite fields. While solving generic multi-variate polynomial systems is known to be NP-hard, a number of polynomial systems arising from cryptographic constructions have been efficiently solved. A well-known example is the polynomial system from the HFE cryptosystem [1]. As such, it is important to have a better understanding of the existing methods to solve multi-variate polynomial systems constructed from cryptosystems.

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 518–536, 2017.
DOI: 10.1007/978-3-319-61204-1_26

Various methods to solve multivariate polynomial systems exist. The most common approach to solve generic polynomial systems over finite fields is Gröbner basis algorithms [2–4]. Typically, a Gröbner basis with respect to the degree reverse lexicographical ordering is first computed via algorithms F_4 or F_5 [3,4]. It is then converted to a Gröbner basis with respect to the lexicographical ordering by algorithms such as the FGLM algorithm [5] which contains equations where variables are eliminated. This enables the variables to be solved one at a time. See [6] for more details.

Another approach to solve multivariate polynomial systems is the XL algorithm and its variants [7–10]. This class of algorithms performs well when the system under consideration is overdetermined, that is, the number of equations far exceeds the number of variables. Briefly, this method works by considering multiples of the generating polynomials bounded by some degree for which the number of independent equations exceeds the number of monomials. Common linear algebra techniques can then be applied to solve this resulting system.

In both the Gröbner basis and XL type approaches, new polynomials are gradually added with degrees larger than the degree of the generating polynomials. This often causes the number of monomials to increase very rapidly, thereby resulting in excessive memory requirements. The ElimLin algorithm was first proposed by Courtois to attack DES [11,12]. Essentially, the ElimLin algorithm seeks to successively find linear equations in the vector space generated by the original equations and subsequently, eliminating variables using these linear equations. Observe that his process constructs new polynomial systems with fewer variables without an increase in the degree. In fact, the "ElimLin" subroutine is incorporated into most Gröbner basis implementations.

All the above methods have been exploited in algebraic cryptanalysis of ciphers to solve for the unknown key variables in the polynomial system representing the ciphers. In general, block ciphers are less vulnerable to algebraic cryptanalysis as the polynomials usually have very high degrees or involve a large number of variables. Nonetheless, some block ciphers had been broken by algebraic attacks, including Keeloq [13]. As another example, ElimLin was used in [14] to solve 5 rounds of Present with half of the keybits fixed and 5 known plaintext/ciphertext pairs.

1.2 The Characteristic Set Algorithm

The characteristic set algorithm (or CSA for short) had been well-established to analyze algebraic properties of polynomial systems over algebraically-closed fields of characteristic 0, see [15,16]. In [17,18], the authors adapted this approach to solve polynomial systems over any finite field. Specifically, CSA seeks to decompose a polynomial system into monic triangular sets of polynomials with disjoint solution sets. Thus, the problem now boils down to solving a monic triangular set of polynomials which can be easily accomplished. In the case of a binary field, they further refined the decomposition algorithm so that the degrees of newly generated polynomials do not increase. The authors of [17] provided some experimental evidence which demonstrated that CSA seems to be more effective on sparse polynomial systems, particularly those exhibiting a block triangular structure. Examples of such systems include polynomials arising from linear feedback shift registers with nonlinear combining functions used in stream ciphers.

1.3 Our Contributions

In this paper, we propose some novel techniques to further enhance the binary CSA. The main features can be summarized as follows:

- we incorporate ElimLin into the binary CSA.
- we apply some statistical learning techniques to choose the next "splitting polynomial".
- we also adaptively choose the next polynomial set to process.
- we propose a preprocessing phase where the variables can be sorted.
- we allow for some flexibility in choosing the combination of features suitable for each polynomial system.

We call our enhanced version the enhanced binary characteristic set algorithm (EBCSA for short). In this paper, we present our experimental results on the block ciphers Present and Prince, as well as the stream ciphers Canfil and Toyocrypt. First, we benchmark EBCSA against prior versions of the characteristic set algorithms on Canfil ciphers. For Present cipher, we show that EBCSA outperforms previous algebraic methods as 5 rounds can be solved with fewer fixed keybits and known plaintext/ciphertext pairs. Moreover, we provide the first algebraic attack on 6 rounds of Present. As for the block cipher Prince, we carried out the first algebraic attack and our results showed that EBCSA is more effective than brute force search for 6 rounds of Prince-core (i.e. without whitening). Finally, we show that the stream cipher Toyocrypt can be solved by EBCSA in a more realistic setting compared to previous attacks, namely, using re-synchronization with sufficient random IV's for a small number of keystream bits.

1.4 Organization of This Paper

This paper is organized as follows. In Sect. 2, we set out the notations and terminologies. We then provide a general description of the binary characteristic set algorithm [17] and its variant [18]. In Sect. 3, we describe our new techniques and present our enhanced binary characteristic set algorithm. Our experimental results are then provided in Sect. 4. Finally, we wrap up with some concluding remarks and suggestions for future research.

2 A Description of the Characteristic Set Algorithm

2.1 Notations and the Set-Up

Let $F = \mathcal{F}_2$ denote the binary field. For a positive integer n, let $R = F[x_0, x_1, \ldots, x_{n-1}]$ be the multivariate polynomial ring in n variables. Since

$x^2 = x$ for all $x \in F$, we are interested in the quotient ring $\overline{R} = R/\{x_i^2 + x_i : i = 0, 1, \ldots, n-1\}$. In particular, every polynomial in R is equivalent to a representative $p \in \overline{R}$ such that the degree of each variable in p is at most 1. In the following, we will represent all the polynomials in this form.

Definition 1. *Fix σ to be a permutation on $\{0, 1, \ldots, n-1\}$. Let P be a polynomial in $\overline{R}$. We define the class of P with respect to σ to be the smallest i such that $x_{\sigma(i)}$ occurs in P. We denote the class of P by $\mathrm{cls}_\sigma(P)$. If σ is clear in the context, we will simply denote it by $\mathrm{cls}(P)$.*

Remark 1.
- *For a constant polynomial $P = 0$ or 1, we define $\mathrm{cls}(P) = -\infty$.*
- *Let σ be the identity permutation, that is, $\sigma(i) = i$ for $i = 0, 1, \ldots, n-1$. Then $\mathrm{cls}_\sigma(P)$ is the smallest i such that x_i occurs in P. On the other hand, suppose that $\sigma(i) = n-1-i$ for $i = 0, 1, \ldots, n-1$. Then $\mathrm{cls}_\sigma(P)$ is the largest i such that x_i occurs in P. This coincides with the definition in [17].*

For a polynomial $P \in \overline{R}$, let c be its class with respect to a permutation σ. Then P can be expressed as $P = Ix_{\sigma(c)} + U$, where $x_{\sigma(c)}$ does not occur in both I and U. We term I and U as the left child and right child of P, respectively. By the definition of c, it follows that both $\mathrm{cls}_\sigma(I)$ and $\mathrm{cls}_\sigma(U)$ are strictly greater than c. Note that a monic polynomial is one in which $I = 1$.

Definition 2. *Let $\mathcal{A}$ be a finite set of polynomials in $\overline{R}$. $\mathcal{A}$ is called triangular if the classes of the polynomials in $\mathcal{A}$ are all distinct. Moreover, $\mathcal{A}$ is monic triangular if every polynomial in $\mathcal{A}$ is monic.*

The Main Problem: Let $\mathcal{P}$ be a finite set of boolean polynomials from $\overline{R}$. We wish to find the zero set of $\mathcal{P}$, that is, to find $Z(\mathcal{P}) = \{(x_0, x_1, \ldots, x_{n-1}) \in F^n : P(x_0, x_1, \ldots, x_{n-1}) = 0 \text{ for all } P \in \mathcal{P}\}$.

2.2 The Basic Structure of Binary CSA

In this section, we describe the main principles underlying binary CSA to help solve the main problem stated above. Our description is more generic than that in [17] in the sense that we allow for a variable ordering on the variables. Note that we will focus on the multiplication-free CSA or MFCS presented in [17] as it was already shown to result in the best performance for the binary case. As such, CSA will refer to MFCS from now on. Throughout this section, we fix a permutation σ and simply write $\mathrm{cls}(P)$ for the class of P. Essentially, the approach of binary CSA is a result of the following lemma.

Lemma 1. *Let P be a polynomial of $\overline{R}$ with class c and write $P = Ix_{\sigma(c)} + U$. Then $Z(\{P\})$ can be split into $Z(\{P\}) = Z(\mathcal{P}_1) \cup Z(\mathcal{P}_2)$ such that: (1) $Z(\mathcal{P}_1)$ and $Z(\mathcal{P}_2)$ are disjoint; (2) One of $\mathcal{P}_1$ or $\mathcal{P}_2$ contains a monic polynomial with class c.*

Proof. As we are in the binary setting, the left child of P, namely I, can only take values 0 or 1 and $Z(\{I\})$ and $Z(\{I+1\})$ are disjoint. When $I = 0$, we have $U = P + Ix_{\sigma(c)} = 0$. On the other hand, if $I = 1$, we have $x_{\sigma(c)} + U = Ix_{\sigma(c)} + U = P = 0$. Let $\mathcal{P}_1 = \{I, U\}$ and $\mathcal{P}_2 = \{I + 1, x_{\sigma(c)} + U\}$. The lemma now follows.

Suppose that $\mathcal{A}$ is a monic triangular set of polynomials. Clearly, $\mathcal{A}$ cannot contain more than n polynomials. Suppose that $0 \leq c_1 < c_2 < \ldots < c_t \leq n - 1$ are the classes of the polynomials in $\mathcal{A}$. Then, $\mathcal{A}$ contains the polynomials $P_i = x_{\sigma(c_i)} + U_i$, $i = 1, 2, \cdots, t$, where $U_1, U_2, \ldots, U_t$ represents the right child of $P_1, \ldots, P_t$, respectively. Observe that since $c_1 < c_2 < \ldots < c_t$, U_t does not contain the variables $x_{\sigma(c_1)}, \ldots, x_{\sigma(c_t)}$. It follows that as long as the variables of U_t are fixed to some values, $x_{\sigma(c_t)}$ is determined. Similarly, U_{t-1} does not contain the variables $x_{\sigma(c_1)}, \ldots x_{\sigma(c_{t-1})}$ which allows one to compute $x_{\sigma(c_{t-1})}$. In this way, we can determine the zeroes of $\mathcal{A}$ by letting x_i, $i \notin \{\sigma(c_1), \ldots, \sigma(c_t)\}$ take values 0 or 1 and computing the corresponding values of $x_{\sigma(c_1)}, \ldots, x_{\sigma(c_t)}$ from the polynomials in $\mathcal{A}$.

In view of the above, the main goal of binary CSA is to decompose $Z(\mathcal{P})$ into disjoint sets based on Lemma 1. Specifically, we have the following theorem:

Theorem 1. [17, Theorems 4.1, 4.2] *Let $\mathcal{P}$ be a finite set of boolean polynomials of $\overline{R}$. The binary CSA decomposes $Z(\mathcal{P})$, in a finite number of steps, into a disjoint union of zero sets, namely,*

$$Z(\mathcal{P}) = \bigcup_{j=1}^{s} Z(\mathcal{A}_j),$$

where $\mathcal{A}_1, \ldots, \mathcal{A}_s$ are all monic triangular sets of boolean polynomials.

The main structure of CSA is described in Algorithm 1 in the Appendix. In order to achieve the goal of Theorem 1, the main function of binary CSA is the triset algorithm which takes in as input a finite set of boolean polynomials and outputs a monic triangular set $\mathcal{A}$ (possibly empty) as well as a set $\mathcal{P}^*$ of sets of boolean polynomials.

Remark 2. Beginning with the set $\mathcal{P}$, the whole CSA can be thought of as building a binary tree of sets ($\mathcal{P}^*$) by adding $\mathcal{Q}^*$ as a branch to the corresponding input set to the triset function. The process stops when all the sets in the tree are processed and the result is a set $\mathcal{A}^*$ of monic triangular sets. Since the polynomial splits by letting $I = 0$ or 1, the binary CSA is essentially a generalization of brute force search where we split by letting the variables take 0 or 1.

Remark 3. Notice that we iteratively split polynomials to obtain sets with disjoint solutions. In this basic version of the triset function, we work on polynomials according to their class, namely, we pick the polynomial with the smallest class. In addition, the next input set to be fed into the triset function is generally taken to be the last generated set in $\mathcal{P}^*$.

2.3 BCSA

In [18], the authors proposed some additional features to improve the efficiency of the binary CSA in [17]. We call it BCSA and briefly describe the techniques here.

Linearization. The authors claimed that this technique was already implemented in the experiments of [17] but it was not explicitly mentioned. This technique applies to the triset function. Suppose that at a certain step, the polynomial $x_{\sigma(c)} + L$ is generated, where $\deg(L) \leq 1$. Then we substitute $x_{\sigma(c)}$ with L in the remaining polynomials and add $x_{\sigma(c)} + L$ to $\mathcal{A}$. Observe that this forms part of the ElimLin process (only the substitution part).

The Choose Function. As discussed in Remark 2, we continue to pick polynomials to split until a monic triangular set is obtained or an inconsistency is reached. In the triset function of CSA, we simply pick the next polynomial to be one with the smallest class in the set. To improve efficiency, it was suggested in [18] to sort the polynomials according to some complexity metric and to choose the "simplest" polynomial among the remaining polynomials to split. The rationale is to seek for more linear polynomials which will help to simplify other polynomials by linearization described above. The complexity metric proposed in BCSA orders the polynomials as follows: A polynomial P is simpler than a polynomial P' if $(\deg(P), \#I(P), \#U(P), n-1-\mathrm{cls}_\sigma(P)) < (\deg(P'), \#I(P'), \#U(P'), n-1-\mathrm{cls}_\sigma(P'))$ in a lexicographical manner, where $I(P)$ (resp. $U(P)$) and $I(P')$ (resp. $U(P')$) represent the left child (resp. right child) of P and P' respectively. Note that the description of the choose function in [18] with respect to the class of the polynomials is in a reverse order due to the difference in the definition of the class of a polynomial.

The Threshold Value. By Lemma (1), whenever we split a polynomial, we obtain a monic polynomial with the same class. In the triset function of CSA, we first split all the polynomials of the same class c to obtain all the monic polynomials with class c. We choose one of the monic polynomials P_0 to be in $\mathcal{A}$ and then add P_0 to all the monic polynomials to eliminate the class c from the remaining polynomials (the adding step).

Since BCSA splits polynomials in order of their complexity instead of their classes, it is not efficient to first obtain all the monic polynomials before adding them. As such, the authors of [18] introduced a threshold value t such that whenever t monic polynomials of the same class are obtained, the adding process will be executed to eliminate the class from $t-1$ of these polynomials.

3 The Enhanced Binary CSA

In this paper, we propose additional techniques to enhance BCSA. Our version will be called EBCSA. Like BCSA, we keep the core of the structure, namely, EBCSA comprises the EBCSA-triset function and the main EBCS function, as shown in Fig. 1, where the new features introduced in EBCSA are marked in bold. Its basic structure is described in Algorithm 2 in the appendix. In the subsequent sections, we will describe the main features of EBCSA.

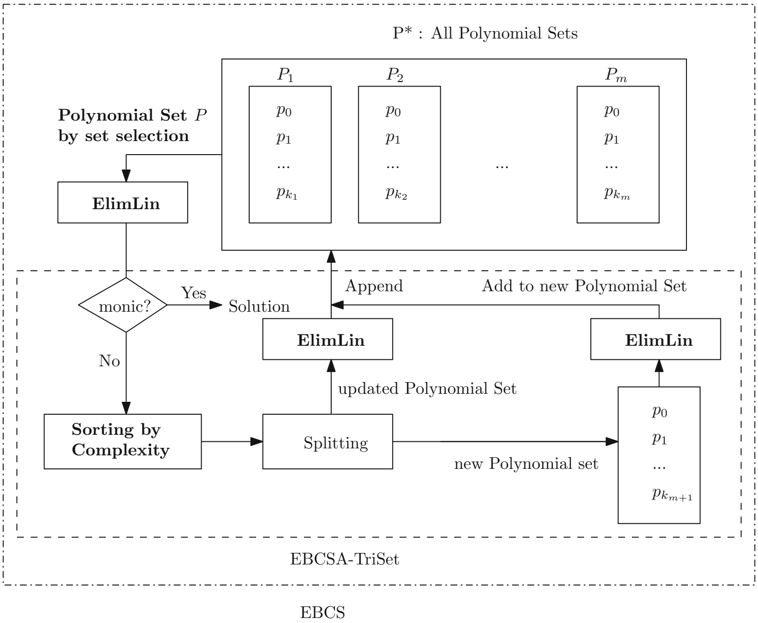


Fig. 1. Principles of the enhanced binary characteristic set algorithm

3.1 The ElimLin Technique

The first main feature of EBCSA is to integrate ElimLin into EBCSA-triset [11,12]. Specifically, whenever a new polynomial is generated in $\mathcal{P}$, we perform the following as long as the generated set contains a linear polynomial. First, order the monomials in decreasing order of degrees (for example, the Grevlex ordering). With respect to this monomial ordering, represent each polynomial as a vector of coefficients. Next, perform Gaussian elimination on the matrix of these vectors. If 1 is obtained, one can terminate EBCSA-triset with $\mathcal{A} = \emptyset$ and $\mathcal{P}^*$. Otherwise, obtain all the linear polynomials $x_i + L_i$ and substitute x_i with L_i in the remaining polynomials of $\mathcal{P}$ for all such i's.

A key merit of this process is that it allows us to detect all the linear polynomials in the vector space spanned by the polynomials [12], thereby simplifying the remaining polynomials more quickly. At the same time, it detects an inconsistent set more rapidly. However, the main drawback is that it generally requires more memory to construct the matrix and may not be feasible when our polynomial set contains too many polynomials.

3.2 An Improved Complexity Metric

In BCSA, a complexity metric was introduced in order to sort the polynomials. With respect to this metric, the next simplest polynomial will be picked to perform the splitting. Their metric compares the degree of the polynomials, the number of terms in the left child and the right child of the polynomials as well as the classes of the polynomials in a linear manner. We propose a better metric based on the Sigmoid function.

For a child $\mathbf{p}$ of a polynomial $\mathbf{P}$, we define its complexity as

$$complex\left(x_1, x_2\right) = x_1 + Sigmoid\left(x_2, K\right) = x_1 + \frac{1 - e^{\frac{x_2}{K}}}{1 + e^{\frac{x_2}{K}}} \tag{1}$$

where $x_1 = \deg\left(\mathbf{p}\right)$ determines the overall complexity and $x_2 = \#\left(\mathbf{p}\right)$ counts the number of terms in $\mathbf{p}$ and it is normalized by the Sigmoid function to the interval $(0, 1)$. Here $K = -100$ is a scaling parameter which is optimized experimentally. Then the complexity of the polynomial $\mathbf{P} = I \cdot x_{\sigma(c)} + U$ is defined as

$$Complex(\mathbf{P}) \ = \ Complex(\mathbf{I}) + \alpha\left(\mathbf{P}\right) \cdot Complex(\mathbf{U}) \tag{2}$$

and

$$\alpha\left(\mathbf{P}\right) = c^{a \cdot \deg(\mathbf{P}) - b} = 10^{3 \cdot \deg(\mathbf{P}) - 8} \tag{3}$$

where $Complex(\mathbf{p}) \ = \ complex(\deg\left(\mathbf{p}\right), \#\left(\mathbf{p}\right))$, and a, b and c are set to be 3, 8 and 10, respectively. $\mathbf{p}$ is taken as either $\mathbf{I}$ or $\mathbf{U}$. Note that as the degree of the polynomial system increases, the complexity of the child $\mathbf{U}$ tends to be more emphasized. Thus the coefficient $\alpha\left(\mathbf{P}\right)$ is defined as an exponential function of the degree. For instance, when $\deg\left(\mathbf{P}\right)$ is 2, 3 and 4, the weighting coefficient $\alpha\left(\mathbf{P}\right)$ is 0.01, 10 and 10000, respectively.

In our model, the Sigmoid function is chosen as a candidate because it has good properties, namely, it is a bounded differentiable real-valued function in $(0, 1)$ that is defined for all real input values and has a positive derivative at each value. The standard Sigmoid model is $Sigmoid\left(x\right) = \frac{1}{1+e^{\frac{x-x_0}{K}}}$, and we are using a modified version $Sigmoid\left(x\right) = \frac{1-e^{\frac{x}{K}}}{1+e^{\frac{x}{K}}}$. The complexity model in Eq. 2 is a combination of linear model and the modified Sigmoid model, and it preserves the good properties of the standard model. Similar logistic functions have been effectively used in population growth modelling, artificial neural networks and distance measure.

To select the best model mentioned above, we have tried many possible parameters for each model. Preliminary experimental results show that the modified Sigmoid model $Sigmoid\left(x\right) = \frac{1-e^{\frac{x}{K}}}{1+e^{\frac{x}{K}}}$ exhibits the best overall performance on our training sets. The training sets include 120 randomly selected polynomial sets from more than 300 available polynomial sets generated from many different ciphers with degrees ranging from 2 to 4, and the rest of the sets are the testing sets. To determine the optimal parameters of each model, a 10-fold cross-validation [19] is applied. Hence, the parameters are dependent on the specific cipher as well as the selected features (degree and number of terms).

In order to reduce the computational cost when selecting the optimal parameters, the parameters are confined to a limited range by preliminary observations.

Then the parameter setting (K, a, b, c) in Eqs. 1, 2 and 3 is determined by the cross-validation procedure as follows: we split the training sets into 10 equal sized parts; using 9 parts we fit the parameters, that is, record the parameter setting that produces the best performance; calculate its performance on the remaining 1 part as the validation set. We repeat these steps by using every part as the validation set. We repeat the whole procedure 3 times. Finally we get an average of the 3 solving times which corresponds to each parameter setting. We choose the setting which corresponds to the minimum solving time. For each of the models mentioned above, we repeat this procedure. As a result, the optimal parameter setting is determined as $(K, a, b, c) = (-100, 3, 8, 10)$. The parameters are not very sensitive, that is, when they are modified by around 10%, the solving time remains within a 10% change. This parameter setting is not guaranteed to be effective for all individual polynomial sets, but is optimal in the sense of overall performance.

By contrast, the choose function in BCSA uses a layered linear model with respect to the essential parameters. This model suffers from the following drawbacks. (1) There are no parameters to control, limiting the possibility to optimize the performance; (2) The "layered" comparison sometimes cannot get reasonable results, e.g. a polynomial with a higher degree but very few terms is not necessarily more complex than a polynomial with a lower degree but the number of terms is huge. As such, the proposed EBCSA complexity model has more flexibility to produce better performance by optimization over a number of polynomial sets. Indeed, we have carried out extensive experiments using different functions and the above function works well in general. We present our experimental results for solving Canfil functions as comparison between using the modified Sigmoid function and the choose function of BCSA in Subsect. 4.1.

3.3 The Set Metric

We introduce a set metric on the sets in the binary tree $\mathcal{P}^*$ to help us find a set with a solution more quickly. Hence, this feature is particularly useful if the system admits a known number of solutions, as well as the case where only a fixed number of solutions is required.

Observe that polynomial sets in the same branch may contain many identical polynomials. Roughly speaking, once many sets in a certain branch terminate without a solution, we like to choose sets from a different branch to increase the chances of a solution. In other words, we will look for a set which is most dissimilar to the previous terminated sets.

For the purpose of practical implementations, we propose the following statistical feature for a set $\mathcal{P}$ of polynomials: $S_{\mathcal{P}} = (m_1, s_1, m_2, s_2)$, where m_1, m_2 (respectively s_1, s_2) represent the mean (respectively the standard deviation) of the degrees and number of terms of the polynomials in $\mathcal{P}$ respectively. Suppose that $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_l$ are inconsistent, that is, terminate without any solution. Let $a(\mathcal{P}_1, \ldots, \mathcal{P}_l) = \frac{s_{\mathcal{P}_1} + \ldots + s_{\mathcal{P}_l}}{l}$, the average of the statistical features of the terminated sets. Then we choose a set $\mathcal{P}$ such that the Euclidean distance between $s_{\mathcal{P}}$ and $a(\mathcal{P}_1, \ldots, \mathcal{P}_l)$ is the largest on the fly.

3.4 Sorting the Variables

In our description of binary CSA, we have introduced the permutation σ to allow for the variables to be rearranged. In general, CSA tends to be more efficient if σ is chosen so that the variables $x_{\sigma(0)}, \ldots, x_{\sigma(x_{n-1})}$ occur in an increasing order of frequency in the polynomial system. The underlying reasons are as follows: recall that when performing linearization, we substitute a variable x_i with a linear polynomial L_i. While this reduces the number of variables, it may result in the polynomial system becoming more complex, particularly when L contains many terms. Hence, it is more advisable to substitute those x_i which do not occur too frequently. In terms of implementation, we simply rename the variables so that $x_0, x_1, \ldots, x_{n-1}$ occur in an ascending order of frequency in the polynomial system.

3.5 Combining the Various Features

We observe from our experiments that not all polynomial systems behave equally well with the same set of features. According to experimental results, some sets may be more efficient with linearization alone while others perform much better with the ElimLin technique. In addition, the set selection may not be necessary if the system has many solutions and we seek to find all of them. In view of the above, EBCSA allows for different combinations of features to yield the best performance. In our current implementation of EBCSA, the following choices are provided: (1) Linearization or ElimLin; (2) Set selection based on statistical feature or set index; and (3) Terminate with one solution or all solutions.

Remark 4. Observe that unlike BCSA, we do not need a threshold value here as whenever we obtain two monic polynomials with the same class, we add them and choose the simpler polynomial according to our complexity metric to be in the set $\mathcal{A}$. In addition, in case the ElimLin option is turned on, polynomials with the same leading term will automatically be added.

4 Experimental Results

We tested EBCSA with many different polynomial systems, specifically those from block and stream ciphers. Our experiments show significant improvements over existing algebraic methods. In this paper, we present three such experiments, namely, the block ciphers Present and Prince as well as the stream cipher Toyocrypt. Experiments on Canfil ciphers were also carried out to compare the efficiency of EBCSA against CSA and BCSA. Since our goal is to find the unknown key which is likely unique, we use the following configurations of EBCSA for all the experiments: Linearization/ElimLin, set selection, and termination when one solution is obtained.

The platform for testing is a personal computer (Ubuntu 14.04, Intel Xeon CPU E5640 2.67 GHz, 24 GB RAM). Only single-core CPU is used. In the following, we further classify EBCSA into EBCSA and EBCSA-GE (short for EBCSA with Gaussian Elimination), where the former refers to EBCSA with linearization and the latter representing EBCSA with the ElimLin feature switched on.

4.1 Experiments on Our Complexity and Set Metrics

In order to benchmark EBCSA against BCSA and CSA, we performed experiments on the Canfil functions similar to those carried out in [17,18]. The Canfil polynomial sets are boolean polynomial systems arising from linear feedback functions with nonlinear Canfil functions. In [17, Table 4], comparisons were made between CSA and the Gröbner basis algorithm for 8 such functions and the results showed that CSA outperformed the Gröbner basis method for these polynomial sets. In [18, Table 2], the authors recorded the number of splits (or branches) resulting from both CSA and BCSA for these same functions and concluded that BCSA was more effective on these sets.

In this paper, we carry out the same experiments using EBCSA and the number of splits (number of new non-terminating sets produced) for each of the 8 Canfil functions is recorded in Table 1. In addition, the timing on the Canfil functions using the approach of EBCSA with set selection is also listed.

Table 1. Comparison between EBCSA and BCSA in terms of the number of splits

#Splits\system	Canfil1	Canfil2	Canfil3	Canfil4	Canfil5	Canfil6	Canfil7	Canfil8
CSA	13719	23881	7251	1657	1086	3331	1551	180710
BCSA	11958	16074	3267	1316	574	671	1852	35547
EBCSA w.o set selection	5218	24616	5402	782	408	400	3970	90040
EBCSA w. set selection	5772	17299	3865	343	100	209	1046	35565
EBCSA time (s)	82.9	629.5	669.7	8.8	56.4	41.7	57.5	501.3

Observe that EBCSA performs better than BCSA in 5 out of the 8 Canfil experiments with significant improvements for Canfil1, Canfil4, Canfil5 and Canfil6. For Canfil7, CSA outperforms BCSA but is less effective than EBCSA (with set selection). Note that we have used a random key in our experiments and it was not stated if a random or a specific key was used in the experiments in [17,18].

The improvements of EBCSA suggest that our choice of a combined continuous Sigmoid function for polynomial complexity is more effective than the layered linear model used in BCSA. In addition, the use of set selection results in finding the unique solution more efficiently.

4.2 Experiments on Present Cipher

Present is a 31-round lightweight block cipher designed by Bogdanov et al. announced in CHES 2007 [20]. It has a block length of 64-bits and key lengths of either 80-bits or 128-bits. It is based on a permutation-substitution network with 16 4-bit S-boxes in each round. In [14], algebraic cryptanalysis was applied to Present with the 80-bit master key. This involves representing the intermediate text by variables and each S-box by 21 quadratic equations. The authors claimed that with 5 known plaintext/ciphertext (KP) pairs, 40 key bits can be recovered by the method of ElimLin.

In the report of [21], results on algebraic attacks on Present with ElimLin and the Polybori Gröbner basis implementation were presented. Even though ElimLin was implemented as a sub-routine in Polybori implementation, it was demonstrated that the stand-alone ElimLin implementation was more effective to solve Present. In addition, it was reported that 6 rounds of Present remained resistant to algebraic cryptanalysis using either ElimLin or Polybori.

As EBCSA-GE incorporates ElimLin into EBCSA, we perform experiments on round-reduced Present with EBCSA-GE. Once again, we consider the implicit representation of each s-box with 21 quadratic equations. Note that we place all the text variables as the front variables while the 80 key variables are placed behind. We perform our experiments with either 1 or 2 KP. We considered r rounds of Present for $r = 5$ and $r = 6$. For experiments with one KP, we did some pre-processing on the polynomial systems before running them on EBCSA-GE. Specifically, we rename the variables so that the text variables and the key variables are sorted in an ascending order of frequency. In each of the experiments, we fix $x_{n-c-1}, x_{n-c}, \ldots, x_{n-1}$, (that is, c most frequent key variables) and record the time taken as well as the number of splits as c varies.

Table 2. 5-round and 6-round present with 1 KP on EBCSA-GE

r	#FixedBits	64	60	56	54	52	50	48	44	40	38	36	34
5	Time (s)	<1	<1	<1	<1	<1	3.89	5.46	58.6	237.3	161.8	2148.7	>4 h
5	#Splits	0	0	0	0	0	3	16	174	710	556	6200	–
6	Time (s)	7.1	81.0	378.9	1979.2	1489.0	>4 h	>4 h	>4 h	>4 h	>4 h	>4 h	>4 h
6	#Splits	24	216	1234	6163	2172	–	–	–	–	–	–	–

Table 3. 5-round present with 2 KP on EBCSA-GE

#FixedBits	40	36	34	32	30	28	26
Time (s)	9.5	300.1	282.9	1562.9	888.4	2120.8	>4 h
#Splits	6	191	172	974	563	1362	–

One sees that EBCSA-GE outperforms both ElimLin and the Polybori implementation of [21] in the following sense. From Table 4 with 40 keybits fixed, EBCSA-GE requires only 1 KP to solve in 4 min, which is of comparable magnitude to the timings of ElimLin and Polybori based on 5 KP's. With just 2 KP's, EBCSA-GE already outperforms ElimLin and Polybori, with a solving time of 9.5s. Next with 36 keybits fixed, EBCSA-GE can solve the system in a few minutes based on just 2 KP's while ElimLin and Polybori both take longer to solve (3 to 4 h) and need more plaintexts (16 KP's). From Table 3 with 2

Table 4. A comparison of algebraic attacks on 5-round present

	Polybori [22]		Elimlin [22]		EBCSA-GE			
#FixedBits	40	36	40	36	40	40	36	36
#KP	5	16	5	16	1	2	1	2
Time (s)	241.2	>4 h	154.8	12.2	237.3	9.5	2148.7	300.1

KP, EBCSA-GE can solve up to 52 keybits (28 fixed keybits) for 5 rounds of Present, which is more keybits than what ElimLin and Polybori can solve from [21]. Finally from Table 2, EBCSA-GE can solve 6 rounds of Present with 1 KP and 52 fixed keybits, while the ElimLin and Polybori experiments from [21] are only conducted up to 5 rounds.

Recall that ElimLin essentially seeks to find linear polynomials in the space spanned by the polynomials and then eliminating their leading variables from the remaining polynomials via substitution. This method can only be successful if there are sufficient linear polynomials in the spanning set. This implies that more polynomials involving the key variables in the original set increases the chance of finding enough linear polynomials and hence, leading to a greater number of KPs as observed. In case there are insufficient linear polynomials, Gröbner basis may not be effective as an increase in the degree of the polynomials with a large number of variables will explode the memory. On the other hand, EBCSA-GE executes the splitting process without a rapid increase in memory and continues to find linear polynomials via ElimLin. By alternating between splitting and ElimLin processes, one sees that EBCSA-GE is more effective for Present cipher.

4.3 Experiments on Prince Cipher

Prince is another lightweight block cipher published in Asiacrypt 2012 [22]. It is a 12-round block cipher with a 64-bit block length and a 128-bit key length. It is optimized with respect to latency for hardware implementations. Its construction satisfies the reflection property and is symmetric about the middle layer. The cryptanalysis of Prince has gained much attention, especially after a challenge was posed by Ruhr-Universität Bochum [23] to find a practical attack for round-reduced Prince. To date, it was shown that round-reduced Prince is vulnerable to various attacks including Differential and Integral cryptanalysis as well as time-memory trade-off and meet-in-the-middle attacks [24–28]. To the best of our knowledge, no algebraic attack has been published on Prince.

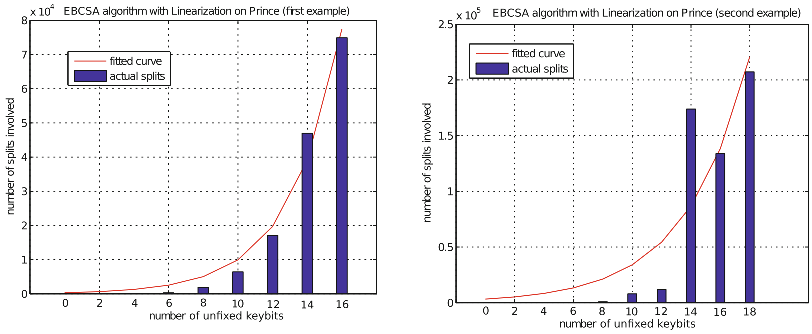
We carry out the first algebraic attack on Prince. Specifically, we consider Prince-core, that is, without whitening. Hence, only the key K_1 is used. As in all block ciphers, we consider the implicit representation of the s-boxes, thereby resulting in 21 quadratic equations for each s-box. In addition, we introduce 64 intermediate variables for each round. Since whitening is not used, the variables for the first round and the last two rounds can be omitted.

Now, consider the 6-round Prince-core. This system has 384 variables (320 text variables and 64 key variables) and 2016 quadratic equations. We place the key variables behind $(x_{320}, \ldots, x_{383})$. We generate 2 different systems using different random keys. For each key, we fix some key bits and run the system with our EBCSA solver. First, Table 5 records the results for Prince with both EBCSA and EBCSA-GE.

Table 5. 6-round Prince-core: EBCSA vs EBCSA-GE

#Fixed bits	64	56	52	48	40
#Splits of EBCSA	53	533	11415	53147	–
EBCSA time (s)	4.3	54	1400	**5900**	>4 h
EBCSA-GE time (s)	**2**	**266**	**784**	>4 h	>4 h

Observe that when more bits are fixed, both EBCSA and EBCSA-GE should be executed in parallel to find the key as it is not certain which of the two versions will be more efficient. However, when fewer bits are fixed, EBCSA-GE tends to require much more memory to generate the large matrices, thereby slowing down the process significantly. In the following tables, we show the results of EBCSA for two random keys by varying the number of fixed keybits. By fitting the results with a suitable graph, we extrapolate the results to estimate the complexity without fixing any key bits. We conclude that the number of splits of Prince-core is generally better than the brute force $f = 2^x$, where x denotes the number of unknown keybits, as shown in Fig. 2. The results are also listed in Table 6, where the complexity of EBCSA is expected to provide an improvement of more than half the key size over the brute force search. We remark that this approach can be applied to test the complexity of other block ciphers as well.



(a) (b)

Fig. 2. Splits of Prince-core. (a) $f = 324.19 \times 2^{0.494x}$ (b) $f = 3287.44 \times 2^{0.337x}$

Table 6. Splits of Prince-core (a) $f = 324.19 \times 2^{0.494x}$ (b) $f = 3287.44 \times 2^{0.337x}$

#Fixed bits	64	62	60	58	56	54	52	50	48	46
(a)										
#Splits	45	105	167	301	1908	6415	17101	46994	74887	–
Linearization	3.51	6.89	11.66	21.44	138.27	471.83	1393.60	4010.22	5699.43	>4 h
(b)										
#Splits	49	106	121	433	888	8012	11893	173878	133736	207162
Linearization	3.52	6.51	8.03	29.97	67.09	599.15	956.47	12309.5	10288.5	15156.6

The comparison with the existing attack on 6-round Prince core from [24] is provided in Table 7. It is obvious that the proposed EBCSA algorithm is more practical in terms of required number of chosen plaintext.

Table 7. Comparison of existing attacks on 6-round Prince-core

Source	Data	Time	$D \times T$	Memory	Attacks
FSE13 [24]	2^{16}	2^{30}	2^{46}	2^{16}	Integral
Proposed EBCSA	1	$T \times K \approx 2^{60}$	2^{60}	2^{15}	First algebraic attack by EBCSA

Remark 5. **D**ata: number of chosen plaintext; **T**ime: equivalent Prince operations; complexity of attacks on Prince is measured by $D \times T < 2^{128}$; Memory: Number of Prince 64-bit WORDs.

4.4 Experiments on the Stream Cipher Toyocrypt

The Toyocrypt [29] is a stream cipher comprising a 128-bit Galois linear feedback shift register (GLFSR) passing through a nonlinear function f of degree 63. Obviously, a naive application of algebraic cryptanalysis will not work due to the high degree of f. In [30], the authors discovered that in fact, $(x_{23} + 1)f$ and $(x_{42} + 1)f$ are cubic polynomials. Further, in [31], the authors showed that with re-synchronization and 4 IV's, the degrees of the equations can be further reduced to 1. Note that in this case, the 4 IV's must span a 2-D vector space.

Several algebraic attacks on Toyocrypt have been proposed. Table 8 summarizes the assumptions and requirements needed to carry out existing algebraic attacks on Toyocrypt.

The attack of [30] requires 2^{18} keystream bits for one session, while the attack of [31] requires 128 keystream bits from 4 chosen IV resynchronizations. These attack scenarios may not be easy to achieve in practice. Here we consider a more realistic scenario where only a few keystream bits can be deduced from some known information about the plaintext. Besides, we allow the IV in different sessions to be selected randomly. Finally, we apply our attack to Toyocrypt with GLFSR as in the original specifications of the cipher unlike previous attacks which only considered Toyocrypt with LFSR as a good approximation.

Table 8. Comparison of assumptions on algebraic attacks on Toyocrypt

Source	Resynchronization needed?	No. of keystream-bits	Remarks on IV
EUROCRYPT 2003 [30], CRYPTO 2004 [32]	No	2^{18}	1 session with random IV
SAC 2004 [33]	Yes	2^{13}	Chosen IVs
CANS 2009 [31]	Yes	128	Chosen IVs
Proposed EBCSA	Yes	≤ 6	Random IVs

To be more precise, we fix a GLFSR with feedback taps $0, 1, 2, 7, 128$. For any 2 IV's,it follows from [31, Theorem 1] that the sum of $(x_{23}+1)f$ (or $(x_{42}+1)f$) at these two IV's result in equations of degree 2. Instead of choosing some fixed IV's, we randomly choose k IV's and construct the corresponding equations for $(x_{23}+1)f + z(x_{23}+1)$ and $(x_{42}+1)f + z(x_{42}+1)$, where z represents the keystream bit. Notice that these are degree 3 equations but the sum of any two of them will give a degree 2 equation.

We perform our experiments with at most 6 keystream bits using EBCSA-GE and record the results of our experiments in Table 9. In the preprocessing stage, we once again arranged the 128 variables ascendingly of their frequencies.

Table 9. Performance of Toyocrypt with EBCSA-GE

#Key stream	2	2	2	2	2	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	6
#IV	128	120	112	104	96	128	120	112	104	96	92	128	120	112	104	96	92	88	84	80	80
#Poly	512	480	448	416	384	768	720	672	624	576	552	1024	960	896	832	768	736	704	672	640	960
#Splits	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	2	2	2
#Elim Lin	6	9	8	7	11	4	8	8	6	8	7	4	6	5	8	5	9	9	13	13	11
Time (s)	7.1	41.0	318.5	24.1	32.7	4.7	19.4	17.1	178.7	59.4	516.4	7.4	134.0	413.0	76.1	79.7	71.1	97.0	542.1	669.3	187.4

5 Conclusions and Future Work

In this paper, we enhanced the binary characteristic set algorithm to work more effectively on both block ciphers and stream ciphers. Using EBCSA, we improve algebraic attacks on various well-known ciphers including Prince, Present and Toyocrypt. EBCSA uses CSA as its core and incorporates ElimLin and some statistical features to improve its performance. In our current version, we can turn ElimLin on or off for each input polynomial system. For future research, we will seek to identify some characteristics of the polynomial system to choose between EBCSA-GE or EBCSA. We can further integrate the criteria into the main algorithm so that ElimLin can be turned on or off for each individual set produced by EBCSA.

Acknowledgements. We are grateful to Dr. Matt Henricksen, Dr. Yap Wun She, Dr. Lee Hian Kiat and Ms. Ivana Thng for their helpful contributions throughout the project.

Appendix: Pseudocodes for CSA and EBCSA

Next, we present the main structure of EBCSA in Algorithm 2. Note that preprocessing of the input set can be carried out to sort the variables in a desired order, that is, we fix a permutation σ beforehand.

Algorithm 1. The CSA (or MFCS) algorithm [**17**]

```
 – Set $\mathcal{P}^* = \emptyset$ and $\mathcal{A}^* = \emptyset$.
 – CSA-triset function – Feed $\mathcal{P}$ into triset function and let $\mathcal{Q}^*$ and $\mathcal{A}$ be outputs:
     • Set $\mathcal{Q}^* = \emptyset$ and $\mathcal{A} = \emptyset$.
     • For $c = 0$ to $n - 1$ do:
         ∗ If $1 \in \mathcal{P}$, return $\mathcal{A} = \emptyset$ and $\mathcal{Q}^*$.
         ∗ Let $\mathcal{Q} = \{P \in \mathcal{P} : \mathrm{cls}(P) = c\}$, set $\mathcal{P} = \mathcal{P} \backslash \mathcal{Q}$, set $\mathcal{Q}_1 = \emptyset$.
         ∗ While $\mathcal{Q} \neq \emptyset$ do
             · Pick $P \in \mathcal{Q}$ and set $\mathcal{Q} = \mathcal{Q} \backslash \{P\}$.
 Splitting step Write $P = Ix_{\sigma(c)} + U$. Add $\mathcal{P} \cup \mathcal{Q} \cup \{I, U\}$ to $\mathcal{Q}^*$.
             · Add $I + 1$ to $\mathcal{P}$ and add $x_{\sigma(c)} + U$ to $\mathcal{Q}_1$.
         ∗ Pick $P \in \mathcal{Q}_1$ and add $P$ to $\mathcal{A}$.
Adding step For every $P' \in \mathcal{Q}_1, P' \neq P$, add $P + P'$ to $\mathcal{P}$.
 – Set $\mathcal{P}^* = \mathcal{P}^* \cup \mathcal{Q}^*$ and add $\mathcal{A}$ to $\mathcal{A}^*$.
 – Repeat the whole procedure till $\mathcal{P}^*$ is empty.
```

Algorithm 2. Algorithm EBCSA

```
 – **Input**: a set of polynomials $\mathbb{P} = \{p_1, \cdots, p_k\}$.
 – **Outputs**: A monic triangular set $\mathbb{A}$ and a super set $\mathbb{P}^*$ for other potential solutions.
 – **EBCSA-triset** function
     • Begin with the simplest $P$, e.g. $P_k = Ix_{c_k} + U \in \mathbb{P}$.
     • Let $\mathbb{P}' = \mathbb{P} \backslash \{P_k = Ix_{c_k} + U\}$. Split $P_k$: $\mathbb{P} \leftarrow \mathbb{P}' \cup \{I + 1\}$, $\mathbb{A} \leftarrow \mathbb{A} \cup \{x_{c_k} + U\}$
       and $\mathbb{P}_{new} \leftarrow \mathbb{P}' \cup I \cup U$, append $\mathbb{P}_{new}$ to $\mathbb{P}^*$.
     • Process each new generated $\mathbb{P}_{new}$ or updated polynomial set $\mathbb{P}$, and terminate
       them as early as possible.
 – Continue with every $P$ until $\mathbb{P}$ is terminated or becomes the monic set $\mathbb{A}$.
 – Select a new set $\mathbb{P}$ from $\mathbb{P}^*$ to apply the above algorithm.
```

References

1. Patarin, J.: Hidden fields equations (HFE) and isomorphisms of polynomials (IP): two new families of asymmetric algorithms. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996). doi:10.1007/3-540-68339-9_4
2. Buchberger, B.: Ein algorithmus zum auffinden der basiselemente des restklassenrings nach einem nulldimensionalen polynomideal. Universitat Innsbruck, Austria, Ph.D. thesis (1965)
3. Faugere, J.C.: A new efficient algorithm for computing gröbner bases (f4). J. Pure Appl. Algebra **139**(1), 61–88 (1999)
4. Faugere, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In: Proceedings of ISSAC, ACM, pp. 75–83 (2002)
5. Faugere, J.C., Gianni, P., Lazard, D., Mora, T.: Efficient computation of zero-dimensional Gröbner bases by change of ordering. J. Symb. Comput. **16**(4), 329–344 (1993)
6. Cox, D., Little, J., O'shea, D.: Ideals, Varieties, and Algorithms, vol. 3. Springer, New York (1992)
7. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE public key cryptosystem by relinearization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 19–30. Springer, Heidelberg (1999). doi:10.1007/3-540-48405-1_2
8. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000). doi:10.1007/3-540-45539-6_27
9. Courtois, N.T., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002). doi:10.1007/3-540-36178-2_17
10. Courtois, N.T., Patarin, J.: About the XL algorithm over $GF(2)$. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 141–157. Springer, Heidelberg (2003). doi:10.1007/3-540-36563-X_10
11. Courtois, N.T., Bard, G.V.: Algebraic cryptanalysis of the data encryption standard. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 152–169. Springer, Heidelberg (2007). doi:10.1007/978-3-540-77272-9_10
12. Courtois, N.T., Sepehrdad, P., Sušil, P., Vaudenay, S.: ElimLin algorithm revisited. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 306–325. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34047-5_18
13. Indesteege, S., Keller, N., Dunkelman, O., Biham, E., Preneel, B.: A practical attack on KeeLoq. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 1–18. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_1
14. Nakahara Jr., J., Sepehrdad, P., Zhang, B., Wang, M.: Linear (Hull) and algebraic cryptanalysis of the block cipher PRESENT. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 58–75. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10433-6_5
15. Aubry, P., Lazard, D., Maza, M.M.: On the theories of triangular sets. J. Symb. Comput. **28**(1), 105–124 (1999)
16. Kalkbrener, M.: A generalized euclidean algorithm for computing triangular representations of algebraic varieties. J. Symb. Comput. **15**(2), 143–167 (1993)
17. Fengjuan, C., Xiao-Shan, G., Chunming, Y.: A characteristic set method for solving boolean equations and applications in cryptanalysis of stream ciphers. J. Syst. Sci. Complex. **21**(2), 191–208 (2008)

18. Huang, Z., Sun, Y., Lin, D.: On the efficiency of solving Boolean polynomial systems with the characteristic set method. arXiv preprint (2014). arXiv:1405.4596
19. Kohavi, R., et al.: A study of cross-validation and bootstrap for accuracy estimation and model selection. Int. Jt. Conf. Artif. Intell. **14**(2), 1137–1145 (1995)
20. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74735-2_31
21. Sepehrdad, P.: Statistical and algebraic cryptanalysis of lightweight and ultra-lightweight symmetric primitives. Ph.D. thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE (2012)
22. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçın, T.: PRINCE – a low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34961-4_14
23. Bochum, R.U.: The PRINCE Challenge. https://www.emsec.rub.de/research/research_startseite/prince-challenge/ (2014). Accessed 18 Jan 2017
24. Jean, J., Nikolić, I., Peyrin, T., Wang, L., Wu, S.: Security analysis of PRINCE. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 92–111. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43933-3_6
25. Dinur, I.: Cryptanalytic time-memory-data tradeoffs for FX-constructions with applications to PRINCE and PRIDE. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 231–253. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46800-5_10
26. Derbez, P., Perrin, L.: Meet-in-the-middle attacks and structural analysis of round-reduced PRINCE. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 190–216. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48116-5_10
27. Canteaut, A., Fuhr, T., Gilbert, H., Naya-Plasencia, M., Reinhard, J.-R.: Multiple differential cryptanalysis of round-reduced PRINCE. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 591–610. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46706-0_30
28. Canteaut, A., Naya-Plasencia, M., Vayssière, B.: Sieve-in-the-middle: improved MITM attacks. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 222–240. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_13
29. Mihaljevic, M.J.: Cryptanalysis of toyocrypt-HS1 stream cipher. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **85**(1), 66–73 (2002)
30. Courtois, N.T., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 345–359. Springer, Heidelberg (2003). doi:10.1007/3-540-39200-9_21
31. Zhang, A., Lim, C.-W., Khoo, K., Wei, L., Pieprzyk, J.: Extensions of the cube attack based on low degree annihilators. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 87–102. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10433-6_7
32. Hawkes, P., Rose, G.G.: Rewriting variables: the complexity of fast algebraic attacks on stream ciphers. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 390–406. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28628-8_24
33. Armknecht, F., Lano, J., Preneel, B.: Extending the resynchronization attack. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 19–38. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30564-4_2

SCRAPE: Scalable Randomness Attested by Public Entities

Ignacio Cascudo[1] and Bernardo David[2,3](✉)

[1] Aalborg University, Aalborg, Denmark
[2] Aarhus University, Aarhus, Denmark
davidm.bernardo@gmail.com
[3] IOHK, Hong Kong, Hong Kong

Abstract. Uniform randomness beacons whose output can be publicly attested to be unbiased are required in several cryptographic protocols. A common approach to building such beacons is having a number parties run a coin tossing protocol with guaranteed output delivery (so that adversaries cannot simply keep honest parties from obtaining randomness, consequently halting protocols that rely on it). However, current constructions face serious scalability issues due to high computational and communication overheads. We present a coin tossing protocol for an honest majority that allows for any entity to verify that an output was honestly generated by observing publicly available information (even after the execution is complete), while achieving both guaranteed output delivery and scalability. The main building block of our construction is the first Publicly Verifiable Secret Sharing scheme for threshold access structures that requires only $O(n)$ exponentiations. Previous schemes required $O(nt)$ exponentiations (where t is the threshold) from each of the parties involved, making them unfit for scalable distributed randomness generation, which requires $t = n/2$ and thus $O(n^2)$ exponentiations.

1 Introduction

The problem of obtaining a reliable source of randomness has been studied since the early days of cryptography. Whereas individual parties can choose to trust locally available randomness sources, it has been shown that local randomness sources can be subverted [BLN16,DPSW16] and many applications require a common public randomness source that is guaranteed to be unbiased by a potential adversary. This necessity inspired the seminal work on Coin Tossing by Blum [Blu81], which allows two or more parties to generate an output that is guaranteed to be uniformly random as long as at least one of the parties is honest (and given that the protocol terminates).

I. Cascudo acknowledges support from the Danish Council for Independent Research, grant no. DFF-4002-00367.

B. David—This project has received funding from the European research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme (grant agreement No. 669255).

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 537–556, 2017.
DOI: 10.1007/978-3-319-61204-1_27

The concept of a public *randomness beacon* that periodically issues fresh unpredictable and unbiased random values was proposed by Rabin [Rab83] in the context of contract signing and has found several other applications such as voting protocols [Adi08], generating public parameters for cryptographic schemes [BDF+15, LW15], privacy preserving instant messaging [WCGFJ12, vdHLZZ15], and anonymous browsing [DMS04, GRFJ14]. More recently, blockchain [Nak08, GKL15] applications such as smart contracts [KMS+16, B+14], sharding [CDE+16] and Proof-of-Stake based consensus protocols [KKR+16] have increased the need for randomness sources [BCG15].

Rabin's concept of randomness beacons fits the above applications very nicely but the proposed implementation in [Rab83] relies on a trusted third party. The goal of this paper is to construct a distributed randomness beacon guaranteeing output delivery and uniformly distributed randomness for the parties that use the beacon as long as a majority of these parties are honest. Moreover, in many of the aforementioned applications, parties that do not necessarily participate in randomness generation but wish to audit the protocol execution must be able to attest *a posteriori* that the randomness source is reliable and unbiased. Hence, we aim at constructing a publicly verifiable randomness beacon and not only a protocol that outputs randomness to the parties actively involved in its execution.

1.1 Related Works

A natural solution for obtaining randomness beacons consists in using a coin tossing protocol as proposed by Blum [Blu81] with its messages posted to a public bulletin board for later verification (or broadcast among the parties). However, it is known that in case half or more of the parties are corrupted, the adversary can bias the output of the protocol or even prevent the honest parties from obtaining any output at all by aborting protocol execution at a given point [Cle86]. Assuming that a majority of the players are honest, it is possible to guarantee output delivery [RBO89] through threshold verifiable secret sharing (VSS) [CGMA85] given that a broadcast channel is available. Basically, given that a majority of n parties are honest, each party can secret share its input into n shares such that $n/2$ are enough to reconstruct the secret, sending one share to each involved party before starting the coin tossing protocol. While the adversary cannot recover any input (since it has at most $n/2-1$ shares of each input), the honest parties collective know at least $n/2$ shares, which they can use to reconstruct the inputs of parties who abort and then finish the protocol.

While a coin tossing protocol with guaranteed output delivery (G.O.D.) with a honest majority based on VSS provides a reliable source of randomness, this approach still has two main issues: 1. most VSS schemes require interaction between the dealer and the other parties, which hinders scalability and 2. only parties who actively participate in the protocol can verify that it was executed correctly. While non-interactive VSS [Fel87] solves the interaction problem, it does not allow the protocol execution to be independently verified by entities that did not actively participate. A natural way to allow for any entity to verify that the outputs produced by such protocols is indeed honestly generated is to substitute traditional VSS by publicly verifiable secret sharing (PVSS) schemes [Sta96], which allow for anybody to verify the validity of shares and reconstructed secrets through information that can be made publicly available without requiring direct interaction between any of the parties. Variations of this approach have been proposed in [KKR+16,SJK+16].

While [KKR+16] instantiates a plain [RBO89]-style G.O.D. coin tossing protocol requiring communication among all parties (through a public ledger), [SJK+16] reduces the communication complexity by partitioning parties into committees that internally run a protocol with publicly verifiable outputs. Later on, a *client* that only communicates to the *leader* of each committee (instead of talking to all parties) can aggregate these outputs to obtain publicly verifiable randomness. However, while the vanilla approach of [KKR+16] achieves security assuming only an honest majority (meaning that the adversary corrupts less than half of all parties), the communication efficient approach of [SJK+16] only achieves security against an adversary that corrupts less than a third of all parties. Moreover, provided that there is an honest majority, the protocol of [KKR+16] guarantees that all parties get output regardless of which parties are corrupted, while in the protocols of [SJK+16], even if only the client is corrupted, it can abort and prevent all other parties from receiving randomness.

Even though coin tossing with G.O.D. built through PVSS can potentially achieve scalability and public verifiability, current PVSS constructions [Sta96,FO98,Sch99,BT99,RV05,HV09,Jha11,JVSN14] suffer from high computational overhead. In general, the parties are required to each compute $O(nt)$ exponentiations to verify n shares of a secret with threshold t, which translates into $O(n^2)$ exponentiations since $t = n/2$ in our randomness beacon application[1]. This computational overhead arises because the main idea behind these schemes is to commit to the coefficients of a polynomial used for a Shamir Secret Sharing [Sha79] and encrypt the shares, later using the commitments to the coefficients to independently compute commitments to the shares, which are proven in zero-knowledge to correspond to the encrypted shares. This approach was originally put forth in [Sch99], which uses the Fiat-Shamir heuristic (and consequently the random oracle model) to obtain the necessary non-interactive zero-knowledge proofs. Later on, variations of this protocol in the plain model were proposed in [RV05,JVSN14], which substitute the zero-knowledge proofs by checks based on Paillier Encryption [Pai99], and in [HV09,Jha11], which propose a pairing based method for checking share validity.

Other approaches for constructing public randomness beacons have been considered in [BCG15,BDF+15,LW15,BLMR14,BGM16]. Public verifiability (or

[1] In fact [Jha11] provides an alternative solution where only $O(n)$ exponentiations and a constant number of pairings are required for verification but $O(n)$ pairings are required for setup and $O(n^2)$ exponentiations in the target group of a bilinear map (more expensive than the other exponentiations performed in the source groups) are required for reconstruction.

auditability) in the context of general multiparty computation protocol has been previously considered in [BDO14,SV15].

1.2 Our Contributions

We introduce SCRAPE, a protocol that implements a publicly verifiable randomness beacon given an honest majority through a PVSS based guaranteed output delivery coin tossing protocol. Our main result lies at the core of SCRAPE: the *first* threshold PVSS scheme that only requires a *linear* number of exponentiations for sharing, verifying and reconstruction, whereas previous schemes only achieve quadratic complexity. This PVSS scheme can be instantiated both under the Decisional Diffie Hellman (DDH) assumption in the Random Oracle Model (ROM) and in the *plain model* under the Decisional Bilinear Square (DBS) assumption [HV09]. While improving on the computational complexity of previous schemes, our PVSS scheme retains a similarly low communication overhead, making it fit for applications with large amounts of users. We remark that our new PVSS schemes can also be used to improve the performance of [SJK+16].

Model: As in previous works [BDO14], we assume that the parties can use a "public bulletin board" to publish information that will be used for posterior verification. In fact, in the applications we are interested in, a *ledger* where messages can be posted for posterior verification is readily available, since the Bitcoin Backbone protocol itself implements such a mechanism (*i.e.* the *distributed ledger* analysed in [GKL15]). Nevertheless, our protocols are compatible with any public ledger, not only with that of [GKL15].

Our Techniques: We improve on Schoenmakers' PVSS scheme [Sch99] and its variants (which require $O(nt)$ exponentiations to verify n shares) by designing a share verification procedure that only requires $O(n)$ exponentiations (or pairings). Our procedure explores the fact that sharing a secret with Shamir Secret Sharing [Sha79] is equivalent to encoding the secret (plus randomness) with a Reed Solomon error correcting code, a fact which was first observed by McEliece and Sarwate in [MS81]. Since shares from Shamir Secret Sharing form a codeword of a Reed Solomon code, computing the inner product of a share vector with a codeword from the corresponding dual code should yield 0 if the shares are correctly computed. As in [Sch99], the dealer in our scheme shares the secret using Shamir Secret sharing, encrypts the shares $s_1, \ldots, s_n$ in ciphertexts of the form $h^{sk_i s_i}$ (where h^{sk_i} is a public key and sk_i is a secret key) but also commits to all shares by computing $v_i = g^{s_i}$, where g, h are two independently chosen generators of a group where the DLOG problem is assumed to be hard. The dealer also provides evidence that the shares in the ciphertexts are the same as the shares in the commitments. To verify the validity of the shares, anybody can sample a random codeword $\boldsymbol{c}^\perp = (\boldsymbol{c}_1^\perp, \ldots, \boldsymbol{c}_n^\perp)$ of the dual code of the Reed Solomon code corresponding to the instance of Shamir Secret sharing that was used, compute the inner product of $\boldsymbol{c}^\perp$ with the share vectors in the exponents of g (by computing $\prod_i v_i^{\boldsymbol{c}_i^\perp} = g^{\sum_i s_i \boldsymbol{c}_i^\perp}$) and check that it is equal to $g^0 = 1$. If the shares are not valid, this check fails with large probability. To prove that the shares in the ciphertexts and in the commitments are the same, the dealer can either use a non-interactive zero-knowledge (NIZK) proof constructed using the Fiat-Shamir heuristic as in [Sch99] (resulting in a construction in the ROM under the DDH assumption) or have the parties do pairing based checks as in [HV09] (resulting in a construction in the plain model under the DBS assumption).

Concrete Efficiency: In the DDH based construction in the ROM, the dealer is required to compute $5n$ exponentiations in the sharing phase, while verification and reconstruction respectively require $4n$ and $5t + 3$ exponentiations (given that all n shares are verified but only t shares are used in reconstruction). In the DBS based construction in the plain model, the dealer is required to compute $2n$ exponentiations in the sharing phase, while verification requires $2n$ pairings and reconstruction requires $2n$ pairings and $t + 1$ exponentiations (given that n decrypted shares are verified but only t shares are used in reconstruction). Previous results required nt extra exponentiations in the verification phase, resulting in $n^2/2$ extra exponentiations in the randomness beacon application, which requires $t = n/2$. In the random oracle model construction, extra NIZK data is needed, amounting to a total of $2n$ group elements and $n + 1$ ring elements published by the dealer. In the construction in the plain model, the dealer saves on the NIZK data and only posts $2n$ group elements, while requiring more expensive computation (*i.e.* pairings).

2 Preliminaries

In this section, we establish notation and introduce definitions that will be used throughout the paper. We denote uniformly sampling a random element x from a finite set $\mathcal{D}$ by $x \leftarrow \mathcal{D}$. We denote vectors as $\boldsymbol{x} = (x_1, \ldots, x_n)$. We denote the inner product of two vectors $\boldsymbol{x}, \boldsymbol{y}$ as $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{1 \leq i \leq n} x_i \cdot y_i$. For the sake of notation, the integer n will always be considered to be even, so that $n/2$ is an integer. In this paper q will always denote a prime number. We denote by $\mathbb{Z}_q$ the ring of integers modulo q and by $\mathbb{G}$ a finite multiplicative group of order q. Since q is prime, $\mathbb{Z}_q$ is a finite field and $\mathbb{G}$ is a cyclic group where every element $g \neq 1$ is a generator. We denote by $\mathbb{Z}_q[x]$ the ring of polynomials in one variable with coefficients in $\mathbb{Z}_q$. We denote by $log_g e$ the discrete logarithm of an element $e \in \mathbb{G}$ with respect to generator $g \in \mathbb{G}$.

2.1 Coding Theory

We define a $[n, k, d]$ code C to be a linear error correcting code over $\mathbb{Z}_q$ of length n, dimension k and minimum distance d. Its dual code $C^\perp$ is the vector space which consists of all vectors $\boldsymbol{c} \in \mathbb{Z}_q^n$ such that $\langle \boldsymbol{c}, \boldsymbol{c}^\perp \rangle = 0$ for all $\boldsymbol{c}$ in C. The dual code $C^\perp$ of an $[n, k, d]$ code C is an $[n, n-k, d^\perp]$ code (for some $d^\perp$). In this work, we will use the following basic linear algebra fact.

Lemma 1. *If $\boldsymbol{v} \in \mathbb{Z}_q^n \setminus C$, and $\boldsymbol{c}^\perp$ is chosen uniformly at random in $C^\perp$ then the probability that $\langle \boldsymbol{v}, \boldsymbol{c}^\perp \rangle = 0$ is exactly 1/q.*

Proof. By linearity, a $\boldsymbol{c}^\perp \in C^\perp$ is orthogonal to $\boldsymbol{v}$ if only if it is also orthogonal to every vector in the code D spanned by $\boldsymbol{v}$ and C, i.e., if and only if $\boldsymbol{c}^\perp \in D^\perp$. Since $\boldsymbol{v} \notin C$, then the dimension of D is $k+1$ and hence the space $D^\perp$ has dimension $n-k-1$. Therefore if $\boldsymbol{c}^\perp$ is chosen uniformly at random in $C^\perp$ the probability that $\langle \boldsymbol{v}, \boldsymbol{c}^\perp \rangle = 0$ is

$$\frac{\#(D^\perp)}{\#(C^\perp)} = \frac{q^{n-k-1}}{q^{n-k}} = \frac{1}{q}.$$

Moreover, in this work we will always be under the assumption $n < q$ and we will use Reed-Solomon codes C of the following form

$$C = \{(p(1), p(2), \ldots, p(n)) : p(x) \in \mathbb{Z}_q[x], \deg p(x) \leq k-1\}$$

where $p(x)$ ranges over all polynomials in $\mathbb{Z}_q[x]$ of degree at most $k-1$. This is an $[n, k, n-k+1]$-code. Its dual $C^\perp$ is an $[n, n-k, k+1]$-code, which can be defined as follows

$$C^\perp = \{(v_1 f(1), v_2 f(2), \ldots, v_n f(n)) : f(x) \in \mathbb{Z}_q[x], \deg f(x) \leq n-k-1\}$$

for the coefficients $v_i = \prod_{j=1, j\neq i}^{n} \frac{1}{i-j}$.

2.2 Shamir Secret Sharing

An (n,t) threshold secret sharing scheme allows a *dealer* D to split a secret s into n shares $\boldsymbol{S} = (s_1, \ldots, s_n)$ distributed among n parties $P_1, \ldots, P_n$ such that it is possible to reconstruct the secret given t of the shares but no information at all is revealed if less shares are known. We refer to $\boldsymbol{S}$ as the *share vector* of the secret sharing scheme. The first threshold secret sharing scheme was introduced by Shamir in [Sha79]. In order to split a secret $s \in \mathbb{Z}_q$, the dealer samples $t-1$ random coefficients $c_1, \ldots, c_{t-1} \leftarrow \mathbb{Z}_q$ and constructs a polynomial $p(x) = s + c_1 x + c_2 x^2 + \cdots + c_{t-1} x^{t-1}$. The shares are computed as $s_i = p(i)$ for $1 \leq i \leq n$. A party who possesses t shares can use Lagrange interpolation to recover the polynomial $p(x)$ and thus obtain s. On the other hand, a party who knows less than t shares has no information about the secret. McEliece *et al.* first observed that sharing a secret into n shares with Shamir Secret Sharing is equivalent to encoding the message $(x, c_1, \ldots, c_{t-1})$ under a $[n, t, n-t+1]$ Reed Solomon code, implying that the share vector $\boldsymbol{S}$ is a codeword of such Reed Solomon code.

2.3 Assumptions

One of our constructions is proven in the Random Oracle Model [BR93], where it is assumed that the parties are given access to a function $H(x)$ that takes inputs of any size and returns unique uniformly random outputs of fixed size (returning the same output every time the input is the same). Such a function can be instantiated in practice by a cryptographic hash function. In this model we prove security of our protocols under the DDH assumption, that states given g, g^{α}, g^{β} it is hard for a PPT adversary to distinguish between $g^{\alpha\beta}$ from g^{r}, where g is a generator of a group $\mathbb{G}$ of order q and $\alpha, \beta, r \leftarrow \mathbb{Z}_q$.

We assume efficient non-degenerate bilinear groups described by $\Lambda := (q, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are groups of order q and $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a bilinear map. We use symmetric pairings to describe the construction for the sake of clarity but remark that our construction can also be easily converted to asymmetric bilinear groups [AHO16], for which state-of-the-art pairing friendly curves [BN06] for which more efficient algorithms for computing pairings [AKL+11] are known. We prove our pairings based protocol secure under the Decisional Bilinear Square (DBS) assumption [HV09] that was shown in that paper to be equivalent to the Decisional Bilinear Quotient assumption [LV08] and related to the Decisional Bilinear Diffie Hellman assumption.

Assumption 1 ***Decisional Bilinear Square (DBS)*** *[HV09]***.** *Let $\Lambda := (q, \mathbb{G}, \mathbb{G}_T, e)$ be a bilinear group. For a generator $g \in \mathbb{G}$, random values $\mu, \nu, s \leftarrow \mathbb{Z}_q$ and given $u = g^{\mu}$ and $v = g^{\nu}$, the following probability distributions are computationally indistinguishable: $D_0 = (g, u, v, T_0 = e(u, u)^{\nu})$ and $D_1 = (g, u, v, T_1 = e(u, u)^{s})$.*

Adversarial Model. We prove the security of our protocols in the stand alone setting against malicious adversaries, who may deviate from the protocol in any arbitrary way. We consider static adversaries, who have to choose which parties to corrupt before protocol execution begins.

Public Ledger and Broadcast Channel. We assume that the parties have access to a public ledger with *Liveness*, meaning that an adversary cannot prevent honest parties from adding information and agreeing on it, and *Persistence*, meaning that the information cannot be modified or removed a posteriori. It is known that such a ledger can be implemented by the Bitcoin backbone protocol assuming an honest majority, digital signatures and a Random Oracle [GKL15]. However, we remark that our protocols do not rely on any properties that are unique to the ledger of [GKL15], meaning that our constructions can also be instantiated over public ledgers in the plain model. Notice that access to a broadcast protocol is commonly assumed in multiparty protocols for an honest majority [RBO89] and that the same effect of broadcasting a messages can be achieved by writing it to the ledger. We also remark that the availability of a *public bulletin board* has been assumed in previous works on public verifiability for multiparty protocols [BDO14,SV15].

2.4 Publicly Verifiable Secret Sharing

We adopt the general model for PVSS schemes of [Sch99] and the security definitions of [RV05,HV09] (with some differences that we remark below). We consider a set of n parties $\mathcal{P} = \{P_1, \dots, P_n\}$ and a dealer D who shares a secret among all the parties in $\mathcal{P}$. We will construct schemes for (n, t)-threshold access structures, meaning that the secret is split in n shares in such a way that knowing at most $t - 1$ shares reveals no information but a collection of t shares allows for secret reconstruction. Additionally, any external verifier V can check that the D is acting honest without learning any information about the shares or the secret. A PVSS protocol has four phases described below:

- **Setup:** The dealer D generates and publishes the parameters of the scheme. Every party P_i publishes a public key pk_i and withholds the corresponding secret key sk_i.
- **Distribution:** The dealer creates shares $s_1, \dots, s_n$ for the secret s, encrypts share s_i with the public key pk_i for $i = 1, \dots, n$ and publishes these encryptions $\hat{s}_i$, together with a proof $PROOF_D$ that these are indeed encryptions of a valid sharing of some secret.
- **Verification:** In this phase, any external V (not necessarily being a participant in the protocol) can verify non-interactively, given all the public information until this point, that the values $\hat{s}_i$ are encryptions of a valid sharing of some secret.
- **Reconstruction:** This phase is divided in two.
  Decryption of the shares: This phase can be carried out by any set $\mathcal{Q}$ of t or more parties. Every party P_i in $\mathcal{Q}$ decrypts the share s_i from the ciphertext $\hat{s}_i$ by using its secret key sk_i, and publishes s_i together with a (non-interactive) zero-knowledge proof $PROOF_i$ that this value is indeed a correct decryption of $\hat{s}_i$.
  Share pooling: Any external verifier V (not necessarily being a participant in the protocol) can now execute this phase. V first checks whether the proofs $PROOF_i$ are correct. If the check passes for less than t parties in $\mathcal{Q}$ then V aborts; otherwise V applies a reconstruction procedure to the set s_i of shares corresponding to parties P_i that passed the checks.

A PVSS scheme must provide three security guarantees: Correctness, Verifiability and IND1-Secrecy. These properties are defined below:

- **Correctness:** If the dealer and all players in $\mathcal{Q}$ are honest, then all verification checks in the verification and reconstruction phases pass and the secret can be reconstructed from the information published by the players in $\mathcal{Q}$ in the reconstruction phase.
- **Verifiability:** If the check in the verification step passes, then with high probability the values $\hat{s}_i$ are encryptions of a valid sharing of some secret. Furthermore if the check in the Reconstruction phase passes then the communicated values s_i are indeed the shares of the secret distributed by the dealer.
- **IND1-Secrecy:** Prior to the reconstruction phase, the public information together with the secret keys sk_i of any set of at most $t - 1$ players gives no information about the secret. Formally this is stated as in the following indistinguishability based definition adapted from [RV05,HV09]:

Definition 1 ***Indistinguishability of secrets (IND1-secrecy).*** *We say that the PVSS is IND1-secret if for any polynomial time adversary $\mathcal{A}_{Priv}$ corrupting at most $t-1$ parties, $\mathcal{A}_{Priv}$ has negligible advantage in the following game played against a challenger.*

1. *The challenger runs the Setup phase of the PVSS as the dealer and sends all public information to $\mathcal{A}_{Priv}$. Moreover, it creates secret and public keys for all uncorrupted parties, and sends the corresponding public keys to $\mathcal{A}_{Priv}$.*
2. *$\mathcal{A}_{Priv}$ creates secret keys for the corrupted parties and sends the corresponding public keys to the challenger.*
3. *The challenger chooses values x_0 and x_1 at random in the space of secrets. Furthermore it chooses $b \leftarrow \{0,1\}$ uniformly at random. It runs the Distribution phase of the protocol with x_0 as secret. It sends $\mathcal{A}_{Priv}$ all public information generated in that phase, together with x_b.*
4. *$\mathcal{A}_{Priv}$ outputs a guess $b' \in \{0,1\}$.*

The advantage of $\mathcal{A}_{Priv}$ is defined as $|\Pr[b=b'] - 1/2|$.

The IND1-secrecy definition is the one used in [RV05,HV09], except for the fact that we do not impose any privacy requirement after the Reconstruction phase. The difference stems from the fact that in [RV05,HV09] it was required that nobody learns the secret but the parties interacting during the reconstruction, while in our random beacon application the secret must be publicly reconstructed and published. We remark that our scheme can achieve both the relaxed definition required by the random beacon application and the stronger secrecy guarantees of [RV05,HV09] (through the use of private channels between parties or through the technique of [HV09] that requires extra data to be posted to the ledger). We also remark that our schemes can achieve the stronger secrecy notion formalized as IND2-secrecy in [RV05,HV09], which allows the adversary to choose arbitrary secrets. This is done by a black box transformation to the protocols that allows for sharing arbitrary secrets instead of random ones by using the random shared secret as a "one time pad" to encrypt an arbitrary secret, which is formally proven in [RV05,HV09].

2.5 Zero-Knowledge Proofs of Discrete Logarithm Knowledge

In our construction based on the DDH assumption in the random oracle model we will need a zero-knowledge proof of knowledge of a value $\alpha \in \mathbb{Z}_q$ such that $x = g^\alpha$ and $y = h^\alpha$ given g, x, h, y. We denote this proof by $DLEQ(g, x, h, y)$. Chaum and Pedersen constructed a sigma protocol to perform this proof in [CP93], their protocol works as follows:

1. The prover computes $a_1 = g^w$ and $a_2 = h^w$ where $w \leftarrow \mathbb{Z}_q$ and sends a_1, a_2 to the verifier.
2. The verifier sends a challenge $e \leftarrow \mathbb{Z}_q$ to the prover.
3. The prover sends a response $z = w - \alpha e$ to the verifier.

4. The verifier checks that $a_1 = g^z x^e$ and $a_2 = h^z y^e$ and accepts the proof if this holds.

This proof has the properties of completeness, soundness and zero-knowledge. In our proofs, we will specifically reference the soundness property, which means that a prover cannot convince a verifier of a fake statement except with a negligible *soundness error* ϵ. Notice that this sigma protocol can be transformed into a non-interactive zero-knowledge proof of knowledge of α in the random oracle model through the Fiat-Shamir heuristic [FS87,PS96]. We remark that, as in [Sch99], we need to compute this proof in parallel for n distinct pairs of values $(x_1, y_1), \ldots, (x_n, y_n)$. In this case, a single challenge e is computed by the prover as $e = H(x_1, y_1, \ldots, x_n, y_n, a_{1,1}, a_{2,1}, \ldots, a_{1,n}, a_{2,n})$, where the values $a_{1,i}, a_{2,i}$ are computed according to x_i, y_i as described above and $H(\cdot)$ is a random oracle (that can be of course substituted by a cryptographic hash function). The proof then consists of the challenge e along with responses z_i computed according to each x_i, y_i. The verifier can check the proof by computing $a'_{1,i} = g^{z_i} x_i^e$ and $a'_{2,i} = h^{z_i} y_i^e$, and verifying that $H(x_1, y_1, \ldots, x_n, y_n, a'_{1,1}, a'_{2,i}, \ldots, a'_{1,n}, a'_{2,n}) = e$.

3 PVSS Based on the DDH Assumption in the ROM

In this section, we construct a PVSS protocol secure under the DDH assumption in the Random Oracle Model. Our general approach is similar to that of Schoenmakers [Sch99] but differs significantly in the procedure used for share verification, which represents the main overhead in Schoenmakers' scheme.

3.1 Security Analysis

Notice that the Setup and Reconstruction phases are exactly equal to those of [Sch99], while our protocol differs in the Distribution and Verification phases, where we apply our new technique. The key observation is that maliciously generated encrypted shares $\hat{s}_1, \ldots, \hat{s}_n$ will only pass the verification procedure with probability $1/q$ plus the soundness error of the DLEQ proof, while $v_1, \ldots, v_n$ reveal no information about the secret h^s under the DDH assumption (by an argument similar to that of [Sch99]).

We formalize these observations below. First we consider IND1-secrecy. We remark that while we use our relaxed IND1-secrecy notion or our randomness beacon application (where no secrecy is preserved after reconstruction), Protocol π_{DDH} achieves the original stronger IND1-secrecy notion of [RV05,HV09] (where secrecy against parties outside the qualified set is guaranteed even after the reconstruction) if the reconstruction is carried out through private channels between the parties in the qualified set.

Theorem 1. *Under the decisional Diffie-Hellman assumption, the protocol π_{DDH} is IND1-secret against a static PPT adversary (Fig. 1).*

Protocol π_{DDH}

Let g and h be two independently chosen generators of a group $\mathbb{G}_q$ of order q. Let $H(\cdot)$ be a random oracle. Let C be the linear error correcting code corresponding to the (n, t)-threshold Shamir Secret Sharing scheme and let $C^{\perp}$ be its dual code. Protocol π_{DDH} is run between n parties $P_1, \ldots, P_n$, a dealer D and an external verifier V (in fact any number of external verifiers) who have access to a public ledger where they can post information for later verification. The protocol proceeds as follows:

1. **Setup:** Party P_i generates a secret key $sk_i \leftarrow \mathbb{Z}_q$, a public key $pk_i = h^{sk_i}$ and registers the public key pk_i by posting it to the public ledger, for $1 \leq i \leq n$.
2. **Distribution** The dealer D first samples $s \leftarrow \mathbb{Z}_q$. The secret is defined to be $S = h^s$. D chooses $t-1$ coefficients $c_1, \ldots, c_{t-1} \leftarrow \mathbb{Z}_q$. D constructs a polynomial $p(x) = s + c_1 x + c_2 x^2 + \cdots + c_{t-1} x^{t-1}$ and computes shares $s_i = p(i)$ for $1 \leq i \leq n$. D encrypts the shares as $\hat{s}_i = pk_i^{s_i}$, computes commitments $v_i = g^{s_i}$ and computes $DLEQ(g, v_i, pk_i, \hat{s}_i)$, for $1 \leq i \leq n$ from a single challenge e as described in Section 2.5, obtaining $(e, z_1, \ldots, z_n)$. D publishes in the public ledger the encrypted shares $(\hat{s}_1, \ldots, \hat{s}_n)$ along with $PROOF_D = (v_1, \ldots, v_n, e, z_1, \ldots, z_n)$.
3. **Verification:** The verifier V first checks that the $DLEQ(g, v_i, pk_i, \hat{s}_i)$ provided by D is valid as described in Section 2.5. If the proof is valid, V samples a random codeword $\boldsymbol{c}^{\perp} = (\boldsymbol{c}_1^{\perp}, \ldots, \boldsymbol{c}_n^{\perp})$ of the dual code $C^{\perp}$ corresponding to the instance of Shamir's (n, t)-threshold secret sharing used by D and considers the shares valid if and only if the following expression is true:
$$\prod_{i=1}^{n} v_i^{\boldsymbol{c}_i^{\perp}} = 1.$$
4. **Reconstruction:** If a set of t or more parties $\mathcal{Q}$ wishes to reconstruct the secret, each party $P_i \in \mathcal{Q}$ starts by publishing in the public ledger its decrypted share $\tilde{s}_i = \hat{s}_i^{\frac{1}{sk_i}} = h^{s_i}$ and $PROOF_i = DLEQ(h, pk_i, \tilde{s}_i, \hat{s}_i)$ (showing that the decrypted share $\tilde{s}_i$ corresponds to $\hat{s}_i$). Once every party in $\mathcal{Q}$ publishes their decrypted shares and $PROOF_i$, they first verify that the proofs are valid and, if this check succeeds, reconstruct the secret $S = h^s$ by Lagrange interpolation:
$$\prod_{P_i \in \mathcal{Q}} (\tilde{s}_i)^{\lambda_i} = \prod_{P_i \in \mathcal{Q}} h^{p(i)\lambda_i} = h^{p(0)} = h^s,$$
where $\lambda_i = \prod_{j \neq i} \frac{j}{j-i}$ are the Lagrange coefficients.

Fig. 1. Protocol π_{DDH}

Proof. We show that, if there exists an adversary $\mathcal{A}_{\text{Priv}}$ which can break the IND1-secrecy property of protocol π_{DDH}, then there exists an adversary $\mathcal{A}_{\text{DDH}}$ which can use $\mathcal{A}_{\text{Priv}}$ to break the decisional Diffie-Hellman assumption with the same advantage. Without loss of generality we assume $\mathcal{A}_{\text{Priv}}$ corrupts the $t-1$ first parties.

Let $(g, g^{\alpha}, g^{\beta}, g^{\gamma})$ be an instance of the DDH problem. Obviously if $\alpha = 0$ or $\beta = 0$ then the problem is trivial, so we assume these values are nonzero. Now $\mathcal{A}_{\text{DDH}}$, using $\mathcal{A}_{\text{Priv}}$, can simulate an IND1 game as follows:

1. The challenger sets $h = g^{\alpha}$ and runs the Setup phase of π_{DDH}. For $t \leq i \leq n$, $\mathcal{A}_{\text{DDH}}$ selects uniformly random values $u_i \leftarrow \mathbb{Z}_p$ (these can be thought of implicitly defining sk_i as $sk_i = u_i/\alpha$) and sends the values $pk_i = g^{u_i}$ to $\mathcal{A}_{\text{Priv}}$.
2. For $1 \leq i \leq t-1$, $\mathcal{A}_{\text{Priv}}$ chooses uniformly random values $sk_i \leftarrow \mathbb{Z}_q$ and sets $pk_i = h^{sk_i}$ and sends this to the challenger.
3. For $1 \leq i \leq t-1$, the challenger chooses uniformly random values $s_i \leftarrow \mathbb{G}_q$ and sets $v_i = g^{s_i}$ and $\hat{s}_i = pk_i^{s_i}$.
   For $t \leq i \leq n$, it generates the values $v_i = g^{p(i)}$ where $p(x)$ is the unique polynomial of degree at most t determined by $p(0) = \beta$ and $p(i) = s_i$ for $i = 1, \ldots, t-1$. Note that $\mathcal{A}_{\text{DDH}}$ does not know β, but it does know $g^{\beta} = g^{p(0)}$ and $g^{s_i} = g^{p(i)}$ for $1 \leq i \leq t-1$, so it can use Lagrange interpolation in the exponent to compute the adequate v_i. It also creates the values $\hat{s}_i = v_i^{u_i}$. Note that then $\hat{s}_i = g^{u_i \cdot p(i)} = pk_i^{p(i)}$. From all the computed values, the challenger now creates the DLEQ proofs as the dealer does in the PVSS protocol. Finally it sends all this information together with the value g^{γ} (which plays the role of x_b in the IND game) to $\mathcal{A}_{\text{Priv}}$.
4. $\mathcal{A}_{\text{Priv}}$ makes a guess b'.

If $b' = 0$, $\mathcal{A}_{\text{DDH}}$ guesses that $\gamma = \alpha \cdot \beta$. If $b' = 1$, $\mathcal{A}_{\text{DDH}}$ guesses that γ is a random element in $\mathbb{Z}_p$.

The information that $\mathcal{A}_{\text{Priv}}$ receives in step 3. is distributed exactly like a sharing of the value $h^{\beta} = g^{\alpha \cdot \beta}$ with the PVSS. Consequently, $\gamma = \alpha \cdot \beta$ if and only if the value g^{γ} sent to $\mathcal{A}_{\text{Priv}}$ is the secret shared by the PVSS. It is now easy to see that the guessing advantage of $\mathcal{A}_{\text{DDH}}$ is the same as the advantage of $\mathcal{A}_{\text{Priv}}$.

The following two theorems guarantee the verifiability property of π_{DDH}.

Theorem 2. *If the dealer does not construct values $(v_i, \hat{s}_i)$ of the right form in the Distribution phase (i.e. either $log_g v_i \neq log_{pk_i} \hat{s}_i$ for some i, or $log_g v_i = log_{pk_i} \hat{s}_i = s_i$ for all i but the values s_i do not constitute a valid sharing of some secret in $\mathbb{Z}_q$ with the (n, t)-threshold Shamir secret sharing scheme), then this is detected in the verification step with probability at least $1 - \epsilon - 1/q$, where ϵ is the soundness error of the proof DLEQ.*

Proof. If the verification of DLEQ passes, then we have that, except with probability ϵ, for every $1 \leq i \leq n$, there exists s_i with $v_i = g^{s_i}$ and $\hat{s}_i = pk_i^{s_i}$. Now the values s_i are a valid sharing with the (n, t)-threshold Shamir secret sharing scheme if and only if the vector $\boldsymbol{v} = (s_1, \ldots, s_n) \in C$. Suppose that $(s_1, \ldots, s_n) \notin C$. Then by Lemma 1, since $\boldsymbol{c}^{\perp}$ is sampled uniformly at random then $\langle \boldsymbol{v}, \boldsymbol{c}^{\perp} \rangle \neq 0$ except with probability $1/q$. But then $\prod_{i=1}^{n} v_i^{\boldsymbol{c}_i^{\perp}} = \prod_{i=1}^{n} g^{s_i \cdot \boldsymbol{c}_i^{\perp}} = g^{\langle \boldsymbol{v}, \boldsymbol{c}^{\perp} \rangle} \neq 1$. Hence if the values s_i are not a valid Shamir sharing, then the check fails with probability $1 - 1/q$.

Theorem 3. *If a party in $\mathcal{Q}$ communicates an erroneous decryption share $\tilde{s}_i$ in the Reconstruction phase, then this is detected by the verifier with probability $1 - \epsilon$, where ϵ is the soundness error of the DLEQ proof.*

Proof. This is straightforward by definition since an adversary that succeeds in providing a DLEQ proof for an invalid decrypted share breaks DLEQ's soundness property.

4 PVSS Based on Pairings in the Plain Model

In this section, we construct a PVSS scheme based on the DBS assumption in the plain model (without requiring random oracles). This scheme uses the techniques of [HV09] to eliminate the need for the random oracle based NIZKs and instead use pairings to check that the encrypted shares $\hat{s}_i$ correspond to the committed shares v_i and, later on, check that the decrypted shares correspond to $\hat{s}_i$. We use the same information theoretical verification procedure as in the DDH based scheme. The protocol is described in Fig. 2.

4.1 Security Analysis

As in the DDH based protocol, notice that the Setup and Reconstruction phases are exactly equal to those of [HV09], while our protocol differs in the Distribution and Verification phases. Maliciously generated encrypted shares $\hat{s}_1, \ldots, \hat{s}_n$ will only pass the verification procedure with probability of $1/q$, while $v_1, \ldots, v_n$ again reveal no information about the secret $e(h,h)^s$ but this time under the BDS assumption. Again we remark that Protocol π_{DBS} achieves the original stronger IND1-secrecy notion of [RV05,HV09] (where secrecy against parties outside the qualified set is guaranteed even after the reconstruction) if the reconstruction is carried out through private channels between the parties in the qualified set. The proofs of Theorems 4 and 5 are similar to those of Theorems 1 and 2, respectively, and are presented in the full version of this work [CD17].

Theorem 4. *Under the DBS assumption, protocol π_{DBS} is IND1-secret against a static PPT adversary.*

Theorem 5. *If the dealer does not construct values $(v_i, \hat{s}_i)$ of the right form in the Distribution phase (i.e. either $log_g v_i \neq log_{pk_i} \hat{s}_i$ for some i, or $log_g v_i = log_{pk_i} \hat{s}_i = s_i$ for all $1 \leq i \leq n$ but the values s_i do not constitute a valid sharing of some secret in $\mathbb{Z}_q$ with the (n,t)-threshold Shamir secret sharing scheme), then this is detected in the verification step with probability at least $1 - 1/q$.*

Theorem 6. *If a party in $\mathcal{Q}$ communicates an erroneous decryption share $\tilde{s}_i$ in the Reconstruction phase, then this is detected by the verifier with probability 1.*

Proof. If $\tilde{s}_i = h^a$ with $a \neq s_i$ then $e(pk_i, \tilde{s}_i) = e(pk_i, h)^a \neq e(pk_i, h)^{s_i} = e(\hat{s}_i, h)$.

Protocol π_{DBS}

Let $\Lambda := (q, \mathbb{G}, \mathbb{G}_T, e)$ be a description of a bilinear group and g, h be two independently chosen generators of $\mathbb{G}$. Let C be the linear error correcting code equivalent to the (n, t)-threshold Shamir Secret Sharing scheme and $C^{\perp}$ its dual code. Protocol π_{DBS} is run between n parties $P_1, \ldots, P_n$, a dealer D and an external verifier V (in fact any number of external verifiers) who have access to a public ledger where they can post information for later verification. The protocol proceeds as follows:

1. **Setup:** Party P_i generates a secret key $sk_i \leftarrow \mathbb{Z}_q$, a public key $pk_i = h^{sk_i}$ and registers the public key pk_i by posting it to the public ledger, for $1 \leq i \leq n$.
2. **Distribution** The dealer D first samples $s \leftarrow \mathbb{Z}_q$. The secret is defined to be $S = e(h, h)^s$. D chooses $t-1$ coefficients $c_1, \ldots, c_{t-1} \leftarrow \mathbb{Z}_q$. D constructs a polynomial $p(x) = s + c_1 x + c_2 x^2 + \cdots + c_{t-1} x^{t-1}$ and computes shares $s_i = p(i)$ for $1 \leq i \leq n$. D encrypts the shares as $\hat{s}_i = pk_i^{s_i}$ and computes commitments $v_i = g^{s_i}$, for $1 \leq i \leq n$. D posts in the public ledger the encrypted shares $\hat{s}_1, \ldots, \hat{s}_n$ and $PROOF_D = (v_1, \ldots, v_n)$.
3. **Verification:** The verifier V first checks that $e(\hat{s}_i, g) = e(pk_i, v_i)$, for $1 \leq i \leq n$. If this check succeeds, V samples a random codeword $\boldsymbol{c}^{\perp} = (\boldsymbol{c}_1^{\perp}, \ldots, \boldsymbol{c}_n^{\perp})$ of the dual code $C^{\perp}$ corresponding to the instance of Shamir's (n, t)-threshold secret sharing used by D and considers the shares valid if and only if the following expression is true:
$$\prod_{i=1}^{n} v_i^{\boldsymbol{c}_i^{\perp}} = 1.$$
4. **Reconstruction:** If a set of t or more parties $\mathcal{Q}$ wishes to reconstruct the secret, each party $P_i \in \mathcal{Q}$ starts publishing in the public ledger its decrypted share $\tilde{s}_i = \hat{s}_i^{\frac{1}{sk_i}} = h^{s_i}$ (here $PROOF_i$ is an empty string). Once every party in $\mathcal{Q}$ publishes their decrypted shares, they first verify that $e(pk_i, \tilde{s}_i) = e(\hat{s}_i, h)$ for every $P_i \in \mathcal{Q}$. If this check succeeds, they reconstruct the value h^s by Lagrange interpolation:
$$\prod_{P_i \in \mathcal{Q}} (\tilde{s}_i)^{\lambda_i} = \prod_{P_i \in \mathcal{Q}} h^{p(i)\lambda_i} = h^{p(0)} = h^s,$$
where $\lambda_i = \prod_{j \neq i} \frac{j}{j-i}$ are the Lagrange coefficients. The secret is then computed as $S = e(h^s, h)$.

Fig. 2. Protocol π_{DBS}

5 Building the SCRAPE Randomness Beacon

Publicly verifiable secret sharing schemes have a multitude of applications as discussed in [Sch99], among them universally verifiable elections, threshold versions of El Gamal encryption and threshold software key escrow. However, we are specially interested in constructing SCRAPE, a protocol that implements a distributed randomness beacon that is guaranteed to be secure given an honest majority, a PVSS scheme and a public ledger. SCRAPE is basically a coin tossing protocol with guaranteed output delivery (G.O.D.), meaning that an adversary

Protocol π_{SCRAPE}

Protocol π_{DDH} is run between n parties $P_1, \ldots, P_n$ who have access to a public ledger where they can post information for later verification. A PVSS protocol is used as a sub protocol and it is assumed that the Setup phase is already done and the public keys pk_i of each party P_i are already registered in the ledger.The protocol proceeds as follows:

1. **Commit:** For $1 \leq j \leq n$, party P_j executes the Distribution phase of the PVSS sub protocol as the Dealer with threshold $t = \frac{n}{2}$, publishing the encrypted shares $\hat{s}_1^j, \ldots, \hat{s}_n^j$ and the verification information $PROOF_D^j$ on the public ledger, and also learning the random secret h^{s^j} and s^j. P_j also publishes a commitment to the secret exponent $\mathsf{Com}(s^j, r_j)$ (with fresh randomness $r_j \leftarrow \mathbb{Z}_q$), for $1 \leq j \leq n$.
2. **Reveal:** For every set of encrypted shares $\hat{s}_1^j, \ldots, \hat{s}_n^j$ and the verification information $PROOF_D^j$ published in the public ledger, all parties run the Verification phase of the PVSS sub protocol. Let $\mathcal{C}$ be the set of all parties who published commitments and valid shares. Once $\frac{n}{2}$ parties have posted their commitments and valid shares on the ledger, party P_j opens its commitment, posting $\mathsf{Open}(s^j, r_j)$ on the ledger, for $j \in \mathcal{C}$.
3. **Recovery:** For every party $P_a \in \mathcal{C}$ that does not publish $\mathsf{Open}(s^a, r_a)$ in the Reveal phase, party P_j runs the Reconstruction phase of the PVSS protocol posting $\tilde{s}_j^a$ and $PROOF_j^a$ to the public ledger, for $1 \leq j \leq n$. Once $\frac{n}{2}$ valid decrypted shares are published, every party reconstructs h^{s^a}.

The final randomness is computed as $\rho = \prod_{j \in \mathcal{C}} h^{s^j}$.

Fig. 3. Protocol π_{SCRAPE}

cannot prevent honest parties from obtaining a correct output (*e.g.* by aborting before the execution is finished). Moreover, SCRAPE is publicly verifiable, meaning that anybody can analyse past (and current) protocol transcripts to verify that the protocol is being correctly executed. The reason we aim at guaranteed output delivery is twofold: 1. Protecting against particularly adversarial behavior and 2. Tolerating non-byzantine failures in users after the commitment phase (*e.g.* power outage). When used to bootstrap blockchain based consensus protocols such as in [KKR+16], π_{SCRAPE} has to tolerate adversaries that can force a temporary loss of consensus making the users end up with conflicting random outputs or temporarily "disconnecting" users from the network or public ledger. We follow the general approach of [RBO89] to obtain guaranteed output delivery based on verifiable secret sharing, building on our PVSS schemes to achieve public verifiability for the final coin tossing protocol. More specifically, we use our PVSS protocols to instantiate the construction of [KKR+16], which proposed to combine a PVSS scheme with a public ledger to obtain publicly verifiable G.O.D. coin tossing. The protocol is described in Fig. 3. The security of π_{SCRAPE} follows from the security of the general construction of [KKR+16] and the security of our protocols that was proven in the previous sections. We refer the reader to [KKR+16] for a detailed discussion on the general protocol.

6 Concrete Complexity and Experiments

In this section, we discuss the concrete computational and communication complexity of our schemes, comparing them with previous work. We first start by discussing the computational complexity of our PVSS protocols, which is compared to that of the protocols of [Sch99, HV09] in terms of numbers of exponentiations and pairings required for each phase in Table 1. Notice that for our randomness beacon application we need $t = n/2$, which translates into an extra overhead of $n^2/2$ exponentiations required for the verification phase of previous protocols. Our protocols eliminate this quadratic overhead, resulting in much better scalability. For example, if 10000 users run SCRAPE based on [Sch99], 50004000 exponentiations are required in the verification phase, while our instantiation

Table 1. Concrete computational complexity in terms of numbers of exponentiations (Exp.) and pairings (Pair.) needed for each phase, considering that n shares are generated and t shares are used in reconstruction.

	Distribution	Verification		Reconstruction	
	Exp.	Exp.	Pair.	Exp.	Pair.
[HV09]	$n+t$	nt	$2n$	$t+1$	$2t+1$
Protocol π_{DBS}	$2n$	n	$2n$	$t+1$	$2t+1$
[Sch99]	$3n+t$	$nt+4n$	-	$5t+3$	-
Protocol π_{DDH}	$4n$	$5n$	-	$5t+3$	-

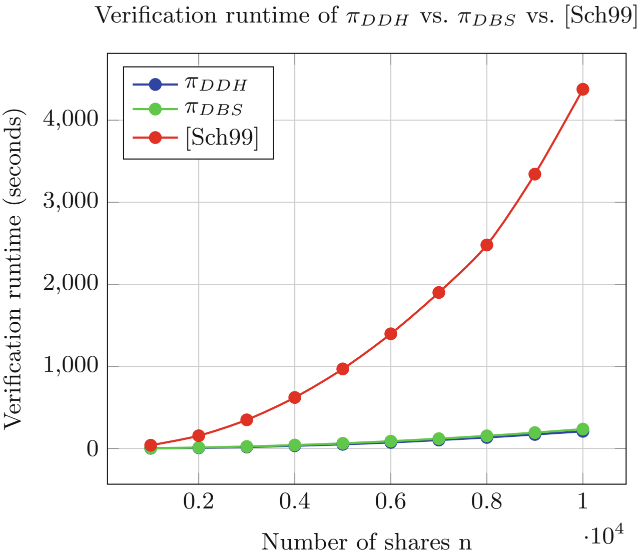


Fig. 4. Execution time of the verification phases of π_{DDH} vs. π_{DBS} vs. Schoenmakers' PVSS [Sch99] for a number of shares n from 1000 to 10000 and threshold $t = \frac{n}{2}$.

of SCRAPE would require only 50000 exponentiations, achieving a performance gain of over 100 times. In terms of communication complexity, our protocols match previous results [Sch99,HV09] in the reconstruction phase while requiring $2n$ group elements to be published in the distribution phase. In the randomness beacon application, this represents an overhead of only $0.5n$ in relation to [Sch99,HV09]. In order to evaluate the concrete performance of our proposed protocols, we have conducted experiments with implementations of π_{DDH}, π_{DBS} and the PVSS scheme of [Sch99], all written in Haskell. The experiments were run on a single core of a Linux machine with a Intel(R) Core(TM) i7-3770K CPU @ 3.50 GHz and 32 GB of RAM. Figure 4 compares the runtime of the verification phases of π_{DDH} and [Sch99], which represents the main improvement of π_{DDH}. Further discussion on concrete complexity and experimental results can be found in the full version of this work [CD17].

Acknowledgements. We thank Vincent Hanquez and Andrzej Rybczak for implementing π_{DDH} and π_{DBS}, respectively.

References

Adi08. Adida, B.: Helios: web-based open-audit voting. In: van Oorschot, P.C. (ed.) Proceedings of 17th USENIX Security Symposium, 28 July–1 August 2008, San Jose, CA, USA, pp. 335–348. USENIX Association (2008)

AHO16. Abe, M., Hoshino, F., Ohkubo, M.: Design in type-I, run in type-III: fast and scalable bilinear-type conversion using integer programming. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 387–415. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53015-3_14

AKL+11. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López, J.: Faster explicit formulas for computing pairings over ordinary curves. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 48–68. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20465-4_5

B+14. Buterin, V., et al.: A next-generation smart contract and decentralized application platform. White paper (2014)

BCG15. Bonneau, J., Clark, J., Goldfeder, S.: On bitcoin as a public randomness source. Cryptology ePrint Archive, Report 2015/1015 (2015). http://eprint.iacr.org/2015/1015

BDF+15. Baignères, T., Delerablée, C., Finiasz, M., Goubin, L., Lepoint, T., Rivain, M.: Trap me if you can - million dollar curve. Cryptology ePrint Archive, Report 2015/1249 (2015). http://eprint.iacr.org/2015/1249

BDO14. Baum, C., Damgård, I., Orlandi, C.: Publicly auditable secure multiparty computation. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 175–196. Springer, Cham (2014). doi:10.1007/978-3-319-10879-7_11

BGM16. Bentov, I., Gabizon, A., Mizrahi, A.: Cryptocurrencies without proof of work. In: Clark et al. [CMR+16], pp. 142–157 (2016)

BLMR14. Bentov, I., Lee, C., Mizrahi, A., Rosenfeld, M.: Proof of activity: extending bitcoin's proof of work via proof of stake [extended abstract]y. SIGMETRICS Perform. Eval. Rev. **42**(3), 34–37 (2014)

BLN16. Bernstein, D.J., Lange, T., Niederhagen, R.: Dual EC: a standardized back door. In: Ryan, P.Y.A., Naccache, D., Quisquater, J.-J. (eds.) The New Codebreakers. LNCS, vol. 9100, pp. 256–281. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49301-4_17

Blu81. Blum, M.: Coin flipping by telephone. In: Gersho, A. (ed.) CRYPTO 1981, vol. ECE report 82-04, pp. 11–15. U.C. Santa Barbara, Department of Electrical and Computer Engineering (1981)

BN06. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006). doi:10.1007/11693383_22

BR93. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, pp. 62–73. ACM Press, November 1993

BT99. Boudot, F., Traoré, J.: Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In: Varadharajan, V., Mu, Y. (eds.) ICICS 1999. LNCS, vol. 1726, pp. 87–102. Springer, Heidelberg (1999). doi:10.1007/978-3-540-47942-0_8

CD17. Cascudo, I., David, B.: Scrape: scalable randomness attested by public entities. Cryptology ePrint Archive, Report 2017/216 (2017). http://eprint.iacr.org/2017/216

CDE+16. Croman, K., Decker, C., Eyal, I., Gencer, A.E., Juels, A., Kosba, A.E., Miller, A., Saxena, P., Shi, E., Sirer, E.G., Song, D., Wattenhofer, R.: On scaling decentralized blockchains - (a position paper). In: Clark et al. [CMR+16], pp. 106–125 (2016)

CGMA85. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In: 26th FOCS, pp. 383–395. IEEE Computer Society Press, October 1985

Cle86. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: Hartmanis, J. (ed.) Proceedings of 18th Annual ACM Symposium on Theory of Computing, 28–30 May 1986, Berkeley, California, USA, pp. 364–369. ACM (1986)

CMR+16. Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D.S., Brenner, M., Rohloff, K. (eds.): FC 2016 Workshops. LNCS, vol. 9604. Springer, Heidelberg (2016)

CP93. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993). doi:10.1007/3-540-48071-4_7

DMS04. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: Proceedings of 13th Conference on USENIX Security Symposium, SSYM 2004, Berkeley, CA, USA, vol. 13, p. 21. USENIX Association (2004)

DPSW16. Degabriele, J.P., Paterson, K.G., Schuldt, J.C.N., Woodage, J.: Backdoors in pseudorandom number generators: possibility and impossibility results. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 403–432. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53018-4_15

Fel87. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th FOCS, pp. 427–437. IEEE Computer Society Press, October 1987

FO98. Fujisaki, E., Okamoto, T.: A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 32–46. Springer, Heidelberg (1998). doi:10.1007/BFb0054115

FS87. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). doi:10.1007/3-540-47721-7_12

GKL15. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_10

GRFJ14. Ghosh, M., Richardson, M., Ford, B., Jansen, R.: A torpath to torcoin: proof-of-bandwidth altcoins for compensating relays. Technical report, DTIC Document (2014)

HV09. Heidarvand, S., Villar, J.L.: Public verifiability from pairings in secret sharing schemes. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 294–308. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04159-4_19

Jha11. Jhanwar, M.P.: A practical (non-interactive) publicly verifiable secret sharing scheme. In: Bao, F., Weng, J. (eds.) ISPEC 2011. LNCS, vol. 6672, pp. 273–287. Springer, Heidelberg (2011). doi:10.1007/978-3-642-21031-0_21

JVSN14. Jhanwar, M.P., Venkateswarlu, A., Safavi-Naini, R.: Ayineedi venkateswarlu, and reihaneh safavi-naini. paillier-based publicly verifiable (non-interactive) secret sharing. Des. Codes Crypt. **73**(2), 529–546 (2014)

KKR+16. Kiayias, A., Konstantinou, I., Russell, A., David, B., Oliynykov, R.: A provably secure proof-of-stake blockchain protocol. Cryptology ePrint Archive, Report 2016/889 (2016). http://eprint.iacr.org/2016/889

KMS+16. Kosba, A.E., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. In: 2016 IEEE Symposium on Security and Privacy, pp. 839–858. IEEE Computer Society Press, May 2016

LV08. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78440-1_21

LW15. Lenstra, A.K., Wesolowski, B.: A random zoo: sloth, unicorn, and trx. Cryptology ePrint Archive, Report 2015/366 (2015). http://eprint.iacr.org/2015/366

Mau96. Maurer, U.M. (ed.): EUROCRYPT 1996. LNCS, vol. 1070. Springer, Heidelberg (1996). doi:10.1007/3-540-68339-9

MS81. McEliece, R.J., Sarwate, D.V.: On sharing secrets and reed-solomon codes. Commun. ACM **24**(9), 583–584 (1981)

Nak08. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)

Pai99. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). doi:10.1007/3-540-48910-X_16

PS96. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer [Mau96], pp. 387–398 (1996)

Rab83. Rabin, M.O.: Transaction protection by beacons. J. Comput. Syst. Sci. **27**(2), 256–267 (1983)

RBO89. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: 21st ACM STOC, pp. 73–85. ACM Press. May 1989

RV05. Ruiz, A., Villar, J.L.: Publicly verifiable secret sharing from Paillier's cryptosystem. In: Wolf, C., Lucks, S., Yau, P.-W. (eds.) WEWoRC 2005 - Western European Workshop on Research in Cryptology. Leuven, Belgium, 5–7 July 2005. LNI, vol. 74, pp. 98–108. GI (2005)

Sch99. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 148–164. Springer, Heidelberg (1999). doi:10.1007/3-540-48405-1_10

Sha79. Shamir, A.: How to share a secret. Commun. Assoc. Comput. Mach. **22**(11), 612–613 (1979)

SJK+16. Syta, E., Jovanovic, P., Kokoris Kogias, E., Gailly, N., Gasser, L., Khoffi, I., Fischer, M.J., Ford, B.: Scalable bias-resistant distributed randomness. Cryptology ePrint Archive, Report 2016/1067 (2016). (To appear at IEEE Security & Privacy 2017). http://eprint.iacr.org/2016/1067

Sta96. Stadler, M.: Publicly verifiable secret sharing. In: Maurer [Mau96], pp. 190–199 (1996)

SV15. Schoenmakers, B., Veeningen, M.: Universally verifiable multiparty computation from threshold homomorphic cryptosystems. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 3–22. Springer, Cham (2015). doi:10.1007/978-3-319-28166-7_1

vdHLZZ15. van den Hooff, J., Lazar, D., Zaharia, M., Zeldovich, N.: Vuvuzela: scalable private messaging resistant to traffic analysis. In: Proceedings of 25th Symposium on Operating Systems Principles, SOSP 2015, pp. 137–152. ACM, New York (2015)

WCGFJ12. Wolinsky, D.I., Corrigan-Gibbs, H., Ford, B., Johnson, A.: Dissent in numbers: making strong anonymity scale. In: Proceedings of 10th USENIX Conference on Operating Systems Design and Implementation, OSDI 2012, pp. 179–192. USENIX Association, Berkeley (2012)

cMix: Mixing with Minimal Real-Time Asymmetric Cryptographic Operations

David Chaum[1], Debajyoti Das[2](✉), Farid Javani[3], Aniket Kate[2], Anna Krasnova[4], Joeri De Ruiter[4], and Alan T. Sherman[3]

[1] Voting Systems Institute, Los Angeles, USA
das48@purdue.edu
[2] Purdue University, West Lafayette, USA
[3] Cyber Defense Lab, UMBC, Baltimore, USA
sherman@umbc.edu
[4] Radboud University, Nijmegen, Netherlands

Abstract. We introduce cMix, a new approach to anonymous communications. Through a precomputation, the core cMix protocol eliminates all expensive real-time public-key operations—at the senders, recipients and mixnodes—thereby decreasing real-time cryptographic latency and lowering computational costs for clients. The core real-time phase performs only a few fast modular multiplications.

In these times of surveillance and extensive profiling there is a great need for an anonymous communication system that resists global attackers. One widely recognized solution to the challenge of traffic analysis is a mixnet, which anonymizes a batch of messages by sending the batch through a fixed cascade of mixnodes. Mixnets can offer excellent privacy guarantees, including unlinkability of sender and receiver, and resistance to many traffic-analysis attacks that undermine many other approaches including onion routing. Existing mixnet designs, however, suffer from high latency in part because of the need for real-time public-key operations. Precomputation greatly improves the real-time performance of cMix, while its fixed cascade of mixnodes yields the strong anonymity guarantees of mixnets. cMix is unique in not requiring any real-time public-key operations by users. Consequently, cMix is the first mixing suitable for low latency chat for light-weight devices.

Our presentation includes a specification of cMix, security arguments, anonymity analysis, and a performance comparison with selected other approaches. We also give benchmarks from our prototype.

1 Introduction

Digital messaging has become a significant form of human communication, yet currently such systems do not provide basic protections of untraceability and unlinkability of messages. These protections are fundamental to freedom of inquiry, to freedom of expression, and increasingly to online privacy. Grave threats to privacy exist from global adversaries who construct traffic-analysis graphs detailing who communicates with whom.

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 557–578, 2017.
DOI: 10.1007/978-3-319-61204-1_28

We introduce cMix, a new approach to anonymous communications. cMix is a new variant of fixed-cascade mixing networks (mixnets). cMix uses a precomputation phase to avoid computationally intensive public-key cryptographic operations in its core real-time protocol. Senders/clients participate only in the real-time phase. Thus, senders never perform any public-key operations. cMix has drastically lower real-time cryptographic latency than do all other mixnets. Through its use of precomputation, and through its novel key management, cMix is markedly different from all previous mixnets. Due to its lack of public-key operations in its core real-time phase, it is very well suited for applications running on light-weight clients, including chat messaging systems running on smartphones, and for applications on low-power devices.

To provide anonymity online, an alternative approach to mixnets is onion routing, such as implemented in the widely used system TOR [46]. Onion-routing systems, however, have limitations on the level of anonymity achievable: most significantly, because they route different sessions of messages along different paths and they do not perform random permutations of messages, they are vulnerable to a variety of traffic-analysis attacks—for example [12,43], as well as intersection attacks [6,13,14].

By contrast, mixnets hold fundamentally greater promise to achieve higher levels of anonymity than can onion-routing systems because mixnets are resilient to traffic-analysis attacks. Specifically, since all mixnet messages travel through the same fixed cascade of mixnodes, observing the communication paths of messages within a mixnet is not useful to the adversary. Also, mixnets can process larger batches of messages than can onion-routing systems, which is important because the batch size is the size of the anonymity set. Using a fixed cascade achieves resilience against intersection attacks [6]. The main disadvantage of current mixnets is their performance, which is throttled by their use of real-time public-key cryptographic operations, which are much slower than symmetric-key operations.

cMix is a practical solution to the cryptographic latency problem. It also provides resistance to traffic analysis and intersection attacks, as do other fixed-route mixnet designs. cMix scales linearly in number of users. Our prototype implementation on Android clients demonstrates the practicality of cMix.

The main novel and significant contribution of cMix is that, by using precomputation, cMix eliminates all expensive real-time public-key operations by sender, receiver, and mixnodes in its core protocol. In cMix, the clients never perform any public-key operations when sending messages. By splitting the computational load over a computationally intensive precomputation phase and a fast real-time phase, cMix's approach can also increase throughput, when demand is non-uniform. No other mixnet has achieved these performance characteristics. Thus, cMix greatly improves real-time performance (especially latency) over all existing traditional mixnets and all re-encryption mixnets, while enjoying their strong anonymity properties.

Our main contributions are the design, preliminary analysis, and prototype implementation of cMix, a new mixnet variant that, through precomputation, achieves lower real-time computational latency than do all existing mixnets (traditional and re-encryption), while still benefiting from the strong anonymity properties of mixnets over onion-routing systems.

In the rest of this paper we review related work, provide an overview of cMix, describe the core cMix protocol, explain some protocol enhancements, provide security arguments, compare cMix's performance with that of other mixnets, present benchmarks from our prototype implementation, discuss several issues raised by cMix, and present our conclusions.

2 Related Work

We briefly review selected background and related work on mixnets, onion routing, and precomputation for mixnets.

Mix Networks. In 1981, Chaum [9] introduced the concept of mixnets (often referred to as *decryption mixnets*) and gave basic cryptographic protocols whereby messages from a set of users are relayed by a sequence of trusted intermediaries, called *mixnodes* or *mixes*. A mixnode is simply a message relay (or proxy) that accepts a batch of encrypted messages, decrypts and randomly permutes them, and sends them on their way forward. The sender in a decryption mixnet must perform a number of public-key operations equal to the number of mixnodes. The length of the encrypted message is proportional to this number, and the length of the plaintext message is restricted for performance reasons.

Hybrid mixnets allow plaintext messages to have arbitrary length, by combining asymmetric and symmetric cryptography. First proposed in 1985 by Pfitzmann and Waidner [37], hybrid mixnets share a session key in the first message, and then use a stream cipher to encrypt further messages. Various proposals based on block ciphers followed [26,33]. The recent system called Riffle [31] provides both sender and receiver anonymity by using verifiable shuffles and private information retrieval. Periodocially, a client and all servers perform verifiable shuffles to exchange session keys, which are then used for several rounds in a manner similar to that performed by other hybrid networks.

In 1994, Park et al. [36] introduced *re-encryption mixnets*. Taking advantage of homomorphic properties of El-Gamal encryption, each mixnode re-encrypts the incoming message instead of decrypting it as in the original mixnet. Doing so reduces and fixes the size of the ciphertext message traveling through the mixnet. *Universal re-encryption mixnets* [23] do not require mixnodes to know public keys for re-encryption. Because the sender encrypts each message using the public key of the receiver, only the receiver is able to read the plaintext. Consequently, unlike other mixnets, universal re-encryption mixnets provide sender anonymity only against external observers, and not against message recipients.

Precomputation mixnets introduce a precomputation phase to decrease latency during message delivery. Jakobsson [25] introduced precompution to reduce the cost of node operations in re-encryption mixnets, though client costs remain the same. Adida and Wikström [1] considered an offline/online approach to mixing. Their protocol still requires several public-key operations in the online phase, and senders have to perform public-key operations. A notable distinction of the cMix precomputation mixnet is its shifting of *all* public-key operations in its core protocol to the precomputation phase. Moreover, only the mixnodes perform these public-key operations; no sender is involved.

Onion Routing. Higher latency of traditional mixnets can be unsatisfactory for several communication scenarios, such as web search or instant messaging. Over the past several years, a significant number of *low-latency* anonymity networks have been proposed [2,3,8,11,20,28,29,34,41], and some have been extensively employed in practice [15,46].

Common to many of them is *onion routing* [21,35], a technique whereby a message is wrapped in multiple layers of encryption, forming an *onion*. A common realization of an onion-routing system is to arrange a collection of onion routers (abbreviated ORs, also called hops or nodes) that relay traffic for users of the system. Users then randomly choose a path with few edges through the network of ORs and construct a circuit—a sequence of nodes that will route traffic. After the OR circuit is constructed, each of the nodes in the circuit shares a symmetric key with the anonymous user, which key is used to encrypt the layers of future onions. Upon receiving an onion, each node decrypts one of the layers, and forwards the message to the next node. Onion routing as it typically exists can be seen as a form of three-node mixing.

Low-latency anonymous communication networks based on onion routing [18,27,32,45], such as TOR [46], are susceptible to a variety of traffic-analysis attacks. By contrast, mixnet methodology ensures that the anonymity set of a user remains the same through the communication route and makes our protocol resistant to these network-level attacks.

There are similarities between our precomputation phase, which uses public-key operations, and the circuit-construction phase of onion routing. Similarly, there are similarities between our real-time phase, which uses symmetric-key operations, and the onion wrapping and unwrapping phases.

Unlike onion routing, however, our precomputation phase requires no participation from the senders. During enrollment, each of our senders establishes a separate shared secret with each mixnode, but this key establishment is performed infrequently. Furthermore, in contrast with onion routing, our senders do not perform anonymous key agreement [3,15,20,29] using a telescoping approach or layered public-key encryption; they can establish these keys using a Diffie-Hellman key exchange. These differences result in a significant reduction in the computation that the users need to perform and make our system more attractive to energy-constrained devices such as smartphones.

3 System Overview

Before defining cMix's core protocol, we first explain our architecture and communication model, adversarial model, and security goals.

3.1 Architecture and Communication Model

cMix is a new mixnet protocol that provides anonymous communication for its users (senders and receivers). The main goal is to ensure *unlinkability* of messages entering and leaving the system, though it is known which users are active at any given moment.

cMix has n mixnodes that compose a *fixed cascade*: all nodes are organized in a fixed order from the first node to the last. Within the cMix system this order can be systematically changed and rotated, without affecting users in any way. Any message sent by a user is forwarded through all n servers. As with any mixnet, cMix collects a certain number of messages in a batch before processing them. Section 8 discusses our strategy for assembling batches, though details may depend on the application.

To become a cMix user, one must first establish for each mixnode a shared key. Section 4.2 provides more details on how these keys can be established. *Round keys* derived from the shared keys are used in each *round* of communication. A round begins with the start of batch processing.

For each round, β messages are collected and randomly ordered. Each message must have the same length, and all messages in a batch are processed simultaneously. The other messages are not accepted and are sent in a subsequent round. To process messages quickly in real time, cMix performs precomputations that do not involve any user. The precomputations are performed in a separate phase during which cMix executes all public-key encryptions, enabling the real-time computations to be carried using only fast multiplications.

3.2 Adversarial Model

We assume authenticated communication channels between all mixnodes. Thus, an adversary can eavesdrop, forward, and delete messages between mixnodes, but not modify, replay, or inject new ones, without detection. For any communication not among mixnodes, we assume the adversary can eavesdrop, modify, or inject messages at any point of the network.

An adversary can also compromise users; however, we assume that at least two users are honest. Mixnodes can also be compromised, but at least one of them needs to be honest for the system to be secure. We assume compromised mixnodes to be malicious but cautious: they aim not to get caught violating the protocol.

The goal of the adversary is to compromise the anonymity of the communication initiator, or to link inputs and outputs of the system. As with other mixnets [1,9,23,25,31,36,37], we do not consider adversaries whose sole purpose is to launch denial-of-service (DoS) attacks.

We envision a deployment model in which there are dedicated trusted data centers serving as the mixnodes (perhaps competitively awarded). As such, they are incentivized not to be expelled. By contrast, some mixnets allow the mixnodes to enter and leave with low cost.

An implication of our deployment model is that it is sufficient to be able to detect a cheating node with sufficient probability at some point. By contrast, in more flexible deployment models, the nodes should prove more stringently that they have computed correctly before the output is opened. cMix does require that the exit node commit to its output prior to being able to read the system output, which protects against certain adaptive attacks.

3.3 Security Goals

cMix aims to satisfy each of the following two security properties:

- **Anonymity:** A protocol provides *anonymity* if the adversary cannot map any input message to the corresponding output message, with a probability significantly better than that of random guessing, even if the adversary compromises all but two users and all but one mixnode.
- **Integrity:** A protocol provides *integrity* if at the end of every run involving β honest users:
  1. either, the β messages from the honest users are delivered unaltered to the intended recipients,
  2. or, a malicious mixnode is detected with a non-negligible probability and no honest party is proven malicious.

4 The Core Protocol

We now present the core cMix protocol, beginning with some preliminary notations and concepts, followed by a detailed specification.

4.1 Preliminaries

We introduce the primitives and notations used to describe the protocol. There are n mixnodes that process β messages per batch. For simplicity we assume here that the system already knows for each sender what slot to use out of the β slots. When implementing the system this assignment can, for example, be achieved by including the sender's identity (possibly a pseudonym) when sending a message to the system.

All computations are performed in a prime-order cyclic group $\mathbf{G}$ satisfying the decision Diffie-Hellman (DDH) assumption. The order of the group is p, and g is a generator for this group. Let $\mathbf{G}^*$ be the set of non-identity elements of $\mathbf{G}$.

cMix uses a multi-party group-homomorphic cryptographic system. We make use of a system based on ElGamal, described by Benaloh [4], though any such system could be used. This system works as follows:

- $d_i \in \mathbb{Z}_p^*$: the secret share for mixnode i of the secret key d.
- e: the public key of the system, based on the mixnode shares of the secret key: $e = \prod_i g^{d_i}$.

- $\mathcal{E}_e(m) = (g^x, m \cdot e^x)$, $x \in_R \mathbb{Z}_p^*$: encryption of message m under the system's public key e. We call g^x the *random component* and $m \cdot e^x$ the *message component* of the ciphertext. When applying encryption on a vector of values, each value in the vector is encrypted individually—each with a fresh random value—and the result is a vector of ciphertexts.
- $\mathcal{D}_{d_i}(g^x) = (g^x)^{-d_i}$: the decryption share for mixnode i computed from the random component of a ciphertext using the mixnode's share of the secret key. As with encryption, applying this function on a vector of random values results in a vector of corresponding decryption shares.

To decrypt a ciphertext $(g^x, m \cdot e^x)$, all parties need to cooperate because the decryption shares for all mixnodes are required to retrieve the original message:

$$m \cdot e^x \cdot \prod_{i=1}^{n} \mathcal{D}_{d_i}(g^x) = m \cdot (\prod_{i=1}^{n} g^{d_i})^x \cdot \prod_{i=1}^{n} (g^x)^{-d_i} = m.$$

The cMix protocol uses the following values:

- $r_{i,a}, s_{i,a} \in \mathbf{G}^*$: random values (freshly generated for each round) of mixnode i for slot a. Thus, $\mathbf{r}_i = (r_{i,1}, r_{i,2}, \ldots, r_{i,\beta})$ is a vector of random values for the β slots in the message map at mixnode i. Similarly, $\mathbf{s}_i$ is also a vector of random values for mixnode i.
- π_i: a random permutation of the β slots used by mixnode i. The inverse of the permutation is denoted by π_i^{-1}.
- $k_{i,j} \in \mathbf{G}^*$: a group element shared between mixnode i and the sending user for slot j. These values are used as keys to blind messages.
- $M_j \in \mathbf{G}^*$: the message sent by user j. Like other values in the system, these values are group elements. They can be easily converted from, for example, an ASCII-encoded string. The group size determines the length of an individual message that can be sent.

For readability we introduce the following shorthand notations:

- $\mathbf{R}_i$: the product of all local random $\mathbf{r}$ values through mixnode i; i.e., $\mathbf{R}_i = \prod_{j=1}^{i} \mathbf{r}_j$.
- $\mathbf{S}_i$: the product and permutation of all local random s values:

$$\mathbf{S}_i = \begin{cases} \mathbf{s}_1 & i = 1 \\ \pi_i(\mathbf{S}_{i-1}) \times \mathbf{s}_i & 1 < i \leq n. \end{cases}$$

- $\boldsymbol{\Pi}_i(a)$: the permutation performed by cMix through mixnode i, i.e., the composition of all individual permutations:

$$\boldsymbol{\Pi}_i(a) = \begin{cases} \pi_1(a) & i = 1 \\ \pi_i(\boldsymbol{\Pi}_{i-1}(a)) & 1 < i \leq n. \end{cases}$$

- $\mathbf{k}_i$ and $\mathbf{k}_i^{-1}$: the vector of keys shared between mixnode i and the users for all β slots and their inverses, respectively; $\mathbf{k}_i = (k_{i,1}, k_{i,2}, \ldots, k_{i,\beta})$ and $\mathbf{k}_i^{-1} = (k_{i,1}^{-1}, k_{i,2}^{-1}, \ldots, k_{i,\beta}^{-1})$.

- K_j: the product of all shared keys of the sending user for slot j: $K_j = \prod_{i=1}^{n} k_{i,j}$.
- $\mathbf{K}$ is a vector of products of shared keys for the β slots; $\mathbf{K} = (K_1, K_2, \ldots, K_\beta)$.

4.2 Protocol Description

We now present the core protocol. In this explanation we focus on simplicity and clarity; see Sect. 5 and our extended paper [10] for a discussion of possible security issues and enhancements. We separately discuss each of the three protocol phases: setup, precomputation, and real time.

Setup. In the setup phase, the mixnodes establish their secret shares d_i and the shared public key e, which are used for the multi-party homomorphic encryption system.

The users also establish their keys $k_{i,j}$, which they share with all mixnodes. This can be done using any (offline) key distribution method. One way to derive these keys is using a Diffie-Hellman key exchange. The resulting key can be used as a seed to derive unique values for $k_{i,j}$ for every session. Depending on the chosen key distribution protocol, this would be the only time a user is possibly required to perform an asymmetric cryptographic operation. The key exchange must be performed once for each user, and this exchange can be carried during the user's enrollment into the system.

Precomputation. The precomputation phase is performed only by the mixnodes, without any involvement from the users. It is performed once for each real-time phase. The mixnodes establish shared values to circumvent the need for public-key operations during the real-time phase. The precomputation phase comprises three different steps given below. The goal of the precomputation phase is to compute the values $\boldsymbol{\Pi}_n(\mathbf{R}_n) \times \mathbf{S}_n$, which are used in the real-time phase. Figure 1 shows a schematic example of the first two steps of the precomputation phase.

Step 1 - Preprocessing. The mixnodes start by generating fresh $\mathbf{r}$, $\mathbf{s}$, $\boldsymbol{\pi}$ values. Then they collectively compute the product of all of their individual $\mathbf{r}$ values under encryption using the public key e of the system, which was computed during the setup phase. This computation takes place by each mixnode i sending the following message to the next mixnode:

$$\mathcal{E}_e(\mathbf{R}_i) = \begin{cases} \mathcal{E}_e(\mathbf{r}_1) & i = 1 \\ \mathcal{E}_e(\mathbf{R}_{i-1}) \times \mathcal{E}_e(\mathbf{r}_i) & 1 < i \leq n. \end{cases}$$

Each mixnode encrypts its own $\mathbf{r}$ values and uses the homomorphic property of the encryption system to compute the multiplication of this ciphertext with the input it receives from the previous mixnode. Eventually, the last mixnode sends the final values $\mathcal{E}_e(\mathbf{R}_n)$ to the first mixnode as input for the next step.

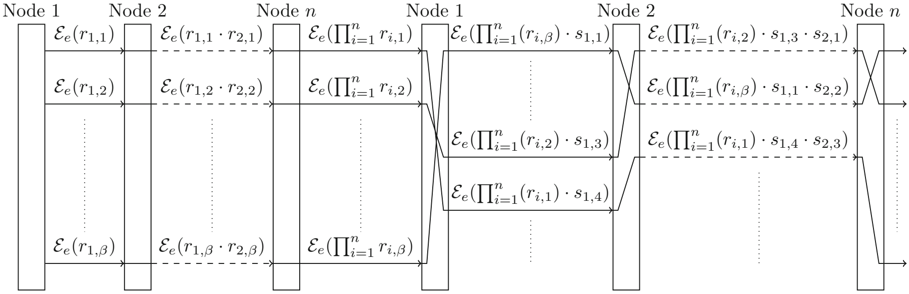

Node 1
Node 2
Node n
Node 1
Node 2
Node n
$\mathcal{E}_e(r_{1,1})$
$\mathcal{E}_e(r_{1,2})$
$\mathcal{E}_e(r_{1,\beta})$
$\mathcal{E}_e(r_{1,1} \cdot r_{2,1})$
$\mathcal{E}_e(r_{1,2} \cdot r_{2,2})$
$\mathcal{E}_e(r_{1,\beta} \cdot r_{2,\beta})$
$\mathcal{E}_e(\prod_{i=1}^n r_{i,1})$
$\mathcal{E}_e(\prod_{i=1}^n r_{i,2})$
$\mathcal{E}_e(\prod_{i=1}^n r_{i,\beta})$
$\mathcal{E}_e(\prod_{i=1}^n (r_{i,\beta}) \cdot s_{1,1})$
$\mathcal{E}_e(\prod_{i=1}^n (r_{i,2}) \cdot s_{1,3})$
$\mathcal{E}_e(\prod_{i=1}^n (r_{i,1}) \cdot s_{1,4})$
$\mathcal{E}_e(\prod_{i=1}^n (r_{i,2}) \cdot s_{1,3} \cdot s_{2,1})$
$\mathcal{E}_e(\prod_{i=1}^n (r_{i,\beta}) \cdot s_{1,1} \cdot s_{2,2})$
$\mathcal{E}_e(\prod_{i=1}^n (r_{i,1}) \cdot s_{1,4} \cdot s_{2,3})$

Fig. 1. A schematic example of the first two steps of the precomputation phase, which result in the values $\mathcal{E}_e(\boldsymbol{\Pi}_n(\mathbf{R}_n) \times \mathbf{S}_n)$.

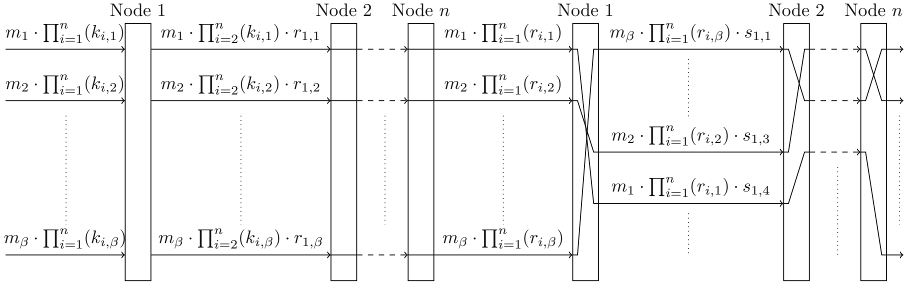

Node 1
Node 2
Node n
Node 1
Node 2
Node n
$m_1 \cdot \prod_{i=1}^n (k_{i,1})$
$m_2 \cdot \prod_{i=1}^n (k_{i,2})$
$m_\beta \cdot \prod_{i=1}^n (k_{i,\beta})$
$m_1 \cdot \prod_{i=2}^n (k_{i,1}) \cdot r_{1,1}$
$m_2 \cdot \prod_{i=2}^n (k_{i,2}) \cdot r_{1,2}$
$m_\beta \cdot \prod_{i=2}^n (k_{i,\beta}) \cdot r_{1,\beta}$
$m_1 \cdot \prod_{i=1}^n (r_{i,1})$
$m_2 \cdot \prod_{i=1}^n (r_{i,2})$
$m_\beta \cdot \prod_{i=1}^n (r_{i,\beta})$
$m_\beta \cdot \prod_{i=1}^n (r_{i,\beta}) \cdot s_{1,1}$
$m_2 \cdot \prod_{i=1}^n (r_{i,2}) \cdot s_{1,3}$
$m_1 \cdot \prod_{i=1}^n (r_{i,1}) \cdot s_{1,4}$

Fig. 2. A schematic example of the first two steps of the real-time phase, which result in the values $\boldsymbol{\Pi}_n(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_n$.

Step 2 - Mixing. In the second step, the mixnodes together mix the values and compute the results $\boldsymbol{\Pi}_n(\mathbf{R}_n) \times \mathbf{S}_n$, under encryption. The mixnodes perform this mixing by having each mixnode i send the following message to the next mixnode:

$$\mathcal{E}_e(\boldsymbol{\Pi}_i(\mathbf{R}_n) \times \mathbf{S}_i) = \begin{cases} \pi_1(\mathcal{E}_e(\mathbf{R}_n)) \times \mathcal{E}_e(\mathbf{s}_1) & i = 1 \\ \pi_i(\mathcal{E}_e(\boldsymbol{\Pi}_{i-1}(\mathbf{R}_n) \times \mathbf{S}_{i-1})) \times \mathcal{E}_e(\mathbf{s}_i) & 1 < i \leq n. \end{cases}$$

As with the first step, the last mixnode sends the final encrypted values $\mathcal{E}_e(\boldsymbol{\Pi}_n(\mathbf{R}_n) \times \mathbf{S}_n)$ to the first mixnode. These final values now must be decrypted together by all mixnodes, which happens in the last step of the precomputation.

Step 3 - Postprocessing. To complete the precomputation, the mixnodes decrypt the precomputed values. Each mixnode i computes its decryption shares $\mathcal{D}_{d_i}(\mathbf{x})$, where $(\mathbf{x}, \mathbf{c}) = \mathcal{E}_e(\boldsymbol{\Pi}_n(\mathbf{R}_n) \times \mathbf{S}_n))$. The message parts $\mathbf{c}$ are multiplied with all the decryption shares to retrieve the plaintext values $\boldsymbol{\Pi}_n(\mathbf{R}_n) \times \mathbf{S}_n$. This computation can be carried out either using another pass through the network (in which every mixnode multiplies in its own decryption share), or by having all mixnodes send their encryption shares to the last mixnode, which can then perform the multiplication. The last mixnode to be used in the real-time phase stores the decrypted precomputed values.

Real Time. For the real-time phase, each user constructs its input by taking its message m and multiplying it with its combined shared key k to compute the blinded message $m \times k$. This blinded message is then sent to the mixnet. One option would be to send the blinded messages to the first mixnode. Once the first mixnode receives enough blinded messages, it combines those messages to yield the vector $\mathbf{M} \times \mathbf{K}$. As in the precomputation phase, the real-time phase can again be split into three steps. Figure 2 gives a schematic example of the first two steps.

Step 1 - Preprocessing. During the preprocessing step, the mixnodes take out the keys $\mathbf{k}$ they share with the users and add in their $\mathbf{r}$ values to blind the original messages. This computation is performed by each mixnode i sending the following to the next mixnode:

$$\mathbf{M} \times \mathbf{K} \times (\prod_{j=1}^{i} \mathbf{k}_j^{-1} \times \mathbf{r}_j) = \mathbf{M} \times \mathbf{K} \times (\prod_{j=1}^{i-1} \mathbf{k}_j^{-1} \times \mathbf{r}_j) \times \mathbf{k}_i^{-1} \times \mathbf{r}_i.$$

The last mixnode sends the final values $\mathbf{M} \times \mathbf{R}_n = \mathbf{M} \times \mathbf{K} \times \prod_{j=1}^{n} \left(\mathbf{k}_i^{-1} \times \mathbf{r}_i\right)$, which are the blinded versions of the original messages, to the first mixnode as input for the next step. Now the user-specific keys $\mathbf{k}$ are taken out and replaced by the user-independent values $\mathbf{r}$.

Step 2 - Mixing. The second step performs the mixing to hide the association between sender and receiver. The $\mathbf{s}$ values are added in to hide which input message corresponds to which output message. Each mixnode i (except the last mixnode) sends the following message to the next mixnode:

$$\boldsymbol{\Pi}_i(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_i = \begin{cases} \pi_1(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{s}_1 & i = 1 \\ \pi_i(\boldsymbol{\Pi}_{i-1}(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_{i-1}) \times \mathbf{s}_i & 1 < i < n. \end{cases}$$

Finally, the last mixnode computes:

$$\boldsymbol{\Pi}_n(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_n = \pi_n(\boldsymbol{\Pi}_{n-1}(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_{n-1}) \times \mathbf{s}_n.$$

Now every mixnode has performed its mixing, destroying the associations between senders and receivers.

Step 3 - Postprocessing. The last mixnode can perform the final step. This mixnode retrieves the locally stored precomputed values $\boldsymbol{\Pi}_n(\mathbf{R}_n) \times \mathbf{S}_n$. To retrieve the permuted messages it now needs only to perform the following computation, using the result from the previous mixing step:

$$\boldsymbol{\Pi}_n(\mathbf{M}) = \boldsymbol{\Pi}_n(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_n \times (\boldsymbol{\Pi}_n(\mathbf{R}_n) \times \mathbf{S}_n)^{-1}.$$

How the messages are then delivered to the recipients depends on the application and is independent from cMix. This step concludes the real-time phase, in which no public-key operations are performed.

5 Protocol Integrity

The cryptographic construction presented in Sect. 4 expects the mixnodes and users to be honest but curious. As for most mixnet protocols [9,26,33,36], cMix requires additional measures to ensure that mixnodes cannot tamper with the messages nor with the flow without detection. In this section, we augment the protocol to protect its anonymity and integrity against malicious attacks by users and by compromised mixnodes, which we shall call "adversarial mixnodes." The overall strategy relies on the assumption that compromised mixnodes are malicious but cautious.

5.1 Integrity of Values and Messages

To enable honest mixnodes to verifiably detect any malicious mixnodes that employ incorrect values or permutations in the precomputation or real-time phase, cMix augments communications with proofs. To this end, all messages exchanged between mixnodes are signed using a digital signature system with existential unforgeability under an adaptive chosen-message attack [22]. In addition, during precomputation, each mixnode commits to the permutation π_j it applies to the incoming slot j using a perfectly hiding commitment (Commit) scheme [24] and signs that commitment. Each node broadcasts its signed commitment using a reliable broadcast ($\mathsf{Broadcast}$) protocol [17,44]. Doing so makes it possible to reconstruct and verify all individual values $r_{i,j}$, $s_{i,j}$ and $\pi_{i,j}$ that mixnode j applies to slot i.

In the real-time phase, the users become involved, making the process more complicated because they do not perform any public-key operations during the real-time phase. The values that we need to verify that depend on the users are the $k_{i,j}$ values (shared between a mixnode i and the user in slot j), and the $m_j \cdot K_j$ values (the blinded message that a user sends in slot j).

Because the k values are shared between a mixnode and a user, we need them to agree on the commitments to these values. If we detect an anomaly involving the k values, and a mixnode and user disagree on the value used, the commitment needs to provide proof of who misbehaved.

For this task, the following procedure can be followed in case the k values are derived from a seed that is agreed upon by a mixnode and user during the setup phase. After establishing this seed, the mixnode will compute a commitment of the seed and provide this commitment to the user. The mixnode generates the commitment in a way that enables the user to reveal the commitment and prove to other parties that the mixnode generated the commitment. The user must verify the commitment during the setup phase. This verification requires an additional public-key operation, though it will be performed only once provided the protocol runs without any disturbance.

Message Integrity at Entry. Messages at entry to cMix need to arrive at cMix and pass through the non-permuted part of the protocol without any undetected modification. Providing integrity of messages at this point is a challenge if one wishes to keep clients free from using any asymmetric cryptography during the real-time phase. We propose a construction where message authentication codes (MACs) are generated over the input message to the cMix system. A shared-key MAC is sufficient here because the communicants (sender, mixnode) do not need to convince any third party.

To accomplish this goal, we introduce additional key values $l_{i,j}$, shared between mixnode i and the user for slot j, established and committed to in a similar way as for the k values. When the user sends the blinded message $m_j \cdot K_j$ to the system, the user also sends the following messages:

$$h_j = \text{Hash}(m_j \cdot K_j) \text{ and } (MAC_{l_{1,j}}(h_j), \ldots, MAC_{l_{n,j}}(h_j)).$$

During the real-time phase, the first mixnode starts by checking its corresponding MAC value in the list. If it is incorrect, the mixnode informs the other mixnodes and does not forward the computed value for this slot. If the MAC value is correct, it forwards the h values and the MAC values for the other mixnodes, together with its basic computed values.

Each subsequent mixnode follows the same procedure. At the end of the first step of the real-time phase, all mixnodes have checked their MAC values on the received h values, or the value is not processed any further.

5.2 Message Tagging Detection

A message-tagging attack is an attack where the adversary can mark a message at some point during the process, such that it is recognizable when it is output, compromising unlinkability between the inputs and outputs [39]. To perform a tagging attack unnoticed, the tag should also be removed before the messages are output. Tagging attacks are a threat to all mixnets that use some form of malleable encryption, such as homomorphic encryption or group multiplications, where valid messages can be recognized when output by the mixnet. For example, Pfitzmann [38] presents such an attack on re-encryption mixnets.

A simple example of a tagging attack is the following: The last mixnode multiplies the blinded message in one of the slots j with an additional factor t in the first step of the real-time phase. Now the blinded message in slot j will be $m_j \cdot \prod_{i=1}^{n} r_{i,j} \cdot t$ at the end of the first step, whereas the values in the other slots stay the same. When the last mixnode performs Step 3, it will see the final messages before it outputs them. If the messages are recognizable as valid outputs, it will observe that one of the messages does not seem to be valid. If this invalid message becomes valid when multiplied with t^{-1}, this message is likely the tagged one. The mixnode can now link the message, and possibly the recipient, to the sender. The mixnode can remove the tag and output the original messages, making it unobservable to the users and other mixnodes that a tagging attack took place.

Detection. To protect against these kinds of attacks and make them detectable, only small changes are needed to the protocol:

- **Precomputation Phase - Step 3:** The mixnodes no longer send out their decryption shares to retrieve the precomputed values. Instead, they keep their shares secret and publish only a commitment to them. The last mixnode also publishes a commitment to the message component of the ciphertext. The commitments can be computed, for example, using only one signature per mixnode for the decryption shares for all slots simultaneously. The plaintext results of the precomputation phase are thus no longer retrieved in that phase itself.
- **Real-Time Phase - Step 3:** The output of the mixing step $\boldsymbol{\Pi}_n(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_n$ is published by the last mixnode. Afterwards, all mixnodes release their decryption shares $\mathcal{D}_{d_i}(\mathbf{x})$ and the message component of the ciphertext $\mathbf{c}$. The output messages are then computed as follows, where $(\mathbf{x}, \mathbf{c}) = \mathcal{E}_e(\boldsymbol{\Pi}_n(\mathbf{R}_n) \times \mathbf{S}_n)$:

$$\boldsymbol{\Pi}_n(\mathbf{M} \times \mathbf{R}_n) \times \mathbf{S}_n \times \mathbf{c} \times \prod_{i=1}^{n} \mathcal{D}_{d_i}(\mathbf{x}).$$

Because the mixnodes committed to all of the values necessary to retrieve the precomputed value, they cannot change these values to take out a possible tag anymore. The output of the mixing step does not reveal anything yet about the messages, because the r and s values are still included. The precomputed value to take out these values is retrieved only after the output of the mixing step is made public. Therefore, the last mixnode would not know from which output slot a possible tag should be removed before the output of the mixing step is made public. Once the output is made public, the tag cannot be removed because all computations for the third step are fixed and can be verified by anyone.

5.3 Sending and Verifying Trap Messages

To ensure the protocol is functioning correctly, we let users send trap messages with a certain probability, and request to open message paths of these traps. Opening of a path includes verification of all messages exchanged between mixnodes, the incoming message from the user, as well as intermediate values and permutations.

Sending Trap Messages and Requesting to Open Them: To send a trap, a user starts by forming a string x with a round ID, user ID, and a statement that this is the dummy message. She calculates a MAC of x with every key $l_{i,j}$ shared between mixnode i and the user for slot j. The trap message is then the string together with its MAC values, i.e., $m = (x, MAC_{l_{1,j}}(x), \ldots, MAC_{l_{n,j}}(x))$. Note that these MAC values are different from the MAC values mentioned in Sect. 5.1, even though we are using the same keys in both places.

After the round is over, the trap message appears at the output. Each mixnode verifies the correctness of her associated user ID, round ID, and $MAC_{l_{i,j}}$

values. If any mixnode i detects that any of these values is incorrect, she stops the round. Otherwise, the mixnode sends an authenticated message to user j notifying that the dummy message is received correctly, including an incremented counter of all dummies received from this user.

If a user actually sent a dummy message, but she does not receive a notification message from each mixnode, she initiates the verification procedure. The verification procedure consists of the following steps:

(i) User j sends a request, tagged with MAC values, to open the path with her dummy message to all mixnodes.
(ii) A mixnode can either agree to participate in the path opening, or can dispute the MAC of the path-opening request.
(iii) If mixnode i disputes the MAC intended for her, both the user and mixnode open their local $l_{i,j}$ values to validate the MAC.
(iv) If both the user and the mixnode are using the same $l_{i,j}$ value, then the $MAC_{l_{i,j}}()$ values computed by them should be same, and the correctness of the request sent by the user can be easily validated. If the user has sent an invalid request, the request is dropped.
(v) If the user and the mixnode dispute over the $l_{i,j}$ value, then the seed of $l_{i,j}$ and the commitment to that seed (made by the mixnode and verified by the user during setup) are opened. The user and mixnode each recomputes $l_{i,j}$ from the seed. Two possibilities follow:
  (a) If the recomputed $l_{i,j}$ value is not equal to what the user claims, the path-opening request is dropped.
  (b) If the recomputed $l_{i,j}$ value is not equal to what the mixnode claims, she must either agree to participate in the path-opening, or be considered malicious.
(vi) Once all the mixnodes agree to participate in the path-opening, they first open the path of the precomputation phase, and then the real-time phase (both are explained below).

Path Opening for the Precomputation Phase: For the input slot j that we want to verify, the mixnodes have to decrypt exchanged messages for this slot and compute the r values in the non-permuted part. For the permuted part, mixnodes subsequently reveal the corresponding permutations and decrypt exchanged messages to obtain the s values. Doing so we can follow the computation through the mixnet and verify whether the precomputed value that was output was correctly computed.

To check, for example, whether mixnode i performed its computation correctly, that mixnode needs to present the signature from the previous mixnode on the values it received. The next mixnode will also have to present the signature it received from mixnode i to obtain proof what values mixnode i output. Once all information about the input, output, and values used in the expected computation are known, due to the signatures, commitments, and threshold decryption, it can be verified whether mixnode i performed the computation as expected or not.

Path Opening for the Real-Time Phase: Malicious mixnodes can also employ incorrect messages, values, and permutations in the real-time phase. We wish to make the mixnodes accountable for their behavior. One challenge is that malicious users may try to victimize some honest mixnodes by providing inconsistent inputs and later deny having done so.

First, the mixing step is verified. This check is performed in a similar fashion as for the precomputation phase. For the preprocessing step, a comparable approach is followed. To verify whether the correct input from the user to the system is used, all the keys l for the MAC values for the suspicious slot are output. The purpose is to detect if a mixnode or user changed the input. Because the values were processed in the mixing step, during the first step of the real-time phase, all mixnodes accepted the MAC values and thus should be able to provide a correct key. The mixnodes also reveal their r and k values for the corresponding slot, and the mixnodes check whether they performed their computations correctly. Although the mixnodes committed to the r values, they have not committed to individual k values. Therefore the sender must be involved in this process. The sender will also release all the k values it used for this message.

If a k value released by the sender does not match the one released by the corresponding mixnode, either the mixnode or the sender is misbehaving. The sender might do this to blame a mixnode of misbehaving to cause it to be removed from the system. This dispute needs to be resolved by having the user reveal the commitment by the mixnode on the shared keys. Doing so might also reveal k values used in previous sessions, but these values are only one of the components of K and therefore do not leak the original messages. There are two possibilities: either the user was malicious, in which case we do not care about his or her previous messages, or the mixnode was malicious, in which case we consider the k values to be compromised already. This procedure will reveal who acted maliciously and modified the output message.

5.4 Security Analysis

We now briefly highlight some of the security properties of cMix. For details, see the extended version of our paper [10].

As described in Sect. 3.3, we intend our protocol to have the following property: if any of the messages are altered by any node, (a) no honest party can be proven malicious, and (b) at least one malicious mixnode is detected with non-negligible probability. In [10], we argue how the above described integrity measures (integrity of values and messages, message tagging detection, and trap messages) achieve the integrity property.

We also claim: if $\mathcal{E}$ is a CPA-secure group-homomorphic encryption system, and Commit is a perfectly-hiding non-interactive commitment scheme, and if the protocol maintains integrity, then cMix offers anonymity. For a detailed anonymity analysis, see [10].

In [10] we also explain how cMix resists well-known attacks on mixnets [6,9,12–14,16,39,40,43].

Galteland et al. [19] propose a tagging attack and an insider attack against the cMix protocol, as described in the preliminary cMix eprint. But security mechanisms specified in this preliminary cMix eprint prevent both attacks, as do alternative integrity mechanisms (e.g., trap messages) specified in the current cMix paper and its extended version [10]. In particular, as presented in the preliminary cMix eprint, cryptographic commitments enforced by Random Partial Checking (RPC) [30] prevent both attacks. Thus, these purported "Norwegian Attacks" do not work.

6 Comparison with Other Mixnets

We compare cMix with well-known fixed-cascade mixnet approaches based on performance. Specifically, as summarized in Table 1, we compare the performance of the core cMix protocol with that of each of the following three competing approaches: original mixnet and hybrid mixnets, re-encryption mixnet, and re-encryption mixnet with precomputation.

For each approach, we compare the precomputation and real-time costs, further compared by number of single-party public-key operations, multi-party public-key operations, and multiplications (each by client and by mixnodes). Note that, when using ElGamal encryption, the multiplication of two ciphertexts requires two multiplications.

Table 1. Performance comparison of the core cMix protocol with three competing mixnet [9, 36] approaches: number of multiplication (Mult) and public-key (PK) operations performed for a batch of β messages processed by an n-node mixnet. One multi-party public-key operation (ops) requires all nodes to participate.

	Precomputation (ops for all mixnodes)			Real time (ops per client; ops for all mixnodes)		
	PK ops	Multi-party PK ops	Mult	PK ops	Multi-party PK ops	Mult
cMix (core)	$2n\beta$	β	$4n - 2\beta$	0; 0	0; 0	n; $3n\beta$
Original mix	-	-	-	n; $n\beta$	0; 0	0; 0
Re-encryption mix	-	-	-	1; $n\beta$	0; β	0; $2n\beta$
Re-encryption mix (precomputation)	$n\beta$	0	0	1; 0	0; β	0; $2n\beta$

The original mixnet [9] requires each sender to perform n encryptions; each of the n mixnodes in the cascade performs one decryption per message and β decryptions per batch. In total, all mixnodes perform n decryptions per message and βn decryptions per batch. Hybrid mixnets [26, 33] require the same amount of asymmetric encryptions, but on a smaller plaintext.

In re-encryption mixnets [36], each client performs one encryption of its message using the mixnet's shared public key. Each node re-randomizes every message, instead of decrypting each one as with original mixnets, resulting in one public-key operation and one multiplication of ciphertexts per message per node. In addition, the nodes need to decrypt the output of the mixnet in a multi-party computation.

Re-encryption mixnets can be improved in a straightforward way using precomputation to perform the public-key operations necessary for the re-randomization, similarly to cMix's strategy. As shown in Table 1, however, with regard to real-time computation, cMix outperforms re-encryption mixnets with precomputation.

Universal re-encryption mixnets perform much more slowly because they require senders to encrypt messages with public keys of their recipients.

7 Proof of Concept

We implemented a proof-of-concept protoype in Python, including tagging detection as discussed in Sect. 5.2. Towards reducing the communication latency, we have also introduced a network handler.

Introducing an untrusted *network handler* reduces latency. In Steps 1 and 3 of the precomputation and real-time phases, only products of values known by the individual nodes are computed (see Sect. 4.2). To compute these products, it is not necessary to make a full pass through the mixnet. Instead, each node can send its values to an untrusted third party, which we call the network handler, who can compute the products and return the results to the mixnet. Doing so reduces latency of the network significantly because each node can send its values simultaneously to the handler, instead of forwarding its local result to the next node sequentially. The network handler does not learn any secret value, and it computes only values that would anyway become public. The network handler, and each of the mixnodes, is a single point of failure. In the event of failure, however, because the handler performs only public operations, it can be easily replaced by another entity—for example, by one of the mixnodes.

Each mixnode includes a keyserver (to establish shared keys with the users) and a mixnet server (to carry out the precomputations and real-time computations). We use Ed25519 [5] signatures to implement authenticated channels between mixnodes. For the precomputation, each mixnode uses parallel processes for the computation of the encryptions and the decryption shares. In the real-time phase, all operations are performed in a single thread.

We performed experiments by running the protoype on Amazon Web Services (AWS) instances, with each node comprising a c3.large with two virtual processors and 3.75 GB of RAM. For all values, we used a prime-order group of 2048 bits.

On the AWS instances, each 2048-bit ElGamal encryption took approximately 10 ms on average, and the computation of a decryption share took approximately 5 ms. Multiplication of group elements took only a fraction of a millisecond.

For our experiments we performed 100 precomputation and real-time phases for selected batch sizes up to 1000 with five mixnodes. Table 2 gives observed

Table 2. Timings measured on the network handler from the start of the phases until the final values or responses are computed. Timings are in seconds (wall clock) for 100 runs of the precomputation and real-time phases, for various batch sizes using five mixnodes.

Batch	Precomputation		Real time	
Size	Mean	Std. dev.	Mean	Std. dev.
10	0.48	0.07	0.07	0.02
50	1.99	0.04	0.21	0.04
100	4.00	0.30	0.38	0.06
200	7.74	0.09	0.75	0.08
300	11.46	0.13	1.09	0.08
400	15.24	0.11	1.44	0.08
500	19.08	0.23	1.80	0.11
1000	37.94	0.19	3.58	0.12

Table 3. Mean timings in seconds (CPU and wall clock) for 100 runs of the real-time phase of the cMix protocol measured on the mixnodes and network handler, for various batch sizes using five mixnodes. For the mixnodes the mean time is taken over all mixnodes.

Batch	Mixnode		Network handler	
Size	CPU	Wall	CPU	Wall
10	0.01	0.04	0.01	0.07
50	0.04	0.16	0.02	0.21
100	0.07	0.30	0.02	0.38
200	0.14	0.60	0.04	0.75
300	0.22	0.87	0.06	1.09
400	0.29	1.15	0.07	1.44
500	0.36	1.45	0.09	1.80
1000	0.73	2.88	0.19	3.58

timings on the network handler for selected batch sizes using five mixnodes, without any enhanced security mechanisms mentioned in Sect. 5. We measured elapsed time on the network handler from the time it instructed the nodes to start until either the precomputation finished successfully, or until it computed the final responses to be sent to the users in the real-time phase. Table 3 gives timings for the real-time phase per node and for the network handler, in both CPU and wall clock time. These timings show the low computational load on the nodes during this phase.

These timings demonstrate the high performance of the system in the real-time phase. The precomputation can be easily accelerated by performing more computations in parallel. Additional processors would significantly improve the time it takes to compute all necessary encryptions and decryption shares. For the real-time phase, a network connection with low latency would improve the timings.

8 Extensions, Discussion, and Future Work

Here, we first describe how receivers can send immediate responses. We then briefly discuss how to arrange messages into batches and how to deal with node failures. We also outline some of our future plans.

Return Path. It is easy to extend cMix to enable a receiver to send an immediate response through the mixnet, for example, to acknowledge receiving a message. To accomplish this goal, the nodes generate additional random values

$\mathbf{s}'$ and compute the permuted products $\mathbf{S}'$ during the precomputation phase. The nodes and users also generate fresh keys $\mathbf{k}'$, which will be used to encrypt the return message.

For a return path, the mixnodes apply the inverse permutations π^{-1} so that the responses will arrive at the original senders. Unless the recipient who sends a response shares keys with the system, no fresh $\mathbf{r}'$ values are needed because the message would not enter the system blinded and hence Step 1 of the real-time phase could be skipped. In Step 2, the system applies the inverse permutations π^{-1} and fresh $\mathbf{s}'$ values. In Step 3, to encrypt the response to the original sender, instead of multiplying only with its decryption component, each node multiplies with the product of its decryption component and $\mathbf{k}'$ value.

Batch Strategy. cMix follows the "threshold and timed mixing strategy" from Serjantov et al. [42], where a new round is started every t seconds only if there are at least β messages in the buffer. We expect at least β users to be active at any given time. When a smaller number of users is active, this strategy can lead to increased latency or even disruption. At the cost of increased energy consumption, one design choice is to inject dummy messages when needed to ensure enough traffic to have β messages every t seconds. Alternatively, empty slots can be used to verify if the precomputation was performed correctly, by revealing all committed values for these slots, where empty slots to use should be chosen at random.

Node Failure. Because cMix uses a fixed cascade of nodes, it is important to consider what happens if a node fails. First, we consider a node failure to be a highly rare event because we expect each node to be a highly reliable computing service that is capable of seamlessly handling failures. Second, the system will detect node failure and notify the senders and the other nodes; senders will be instructed to resend using a new cascade (e.g., the old cascade without the failed node). Each node can detect failures by listening for periodic "pings" from the other nodes.

To minimize possible disruption caused by a single failure, at the cost of increasing the precomputations, the following option can be deployed: Each node can have a reserve of precomputations ready to use for certain alternative cascades. For example, this reserve can include each of the alternative cascades formed by removing any one node from the current cascade.

Future Steps. Tasks we plan to work on in the future include the following: First, we would like to deploy cMix, including implementing and refining different applications. We would also like to carry out more performance studies. Second, we plan to explore alternative and even more efficient approaches for enforcing integrity of the nodes, to ensure that they cannot modify any message without detection. Third, currently, message length is restricted by the group modulus. We have begun to work out how to apply key-homomorphic pseudorandom functions [7] and an appropriate additive homomorphic encryption system to allow any length message. Fourth, we would like to explore possible ways of reusing a precomputation in a secure way.

9 Conclusion

cMix offers a promising new approach to anonymous communications, building on the strong anonimity properties of mixnets, and improving real-time cryptographic performance by eliminating real-time public-key operations in its core protocol. By replacing real-time public-key operations with precomputations, and by avoiding the user's direct involvement with the construction of the path through the mixnodes, cMix scales well for deployment with large anonymity sets and large numbers of mixnodes. Even though the adversary may know all senders and receivers in each batch, she cannot link any sender and receiver unless all mixnodes are compromised. cMix offers the potential for real-time performance improvements over existing mixnets, without losing any of their security guarantees.

Acknowledgments. We thank the anonymous reviewers for their comments. We also thank the following people for helpful suggestions: David Delatte, Russell Fink, Bryan Ford, Moritz Neikes, and Dhananjay Phatak.

Sherman was supported in part by the National Science Foundation under SFS grant 1241576 and a subcontract of INSuRE grant 1344369, and by the Department of Defense under CAE-R grant H98230-15-10294. Krasnova conducted this research within the Privacy and Identity Lab (PI.lab, http://www.pilab.nl) funded by SIDN.nl (http://www.sidn.nl/).

References

1. Adida, B., Wikström, D.: Offline/online mixing. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 484–495. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73420-8_43
2. Backes, M., Goldberg, I., Kate, A., Mohammadi, E.: Provably secure and practical onion routing. In: Proceeding of the 25th IEEE Computer Security Foundations Symposium (CSF), pp. 369–385 (2012)
3. Backes, M., Kate, A., Mohammadi, E.: Ace: an efficient key-exchange protocol for onion routing. In: Proceeding of the 11th ACM Workshop on Privacy in the Electronic Society (WPES), pp. 55–64 (2012)
4. Benaloh, J.: Simple verifiable elections. In: Proceeding of USENIX/Accurate Electronic Voting Technology Workshop (EVT), p. 5 (2006)
5. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. J. Cryptogr. Eng. **2**(2), 77–89 (2012)
6. Berthold, O., Pfitzmann, A., Standtke, R.: The disadvantages of free MIX routes and how to overcome them. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 30–45. Springer, Heidelberg (2001). doi:10.1007/3-540-44702-4_3
7. Boneh, D., Lewi, K., Montgomery, H., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_23
8. Camenisch, J., Lysyanskaya, A.: A formal treatment of onion routing. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 169–187. Springer, Heidelberg (2005). doi:10.1007/11535218_11

9. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **4**(2), 84–88 (1981)
10. Chaum, D., Das, D., Javani, F., Kate, A., Krasnova, A., Ruiter, J.D., Sherman, A.T.: cMix: mixing with minimal real-time asymmetric cryptographic operations. Cryptology ePrint Archive, Report 2016/008 (2016). https://eprint.iacr.org/2016/008.pdf
11. Chen, C., Asoni, D.E., Barrera, D., Danezis, G., Perrig, A.: HORNET: high-speed onion routing at the network layer. In: Proceeding of the 22nd ACM Conference on Computer and Communications Security, pp. 1441–1454 (2015)
12. Danezis, G.: The traffic analysis of continuous-time mixes. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 35–50. Springer, Heidelberg (2005). doi:10.1007/11423409_3
13. Danezis, G., Diaz, C., Troncoso, C.: Two-sided statistical disclosure attack. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 30–44. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75551-7_3
14. Danezis, G., Serjantov, A.: Statistical disclosure or intersection attacks on anonymity systems. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 293–308. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30114-1_21
15. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: Proceeding of the 13th USENIX Security Symposium, pp. 303–320 (2004)
16. Dingledine, R., Shmatikov, V., Syverson, P.: Synchronous batching: from cascades to free routes. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 186–206. Springer, Heidelberg (2005). doi:10.1007/11423409_12
17. Dolev, D., Reischuk, R., Strong, H.R.: Early stopping in byzantine agreement. J. ACM **37**(4), 720–741 (1990)
18. Evans, N.S., Dingledine, R., Grothoff, C.: A practical congestion attack on tor using long paths. In: Proceeding of the 18th USENIX Security Symposium, pp. 33–50 (2009)
19. Galteland, H., Mjølsnes, S.F., Olimid, R.F.: Attacks on cMix - some small overlooked details. Cryptology ePrint Archive, Report 2016/729 (2016). http://eprint.iacr.org/2016/729
20. Goldberg, I., Stebila, D., Ustaoglu, B.: Anonymity and one-way authentication in key exchange protocols. Des. Codes Crypt. **67**(2), 245–269 (2013)
21. Goldschlag, D.M., Reed, M.G., Syverson, P.F.: Onion routing. Commun. ACM **42**(2), 39–41 (1999)
22. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. **17**(2), 281–308 (1995)
23. Golle, P., Jakobsson, M., Juels, A., Syverson, P.: Universal re-encryption for mixnets. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 163–178. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24660-2_14
24. Halevi, S., Micali, S.: Practical and provably-secure commitment schemes from collision-free hashing. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 201–215. Springer, Heidelberg (1996). doi:10.1007/3-540-68697-5_16
25. Jakobsson, M.: Flash mixing. In: Proceedings of 18th Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 83–89 (1999)
26. Jakobsson, M., Juels, A.: An optimally robust hybrid mix network. In: Proceedings of 20th Annual ACM Symposium on Principles of Distributed Computing, pp. 284–292 (2001)

27. Jansen, R., Tschorsch, F., Johnson, A., Scheuermann, B.: The sniper attack: anonymously deanonymizing and disabling the Tor network. In: Proceedings of Network and Distributed System Security Symposium (NDSS 2014) (2014)
28. Kate, A., Goldberg, I.: Using Sphinx to improve onion routing circuit construction. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 359–366. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14577-3_30
29. Kate, A., Zaverucha, G.M., Goldberg, I.: Pairing-based onion routing with improved forward secrecy. ACM Trans. Inf. Syst. Secur. **13**(4), 29:1–29:32 (2010)
30. Khazaei, S., Wikström, D.: Randomized partial checking revisited. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 115–128. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36095-4_8
31. Kwon, A., Lazar, D., Devadas, S., Ford, B.: Riffle: an efficient communication system with strong anonymity. PoPETs **2016**(2), 115–134 (2016)
32. Murdoch, S.J., Danezis, G.: Low-cost traffic analysis of Tor. In: Proceedings of 26th IEEE Symposium on Security and Privacy, pp. 183–195 (2005)
33. Ohkubo, M., Abe, M.: A length-invariant hybrid mix. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 178–191. Springer, Heidelberg (2000). doi:10.1007/3-540-44448-3_14
34. Øverlier, L., Syverson, P.: Improving efficiency and simplicity of tor circuit establishment and hidden services. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 134–152. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75551-7_9
35. Øverlier, L., Syverson, P.F.: Locating hidden servers. In: Proceedings of 27th IEEE Symposium on Security and Privacy, pp. 100–114 (2006)
36. Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/nothing election scheme. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994). doi:10.1007/3-540-48285-7_21
37. Pfitzmann, A., Waidner, M.: Networks without user observability – design options. In: Pichler, F. (ed.) EUROCRYPT 1985. LNCS, vol. 219, pp. 245–253. Springer, Heidelberg (1986). doi:10.1007/3-540-39805-8_29
38. Pfitzmann, B.: Breaking an efficient anonymous channel. In: Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 332–340. Springer, Heidelberg (1995). doi:10.1007/BFb0053448
39. Raymond, J.-F.: Traffic analysis: protocols, attacks, design issues, and open problems. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 10–29. Springer, Heidelberg (2001). doi:10.1007/3-540-44702-4_2
40. Reed, M., Syverson, P., Goldschlag, D.: Anonymous connections and onion routing. IEEE J-SAC **16**(4), 482–494 (1998)
41. Ruffing, T., Moreno-Sanchez, P., Kate, A.: P2P mixing and unlinkable Bitcoin transactions. In: NDSS 2017 (2017)
42. Serjantov, A., Dingledine, R., Syverson, P.: From a trickle to a flood: active attacks on several mix types. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 36–52. Springer, Heidelberg (2003). doi:10.1007/3-540-36415-3_3
43. Serjantov, A., Sewell, P.: Passive attack analysis for connection-based anonymity systems. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 116–131. Springer, Heidelberg (2003). doi:10.1007/978-3-540-39650-5_7
44. Srikanth, T.K., Toueg, S.: Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. Distrib. Comput. **2**(2), 80–94 (1987)
45. Sun, Y., Edmundson, A., Vanbever, L., Li, O., Rexford, J., Chiang, M., Mittal, P.: Raptor: routing attacks on privacy in Tor. In: Proceedings of 24th USENIX Security Symposium, pp. 271–286 (2015)
46. The Tor project (2003). https://www.torproject.org/. Accessed April 2017

Almost Optimal Oblivious Transfer from QA-NIZK

Olivier Blazy[1(✉)], Céline Chevalier[2], and Paul Germouty[1]

[1] Université de Limoges, XLim, Limoges, France
{olivier.blazy,paul.germouty}@unilim.fr
[2] CRED, Université Panthéon-Assas Paris II, Paris, France
celine.chevalier@u-paris2.fr

Abstract. We show how to build a UC-Secure Oblivious Transfer in the presence of Adaptive Corruptions from Quasi-Adaptive Non-Interactive Zero-Knowledge proofs. Our result is based on the work of Jutla and Roy at Asiacrypt 2015, where the authors proposed a constant-size very efficient PAKE scheme. As a stepping stone, we first show how a two-flow PAKE scheme can be generically transformed in an optimized way, in order to achieve an efficient three-flow Oblivious-Transfer scheme. We then compare our generic transformations to existing OT constructions and see that we manage to gain at least a factor 2 to the best known constructions. To the best of our knowledge, our scheme is the first UC-secure Oblivious Transfer with a constant size flow from the receiver, and nearly optimal size for the server.

Keywords: OT · UC · QA-NIZK · Pairing-based cryptography

1 Introduction

Sharing data securely and efficiently is at the core of modern cryptography. The concept of Oblivious Transfer was introduced in 1981 by Rabin [41]. It allows to retrieve information from a database without privacy risk. Such protocols allow a receiver to ask and have access to a single line of a database. This exchange is oblivious in two ways. On the one hand, it allows to preserve the privacy of the receiver by preventing the sender from learning which part of the database has been asked. On the other hand, it allows the receiver to recover only one line, the one he previously asked for. The receiver is said to be oblivious to the values he is not allowed to get and the sender is oblivious to the information received. Several instantiations and optimizations of such protocols have appeared in the literature, like [21,38]. More recently, new instantiations have been proposed, trying to reach round-optimality [29], or low communication costs [40]. Recent schemes like [1,9,10] manage to achieve round-optimality while maintaining a small communication cost. Choi *et al.* [22] also propose a generic method and an efficient instantiation secure against adaptive corruptions in the CRS model with erasures, but it is only 1-out-of-2 and it does not scale to 1-out-of-n OT, for $n > 2$.

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 579–598, 2017.
DOI: 10.1007/978-3-319-61204-1_29

In *Password-Authenticated Key Exchange (*PAKE*)* protocols, proposed in 1992 by Bellovin and Merritt [7], authentication is done using a simple password, possibly drawn from a small space subject to exhaustive search. Since then, many schemes have been proposed and studied in various models, and ultimately proven in the UC framework.

Smooth Projective Hash Functions [23,26] have found several applications in the context of UC Secure protocols [3,8,34], and also more particularly in the context of Password Authenticated Key Exchange and Oblivious-Transfer Proposals [1,9]. These functions are defined such that their output can be computed in two different ways if the input belongs to a particular subset (the *language*), either using a private hashing key or a public projection key along with a private witness ensuring that the input belongs to the language. The value obtained is indistinguishable from random when the input does not belong to the language (this is called the *smoothness*) and in case the input does belong to the language but no witness is known (this is called the *pseudo-randomness*).

[28] proposed a technique to achieve *Non-Interactive Zero-Knowledge Proofs of Knowledge*. Since then, a tremendous amount of work has been dedicated to improving those proofs [13,24,27]. And more recently, a trend studied Quasi-Adaptive NIZK allowing more efficient protocols [2,30–32,35,36]. In this last paper, the authors proposed one of the first PAKE UC-Secure even in the presence of adaptive corruptions, that manages to have flows independent of the password length.

Both Smooth Projective Hash Functions and Quasi-Adaptive Non-Interactive Zero-Knowledge allow to produce very efficient protocols, with a very similar structure, and thus security arguments.

Contributions: Our paper proposes a generic transformation from Password-Authenticated Key Exchange to Oblivious Transfer. As the reverse transformation was already studied in [18], this allows to study one protocol for the other. Our framework allows to transform a UC-secure PAKE into a UC-secure OT with the same level of security (for instance resistance to adaptive corruptions).

We choose to focus on the transformation of a 2-round PAKE to a 3-round OT since this kind of PAKE is the most commonly encountered and the most efficient one. Furthermore, this allows us to give a generic optimization of our transformation in this specific case, exploiting the fact that the server does not need to hide his "password", since his password is a (public) number of line in the database. Note that our generic transformation could allow to transform an m-round PAKE into an $(m + 1)$-round OT, but the optimization would then have to be tailored to the considered protocol.

After showing an application of our technique on an already existing PAKE scheme from [1], which allows us to immediately recover the associated OT scheme given in their article, we show that our transformation also applies to the PAKE scheme from [32], allowing us to give a nearly optimal OT scheme.

Note that our technique works with every possible PAKE scheme, be they elliptic-curve-based or not. Furthermore, also note that it also allows to transform a BPR-secure PAKE [6] into an OT secure in a game-based security model.

In order to illustrate these last two points, we apply our framework to a (non-UC) secure lattice-based PAKE scheme proposed by [33], giving us a (non-UC) secure lattice-based OT scheme (see the paper full version [12]).

Comparison with Other Existing OT Schemes: The table given in Fig. 1 shows the costs of various known OT schemes based on elliptic curves.

While Barreto-Naehrig curves [5] are considered less and less secure for efficient parameters, we consider for the asymptotic scaling that elements in $\mathbb{G}_2$ are smaller than those in $\mathbb{G}_1$, hence, we switched some elements to $\mathbb{G}_2$ for big enough values of n.

Thanks to the use of a QA-NIZK technique, we manage to get rid of the extra logarithm overhead. And we end up being approximately four times more efficient than existing solutions without this overhead.

Communication cost comparisons of various Elliptic Curve based OT schemes

Paper	Assumption	# Group elements	# Rounds
Static Security (1-out-of-2)			
[40] + [25]	SXDH	51	8
[22]	SXDH	$26 + 7\ \mathbb{Z}_p$	4
Adaptive Security (1-out-of-2)			
[1]	SXDH	$12\ \mathbb{G}_1 + 1\ \mathbb{G}_2 + 2\mathbb{Z}_p$	3
[9][1]	DDH	$15\ \mathbb{G} + 2\ \mathbb{Z}_p$	3
[10]	SXDH	$12\ \mathbb{G}_1 + 4\ \mathbb{G}_2 + 2\ \mathbb{Z}_p$	3
This paper	SXDH	$6\ \mathbb{G}_1 + 2\ \mathbb{G}_2 + 2\ \mathbb{Z}_p$	3
Adaptive Security (1-out-of-n)			
[1]	SXDH	$\log n\ \mathbb{G}_1 + (n + 8 \log n)\ \mathbb{G}_2 + n\ \mathbb{Z}_p$	3
[9][1]	DDH	$(n + 9 \log n + 4)\ \mathbb{G} + 2n\ \mathbb{Z}_p$	3
[10]	SXDH	$4\ \mathbb{G}_1 + (4n + 4)\ \mathbb{G}_2 + n\ \mathbb{Z}_p$	3
This paper	SXDH	$4\ \mathbb{G}_1 + (n + 2)\ \mathbb{G}_2 + n\ \mathbb{Z}_p$	3

Fig. 1. Comparison to existing Oblivious Transfer. [1]It should be noted that the [9] OT does not rely on pairings. Classical implementations suggest that the elements in one of the groups in our scheme will be at least twice as big as those from the non-pairing curve, which would give roughly the same efficiency between both schemes in the 1-out-of-2 case, with an edge for the [9] construction in term of computational requirements. However, the scaling in their paper is then worse when increasing the number of lines (see the 1-out-of-n case).

Organization of the Paper: To present our fast Oblivious Transfer scheme in 3 flows, we first recall basic definitions and security notions needed for the understanding of the paper. Then, we propose our generic transformation from a 2-round PAKE scheme to a 3-round OT scheme and show how it can be generically improved by some optimizations. We then apply our framework to the PAKE from [1], and show it directly leads to their existing OT. We then recall the PAKE scheme described in [32], and show how to construct from it our new nearly optimal OT by applying our framework.

Finally, we sketch in the paper full version [12] a brief example to show how our framework behave on a non-UC lattice-based PAKE and how we get a game-based OT scheme from the BPR-secure PAKE scheme presented in [33].

2 Preliminaries

2.1 Definitions and Notations

In the following, in asymmetric protocols the sender is going to be called $\mathcal{S}$ while the receiver will be noted $\mathcal{R}$. For symmetrical ones like PAKE, we will rather use P_i and P_j. We will note the security parameter $\mathfrak{K}$, and n the size of the database (number of lines) for Oblivious Transfer schemes.

We are going to work in an asymmetric bilinear setting denoted as $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic groups of order p for a $\mathfrak{K}$-bit prime p, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is an efficiently computable (non-degenerated) bilinear map. Define $g_T := e(g_1, g_2)$, which is a generator in $\mathbb{G}_T$.

Definition 1 (DDH assumption). *Assuming a multiplicative group $\mathbb{G}$ of prime order p and generator g, we say that the* DDH *assumption is hard in $\mathbb{G}$ if, given a tuple $(g^x, g^y, g^z) \in \mathbb{G}^3$, it is hard to decide whether $z = xy$ or not.*

Definition 2 (Symmetric External Diffie Hellman (SXDH [4]) assumption). *This variant of* DDH*, used mostly in bilinear groups $(\mathbb{G}_1, \mathbb{G}_2)$ in which no computationally efficient homomorphism exists from $\mathbb{G}_2$ to $\mathbb{G}_1$ or $\mathbb{G}_1$ to $\mathbb{G}_2$, states that the* DDH *assumption is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

Definition 3 ((Labelled) Quasi Adaptive Non-Interactive Zero Knowledge Argument). *Assuming a randomness distribution $\mathcal{D}_{\mathsf{param}}$ (for a public set of parameters* param*) and a set of languages $\{\mathfrak{L}_\rho\}_\rho$ parameterized by a randomness $\rho \leftarrow \mathcal{D}_{\mathsf{param}}$ and associated with a relation $\mathcal{R}_\rho$ (meaning that a word x belongs to $\mathfrak{L}_\rho$ if and only if there exists a witness w such that $\mathcal{R}_\rho(x, w)$ holds), a* QA-NIZK *argument Π for this set of languages $\{\mathfrak{L}_\rho\}_\rho$ consists of five probabilistic polynomial time (PPT) algorithms $\Pi = (\mathsf{Gen}_{\mathsf{param}}, \mathsf{Gen}_{\mathsf{crs}}, \mathsf{Prove}, \mathsf{Sim}, \mathsf{Ver})$ defined as follows:*

- $\mathsf{Gen}_{\mathsf{param}}(\mathfrak{K})$ *returns the public parameters* param*.*
- $\mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho)$ *returns a common reference string* crs *and a trapdoor* tk*. We assume that* crs *contain the parameters* param*, a description of the language $\mathfrak{L}_\rho$ and a description of a label space $\mathcal{L}$.*
- Prove *takes as input* crs*, a label $\ell \in \mathcal{L}$, a word x of the language $\mathfrak{L}_\rho$ and a witness w corresponding to this word. If $(x, w) \notin \mathcal{R}_\rho$, it outputs failure. Otherwise, it outputs a proof π.*
- Ver *takes as input* crs*, a label ℓ, a word x and a proof π. It outputs a bit b (either 1 if the proof is correct, or 0 otherwise).*

- Sim *takes as input* crs*, a trapdoor* tk*, a label* $\ell \in \mathcal{L}$ *and a word* x*. It outputs a proof* π *for* x *(not necessarily in* $\mathfrak{L}_\rho$*) with respect to the label* ℓ*.*

These algorithms must satisfy the following properties

Perfect Completeness: *for all PPT adversary* $\mathcal{A}$*, all security parameter* $\mathfrak{K}$*, all public parameters* $\mathsf{param} \leftarrow \mathsf{Gen}_{\mathsf{param}}(\mathfrak{K})$*, all randomness* $\rho \leftarrow \mathcal{D}_{\mathsf{param}}$*, all label* $\ell \in \mathcal{L}$ *and all* $(x, w) \in \mathcal{R}_\rho$*, the following holds:*

$$\Pr[(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho); \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \ell, x, w) : \mathsf{Ver}(\mathsf{crs}, \ell, x, \pi) = 1] = 1$$

Computational Adaptive Soundness: *For a given security parameter* $\mathfrak{K}$*, a scheme achieves computational adaptive soundness if for all PPT adversary* $\mathcal{A}$*, the probability*

$$\Pr[\mathsf{param} \leftarrow \mathsf{Gen}_{\mathsf{param}}(\mathfrak{K}); \rho \leftarrow \mathcal{D}_{\mathsf{param}}; (\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho);$$
$$(\ell, x, \pi) \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{crs}, \rho) : x \notin \mathfrak{L}_\rho \wedge \mathsf{Ver}(\mathsf{crs}, \ell, x, \pi) = 1]$$

is negligible.

Perfect Zero-Knowledge: *for all* $\mathfrak{K}$*, all* $\mathsf{param} \leftarrow \mathsf{Gen}_{\mathsf{param}}(\mathfrak{K})$*, all* $\rho \leftarrow \mathcal{D}_{\mathsf{param}}$*, all* $(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho)$*, all label* $\ell \in \mathcal{L}$ *and all* $(x, w) \in \mathcal{R}_\rho$*, the distributions* $\mathsf{Prove}(\mathsf{crs}, \ell, x, w)$ *and* $\mathsf{Sim}(\mathsf{crs}, \mathsf{tk}, \ell, x)$ *are the same.*

2.2 Security Models

UC Framework. The goal of the UC framework [16] is to ensure that UC-secure protocols will continue to behave in the ideal way even if executed in a concurrent way in arbitrary environments. It is a simulation-based model, relying on the indistinguishability between the real world and the ideal world. In the ideal world, the security is provided by an ideal functionality $\mathcal{F}$, capturing all the properties required for the protocol and all the means of the adversary. In order to prove that a protocol Π emulates $\mathcal{F}$, one has to construct, for any polynomial adversary $\mathscr{A}$ (which controls the communication between the players), a simulator $\mathscr{S}$ such that no polynomial environment $\mathcal{Z}$ can distinguish between the real world (with the real players interacting with themselves and $\mathscr{A}$ and executing the protocol π) and the ideal world (with dummy players interacting with $\mathscr{S}$ and $\mathcal{F}$) with a significant advantage. The adversary can be either *adaptive*, *i.e.* allowed to corrupt users whenever it likes to, or *static*, *i.e.* required to choose which users to corrupt prior to the execution of the session sid of the protocol. After corrupting a player, $\mathscr{A}$ has complete access to the internal state and private values of the player, takes its entire control, and plays on its behalf.

Simple UC Framework. Canetti et al. formalized a simpler variant in [17], that we use here. This simplifies the description of the functionalities for the following reasons (in a nutshell): All channels are automatically assumed to be authenticated (as if we worked in the $\mathcal{F}_{\text{AUTH}}$-hybrid model); There is no need for *public delayed outputs* (waiting for the adversary before delivering a message to a party), neither for an explicit description of the corruptions. We refer the interested reader to [17] for details.

Password Authenticated Key Exchange. Before the UC framework, the classical security model for PAKE protocols was the so-called BPR model [6]. The first ideal functionality for PAKE protocols in the UC framework [16,20] was proposed by Canetti *et al.* [19], who showed how a simple variant of the Gennaro-Lindell methodology [26] could lead to a secure protocol.

We present the PAKE ideal functionality $\mathcal{F}_{pwKE}$ on Fig. 2. It was described in [19].

The functionality $\mathcal{F}_{\mathrm{pwKE}}$ is parameterized by a security parameter k. It interacts with an adversary $\mathscr{S}$ and a set of parties P_1,…,P_n via the following queries:

- **Upon receiving a query** $(\mathtt{NewSession}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ **from party** P_i**:**
  Send $(\mathtt{NewSession}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to $\mathscr{S}$. If this is the first `NewSession` query, or if this is the second `NewSession` query and there is a record $(\mathsf{sid}, \mathsf{ssid}, P_j, P_i, \mathsf{pw}')$, then record $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ and mark this record `fresh`.
- **Upon receiving a query** $(\mathtt{TestPwd}, \mathsf{sid}, \mathsf{ssid}, P_i, \mathsf{pw}')$ **from the adversary** $\mathscr{S}$**:**
  If there is a record of the form (P_i, P_j, pw) which is `fresh`, then do: If $\mathsf{pw} = \mathsf{pw}'$, mark the record `compromised` and reply to $\mathscr{S}$ with "correct guess". If $\mathsf{pw} \neq \mathsf{pw}'$, mark the record `interrupted` and reply with "wrong guess".
- **Upon receiving a query** $(\mathtt{NewKey}, \mathsf{sid}, \mathsf{ssid}, P_i, \mathsf{sk})$ **from the adversary** $\mathscr{S}$**:**
  If there is a record of the form $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$, and this is the first `NewKey` query for P_i, then:
  - If this record is `compromised`, or either P_i or P_j is corrupted, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sk})$ to player P_i.
  - If this record is `fresh`, and there is a record (P_j, P_i, pw') with $\mathsf{pw}' = \mathsf{pw}$, and a key sk' was sent to P_j, and (P_j, P_i, pw) was `fresh` at the time, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sk}')$ to P_i.
  - In any other case, pick a new random key sk' of length $\mathfrak{K}$ and send $(\mathsf{sid}, \mathsf{ssid}, sk')$ to P_i.

  Either way, mark the record $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ as `completed`.

Fig. 2. Ideal Functionality for PAKE $\boldsymbol{\mathcal{F}}_{\mathrm{pwKE}}$

The main idea behind this functionality is as follows: If neither party is corrupted and the adversary does not attempt any password guess, then the two players both end up with either the same uniformly-distributed session key if the passwords are the same, or uniformly-distributed independent session keys if the passwords are distinct. In addition, the adversary does not know whether this is a success or not. However, if one party is corrupted, or if the adversary successfully guessed the player's password (the session is then marked as `compromised`), the adversary is granted the right to fully determine its session key. There is in fact nothing lost by allowing it to determine the key. In case of a wrong guess (the session is then marked as `interrupted`), the two players are given independently-chosen random keys. A session that is nor `compromised` nor `interrupted` is called `fresh`, which is its initial status.

Finally notice that the functionality is not in charge of providing the password(s) to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This guarantees security even in the case where two honest players execute the protocol with two different passwords: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where the passwords are related passwords and/or used in different protocols. Also note that allowing the environment to choose the passwords guarantees forward secrecy.

In case of corruption, the adversary learns the password of the corrupted player; After the `NewKey` -query, it additionally learns the session key.

Oblivious Transfer. The ideal functionality of an Oblivious Transfer (OT) protocol was given in [1,16,22]. We recall it in simple UC in Fig. 3 using the functionality introduced in [11]. Note that the BPR model [6] given for PAKE protocols can be adapted to give a game-based security model for OT schemes but this is well beyond the scope of this paper.

The party P_i is the sender $\mathcal{S}$, while the party P_j is the receiver $\mathcal{R}$. The former is provided with a database consisting of a set of n lines $(L_1, \ldots, L_n)$, while the latter is querying a particular line L_s (with $s \in \{1, \ldots, n\}$). Since there is no communication between them (the functionality deals with everything), it automatically ensures the oblivious property on both sides (the sender does not learn which line was queried, while the receiver does not learn any line other than L_s).

3 Generic Construction of an Oblivious Transfer from a UC-Secure PAKE

There is a trend in proven cryptography that consists in finding the link between primitives and show which primitives can be used to build other ones. A well-known example is the transformation of an IBE encryption scheme into a signature scheme, which was proposed by Naor as recalled in [15], and its reverse transformation from an affine MAC or signature scheme into an IBE scheme given in [14].

A neat result, presented by Canetti et al. in [18], shows how a UC-secure Oblivious Transfer scheme can be transformed into a UC-secure PAKE scheme. In this section we are going to go the other way, and show how a UC-secure PAKE can be transformed into a UC-secure Oblivious Transfer scheme. While

The functionality $\mathcal{F}_{(1,n)\text{-OT}}$ is parametrized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathscr{S}$ and a set of parties $P_1,\ldots,P_N$ via the following queries:

- **Upon receiving an input $(\texttt{Send}, \mathsf{sid}, \mathsf{ssid}, \boldsymbol{P_i}, \boldsymbol{P_j}, (\boldsymbol{L_1}, \ldots, \boldsymbol{L_n}))$ from party $\boldsymbol{\mathcal{P}_i}$**, with $L_i \in \{0,1\}^{\mathfrak{K}}$ for all i: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (L_1, \ldots, L_n))$ and reveal $(\texttt{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathscr{S}$. Ignore further $\texttt{Send}$-message with the same ssid from P_i.
- **Upon receiving an input $(\texttt{Receive}, \mathsf{sid}, \mathsf{ssid}, \boldsymbol{P_i}, \boldsymbol{P_j}, \boldsymbol{s})$ from party $\boldsymbol{P_j}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (L_1, \ldots, L_n))$ is not recorded. Otherwise, reveal $(\texttt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathscr{S}$ and send $(\texttt{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, L_s)$ to P_j and ignore further $\texttt{Receive}$-message with the same ssid from P_j.

Fig. 3. Ideal functionality for 1-out-of-n oblivious transfer $\boldsymbol{\mathcal{F}}_{(1,n)\text{-OT}}$

considering this direction may seem like transforming a very strong primitive into a weaker one, this will allow us to prepare a framework to propose the most efficient Oblivious Transfer scheme to date. This also echoes to the results from [39] on a simulation-based transform.

3.1 Description of a UC-Secure PAKE Scheme

As explained in the introduction, we are going to consider PAKE protocols in two flows, since there already exists a multitude of PAKE schemes satisfying this requirement, and since minimizing the number of flows is one of the most important issues in modern instantiations. This allows us to generically give an optimization, leading to more efficient protocols.

In order to present the framework or our transformation, we formally split a PAKE scheme into the following four algorithms[1].

- Setup: An algorithm that generates the initial crs.
- $\mathsf{flow}_1(\mathsf{pw}_i; \rho_i, \rho'_i)$: The algorithm run by the user P_i, using some randomness ρ_i and ρ'_i and possessing the password pw_i, to generate the first flow of the protocol. We note ρ_i the part of the randomness involved in hiding the password, and ρ'_i the remainder.
- $\mathsf{flow}_2(\mathsf{flow}_1, \mathsf{pw}_j; \rho_j, \rho'_j)$: The algorithm run by the user P_j, using some randomness ρ_j and possessing the password pw_j, after receiving the flow flow_1, to generate the second flow of the protocol. This algorithm also produces P_j's view of the key: $K_i^{(j)}$.
- $\mathsf{Decap}(\mathsf{flow}_2, \mathsf{pw}_i, f(\rho_i, \rho'_i))$: The algorithm run by P_i when receiving the flow flow_2 from P_j, using his password pw_i and a function $f(\rho_i, \rho'_i)$ of the randomness initially used and the remainder (which function can be anything from the identity to the null function), to recover P_i's view of the key: $K_j^{(i)}$.

[1] Since it was shown in [19] that UC-secure PAKE is impossible in the plain model, we focus on the CRS model for the ease of readability.

3.2 Generic Construction of a UC-Secure OT Scheme

We denote by $\mathsf{DB} = (L_1, \ldots, L_n)$ the database of the server containing n lines, and by s the line requested by the user in an oblivious way. We assume the existence of a CPA-Secure encryption scheme, whose public parameters are going to be included in the public common reference string crs generated by the setup algorithm of the PAKE scheme. Such an encryption scheme $\mathcal{E}$ is described through four algorithms $(\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$:

- $\mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters $\mathsf{param}_{\mathrm{cpa}}$ of the scheme;
- $\mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ outputs a pair of keys: a (public) encryption key pk and a (private) decryption key sk;
- $\mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, M; \rho)$ outputs a ciphertext $\boldsymbol{c} = \mathcal{E}(M)$ on the message M, under the encryption key pk, using the randomness ρ;
- $\mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, \boldsymbol{c})$ outputs the plaintext M encrypted in the ciphertext $\boldsymbol{c}$, or $\perp$.

We also assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database.

Using the PAKE scheme, the encryption scheme and the PRG, one can now obtain an OT scheme from a PAKE scheme, as described in Fig. 4. The idea of the protocol is as follows:

- First, a setup phase enables to generate the CRS, both for the PAKE and encryption schemes.
- The pre-flow is a technical requirement for the UC proof.
- When querying for line s, the receiver proceeds in three steps. The first one consists in generating a masking value R (technical requirement for the UC proof), the second one consists in running the first flow of the PAKE scheme for password s to generate the first flow denoted[2] as flow_0, and the third one consists in erasing anything but the needed values, and sending flow_0 to the sender.
- When answering to this query, the sender also proceeds in three steps. The first one consists in recovering the value R and the second one consists in running, for each line $k \in \{1, \ldots, n\}$, the second flow of the PAKE scheme to generate the second flow denoted as flow_k (using k as the password) as well as $\mathcal{S}$'s view of the session key, denoted as $(K_k)_{\mathcal{R}}^{(\mathcal{S})}$. Finally, in the third step, the server computes a xor Q_k of the line L_k and this session key and the value R and sends the set (flow_k, Q_k) back to $\mathcal{R}$.
- When receiving this set of values, the receiver uses flow_s to run the decapsulation algorithm of the PAKE scheme (with password s) to recover his view of the session key, denoted as $(K_s)_{\mathcal{S}}^{(\mathcal{R})}$. He finally uses R, Q_s and $(K_s)_{\mathcal{S}}^{(\mathcal{R})}$ to recover the expected line L_s.

[2] Note that we now denote as flow_0 (and not flow_1) the flow generated by the PAKE algorithm flow_1, in order to avoid the confusion with the flow_1 message generated by the sender (for line 1) during the database answer phase.

The correctness easily follows from the correctness of the PAKE protocol, since the views of $\mathcal{R}$ and $\mathcal{S}$ of the session key for the PAKE scheme executed for password s are the same. The scheme is oblivious with respect to $\mathcal{S}$ since the first flow of the PAKE scheme executed by $\mathcal{R}$ does not reveal any information on s. And it is oblivious with respect to $\mathcal{R}$ since the correctness of the PAKE scheme gives him no information on the session keys $(K_k)_{\mathcal{R}}^{(\mathcal{S})}$ for $k \neq s$. The security is stated in Theorem 4.

CRS generation:
$\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.

Pre-Flow:

1. $\mathcal{S}$ generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by $\mathsf{KeyGen}_{\mathrm{cpa}}$.
2. $\mathcal{S}$ publishes pk.

Index query on s:

1. $\mathcal{R}$ picks a random J, computes $R \leftarrow F(J)$ and sets $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$.
2. $\mathcal{R}$ picks a random (ρ, ρ'), and runs $\mathsf{flow}_1(s; \rho, \rho')$ to obtain flow_0.
3. $\mathcal{R}$ stores $f(\rho, \rho')$ needed for the Decap algorithm and R and completely erases the rest, including the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends (c, flow_0) to $\mathcal{S}$.

Database answer:

1. $\mathcal{S}$ decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $R \leftarrow F(J)$.
2. For each line k:
   - $\mathcal{S}$ picks a random (ρ_k, ρ'_k), runs $\mathsf{flow}_2(\mathsf{flow}_0, k; \rho_k, \rho'_k)$ to generate flow_k and $(K_k)_{\mathcal{R}}^{(\mathcal{S})}$.
   - $\mathcal{S}$ then computes $Q_k = L_k \oplus (K_k)_{\mathcal{R}}^{(\mathcal{S})} \oplus R$.
3. $\mathcal{S}$ erases everything except $(\mathsf{flow}_k, Q_k)_{k \in [\![1,n]\!]}$ and sends it to $\mathcal{R}$.

Data recovery:
$\mathcal{R}$ then using $f(\rho, \perp)$ computes $(K_s)_{\mathcal{S}}^{(\mathcal{R})} = \mathsf{Decap}(\mathsf{flow}_s, s, f(\rho, \rho'))$, sets $L_s = Q_s \oplus (K_s)_{\mathcal{S}}^{(\mathcal{R})} \oplus R$, and erases everything else.

Fig. 4. UC-Secure 1-out-of-n OT from a UC-Secure PAKE

Theorem 4. *Under the* UC-*security*[3] *of the* PAKE *protocol, the existence of a Pseudo-Random Generator (*PRG*) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database, and the* $\mathsf{IND\text{-}CPA}$ *security of the encryption scheme, the transformation presented*

[3] Note that the transformation also allows to obtain an OT scheme from a BPR-secure PAKE scheme in a game-based security model, but since this is of less interest, we do not formally prove it due to lack of space.

in Fig. 4 achieves a UC*-secure 1-out-of-n Oblivious Transfer scheme, with the same handling of corruptions as in the* PAKE *protocol.*

The high-level idea of the proof is quite simple. Recall that in a UC proof, one has to simulate one of the two players in front of a corrupted player. The first flow allows the simulated server to extract the requested line. One will then be able to send random noise on all the lines besides the correct one. The PAKE functionality ensures that this is possible. Then again for honest users, the simulator will replace the remaining line by a random noise. In case of corruptions of the server, one relies on the security of the CPA encryption, to ensure that everything was done honestly (the adversary is not able to test the validity of the retrieved R, and so for the correct line as $L \oplus K \oplus R$ where L and K are the honest values, and R is simply set as $Q \ominus (L \oplus K)$). In case of corruption of the user, we rely on the PAKE handling of user corruption to adapt the memory.

A more detailed proof can be found in the paper full version [12].

3.3 Generic Optimization

It should be noted that in the previous transformation we never used the fact that the first user in the PAKE scheme (here, the receiver) does not learn the password from the second (here, the sender). This is a property required by PAKE but not useful in the transformation (since the sender executes n parallel PAKE schemes, using the (public) number k of the lines as the password). This argument is the key of the optimization we now propose in Fig. 5. The difference is that in the index query, the receiver does not pick the second randomness ρ' and in the database answer, the randomness ρ_k is not used anymore. The proof remains the same as the previous one.

4 Warm-Up: Applying the Framework on [1]

4.1 Notations

Since in [1], the authors proposed both a UC-secure PAKE and a UC-secure Oblivious Transfer constructions, this section serves as an example of application of our framework. More precisely, we apply here our (optimized) transformation to their UC-secure PAKE and show that we obtain exactly their UC-secure Oblivious Transfer.

There protocols make use of Commitment schemes and Smooth Projective Hash Functions. Rigorous definitions can be found in the paper full version [12] but we give here the notations used in the following.

A commitment scheme is a two-party primitive (between a committer and a receiver) divided into two phases. In a first *commit* phase, the committer gives the receiver an analogue of a sealed envelope containing a value m, while in the second *opening* phase, the committer reveals m in such a way that the receiver can verify it was indeed m that was contained in the envelope. It is required that a committer cannot change the committed value (*i.e.*, he should not be

CRS generation:
$\mathsf{crs} \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \stackrel{\$}{\leftarrow} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.

Pre-Flow:

1. $\mathcal{S}$ generates a key pair $(\mathsf{pk}, \mathsf{sk}) \stackrel{\$}{\leftarrow} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by $\mathsf{KeyGen}_{\mathrm{cpa}}$.
2. $\mathcal{S}$ publishes pk.

Index query on s**:**

1. $\mathcal{R}$ picks a random J, computes $R \leftarrow F(J)$ and sets $c \stackrel{\$}{\leftarrow} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$.
2. $\mathcal{R}$ picks a random ρ, and runs $\mathsf{flow}_1(s; \rho, \bot)$ to obtain flow_0.
3. $\mathcal{R}$ stores $f(\rho, \bot)$ needed for the Decap algorithm and R and completely erases the rest, including the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends (c, flow_0) to $\mathcal{S}$.

Database answer:

1. $\mathcal{S}$ decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $R \leftarrow F(J)$.
2. For each line k:
   - $\mathcal{S}$ picks a random ρ'_k, and runs $\mathsf{flow}_2(\mathsf{flow}_0, k; \bot, \rho'_k)$ to generate flow_k and $(K_k)^{(\mathcal{S})}_{\mathcal{R}}$.
   - $\mathcal{S}$ then computes $Q_k = L_k \oplus (K_k)^{(\mathcal{S})}_{\mathcal{R}} \oplus R$.
3. $\mathcal{S}$ erases everything except $(\mathsf{flow}_k, Q_k)_{k \in [\![1,n]\!]}$ and sends it to $\mathcal{R}$.

Data recovery: $\mathcal{R}$ then using $f(\rho, \bot)$ computes $(K_s)^{(\mathcal{R})}_{\mathcal{S}} = \mathsf{Decap}(\mathsf{flow}_s, s, f(\rho, \bot))$, sets $L_s = Q_s \oplus (K_s)^{(\mathcal{R})}_{\mathcal{S}} \oplus R$, and erases everything else.

Fig. 5. Optimized UC-Secure 1-out-of-n OT from a UC-Secure PAKE

able to open to a value different from the one he committed to), this is called the *binding* property. It is also required that the receiver cannot learn anything about m before the opening phase, this is called the *hiding* property.

Formally, a *non-interactive labelled commitment scheme* $\mathcal{C}$ is defined by three algorithms:

- $\mathsf{SetupCom}(1^{\mathfrak{K}})$ takes as input the security parameter $\mathfrak{K}$ and outputs the global parameters, passed through the CRS ρ to all other algorithms;
- $\mathsf{Com}^{\ell}(x; \rho)$ takes as input a label ℓ, a message x and a random ρ, and outputs a pair (C, δ), where C is the commitment of x for the label ℓ, and δ is the corresponding opening data (a.k.a. decommitment information).
- $\mathsf{VerCom}^{\ell}(C, x, \delta)$ takes as input a commitment C, a label ℓ, a message x, and the opening data δ and outputs 1 (true) if δ is a valid opening data for C, x and ℓ. It always outputs 0 (false) on $x = \bot$.

Smooth projective hash functions are defined such as their value can be computed in two different ways if the input belongs to a particular subset (the *language*), either using a private hashing key or a public projection key along with a private witness ensuring that the input belongs to the language. The projection key is derived from the hashing key. The hash value obtained is indistinguishable from random in case the input does not belong to the language (property of *smoothness*) and in case the input does belong to the language but no witness is known (property of *pseudo-randomness*).

Formally, a Smooth Projective Hash Function over a language $\mathfrak{L} \subset X$, onto a set $\mathcal{G}$, is defined by five algorithms $(\mathsf{Setup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$:

- $\mathsf{Setup}(1^{\mathfrak{K}})$ where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme, and the description of an $\mathcal{NP}$ language $\mathfrak{L}$;
- $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$, outputs a hashing key hk for the language $\mathfrak{L}$;
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$, derives the projection key hp from the hashing key hk.
- $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$, outputs a hash value $v \in \mathcal{G}$, thanks to the hashing key hk and W.
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$, outputs the hash value $v' \in \mathcal{G}$, thanks to the projection key hp and the witness w that $W \in \mathfrak{L}$.

4.2 Description of Their UC-Secure PAKE Scheme

Description of the Scheme: Both parties (denoted as P_b and $P_{\overline{b}}$) want to share a key by using a common password pw. This will thus fix a language $\mathfrak{L}_{\mathsf{pw}}$, which is the set of valid commitments of pw, for a smooth projective hash function. Both flows are done simultaneously. In flow_b, each party P_b will generate a pair of hash keys $(\mathsf{hk}_b, \mathsf{hp}_b)$ and then commit to the password pw_b in C_b. Note that P_b stores the witness δ_b that this value C_b is a valid commitment of pw_b and sends (hp_b, C_b) to the other party.

In the decapsulation phase, each party P_b then receives the commitment and the projection key of the other party and computes two hash values: On the one hand, the projected hash value using his own commitment C_b along with the witness δ_b and the projected key $\mathsf{hp}_{\overline{b}}$ sent by the other party, and on the other hand, the (regular) hash value using the received commitment $C_{\overline{b}}$ and the secret hash key hk_b. The shared session key is then defined as the product of these two values.

The commitment scheme used has specific properties explained in [1] well beyond the scope of the paper. It needs to be compatible with SPHF, and both extractable and equivocable.

Completeness: If the two parties used the same password pw, then at the end of the protocol they will share the same value. Thanks to the property of the SPHF, the hash value and the projected value (on the same commitment C_b) will be equal because the commitment belongs to the good language $\mathfrak{L}_{\mathsf{pw}}$ with the good witness δ_b.

4.3 Applying the Framework to Obtain a UC-Secure OT Scheme

Applying our (optimized) framework, we manage to fall back directly to the Oblivious Transfer that was proposed simultaneously in [1] (Fig. 6).

- Setup: $\rho \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}})$.
- $\mathsf{flow}_b(\mathsf{pw}_b; \rho_b)$:
  - Generates $\mathsf{hk}_b \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L}_{\mathsf{pw}_b})$, $\mathsf{hp}_b \leftarrow \mathsf{ProjKG}(\mathsf{hk}_b, \mathfrak{L}_{\mathsf{pw}_b}, \perp)$
  - picks additional ρ'_b and computes $(C_b, \delta_b) \xleftarrow{\$} \mathsf{Com}^{\ell_b}(\mathsf{pw}_b; \rho'_b)$ with $\ell_b = (\mathsf{sid}, \mathsf{ssid}, P_b, P_{\bar{b}}, \mathsf{hp}_b)$. Sets $\rho_b = (\mathsf{hk}_b, \rho'_b)$.
  - stores hk_b, δ_b, completely erases random coins used before and sends (hp_b, C_b).
- $\mathsf{Decap}(\mathsf{flow}_b, \mathsf{pw}_b, f(\rho_b) = \mathsf{hk}_b, \delta_b)$:
  - Computes $H'_b \leftarrow \mathsf{ProjHash}(\mathsf{hp}_{\bar{b}}, \mathfrak{L}_{\mathsf{pw}_b}, (\ell_b, C_b), \delta_b)$ and $H_{\bar{b}} \leftarrow \mathsf{Hash}(\mathsf{hk}_b, \mathfrak{L}_{\mathsf{pw}_b}, (\ell_{\bar{b}}, C_{\bar{b}}))$ with $\ell_{\bar{b}} = (\mathsf{sid}, \mathsf{ssid}, P_{\bar{b}}, P_b, \mathsf{hp}_{\bar{b}})$
  - Computes $\mathsf{sk}_b = H'_b \cdot H_{\bar{b}}$ and erases everything else, except pw_b.

Fig. 6. UC-Secure PAKE from [1]

CRS generation:
$\rho \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.

Pre-flow :

1. $\mathcal{S}$ generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by $\mathsf{KeyGen}_{\mathrm{cpa}}$.
2. $\mathcal{S}$ publishes pk.

Index query on s:

1. $\mathcal{R}$ chooses a random value J, computes $R \leftarrow F(J)$ and encrypts J under pk: $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$
2. $\mathcal{R}$ computes $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(s)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{R}, \mathcal{S})$
3. $\mathcal{R}$ stores δ_s, R, completely erases the random coins used and sends C and c to $\mathcal{S}$

Database answer $(L_1, \ldots, L_n)$**:**

1. $\mathcal{S}$ decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and gets $R \leftarrow F(J)$
2. For each line $k = 1, \ldots, n$,
   - $\mathcal{S}$ computes $\mathsf{hk}_k \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L}_k)$, $\mathsf{hp}_k \leftarrow \mathsf{ProjKG}(\mathsf{hk}_k, \mathfrak{L}_k, (\ell, C))$, $K_k \leftarrow \mathsf{Hash}(\mathsf{hk}_k, \mathfrak{L}_k, (\ell, C))$,
   - $\mathcal{S}$ computes $Q_k \leftarrow R \oplus K_k \oplus L_k$
3. $\mathcal{S}$ erases everything except $(\mathsf{hp}_k, Q_k)_{k=1,\ldots,n}$ and sends it to $\mathcal{R}$.

Data recovery:
Upon receiving $(\mathsf{hp}_k, Q_k)_{k=1,\ldots,n}$, $\mathcal{R}$ computes $K_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_s, \mathfrak{L}_s, (\ell, C), \delta)$ and gets $L_s \leftarrow R \oplus K_s \oplus Q_s$.
Then $\mathcal{R}$ erases everything except L_s.

Fig. 7. UC-Secure 1-out-of-n OT from an SPHF-Friendly Commitment

Theorem 5. *As the* PAKE *was UC-Secure in presence of adaptive corruption under* SXDH*, the Oblivious Transfer scheme presented in Fig. 7 is a secure 1-out-of-n Oblivious Transfer, with the same handling of corruptions as the* PAKE *protocol under* SXDH *and the* IND-CPA *security of the encryption scheme.*

5 New Efficient Oblivious Transfer Based on QA-NIZK

5.1 Description of Their UC-Secure PAKE Scheme

We now consider the instantiation from [32] in which the UC-PAKE allows for each party to hide his own password. As before, we first recall their protocol and then show how to use it in order to instantiate a new UC-secure Oblivious Transfer. This OT scheme ends up being the most efficient to date, as we show by comparing to the recent [10] in Fig. 8.

Paper	Pre-Flow	Query	Answer
[9]	$2\ \mathbb{G}$	$9 \log n + 2\ \mathbb{G}$	$n\ \mathbb{G} + 2n\ \mathbb{Z}_p$
[10]	$2\ \mathbb{G}_2$	$4\ \mathbb{G}_1\ +\ 2\ \mathbb{G}_2$	$4n\ \mathbb{G}_2\ +\ n\ \mathbb{Z}_p$
This paper	$2\ \mathbb{G}_2$	$4\ \mathbb{G}_1$	$n\ \mathbb{G}_2\ +\ n\ \mathbb{Z}_p$

Fig. 8. Comparison to previous schemes

Interestingly this protocol does not use Smooth Projective Hash Function, but Quasi Adaptive Non-Interactive Zero Knowledge proofs. It is presented in Fig. 9.

CRS generation:
$g_1 \xleftarrow{\$} \mathbb{G}_1, g_2 \xleftarrow{\$} \mathbb{G}_2, a, c, o, d, e, u_1, u_2 \xleftarrow{\$} \mathbb{Z}_q, H \xleftarrow{\$} \mathcal{H}$
Set $A = g_1^a, D = g_1^d, E = g_1^e, U_1 = g_1^{u_1}, U_2 = g_1^{u_2} \in \mathbb{G}_1$
And $B = g_2^c, O = g_2^o, V_1 = g_2^{u_1 c - d - oa}, V_2 = g_2^{u_2 c - e} \in \mathbb{G}_2$
$\mathsf{crs} := (g_1, g_2, A, B, O, F, D, E, U_1, U_2, V_1, V_2, H)$

$\mathsf{flow}_b(\mathsf{pw}; (r_b, s_b))$**:**

1. P_b computes $R_b = g_1^{r_b}, S_b = \mathsf{pw} \cdot A^{r_b}, T_b = (D \cdot E^{h_b})^{r_b}, \hat{\rho}_b = B^{s_b}, W_b = (U_1 \cdot U_2^{h_b})^{r_b}$, where $h_b = H(\mathbf{sid}, \mathbf{ssid}, P_b, P_{\bar{b}}, R_1, S_1, \hat{\rho}_b)$
2. P_b erases r_b, sends $R_b, S_b, T_b, \hat{\rho}_b$ and keeps W_b.

Key computing:

1. P_b computes $h_{\bar{b}} = H(\mathbf{sid}, \mathbf{ssid}, P_{\bar{b}}, P_b, R_{\bar{b}}, S_{\bar{b}}, \hat{\rho}_{\bar{b}})$
Checks that all elements are consistent, and computes:
$\rho = g_2^{s_b}, \theta = O^{s_b}, \gamma = (V_1 \cdot V_2^{h_{\bar{b}}})^{s_b}$
$\mathsf{sk} = e(T_{\bar{b}}, \rho) \cdot e(S_{\bar{b}}/\mathsf{pw}, \theta) \cdot e(R_{\bar{b}}, \gamma) \cdot e(W_b, \hat{\rho}_{\bar{b}})$

Fig. 9. UC secure PAKE from [32]

As explained in the original paper, the scheme is loosely-based on Quasi-Adaptive NIZK, in the sense that the flows use computation very close to what would be expected. In the Fig. 9, $\mathcal{H}$ is a family of hash functions.

5.2 Applying the Framework to Obtain a UC-Secure OT Scheme

Using the instantiation from [32] described in Fig. 9, we apply our framework and directly obtain the protocol described in Fig. 10.

CRS generation:
$g_1 \xleftarrow{\$} \mathbb{G}_1, g_2 \xleftarrow{\$} \mathbb{G}_2, a, c, o, d, e, u_1, u_2 \xleftarrow{\$} \mathbb{Z}_p, H \xleftarrow{\$} \mathcal{H}$
Sets $A = g_1^a, D = g_1^d, E = g_1^e, U_1 = g_1^{u_1}, U_2 = g_1^{u_2} \in \mathbb{G}_1$
And $B = g_2^c, O = g_2^o, V_1 = g_2^{u_1 c - d - oa}, V_2 = g_2^{u_2 c - e} \in \mathbb{G}_2$
$\mathsf{crs} := (g_1, g_2, A, B, O, F, D, E, U_1, U_2, V_1, V_2, H)$

Pre-Flow:

1. $\mathcal{S}$ generates $(\mathsf{pk} = f_1 = g_1^\alpha, \mathsf{sk} = \alpha \xleftarrow{\$} \mathbb{Z}_p)$
2. $\mathcal{S}$ erases random coins, keeps sk and sends pk.

Index query on i:

1. $\mathcal{R}$ picks $J \xleftarrow{\$} \mathbb{G}_1, s \xleftarrow{\$} \mathbb{Z}_p$, sets $\mathsf{mask} \leftarrow F(J)$ and $c = (f_1^s J, g_1^s)$
2. $\mathcal{R}$ picks $r \xleftarrow{\$} \mathbb{Z}_p$ computes $R = g_1^r, S = G(i) \cdot A^r, T = (D \cdot E^{h_{\mathcal{R},\mathcal{S}}})^r, W = (U_1 \cdot U_2^{h_{\mathcal{R},\mathcal{S}}})^r$, where $h_{\mathcal{R},\mathcal{S}} = H(\mathbf{sid}, \mathbf{ssid}, \mathcal{R}, \mathcal{S}, R, S, c)$
3. $\mathcal{R}$ erases r, s, J, sends $c, C = (R, S, T)$ and keeps W, mask.

Database answer:

1. $\mathcal{S}$ sets $J = c_1 / c_2^\alpha$, $\mathsf{mask} = F(J)$ and $h_{\mathcal{R},\mathcal{S}} = H(\mathbf{sid}, \mathbf{ssid}, \mathcal{R}, \mathcal{S}, R, S, c)$
2. For $k = 1, \ldots, n$:
   (a) Picks $s_k \xleftarrow{\$} \mathbb{Z}_p$
   (b) Sets $\rho_k = B^{s_k}, \rho'_k = g_2^{s_k}, \theta_k = O^{s_k}, \gamma_k = (V_1 \cdot V_2^{h_{\mathcal{R},\mathcal{S}}})^{s_k}$
   (c) Computes $K_{\mathcal{R},\mathcal{S},k} = e(T, \rho'_k) \cdot e(S/G(k), \theta_k) \cdot e(R, \gamma_k)$
   (d) Sets $Q_k = L_k \oplus \mathcal{H}(K_{\mathcal{R},\mathcal{S},k} \oplus \mathsf{mask})$
3. $\mathcal{S}$ sends all $(Q_k, \rho_k)_{k \in [\![1,n]\!]}$ and erases everything else.

Data recovery:
$\mathcal{R}$ computes $K_{\mathcal{R},\mathcal{S},i} = e(W, \rho_i)$, and recovers $L_i = Q_i \oplus \mathcal{H}(K_{\mathcal{R},\mathcal{S},i} \oplus \mathsf{mask})$

Fig. 10. UC secure OT from QA-NIZK

The function G is assumed to map line numbers to group elements, and it should be collision-resistant and efficiently computable. One can use for example, the G function from Lindell in [37].

Theorem 6. *As the* PAKE *was UC-Secure in presence of adaptive corruption under* SXDH*, and ElGamal is* IND-CPA *under* SXDH *too, the Oblivious Transfer* *scheme presented in Fig. 10 is a secure 1-out-of-n Oblivious Transfer, with the same handling of corruptions as the* PAKE *protocol under* SXDH*.*

The PAKE scheme directly fits in the framework, hence the security is inherited from the initial PAKE proven secure in [32].

The same basic arguments apply: We first start by switching the crs so as to be able to simulate password openings, which will allow an honest receiver to commit to a dummy line value, and retroactively compute the opening value for any possible line s. On the other hand, an honest server will extract the committed line number, and provide dummy answers for all the other ones. If the receiver is honest, then the server will also send a dummy answer for the valid line, since the encrypted value S provides the extra degree of freedom to open to any line value L_k provided belatedly by the environment.

6 Conclusion

We have just presented a framework for generically transforming any PAKE protocol into an Oblivious Transfer protocol. While doing so, we achieved to instantiate the most-efficient Oblivious-Transfer UC-Secure on elliptic curves.

We also propose an Oblivious-Transfer scheme on lattices in the paper full version [12], showing that our framework is not limited to regular elliptic curve cryptography. What remains now, is to find a UC-Secure Lattice based PAKE scheme.

Acknowledgments. This work was supported in part by the French ANR EnBid (ANR-14-CE28-0003) and ID-FIX (ANR-16-CE39-0004) Projects.

References

1. Abdalla, M., Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D.: SPHF-friendly non-interactive commitments. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 214–234. Springer, Heidelberg (2013). doi:10.1007/978-3-642-42033-7_12
2. Abdalla, M., Benhamouda, F., Pointcheval, D.: Disjunctions for hash proof systems: new constructions and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 69–100. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_3
3. Abdalla, M., Chevalier, C., Pointcheval, D.: Smooth projective hashing for conditionally extractable commitments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 671–689. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03356-8_39
4. Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385 (2005). http://eprint.iacr.org/2005/385
5. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006). doi:10.1007/11693383_22

6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000). doi:10.1007/3-540-45539-6_11
7. Bellovin, S.M., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: 1992 IEEE Symposium on Security and Privacy, pp. 72–84. IEEE Computer Society Press, May 1992
8. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New techniques for SPHFs and efficient one-round PAKE protocols. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 449–475. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_25
9. Blazy, O., Chevalier, C.: Generic construction of UC-secure oblivious transfer. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 65–86. Springer, Cham (2015). doi:10.1007/978-3-319-28166-7_4
10. Blazy, O., Chevalier, C.: Structure-preserving smooth projective hashing. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 339–369. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53890-6_12
11. Blazy, O., Chevalier, C., Germouty, P.: Adaptive oblivious transfer and generalization. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 217–247. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53890-6_8
12. Blazy, O., Chevalier, C., Germouty, P.: Almost optimal oblivious transfer from QA-NIZK. Cryptology ePrint Archive, Report 2017/358 (2017). http://eprint.iacr.org/2017/358
13. Blazy, O., Fuchsbauer, G., Izabachène, M., Jambert, A., Sibert, H., Vergnaud, D.: Batch Groth–Sahai. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 218–235. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13708-2_14
14. Blazy, O., Kiltz, E., Pan, J.: (Hierarchical) identity-based encryption from affine message authentication. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 408–425. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44371-2_23
15. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). doi:10.1007/3-540-44647-8_13
16. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: 42nd FOCS, pp. 136–145. IEEE Computer Society Press, October 2001
17. Canetti, R., Cohen, A., Lindell, Y.: A simpler variant of universally composable security for standard multiparty computation. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 3–22. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48000-7_1
18. Canetti, R., Dachman-Soled, D., Vaikuntanathan, V., Wee, H.: Efficient password authenticated key exchange via oblivious transfer. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 449–466. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30057-8_27
19. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005). doi:10.1007/11426639_24
20. Canetti, R., Krawczyk, H.: Universally composable notions of key exchange and secure channels. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 337–351. Springer, Heidelberg (2002). doi:10.1007/3-540-46035-7_22

21. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC, pp. 494–503. ACM Press, May 2002
22. Choi, S.G., Katz, J., Wee, H., Zhou, H.-S.: Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 73–88. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36362-7_6
23. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002). doi:10.1007/3-540-46035-7_4
24. Escala, A., Groth, J.: Fine-tuning Groth-Sahai proofs. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 630–649. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54631-0_36
25. Garay, J.A., Wichs, D., Zhou, H.-S.: Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 505–523. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03356-8_30
26. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003). doi:10.1007/3-540-39200-9_33
27. Ghadafi, E., Smart, N.P., Warinschi, B.: Groth–Sahai proofs revisited. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 177–192. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13013-7_11
28. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78967-3_24
29. Horvitz, O., Katz, J.: Universally-composable two-party computation in two rounds. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 111–129. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74143-5_7
30. Jutla, C.S., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 1–20. Springer, Heidelberg (2013). doi:10.1007/978-3-642-42033-7_1
31. Jutla, C.S., Roy, A.: Switching lemma for bilinear tests and constant-size NIZK proofs for linear subspaces. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 295–312. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44381-1_17
32. Jutla, C.S., Roy, A.: Dual-system simulation-soundness with applications to UC-PAKE and more. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 630–655. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48797-6_26
33. Katz, J., Vaikuntanathan, V.: Smooth projective hashing and password-based authenticated key exchange from lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 636–652. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10366-7_37
34. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 293–310. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19571-6_18
35. Kiltz, E., Wee, H.: Quasi-adaptive NIZK for linear subspaces revisited. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 101–128. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_4

36. Libert, B., Peters, T., Joye, M., Yung, M.: Non-malleability from malleability: simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 514–532. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_29
37. Lindell, Y.: Highly-efficient universally-composable commitments based on the DDH assumption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 446–466. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20465-4_25
38. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: Kosaraju, S.R. (ed.) 12th SODA, pp. 448–457. ACM-SIAM, January 2001
39. Nguyen, M.-H.: The relationship between password-authenticated key exchange and other cryptographic primitives. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 457–475. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30576-7_25
40. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85174-5_31
41. Rabin, M.O.: How to exchange secrets with oblivious transfer. Technical report TR81, Harvard University (1981)

OnionPIR: Effective Protection of Sensitive Metadata in Online Communication Networks

Daniel Demmler, Marco Holz(✉), and Thomas Schneider

Technische Universität Darmstadt, Darmstadt, Germany
{daniel.demmler,thmomas.schneider}@cysec.de, onionpir@marcoholz.de

Abstract. While great effort has been put into securing the *content* of messages transmitted over digital infrastructures, practical protection of *metadata* is still an open research problem. Scalable mechanisms for protecting users' anonymity and hiding their social graph are needed. One technique that we focus on in this work is private information retrieval (PIR), an active field of research that enables private querying of data from a public database without revealing which data has been requested and a fundamental building block for private communication. We introduce two significant improvements for the multi-server scheme RAID-PIR (ACM CCSW'14): precomputing queries using the *Method of four Russians* and optimizing the database layout for parallel queries. We then propose *OnionPIR*, an anonymous messaging service as example application for PIR combined with onion routing that prevents the leakage of communication meta-data. By providing and evaluating a prototype, we show that OnionPIR is usable in practice. Based on our results, we conclude that it is possible to build and deploy such a service today, while its operating expenses are within the order of magnitude of those of traditional messaging services that leak metadata.

Keywords: Private information retrieval · Tor · Privacy · Meta-data protection

1 Introduction

Communication has never been more important for our society than it is today. Large parts of our infrastructure depend on digital computer networks and would collapse when online communication suddenly were not possible anymore. With the rise of instant messengers and the availability of mobile devices, communication shifted from the offline world to online communication platforms. End-to-end encryption is nowadays often deployed to protect the content of exchanged messages such that it cannot be read by an adversary. However, these digital platforms produce large amounts of metadata, i.e., who communicates with whom at what time. Electronic mass surveillance programs strengthen the need for systems providing communication channels without leaking metadata, which has shown to be of tremendous value [Lan15,MMM16], e.g., for people living in

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 599–619, 2017.
DOI: 10.1007/978-3-319-61204-1_30

suppressive regimes, for whom trying to contact government-critical organizations can be a tremendous risk. The right to remain anonymous is a fundamental right in our modern society.

Developing systems that protect the users' privacy and at the same time scale to a large number of users is challenging and a very active research field. Private information retrieval (PIR) can be used as a fundamental building block for those systems and also has many other applications, e.g., private querying of entries from patent registers or medical databases, or improving the scalability of Tor, as proposed in PIR-Tor [MOT+11].

Moreover, building blocks of private and untraceable communication services could potentially also be reused in other privacy-critical applications that are currently under active research, such as electronic voting systems [BV14] or privacy-preserving location-based services [MCA06,HCE11,DSZ14].

Outline and Our Contributions. In Sect. 2, we propose and implement two optimizations for the multi-server PIR scheme RAID-PIR [DHS14] that lead to significant performance improvements. The optimization reduces the server's computation time by using the Method of four Russians for precomputation (Sect. 2.2), thus increasing the overall throughput of the system. The second optimization targets the process of privately retrieving multiple database entries within a single PIR query by adjusting the database layout to increase the number of entries that can be queried in parallel (Sect. 2.3). After introducing multiple generic implementation enhancements (Sect. 2.4), we perform benchmarks to evaluate our optimizations and measure speedups of up to factor 7x over RAID-PIR for many cases when querying through a WAN connection and a performance close to the theoretical optimum when using a DSL connection for the queries (Sect. 2.5).

Based on our improved implementation of PIR, we propose *OnionPIR*, a novel anonymous communication system that combines PIR with onion routing to create a scalable way to privately exchange messages (Sect. 3). OnionPIR is designed to be ready for deployment and provides a way to establish a secure communication channel without the need of exchanging any type of cryptographic keys out-of-band. We built a proof-of-concept implementation and our evaluation demonstrates its practical performance. We will publish our implementations as open-source software upon acceptance of this paper.

Finally, we provide an overview and comparison with related work in Sect. 4.

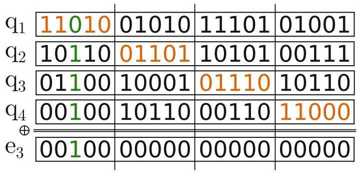
2 Private Information Retrieval and Improvements

Private information retrieval (PIR) is an active research topic and enables the querying of information from one or multiple servers without disclosing which information was requested. Since this technique is a fundamental building block for higher-level protocols and applications, its performance is of prime importance. In this section, we briefly summarize RAID-PIR [DHS14] and introduce and implement two significant performance improvements.

2.1 PIR Background

PIR protocols can be grouped into information-theoretic schemes and computational schemes. RAID-PIR [DHS14], the work that we base our system on, is a multi-server scheme based on the original information-theoretic design by Chor et al. [CKGS95]. Its security guarantees rely on a non-collusion assumption between multiple servers. By using fast XOR operations, it achieves great performance. The alternatives are single-server PIR schemes that rely on computational assumptions, as first introduced in [CGN98] and soon followed by another approach [KO97]. These schemes are typically based on computationally expensive, partially homomorphic encryption. A recent scheme using lattice-based cryptography is presented in [AMBFK16]. Both approaches were combined in [DG14] in a design called hybrid PIR.

Notation. Throughout the paper we use the following notation: The database that may be queried by clients is divided into B blocks of size b bits each and distributed to k servers. When a client wants to query the n-th block from the database, it generates $k - 1$ random bitstrings, called queries, of length B and constructs the k-th bitstring in a way that the XOR of all k queries results in a bitstring where all bits except the n-th are zero. Each of the k queries is then sent to a different server which will XOR all blocks whose corresponding bit i is set in the query. Each server returns the resulting block of XORs of length b bits. To recover the n-th block, the clients computes the XOR of all received blocks.



q1	11010	01010	11101	01001
q2	10110	01101	10101	00111
q3	01100	10001	01110	10110
q4	00100	10110	00110	11000
⊕				
e3	00100	00000	00000	00000

Fig. 1. RAID-PIR Query. The four queries q_i that are sent to the $k = 4$ servers consist of one (orange) *flip chunk* and three (black) randomly generated chunks so that the XOR of all queries is the plaintext query e_3 that retrieves the third database block. (Color figure online)

RAID-PIR improves this scheme by splitting each query into k chunks, as shown in Fig. 1. The queries are constructed in a way that all chunks sent to the l-th server, except the l-th chunk are generated from a seed using a pseudo random generator (PRG). In addition, a redundancy parameter r is introduced as trade-off between security and performance and chosen such that $2 \leq r \leq k$. It allows reducing the downstream bandwidth and server-side computational costs. For that matter, each server will only handle r chunks of the database. This also reduces the number of servers that have to collude in order to retrieve the plaintext of a query from k to r.

Another optimization provides the ability to request multiple blocks of the database within a single query. This is achieved by XORing the requested blocks only within a chunk at the server side and returning one block per chunk instead of one block per query. However, this optimization has the limitation that the requested blocks have to be in different chunks of the database.

In the following sections, we introduce two additional improvements to RAID-PIR that further increase its efficiency.

The first one speeds up the generation of the PIR responses at the server-side by using precomputations (Sect. 2.2), while the second one achieves a significant speedup when querying multiple blocks in parallel by using a database layout where blocks are uniformly distributed (Sect. 2.3).

2.2 Method of Four Russians

The *Method of four Russians* [ADKF70], is a technique for matrix multiplication with a limited number of possible values per entry. Since RAID-PIR makes heavy usage of such kinds of multiplications, this method can be used to reduce the computation time of the RAID-PIR servers for generating responses for PIR queries. This leads to lower latency and higher throughput and comes at the (low) cost of a preprocessing phase for the servers and increased memory requirements.

The Method of four Russians bases on the assumption that the number of possible values for the cells of a matrix is finite. Here, it is assumed that all matrices in the multiplication $\mathsf{X} \cdot \mathsf{Y} = \mathsf{Z}$ are done over the field with two elements ($\mathbb{F}_2$), i.e. only binary elements are being multiplied. Note, that this property is not required for the algorithm (i.e., it is also applicable to numbers in $\mathbb{Z}$) but it reduces the complexity of the algorithm and is sufficient for our use case. In $\mathbb{F}_2$, the multiplicative operation is *AND* and the additive operation is *XOR*.

Crucial for understanding the idea behind the four Russians algorithm, is the fact that the indices of all non-zero elements in row r in the first matrix X indicate a subset of rows in the second matrix Y that have to be XORed in order to produce the r-th row of the output matrix Z. As a naïve approach, it would now be possible to precompute all possible XOR combinations of all rows in Y, hereby creating a lookup table (LUT) that returns the final results for the rows of Z. In other words, a lookup could be performed using each row in X as key to the LUT, returning the corresponding row in Z. This would reduce the computational complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n)$ assuming the lookup is done in constant time for matrices of size $n \times n$. Unfortunately, this naïve approach is impractical due to the exponential computational and memory complexity of the precomputation of $\mathcal{O}(n \cdot 2^n)$, which is worse than the regular approach for $n \geq 4$.

The reason why only n XOR operations have to be performed for each of the 2^n possible combinations of rows of the matrix Y is, that it is possible to use a Gray code to reorder the possible combinations in a way that two consecutive combinations only differ by a single bit that corresponds to only one row of Y. For each of the 2^n combinations, it is now sufficient to XOR the last precomputed result with the row of Y that changed in the Gray code. The binary Gray code g of a number m can be calculated as $g = (m \oplus (m \gg 1))$. In Fig. 2 we depict the calculation of LUT entries for a given example matrix Y^*.

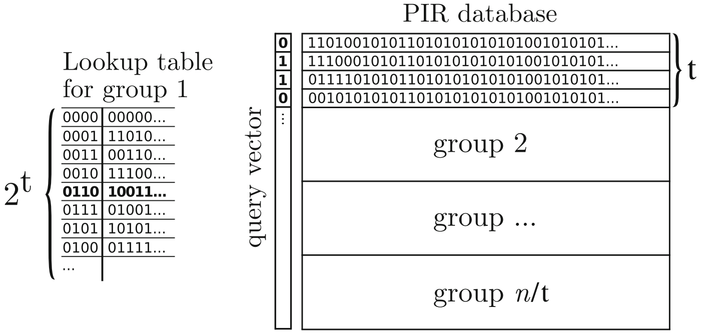
$$\mathsf{Y}^* = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Gray code	decimal	result
0000	0	0000
0001	1	$1011 = 0000 \oplus \mathsf{Y}^*[4,:]$
0011	3	$0010 = 1011 \oplus \mathsf{Y}^*[3,:]$
0010	2	$1001 = 0010 \oplus \mathsf{Y}^*[4,:]$
0110	6	$0101 = 1001 \oplus \mathsf{Y}^*[2,:]$
...	...	...

Fig. 2. Example LUT precomputation for matrix Y^*. $\mathsf{Y}^*[n,:]$ denotes the n-th row of matrix Y^*.

Instead of precomputing a LUT for the full matrix Y, it is divided into groups of $t \in \mathbb{N}$ rows each. The precomputation is then done within these n/t groups using the Gray code as described above, resulting in a computational complexity of $\mathcal{O}\big(2^t \cdot n^2/t\big)$. If $t = \log_2(n)$ the complexity simplifies to $\mathcal{O}\big(n^3/\log_2(n)\big)$, resulting in a speedup of t, compared to traditional matrix multiplication, for all queries that will be answered after the precomputation phase.

While the matrix Y is divided into groups horizontally, the matrix X has to be divided into vertical groups of t columns. To multiply the two $n \times n$ matrices, X is now traversed column-wise, grouped by t columns forming a group each. For each group, the corresponding LUT can now be created by t rows of Y and a lookup can be performed for all t-bit parts of the n rows of X in this group. This procedure is depicted in Fig. 3. We note that this optimization is generic and can potentially be used in any PIR scheme in which queries can be expressed as matrix-matrix or vector-matrix multiplication, e.g., [CKGS95, LG15, Hen16].



0000	00000...
0001	11010...
0011	00110...
0010	11100...
0110	**10011...**
0111	01001...
0101	10101...
0100	01111...
...	

0	11010010101101010101010101001010101...
1	11100010101101010101010101001010101...
1	01111010101101010101010101001010101...
0	00101010101101010101010101001010101...
⋮	

Fig. 3. Precomputation in the PIR database for $t = 4$. A query vector is one *row* in the matrix X. The first 4 bits of the query vector represent the index in the LUT for the first group of Y (the PIR database).

Implementation: An efficient implementation of the Method of four Russians is provided in [ABH10]. However, since it only offers full matrix-matrix multiplications and uses different data structures than RAID-PIR, we implemented our own version and directly integrate it into the RAID-PIR library[1] to avoid costly memory movements and exploit memory locality. While the traditional Method of four Russians assumes the multiplication of two full matrices, this is not the case in RAID-PIR. Often, only a single PIR request has to be answered, making X effectively a vector. Several PIR queries could be batched to create matrix X at the cost of increased latency of the individual queries, similar to [LG15].

In our implementation we do the LUT precomputation once while loading the database and store the LUTs in main memory. For every query we achieve a theoretical speedup of t. The downside of this approach is the increased memory requirement. Without loss of generality, we now assume that Y is a RAID-PIR database of size $B \times b$, i.e., B blocks of b bits each. While a larger t decreases the computation time for the generation of response vectors, it also excessively increases the memory needed to store the LUTs. When t blocks of the PIR database are put in a group, only B/t XOR operations have to be performed per query, resulting in a speedup of factor t. On the other hand, each of the B/t LUTs requires $2^t \cdot b$ bits of memory, where b is the database block size. The choice of $t = 4$ gives a good trade-off between theoretical speedup and memory requirements for larger databases (see Table 1) and is used in our implementation.

Table 1. Comparison of speedup and memory for the Method of four Russians

t (speedup)	2	3	4	5	6	8
$\frac{1}{t} \cdot 2^t$ (memory overhead)	2.00	2.66	4.00	6.40	10.67	32.00

The optimizations were implemented in C and integrated into the existing RAID-PIR library. The precomputation only affects the calculation of the XOR responses by the PIR servers and is completely transparent to PIR clients. This means that clients do not have to be aware of the changes and can send the same queries as before. Furthermore, different PIR server operators could decide independently whether they want to enable the precomputation or not.

In [LG15] a related approach to speedup server computation, based on the Strassen algorithm for matrix multiplication is introduced. This comes at the expense of higher latency for each individual query, as multiple queries are collected and processed together. Furthermore, this approach only introduces a speedup of $q/q^{0.80735} = q^{0.19265}$, where q is the number of queries that are processed together, which is less efficient than our approach (constant speedup t) when q is small (e.g., for $q < 1334$ and $t = 4$).

2.3 Uniform Distribution of Data Entries

In the following we present an optimization that is specific to so-called multi-block (MB) queries as proposed in RAID-PIR, that enable a client to privately request more than one block in a single query. The improvements of RAID-PIR,

[1] https://github.com/encryptogroup/RAID-PIR.

compared to the original CKGS scheme [CKGS95], base on the fact that the PIR blocks are grouped into k chunks that divide the database into equally sized parts. PIR servers receiving an MB query will reply with one block per chunk. The XOR of the corresponding blocks is calculated as before, but only within the boundaries of these chunks. Since only one block per chunk can be queried in one multi-block query, the blocks to retrieve have to be located in different chunks. The performance analysis of RAID-PIR showed that no speedup could be achieved for a large consecutive file residing in a single chunk. However, a large speedup could be measured when querying many smaller files that were distributed among the database. In RAID-PIR, files larger than the block size b are simply placed in adjacent blocks to build the database. While this does not affect the performance of a query when retrieving the blocks consecutively, querying multiple blocks in parallel is often not possible, as they most likely happen to be located in the same chunk.

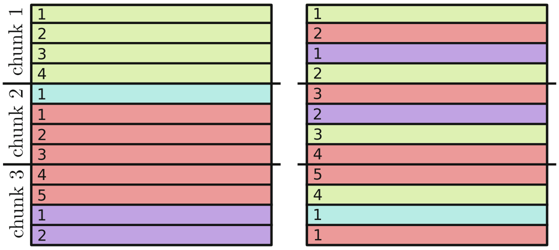


Fig. 4. Uniform distribution of the data entries in the PIR database. While the first file (green) filled up only the first chunk in the original RAID-PIR database layout (left), it is now uniformly distributed over the whole database (right). (Color figure online)

Implementation: To improve the performance of multi-block queries for files larger than the block size b, we change the layout of the database. Instead of placing files in consecutive blocks, they are now uniformly distributed over the whole database, as shown in Fig. 4. For each file, the number of blocks n needed to store its contents is calculated. The file is then placed such that between two of its blocks $B/n - 1$ blocks are used for other files, where B is the total number of blocks. If a chosen block is already assigned to another file, the next possible free block of the database is used instead. The first bytes of the next file will then fill the rest of the previous file's last block. It is guaranteed that all B blocks of the database contain data and no block, except the last, one is only partially filled. Since both the PIR servers and the clients have to know the new locations of the database entries, both client-side and server-side code is extended to support the new layout.

2.4 Generic Implementation Improvements

We also introduced several implementation improvements in RAID-PIR, that are included in all measurements in Sect. 2.5. We pipelined the communication on the client side and decoupled the sending of queries and reception of answers. Furthermore, only a single seed is sent from the client to the servers during a session and a state is kept while the connection is alive. We also introduced SSE2 intrinsics on very wide data types. This feature is available in all x86 CPUs since several years and leads to a more efficient bit-wise XOR computation, which is one of the most crucial operations in RAID-PIR for both clients and PIR servers.

2.5 PIR Benchmark Results

In this section, we evaluate the performance of our modified implementation of RAID-PIR [DHS14]. All measurements include the generic optimizations from Sect. 2.4. We show the influence of the optimizations from Sects. 2.2 and 2.3.

Benchmark Setting. Three PIR servers were deployed as `r3.xlarge` instances on Amazon EC2 with 30.5 GiB RAM, an Intel Xeon E5-2670 v2 processor and a 1 Gbit/s ethernet connection. The PIR queries were performed by a `t2.micro` instance with 1 GiB RAM, one core of an Intel Xeon E5-2676 v3 processor and a 250 Mbit/s ethernet connection (0.5 ms latency) in the WAN setup and a notebook with 8 GiB RAM, an Intel Core i3-3120M processor and a consumer grade DSL Internet connection (4.5 Mbit/s downstream, 400 kbit/s upstream, 30 ms latency) in the DSL setup. The used database consists of 964 files of Ubuntu security updates adding up to a total size of 3.8 GiB. All results are average runtimes of 5 iterations. Note that the y-axes in the figures are in logarithmic scale.

PIR Server Startup Duration. The time to load the database into the PIR servers' RAM is mostly independent of the block size and took ≈40 s for the database of 3.8 GiB. The four Russian precomputation took between 5 s and 7 s. The startup time scales linearly with the database size.

File Query via WAN. In the remainder of this section, the number of servers is set to $k = 3$ and the redundancy parameter is $r = 2$. The block size is varied from $b = 16$ kiB to $b = 4$ MiB.

Small File Query via WAN: The results for querying 10 small files adding up to 2.9 MiB in the WAN setup are depicted in the left part of Fig. 5. Single-block queries are abbreviated as `SB`, multi-block queries as `MB`, the Method of four Russians as `4R` and the uniform distribution of the data entries as `UD`. Here, the multi-block queries are significantly faster than single-block queries even when no optimizations are applied since the requested files are already distributed among the whole database. Therefore, uniform distribution of the data entries does not have a significant impact on the overall performance. Four Russians precomputation improves the runtimes by factor 2 and does not introduce improvements any more when the cache size of the processor is not sufficient for larger block sizes. For a block size of $b = 16$ kiB, the runtime decreases from 10.1 s for the

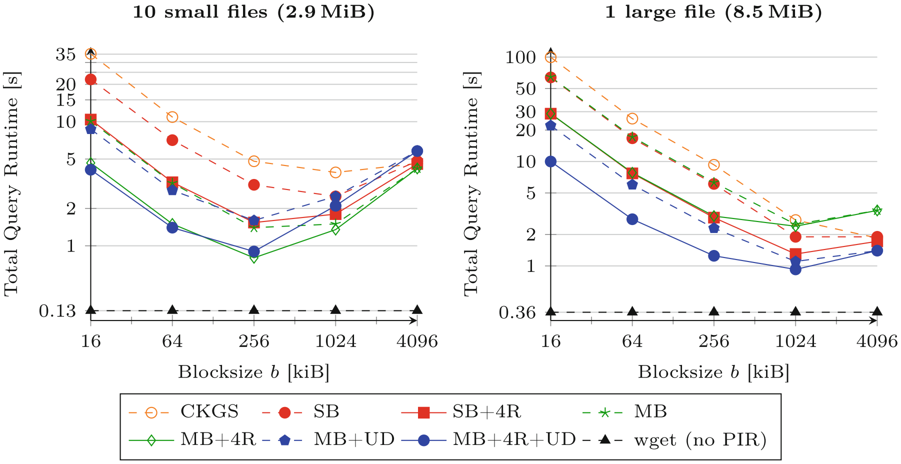


Fig. 5. WAN Benchmarks: Runtimes for varying block size b with $k = 3$ servers, redundancy parameter $r = 2$ for 10 small files (2.9 MiB, left) and a large file (8.5 MiB, right). `CKGS`: [CKGS95], `SB`: Single-Block scheme, `MB`: Multi-Block scheme, `4R`: Four Russians precomputation, `UD`: Uniform distribution of data entries. DB size 3.8 GiB.

originally best performing multi-block queries to 4.1 s when all optimizations are applied. The best performance is obtained for a block size of $b = 256$ kiB where 7 queries are performed to retrieve 18 blocks in 0.8 s.

Large File Query via WAN: In the right part of Fig. 5, one large 8.5 MiB file is requested. As already observed in [DHS14], single-block and multi-block queries take similar time when querying one large file in the WAN setup, where bandwidth and latency typically are not the bottleneck. The four Russians precomputation leads to a significant speedup and effectively improves the runtime of most test cases by factor 2. This is a good result even though the theoretical speedup, that does not cover data locality and communication overhead, is 4. When the CPU cache size is too small to store the precomputed LUTs and all interim results for large block sizes, the Method of four Russians' speedup decreases and is not measurable any more for $b = 4$ MiB.

The speedup gained by uniformly distributing the data entries in the database is even greater and improves the overall runtime by approximately a factor of 3 compared to the original multi-block queries for small block sizes. While all blocks have to be queried from the first chunk for a non-uniformly distributed database, it is now possible to retrieve 3 blocks from 3 different chunks in one multi-block query in parallel. When both optimizations are combined, the runtime decreases from 64 s (or 100 s for the original CKGS scheme) to 10 s for block size $b = 16$ kiB and reaches its minimum for $b = 1$ MiB where the runtime decreases from 1.9 s for the originally best performing single-block queries to 0.9 s for multi-block queries using both optimizations.

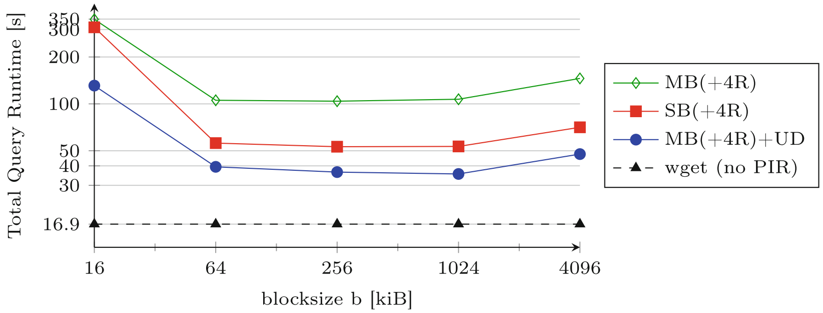


Fig. 6. Large File, DSL: Runtimes for varying block sizes b with $k = 3$ servers, redundancy parameter $r = 2$ and one large file (8.5 MiB). `CKGS`: original PIR scheme [CKGS95], `SB`: Single-Block scheme, `MB`: Multi-Block scheme, `4R`: Four Russians precomputation, `UD`: Uniform distribution of data entries. DB size 3.8 GiB.

Large File Query via DSL. In Fig. 6, the results for querying one large file (8.5 MiB, as in the first test case) over the consumer-grade DSL connection are depicted. The first interesting observation is that the results for single- and multi-block queries with a block size of $b = 16$ kiB do not differ significantly. The reason for this is the low upstream bandwidth of the DSL connection. For small block sizes, the required upstream bandwidth, which depends on the number of blocks, has a larger impact than the downstream bandwidth, which depends on the block size. To query a database containing $B = 246360$ blocks of 16 kiB each, $B/3$ bits have to be transferred to each server (plus additional 16 Bytes for the PRG seeds) resulting in a total transmission time of $3 \cdot (B/3\,\text{bit} + 16\,\text{Byte})/(400\,\text{kbit/s}) \approx 602\,\text{ms}$. This phenomenon can also be observed in the benchmarks run in [DHS14].

Due to this high latency and communication overhead of the PIR queries, the server-side four Russians precomputation does not significantly affect the overall runtime of a client's request. As in the first test case, single-block queries provide better performance than multi-block queries without the uniform distribution. However, when the data entries are distributed uniformly, a significant speedup to multi-block queries can be observed and – as already expected – the multi-block queries supersede the single-block approach because the number of requests reduces by about a third. It was also shown in [DHS14] that larger block sizes have the disadvantage that large parts of some requested blocks contain no relevant data and therefore lead to a slower runtime.

For a redundancy parameter of $r = 2$, the data that needs to be sent to the client is about twice the size of the raw data. The best results are achieved for $b = 1$ MiB using both new optimizations. Retrieving the file without PIR, using `wget` over an unencrypted HTTP connection takes 16.9 s and is only 2.1 times faster, indicating that the runtime of 35.5 s for the PIR approach almost reached the theoretically optimal results for the DSL setup, as we need twice the communication. When the network is the bottleneck, computation has only a minor influence on total runtime. Our results also show that the previously assumed optimal block size of $b = B = \sqrt{\text{DB size}}$ does not lead to good performance, especially in the DSL setting with asymmetric up- and downstream bandwidth. For a database size of 3.8 GiB that would be ≈22 kiB block size, which our results show to be far worse than larger block sizes.

Query Time for Varying Database Size. To demonstrate how the size of the database influences the query runtime we performed the same queries as in Sect. 2.5 using multi block queries with four Russians precomputation and uniformly distributed database blocks. We varied the size of the database and depict our results in Fig. 7. The time that the clients needs to generate the queries to the servers is almost constant and always around or below 50 ms. The communication varies due to queries being sent over a public Internet connection and also remains mostly constant between 200 ms and 300 ms. This is due to the fact, that even though the database and thus the number of blocks increases, the query size only grows from 250 Bytes to 4 kiB, which is negligible compared to the block size $b = 256$ kiB, that the client receives. Server computation time and thus total time increase linearly with the database size.

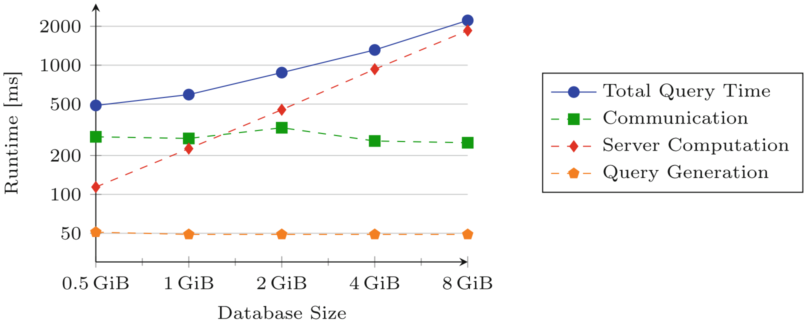


Fig. 7. Varying DB size: Runtimes for varying database size, $k = 3$ servers, redundancy parameter $r = 2$ for querying one large file (8.5 MiB) and block size $b = 256$ kiB. All queries are multi-block (MB) queries using four Russians precomputation and uniformly distributed database blocks. Note the logarithmic scale of the y-axis.

3 OnionPIR

As an application for our previously presented improvements, we introduce a private communication system called *OnionPIR* for anonymous communication. We use RAID-PIR with our optimizations as a building block for public key distribution, and onion routing to get a system efficient enough for practical use.

3.1 Motivation

When two parties register under pseudonymous identities to a classical messaging service and connect to the service by using Tor [DMS04], a malicious server can link those two pseudonyms together. Even if these pseudonyms do not reveal information about the users behind them, it is possible to build a social graph isomorphic to the one built with information from other sources. These two graphs can then be mapped together to reveal information about users. Therefore, Tor alone is not enough to provide protection against metadata leakage. Additionally, it has turned out that the assumption of users willing to establish a shared secret is problematic. In order to provide protection against mass surveillance, a system to establish private communication channels based on already existing contact information such as phone numbers or email addresses is needed.

3.2 System Model and Goals

Existing privacy-preserving communication systems do often not scale for large numbers of users or require the exchange of a shared secret out-of-band, which is error-prone and leads to usability issues most users are not willing to accept. The success of the popular messaging app Signal[2] and the adaption of its protocol in WhatsApp and Facebook Messenger are significantly founded on the combination of strong security and great usability. Since most users do not have an in-depth knowledge of cryptography, it is necessary to build technologies that provide privacy and usability by design and give reasonable defaults for its users.

To combine the strong privacy guarantees of PIR protocols with the efficiency of onion routing protocols, such as Tor [DMS04], the interaction with the server is divided into two phases: an *initialization phase* where the communication channels are established and keys are exchanged via PIR and a *communication phase* where the actual message exchange takes place through Tor.

In the initialization phase, PIR is used to *privately* exchange information between two parties which want to communicate securely. The exchange happens in a way that no one except these two parties will find out that the communication between them happened. This procedure could theoretically be used to exchange all kinds of data. However, when it comes to practical applicability, querying large amounts of information via PIR does not scale well for a larger number of users. Hence, in the communication phase no PIR techniques will be used.

Instead, the information retrieved from PIR is then used to place messages, encrypted using Authenticated Encryption with Associated Data (AEAD), in an anonymous inbox, called “dead drop”. This dead drop can only be read and written at the OnionPIR control server in the communication phase by clients that know its identifier. By using onion routing to hide the identity of the communication partners, the server is not able to determine who sent and received messages. For real-time communication, the users could also establish direct connections using TCP streams or web sockets [FM11] through the server.

The OnionPIR system, depicted in Fig. 8, serves clients who want to communicate with each other and two types of servers. All honest clients form the anonymity set among which a user is anonymous. That means that a potential

[2] downloaded by 1–5 Mio. users according to the Google Play Store, https://play.google.com/store/apps/details?id=org.thoughtcrime.securesms.

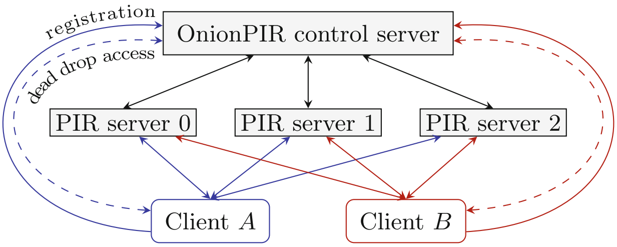


Fig. 8. OnionPIR system. The OnionPIR control server handles user registrations, distributes users' public keys to the PIR servers and holds the dead drops. Clients perform PIR queries to privately receive other users' public keys from the PIR servers. Clients connect to the OnionPIR control server directly or via Tor (dashed lines).

adversary cannot determine which users within this set communicate with each other. The central OnionPIR control server handles user registration and serves as a content provider for the PIR servers. It also acts as a database for the "dead drops" in the communication phase. The PIR servers are used only in the initialization phase to privately distribute the public keys of clients.

3.3 Protocol Description

In this section we describe the OnionPIR protocol. A (simplified) version is depicted in Fig. 9. When a client A registers for the service, it first runs through an account verification process and sends its public key to the OnionPIR control server which will later distribute it to the PIR servers. Another client B, that has an address book entry for A (e.g., a mobile number or e-mail address), will later *privately* query A's public key via PIR. In addition, each user periodically queries for his or her own public key to make sure the OnionPIR control server does not distribute bad keys to the PIR servers (see Sect. 3.4 for details). Note, that a single PIR server is never able to replace keys unless it colludes with other PIR servers because the responses of multiple servers will be combined to

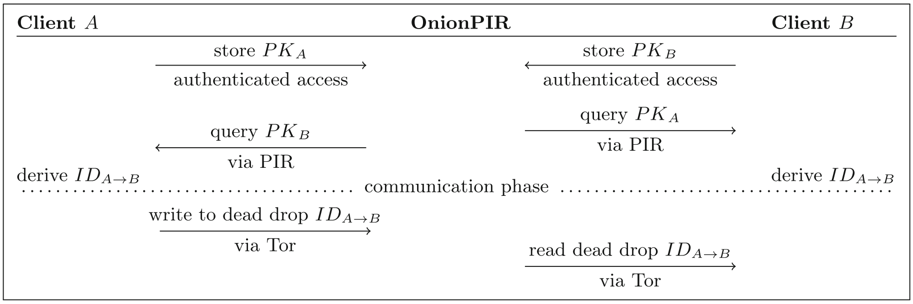


Fig. 9. Simplified OnionPIR protocol. The registration, dead drop and PIR servers were abstracted into one server. Key renewal and the answer of client B are not shown.

retrieve the public key. B will then use his own private key and the received public key to generate a shared secret K_{AB} between A and B by performing an Elliptic Curve Diffie-Hellman (ECDH) key agreement. Since no communication with the server is required to derive the shared secret, this type of key agreement protocols is often also called *private key agreement*. In the same way, A is also able to derive the shared secret K_{AB} using her own private key and the public key of B.

Symmetric Keys. The shared secret K_{AB} and both parties' public keys are used to derive identifiers of the dead drops and keys used to exchange messages. Client A generates two different symmetric keys derived from the initial shared secret K_{AB} and the respective public keys by a keyed-hash message authentication code (HMAC): $K_{A \rightarrow B} = \text{HMAC}_{K_{AB}}(pk_B)$ for sending messages to B and $K_{B \rightarrow A} = \text{HMAC}_{K_{AB}}(pk_A)$ for receiving messages from B. These secrets constantly get replaced by new ones, transmitted alongside every message to provide forward secrecy.

Dead Drop IDs. The identifiers of the dead drop for sending from A to B is built by computing an HMAC with the key $K_{A \rightarrow B}$ of a nonce $N_{A \rightarrow B}$ that increases after a given time period. Hence, the identifier $ID_{A \rightarrow B} = \text{HMAC}_{K_{A \rightarrow B}}(N_{A \rightarrow B})$ changes in a fixed interval, even if no messages were exchanged at all. This prevents the server from identifying clients that disconnect for several days and would otherwise reconnect using the same identifiers. However, a fixed point in time at which the nonce changes (i.e. a unix timestamp rounded to the current day) is not a good choice. Assuming synchronized clocks often is error-prone[3]. If all clients would update their identifiers at a given point in time, e.g., at midnight, the server could detect a correlation between all identifiers of a user whose clock is out of sync. Thus, the nonces will be handled per contact and the secret key $K_{A \rightarrow B}$ is used to generate a point in time at which the nonce $N_{A \rightarrow B}$ will be increased. This leads to a different update time for all identifiers of dead drops.

Sending Messages. When a client A wants to send a message m to B, it encrypts the message using Authenticated Encryption with Associated Data (AEAD) using $K_{A \rightarrow B}$ so that only B can read it. Note, that the ciphertext of the message m does not reveal any information about the sender or the receiver as explained in [Ber09, Sect. 9]. A then stores the encrypted message in the dead drop $ID_{A \rightarrow B}$, which A accesses via Tor. B is now able to fetch and decrypt A's message from this dead drop. The shared secret will only be replaced by a new one transmitted alongside a message when B acknowledges the retrieval of the

[3] The need for secure time synchronization protocols lead to a number of secure time synchronization concepts such as ANTP [DSZ16], NTS (https://tools.ietf.org/html/draft-ietf-ntp-network-time-security-14) or Roughtime (https://roughtime.googlesource.com/roughtime).

new secret in a dead drop derived from the *existing* shared secret. Until then, A will not store any messages in a dead drop derived from the new secret and will retransmit messages until B sends the acknowledgment. This procedure will guarantee that no messages will be lost. Note, that $ID_{A \to B}$ may still change over time because of the used nonce $N_{A \to B}$ which is known by both parties.

Querying for new public keys using PIR is done in a fixed interval, e.g. once a day, to discover new users of the system. It is mandatory not to write to the dead drop specified by the shared secret immediately after receiving a new public key. This would allow an adversary to correlate a PIR request of a given user with (multiple) new requests to the dead drop database, even if the server does not know which public keys were queried. Instead, the first access to the dead drop database for new identifiers is delayed until a random point in time between the current query and the next one derived from the shared key $K_{A \to B}$. This ensures that the server cannot correlate the dead drop access to the PIR request because it could also have been initiated by any other clients that did a PIR request in the fixed interval. Note, that this delay is only necessary when a new contact is discovered. New users joining the service will therefore also have to delay their first interaction with other users.

Initial Contact. If B wants to contact A without A knowing about that (and thus not checking the associated dead drop at $ID_{B \to A}$), an anonymous signaling mechanism is required. We propose a per-user fixed and public dead drop that *all* other users write to, to establish contact. The fixed dead drop ID is $ID_A = \text{HMAC}_0(pk_A)$. Messages into this dead drop are encrypted using hybrid encryption, similar to PGP, where A's public key pk_A is used to encrypt an ephemeral symmetric key, which encrypts the request message. This reveals how many contact requests A receives, which we consider as non-critical. However, this number could be obscured by sending dummy requests.

3.4 Analysis

Correctness. Correctness of RAID-PIR is shown in [DHS14], correctness of Tor is explained in [DMS04] and has already been well-proven in practice. Messages are acknowledged and retransmitted if identifiers change and the message has not been read yet. Therefore it is guaranteed that messages will reach their desired destination. Correctness of the ECDH key exchange is shown in [Ber09]. All operations involved in the private establishment of identifiers for the dead drops are deterministic and therefore result in the same identifiers for both parties.

Security. OnionPIR's security is based on the security guarantees of the underlying protocols and their assumptions. First, we assume that RAID-PIR is secure and does not leak any metadata about the queried information. A security argumentation for this is given in [DHS14]. In particular, it is important that the PIR servers are run by different non-colluding operators. Note, that the security guarantees are still fulfilled if less than r servers collude, where r is the redundancy parameter. A good choice for the operators would be a number of NGOs located in different legal territories. It is also mandatory that the different PIR servers are not in (physical) control of the same data center operator.

Next, we assume that Tor provides anonymity for the users that tunnel their connections through this anonymity network. This assumption implies that there is no global passive adversary which is able to monitor and analyze user traffic and colludes with the operator of the PIR servers or gains unauthorized access to them. Note, that it is necessary to at least de-anonymize two specific Tor connections in order to learn if two users are communicating with each other. Therefore, an attacker would have to be able to de-anonymize all users of the service to gain the full social graph of a given user. OnionPIR does not put any effort in hiding the fact that a user is using the service at all.

Anonymity is guaranteed among all honest users. A malicious user that announces its list of requests to the dead drops, would effectively remove himself from the anonymity set. Note, that no user is able to gain information about communication channels it is not participating in. In addition, no user is able to prove that a communication took place because the used AEAD guarantees repudiability.

OnionPIR relies on a Trust On First Use (TOFU) strategy to lessen the burden of manually exchanging keys. For the purpose of detecting the distribution of faulty keys, a client queries not only for the public keys of its contacts, but also for its own public key. This query can be performed at nearly no cost when querying together with other contacts using RAID-PIR's multi-block query. Since the PIR servers are not able to determine which PIR blocks are being requested, they are not able to manipulate the resulting response in a meaningful way, unless they collude. Of course, additional security can be achieved by adding out-of-band key verification, e.g., by announcing public keys on personal websites. Another interesting option are plausibility checks for the updates of the PIR database in the PIR servers and thereby extending the existing anytrust model.

Access to the dead drops is not protected against any type of manipulation by third parties since identifiers are only known to the involved parties. An adversary that is interested in deleting the messages for a specific client would have to brute-force the identifier of the dead drop which is impossible in practice. Protection against a server deleting messages or blocking access to the system is out of scope of this work and would require a federated or decentralized system.

Protection against malicious clients trying to flood the dead drops with large amounts of data could be achieved by making use of blind signatures [Cha83]. A client could encrypt a number of random tokens and authenticate against the server who will then blindly sign them. The tokens can then be decrypted by the client and sent to the server with each write access to the dead drops. While the server is able to determine that these tokens have a valid signature, it cannot identify the client who generated them. Since a server only signs a fixed number of tokens in a given time interval per client, this approach rate-limits write requests to the database.

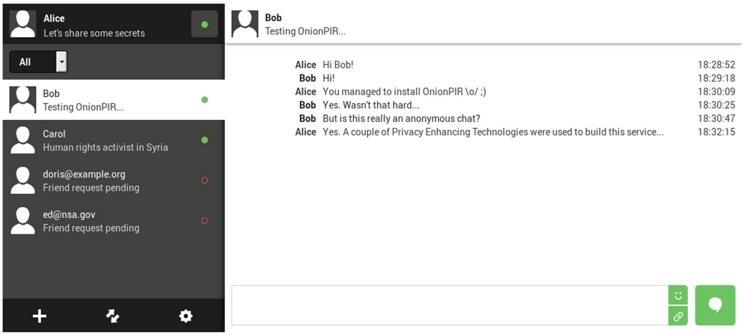
Complexity and Efficiency. OnionPIR aims at providing efficient anonymous communication. Many existing systems (see Sect. 4) require a high communication overhead or high computational costs. The dead drop database is therefore combined with scalable onion routing and can be implemented as simple key-value storage. The servers needed in the communication phase can be deployed with low operating costs, comparable to traditional communication services.

The initialization phase in which the PIR requests are performed is crucial for the scalability of the system. As shown in Sect. 2.5, PIR is a valid choice that offers reasonable performance and allows users to detect malicious actions of the servers. The 3.8 GiB PIR database used for the benchmarks in Sect. 2.5 is sufficient to store public keys of 127 Mio. users, using 256 bit elliptic curve public keys.

The database size and the server's computation will grow linearly in the number of users. The PIR servers' ingress traffic for PIR queries depends on the number of blocks B in the database and therefore also scales linearly. Thanks to the PRG used in [DHS14], the size of a PIR query is $\lceil B/8 \rceil$ Bytes for all servers combined (excluding PRG seeds and overhead for lower level transport protocols). The egress bandwidth is constant since the size of the response only depends on the block size b (and the number of chunks for multi-block queries).

3.5 Implementation

We implemented a desktop application for secure messaging with metadata protection based on OnionPIR. Our implementation is written in Python and C, using our version of RAID-PIR including the optimizations from Sect. 2 for PIR, the Networking and Cryptography library (NaCl) [BLS12] for cryptographic operations, and Stem[4] as controller library for Tor. The system is divided into a client, the OnionPIR control server and PIR servers. The client offers a GUI as depicted in Fig. 10. Our open-source implementation is publicly available.[5]



Alice	Hi Bob!	18:28:52
Bob	Hi!	18:29:18
Alice	You managed to install OnionPIR \o/ ;)	18:30:09
Bob	Yes. Wasn't that hard...	18:30:25
Bob	But is this really an anonymous chat?	18:30:47
Alice	Yes. A couple of Privacy Enhancing Technologies were used to build this service...	18:32:15

Fig. 10. Screenshot of the OnionPIR client GUI.

[4] https://stem.torproject.org/.

[5] https://github.com/encryptogroup/onionPIR.

4 Related Work

Nowadays, end-to-end encryption is available and deployed at large scale. In the past, these technologies were not accessible to a large user base because they required expert knowledge or were just not convenient, and thus only used by enthusiasts. For example, users of OpenPGP [CDF+07] have to manually build a web of trust and should be familiar with public key cryptography, while S/MIME [Ram99] requires certificate management. When the Signal Protocol was integrated into popular messaging services like WhatsApp[6] or Facebook Messenger[7] in 2016, private messaging became available to an extremely large user base. However, protecting not only communication content but also its metadata are still under active research. OnionPIR currently relies on RAID-PIR [DHS14] for public key distribution, but could also employ different PIR schemes. PIR is an active research area and there are other viable candidates [DG14, Hen16, AMBFK16]. Similarly, we rely on Tor to provide anonymity, which can also be achieved by using alternative techniques such as mixing networks [Cha81].

Next, we present system proposals that are related to OnionPIR. *Redphone* was one of the first applications that tackled metadata leakage using Bloom filters [Blo70] for private contact discovery. Redphone's encrypted call features were integrated into Textsecure/Signal, however, without anonymity due to scalability [Ope14]. With *DP5* [BDG15] users can anonymously exchange online presence information based on PIR. DP5 divides time in long-term and short-term epochs which are in the order of days and minutes respectively, which makes it impractical for real-time communication. Users must share symmetric keys before the protocol run, which can be hard to achieve in practice. Recently, *Alpenhorn* [LZ16] was proposed, which is based on identity-based encryption (IBE) for key distribution, a mix network [CJK+16, Cha81] for privacy and a so called keywheel construct for forward secrecy. Alpenhorn was integrated into Vuvuzela [vdHLZZ15], which supports 10 Mio. users using three Alpenhorn servers with an average dial latency of 150 s and a client bandwidth overhead of 3.7 KiB/s. *Riposte* [CGBM15] enables a large user base to privately post public messages to a message board. Its security is based on distributed point functions that are evaluated by a set of non-colluding servers. Time is divided into epochs, in which all users who post messages form an anonymity set. *Ricochet* (https://ricochet.im) is a decentralized private messaging service based on Tor hidden services, whose addresses must be exchanged out-of-band to establish connections. Recent research showed that HSDirs are used to track users [SN16], which might be problematic for Ricochet's privacy. *Riffle* [KLDF16] provides scalable low-latency and low-bandwidth communication through mix networks [Cha81] using verifiable shuffles [BG12] for sender anonymity and PIR for receiver anonymity. In Riffle, time is divided into epochs and each client sends and receives messages

[6] https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf.

[7] https://fbnewsroomus.files.wordpress.com/2016/07/secret_conversations_whitepaper-1.pdf.

even if they do not communicate. *The Pynchon Gate* [SC05] is an anonymous mail system that guarantees only receiver anonymity by using PIR.

Acknowledgments. We thank the anonymous reviewers for their valuable feedback on our paper. This work has been co-funded by the German Federal Ministry of Education and Research (BMBF) and by the Hessen State Ministry for Higher Education, Research and the Arts (HMWK) within CRISP, and by the DFG as part of project S5 within the CRC 1119 CROSSING.

References

[ABH10] Albrecht, M., Bard, G., Hart, W.: Efficient multiplication of dense matrices over GF(2). ACM Trans. Math. Softw. **37**, 9:1–9:14 (2010)

[ADKF70] Arlazarov, V., Dinic, E., Kronrod, M., Faradzev, I.: On economical construction of the transitive closure of a directed graph. USSR Acad. Sci. **134**, 1209–1210 (1970)

[AMBFK16] Aguilar-Melchor, C., Barrier, J., Fousse, L., Killijian, M.-O.: XPIR: private information retrieval for everyone. In: Privacy Enhancing Technologies Symposium (PETS 2016), no. 2, pp. 155–174 (2016)

[BDG15] Borisov, N., Danezis, G., Goldberg, I.: DP5: a private presence service. In: Privacy Enhancing Technologies Symposium (PETS 2015), no. 2, pp. 4–24 (2015)

[Ber09] Bernstein, D.J.: Cryptography in NaCl (2009). https://cr.yp.to/highspeed/naclcrypto-20090310.pdf

[BG12] Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29011-4_17

[Blo70] Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970)

[BLS12] Bernstein, D.J., Lange, T., Schwabe, P.: The security impact of a new cryptographic library. In: Hevia, A., Neven, G. (eds.) LATINCRYPT 2012. LNCS, vol. 7533, pp. 159–176. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33481-8_9

[BV14] Budurushi, J., Volkamer, M.: Feasibility analysis of various electronic voting systems for complex elections. In: International Conference for E-Democracy and Open Government 2014 (2014)

[CDF+07] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., Thayer, R.: OpenPGP message format. RFC 4880, RFC Editor, November 2007. http://www.rfc-editor.org/rfc/rfc4880.txt

[CGBM15] Corrigan-Gibbs, H., Boneh, D., Maziàres, D.: Riposte: an anonymous messaging system handling millions of users. In: IEEE Symposium on Security and Privacy (S&P 2015), pp. 321–338 (2015)

[CGN98] Chor, B., Gilboa, N., Naor, M.: Private information retrieval by keywords. IACR Cryptology ePrint Archive, Report 1998/003 (1998). http://eprint.iacr.org/1998/003

[Cha81] Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**, 84–90 (1981)

[Cha83] Chaum, D.: Blind signature systems. In: Advances in Cryptology - CRYPTO 1983, p. 153 (1983)

[CJK+16] Chaum, D., Javani, F., Kate, A., Krasnova, A., de Ruiter, J., Sherman, A.T., Das, D.: cMix: anonymization by high-performance scalable mixing. IACR Cryptology ePrint Archive, Report 2016/008 (2016). http://eprint.iacr.org/2016/008

[CKGS95] Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. In: Foundations of Computer Science (FOCS 1995), pp. 41–50 (1995)

[DG14] Devet, C., Goldberg, I.: The best of both worlds: combining information-theoretic and computational PIR for communication efficiency. In: Cristofaro, E., Murdoch, S.J. (eds.) PETS 2014. LNCS, vol. 8555, pp. 63–82. Springer, Cham (2014). doi:10.1007/978-3-319-08506-7_4

[DHS14] Demmler, D., Herzberg, A., Schneider, T.: RAID-PIR: practical multi-server PIR. In: ACM Cloud Computing Security Workshop (CCSW 2014), pp. 45–56 (2014)

[DMS04] Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: USENIX Security Symposium 2004, p. 21 (2004)

[DSZ14] Demmler, D., Schneider, T., Zohner, M.: Ad-hoc secure two-party computation on mobile devices using hardware tokens. In: USENIX Security Symposium 2014, pp. 893–908 (2014)

[DSZ16] Dowling, B., Stebila, D., Zaverucha, G.: Authenticated network time synchronization. In: USENIX Security Symposium 2016, pp. 823–840 (2016)

[FM11] Fette, I., Melnikov, A.: The websocket protocol. RFC 6455, RFC Editor, December 2011. http://www.rfc-editor.org/rfc/rfc6455.txt

[HCE11] Huang, Y., Chapman, P., Evans, D.: Privacy-preserving applications on smartphones. In: USENIX Workshop on Hot Topics in Security (HotSec 2011), p. 4 (2011)

[Hen16] Henry, R.: Polynomial batch codes for efficient IT-PIR. In: Privacy Enhancing Technologies Symposium (PETS 2016), pp. 202–218 (2016)

[KLDF16] Kwon, A., Lazar, D., Devadas, S., Ford, B.: Riffle: an efficient communication system with strong anonymity. In: Privacy Enhancing Technologies Symposium (PETS 2016), pp. 115–134 (2016)

[KO97] Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: Foundations of Computer Science (FOCS 1997), pp. 364–373 (1997)

[Lan15] Landau, S.: Mining the metadata: and its consequences. In: International Conference on Software Engineering (ICSE 2015), pp. 4–5 (2015)

[LG15] Lueks, W., Goldberg, I.: Sublinear scaling for multi-client private information retrieval. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 168–186. Springer, Heidelberg (2015). doi:10.1007/978-3-662-47854-7_10

[LZ16] Lazar, D., Zeldovich, N.: Alpenhorn: bootstrapping secure communication without leaking metadata. In: USENIX Symposium on Operating Systems Design and Implementation (OSDI 2016), pp. 571–586 (2016)

[MCA06] Mokbel, M.F., Chow, C.-Y., Aref, W.G.: The new Casper: query processing for location services without compromising privacy. In: International Conference on Very Large Data Bases (VLDB 2006), pp. 763–774 (2006)

[MMM16] Mayer, J., Mutchler, P., Mitchell, J.C.: Evaluating the privacy properties of telephone metadata. Natl. Acad. Sci. **113**(20), 5536–5541 (2016)

[MOT+11] Mittal, P., Olumofin, F., Troncoso, C., Borisov, N., Goldberg, I.: PIR-tor: scalable anonymous communication using private information retrieval. In: USENIX Security Symposium 2011, p. 31 (2011)

[Ope14] Open Whisper Systems. The difficulty of private contact discovery (2014). https://whispersystems.org/blog/contact-discovery/

[Ram99] Ramsdell, B.: S/MIME version 3 message specification. RFC 2633, RFC Editor, June 1999. http://www.rfc-editor.org/rfc/rfc2633.txt

[SC05] Sassaman, L., Cohen, B., Gate, T.P.: A secure method of pseudonymous mail retrieval. In: Workshop on Privacy in the Electronic Society (WPES 2005), pp. 1–9 (2005)

[SN16] Sanatinia, A., Noubir, G.: HOnions: towards detection and identification of misbehaving tor HSDirs. In: Hot Topics in Privacy Enhancing Technologies Symposium (HotPETS 2016) (2016)

[vdHLZZ15] van den Hooff, J., Lazar, D., Zaharia, M., Zeldovich, N.: Vuvuzela: scalable private messaging resistant to traffic analysis. In: Symposium on Operating Systems Principles (SOSP 2015), pp. 137–152 (2015)

Data and Server Security

Accountable Storage

Giuseppe Ateniese[1], Michael T. Goodrich[2], Vassilios Lekakis[3],
Charalampos Papamanthou[5], Evripidis Paraskevas[5(✉)],
and Roberto Tamassia[4]

[1] Department of Computer Science, Stevens Institute of Technology, Hoboken, USA
gatenies@stevens.edu
[2] Department of Computer Science, University of California, Irvine, USA
goodrich@uci.edu
[3] Department of Computer Science, University of Maryland, College Park, USA
lex@cs.umd.edu
[4] Department of Computer Science, Brown University, Providence, USA
rt@cs.brown.edu
[5] Department of Electrical and Computer Engineering, University of Maryland,
College Park, USA
cpap@umd.edu, evripar@terpmail.umd.edu

Abstract. We introduce *Accountable Storage* (AS), a framework enabling a client to outsource n file blocks to a server while being able (any time after outsourcing) to provably compute how many bits were discarded or corrupted by the server. Existing techniques (e.g., proofs of data possession or storage) can address the accountable storage problem, with linear server computation and bandwidth. Instead, our optimized protocols achieve $O(\delta \log n)$ complexity (where δ is the maximum number of corrupted blocks that can be tolerated) through the novel use of invertible Bloom filters and a new primitive called *proofs of partial storage*. With accountable storage, a client can be compensated with a dollar amount proportional to the number d of corrupted bits (that he can now provably compute). We integrate our protocol with Bitcoin, supporting automatic such compensations. Our implementation is open-source and shows our protocols perform well in practice.

1 Introduction

Cloud computing is revolutionizing our digital world, posing new security and privacy challenges. E.g., businesses and individuals are reluctant to outsource their databases for fear of having their data lost or damaged. Thus, they would benefit from technologies that would allow them to manage their risk of data loss, just like insurance allows them to manage their risk of physical or financial losses, e.g., from fire or liability.

As a first step, a client needs a mechanism for verifying that a cloud provider is storing her entire database intact, and fortunately, *Provable Data Possession* (PDP) [3,11,13] and *Proofs of Retrievability* (POR) [10,19,26–28], have been conceived as a solution to the integrity problem of remote databases. PDP and

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 623–644, 2017.
DOI: 10.1007/978-3-319-61204-1_31

POR scheme can verify whether the server possesses the database originally uploaded by the client by having the server generate a proof in response to a challenge.

However, they leave unsettled several risk management issues. Arguably, an important question is

> *What happens if a PDP or POR scheme shows that a client's outsourced database has been damaged?*

The objective of this work is to design new efficient protocols for *Accountable Storage* (AS) that enable the client to reliably and quickly assess the damage and at the same time automatically get compensated using the Bitcoin protocol.

To be precise, suppose Alice outsources her file blocks $b_1, b_2, \ldots, b_n$ to a potentially malicious cloud storage provider, Bob. Since Alice does not trust Bob, she wishes, at any point in time, to be able to compute the amount of damage, if any, that her file blocks have undergone, by engaging in a simple challenge-response protocol with Bob. For instance, she wishes to *provably* compute the value of a *damage metric*, such as

$$d = \sum_{i=1}^{n} w_i \cdot ||b_i \oplus b'_i||, \tag{1}$$

where b'_i is the file currently stored by Bob at the time of the challenge, $||.||$ denotes Hamming distance and w_i is a weight corresponding to file b_i. If $d = 0$, Alice is entitled to no dollar credit. Bob can easily prove to Alice that this is the case through existing protocols, as noted above. If $d > 0$, however, then Alice should receive a compensation proportional to the damage d, which should be provided automatically.[1]

Naive Approaches for AS. A PDP protocol [3,4,11,13,29] enables a server to prove to a client that all of the client's data is stored intact. One could design an AS protocol by using a PDP protocol only for the portion of storage that the server possesses. This could determine the damage, d (e.g., when all weights w_i are equal to 1). However this approach requires using of PDP at the bit level, and in particular computing one 2048-bit tag for each bit of our file collection which is very storage-inefficient.

To overcome the above problem, one could use PDP at the block level, but at the same time keep some redundancy locally. Specifically, before outsourcing the n blocks at the server, the client could store δ extra check blocks locally (e.g., computed with an error-correcting code). The client could then verify through PDP that a set of at most δ blocks have gone missing and retrieve the lost blocks by executing the decoding algorithm on the remote intact $n - \delta$ data blocks and

[1] We highlight that such fine-grained compensation models, which work at the bit level as opposed to at the file block level, allow Alice to better manage her risk for damage even within the same file. For example, compensation for an unusable movie stored by Bob could be larger than that for a usable movie whose resolution has deteriorated by just 5%.

the δ local check blocks (then the recovered blocks can be used to compute d). This procedure has $O(n)$ communication, since the $n - \delta$ blocks at the server must be sent to the client. IRIS [28] is a system along these lines, requiring the whole file system be streamed over to the client for recovery.

Finally we note here that while PDP techniques combined with redundant blocks stored at the client can be used to solve the accountable storage problem (even inefficiently, as shown above), POR techniques cannot. This is because POR techniques (e.g., [26]) cannot provide proofs of retrievability for a certain portion of the file (as is the case with PDP), but only for the whole file—this is partly due to the fact that error-correcting codes are used on top of all the file blocks.

Our AS Protocol. Our protocol for assessing damage d from Relation 1 is based on *recovering the actual blocks* $b_1, b_2, \ldots, b_\delta$ *and XORing them with the corrupted blocks* $b'_1, b'_2, \ldots, b'_\delta$ *returned by the server.* For recovery, we use the invertible Bloom filter (IBF) data structure [12,15]. An IBF is an array of t cells and can store $O(t)$ elements. Unlike a Bloom filter [7], its elements can be enumerated with high probability.

Let $B = \{b_1, b_2, \ldots, b_n\}$ be the set of outsourced blocks and let δ be the maximum number of corrupted blocks that can be tolerated. In preprocessing, the client computes an IBF $\mathbf{T}_B$ with $O(\delta)$ cells, on the blocks $b_1, \ldots, b_n$. $\mathbf{T}_B$ is stored locally. Computing $\mathbf{T}_B$ is similar to computing a Bloom filter: every cell of $\mathbf{T}_B$ is mapped to a XOR over a set of at most n blocks, thus the local storage is $O(\delta)$. To outsource the blocks, the client computes homomorphic tags, $\mathtt{T}_i$ (as in [3]), for each block b_i. The client then stores $(b_i, \mathtt{T}_i)$ with the cloud and deletes $b_1, b_2, \ldots, b_n$ from local storage. In the challenge phase, the client asks the server to construct an IBF $\mathbf{T}_K$ of $O(\delta)$ cells on the set of blocks K the server currently has—this is the "proof" the server sends to the client. Then the client takes the "difference" $\mathbf{T}_L = \mathsf{subtract}(\mathbf{T}_B, \mathbf{T}_K)$ and recovers the elements of the difference $B - K$ (since $|B - K| \leq \delta$ and $\mathbf{T}_L$ has $O(\delta)$ cells). Recovering blocks in $B - K$ enables the client to compute d using Relation 1. Clearly, the bandwidth of this protocol is proportional to δ (due to the size of the IBFs), and not to the total number of outsourced blocks n. Our optimized construction in Sect. 5 achieves sublinear server and client complexities as well.

Fairness Through Integration with Bitcoin. The above protocol assures that Bob (the server) cannot succeed in persuading Alice that the damage of her file blocks is $d' < d$. After Alice is persuaded, compensation proportional to d must be sent to her. But Bob could try to cheat again. Specifically, Bob could try to give Alice a smaller compensation or even worse, disappear. To deal with this problem, we develop a modified version of the recently-introduced timed commitment in Bitcoin [2]. At the beginning of the AS protocol, Bob deposits a large amount, A, of bitcoins, where A is contractually agreed on and is typically higher than the maximum possible damage to Alice's file blocks. The Bitcoin-integrated AS protocol of Sect. 6 ensures that unless Bob fully and timely compensates Alice for damage d, then A bitcoins are automatically and irrevocably transferred to Alice. At the same time, if Alice tries to cheat (e.g., by asking for compensation higher than the contracted amount), our protocol ensures that she gets no compensation at all while Bob gets back all A of his bitcoins.

Structure of the Paper. Section 2 presents background on IBFs and Bitcoin, Sect. 3 gives definitions, and Sects. 4 and 5 present our constructions. We present our Bitcoin protocol in Sect. 6, our evaluation in Sect. 7 and conclude in Sect. 8.

2 Preliminaries

Let τ denote the security parameter, δ denote an upper bound on the number of corrupted blocks that can be tolerated, n denote the number of file blocks, and $b_1, b_2, \ldots, b_n$ denote the file blocks. Each block b_i has λ bits. The first $\log n$ bits of each block b_i are used for storing the index i of the block, which can be retrieved through function $\mathsf{index}()$. Namely $i = \mathsf{index}(b_i)$. Let also $h_1, h_2, \ldots, h_k$ be k hash functions chosen at random from a universal family of functions $\mathcal{H}$ [9] such that $h_i : \{0,1\}^\lambda \rightarrow \{1, 2, \ldots, t\}$ for some parameter t.

Invertible Bloom Filters. An *Invertible Bloom Filter* (*IBF*) [12,15] can be used to compactly store a set of blocks $\{b_1, b_2, \ldots, b_n\}$: It uses a table (array) $\mathbf{T}$ of $t = (k+1)\delta$ cells. Each cell of the IBF's table $\mathbf{T}$ contains the following two fields[2]: (1) $\mathsf{dataSum}$: XOR of blocks b_i mapped to this cell; (2) $\mathsf{hashSum}$: XOR of cryptographic tags (to be defined later) $\mathtt{T}_i$ for all blocks b_i mapped to this cell. As in Bloom filters, we use functions $h_1, \ldots, h_k$ to decide which blocks map to which cells.

An IBF supports simple algorithms for insertion and deletion via algorithm update in Fig. 1. For $B \subseteq A$, one can also take the *difference* of IBFs $\mathbf{T}_A$ and $\mathbf{T}_B$, to produce an IBF $\mathbf{T}_D \leftarrow \mathsf{subtract}(\mathbf{T}_A, \mathbf{T}_B)$ representing the difference set $D = A - B$. Finally, given $\mathbf{T}_D$, we can enumerate its contents by using algorithm $\mathsf{listDiff}$ from [12]:

Lemma 1 (Adjusted from Eppstein et al. [12]). *Let $B \subseteq A$ be two sets having $\leq \delta$ blocks in their difference $A - B$, let $\mathbf{T}_A$ and $\mathbf{T}_B$ be their IBFs constructed using k hash functions and let $\mathbf{T}_D \leftarrow \mathsf{subtract}(\mathbf{T}_A, \mathbf{T}_B)$. All IBFs have $t = (k+1)\delta$ cells and their $\mathsf{hashSum}$ field is computed using a function mapping blocks to at least $k \log \delta$ bits. Then there is an algorithm $\mathsf{listDiff}(\mathbf{T}_D)$ that recovers $A - B$ with probability $1 - O(\delta^{-k})$.*

Bitcoin Basics. Bitcoin [23] is a decentralized digital currency system where transactions are recorded on a public ledger (the blockchain) and are verified through the collective effort of *miners*. A bitcoin address is the hash of an ECDSA public key. Let A and B be two bitcoin addresses. A *standard* transaction contains a signature from A and mandates a certain amount of bitcoins be transferred from A to B. If A's signature is valid, the transaction is inserted into a block which is then stored in the blockchain.

[2] Note that we do not use the count field, as in [12,15].

```
**Algorithm** $\mathbf{T} \leftarrow \mathsf{update}(b_i, \mathbf{T})$
for each $j = 1, \ldots, k$ do
    Set $\mathbf{T}[h_j(b_i)].\mathsf{dataSum} \oplus= b_i$;
    Set $\mathbf{T}[h_j(b_i)].\mathsf{hashSum} \oplus= \mathtt{T}_i$;
return $\mathbf{T}$;
```

```
**Algorithm** $\mathbf{T}_D \leftarrow \mathsf{subtract}(\mathbf{T}_A, \mathbf{T}_B)$
for each $i = 1, \ldots, t$ do
    $\mathbf{T}_D[i].\mathsf{dataSum} = \mathbf{T}_A[i].\mathsf{dataSum} \oplus \mathbf{T}_B[i].\mathsf{dataSum}$;
    $\mathbf{T}_D[i].\mathsf{hashSum} = \mathbf{T}_A[i].\mathsf{hashSum} \oplus \mathbf{T}_B[i].\mathsf{hashSum}$;
return $\mathbf{T}_D$;
```

Fig. 1. Update and subtraction algorithms in IBFs.

Bitcoin allows for more complicated transactions (as we are using here), whose validation requires more than just a signature. In particular, each transaction can specify a *locktime* containing a timestamp t at which the transaction is locked (before time t, even if a valid signature is provided, the transaction is not final). Slightly changing the notation from [2], a Bitcoin transaction T_x can be represented as the table below, where Prev is the transaction (say T_y) that T_x is redeeming, InputsToPrev are inputs that T_x is sending to T_y so that T_y's redeeming can take place, Conditions is a program written in the Bitcoin scripting language (outputting a boolean) controlling whether T_x can be redeemed or not (given inputs from another transaction), Amount is the value in bitcoins, and Locktime is the locktime. For standard transactions, InputsToPrev is a signature with the sender's secret key, and Conditions implements a signature verification with the recipient's public key. Also, standard transactions have locktime set to 0, meaning they are locked and final.

Prev :
InputsToPrev :
Conditions :
Amount :
Locktime :

3 Accountable Storage Definitions

We now define an AS scheme. An AS scheme does not allow the client to compute damage d directly. Instead, it allows the client to use the server's proof to retrieve the blocks $\mathcal{L}$ that are not stored by the server any more (or are stored corrupted). By having the server send the current blocks he stores in the position of blocks in $\mathcal{L}$ (in addition to the proof), computing the damage d is straightforward.

Definition 1 (δ-AS scheme). *A δ-AS scheme $\mathcal{P}$ is the collection of four PPT algorithms:*

1. $\{\mathsf{pk}, \mathsf{sk}, \mathsf{state}, \mathtt{T}_1, \ldots, \mathtt{T}_n\} \leftarrow \mathsf{Setup}(b_1, \ldots, b_n, \delta, 1^\tau)$ *takes as inputs file blocks $b_1, \ldots, b_n$, a parameter δ and the security parameter τ and returns a public key* pk*, a secret key* sk*, tags $\mathtt{T}_1, \ldots, \mathtt{T}_n$ and a client state* state.
2. $\mathsf{chal} \leftarrow \mathsf{GenChal}(1^\tau)$ *generates a challenge for the server;*
3. $\mathcal{V} \leftarrow \mathsf{GenProof}(\mathsf{pk}, \beta_{i_1}, \ldots, \beta_{i_m}, \mathtt{T}_{i_1}, \ldots, \mathtt{T}_{i_m}, \mathsf{chal})$ *takes as inputs a public key* pk*, a collection of $m \leq n$ blocks and their corresponding tags. It returns a proof of accountability $\mathcal{V}$;*
4. $\{\mathsf{reject}, \mathcal{L}\} \leftarrow \mathsf{CheckProof}(\mathsf{pk}, \mathsf{state}, \mathcal{V}, \mathsf{chal})$ *takes as inputs a public key* pk *and a proof of accountability $\mathcal{V}$. It returns a list of blocks $\mathcal{L}$ or* reject.

Relation to Proofs of Storage. A δ-AS scheme is a generalization of proof-of-storage (PoS) schemes, such as [3,19]. In particular, a 0-AS scheme (i.e., where we set $\delta = 0$) is equivalent to PoS protocols, where there is no tolerance for corrupted/lost blocks.

Definition 2 (δ-AS scheme correctness). *Let $\mathcal{P}$ be a δ-AS scheme. Let* $\{\mathsf{pk}, \mathsf{sk}, \mathsf{state}, \mathtt{T}_1\ , \ldots, \mathtt{T}_n\} \leftarrow \mathsf{Setup}(b_1, \ldots, b_n, \delta, 1^\tau)$ *for some set of blocks* $B = \{b_1, \ldots, b_n\}$. *Let now* $\mathcal{L} \subseteq B$ *such that* $|\mathcal{L}| \leq \delta$, $\mathsf{chal} \leftarrow \mathsf{GenChal}(1^\tau)$ *and* $\mathcal{V} \leftarrow \mathsf{GenProof}(\mathsf{pk}, B - \mathcal{L}, \mathtt{T}(B - \mathcal{L}), \mathsf{chal})$, *where* $\mathtt{T}(B - \mathcal{L})$ *denotes the tags corresponding to the blocks in* $B - \mathcal{L}$. *A δ-AS scheme is correct if the probability that* $\mathcal{L} \leftarrow \mathsf{CheckProof}(\mathsf{pk}, \mathsf{state}, \mathcal{V}, \mathsf{chal})$ *is at least* $1 - \mathsf{neg}(\tau)$.[3]

To define the security of a δ-AS scheme, the adversary adaptively asks for tags on a set of blocks $B = \{b_1, b_2, \ldots, b_n\}$ that he chooses. After the adversary gets access to the tags, his goal is to output a proof $\mathcal{V}$, so that if $\mathcal{L}$ is output by algorithm $\mathsf{CheckProof}$, where $|\mathcal{L}| \leq \delta$, then (a) either $\mathcal{L}$ is *not* a subset of the original set of blocks B; (b) or the adversary does *not* store all remaining blocks in $B - \mathcal{L}$ intact.

Such a proof is invalid since it would allow the verifier to either recover the *wrong* set of blocks (e.g., a set of blocks whose Hamming distance from the corrupted blocks is a lot smaller) or to accept a corruption of more than δ file blocks.

Definition 3 (δ-AS security). *Let $\mathcal{P}$ be a δ-AS scheme as in Definition 1 and $\mathcal{A}$ be a PPT adversary. We define security using the following steps.*

1. ***Setup.*** *$\mathcal{A}$ chooses $\delta \in [0, n)$, blocks $B = \{b_1, b_2, \ldots, b_n\}$ and is given* $\mathtt{T}_1, \ldots, \mathtt{T}_n$ *and* pk *output by* $\{\mathsf{pk}, \mathsf{sk}, \mathsf{state}, \mathtt{T}_1, \ldots, \mathtt{T}_n\} \leftarrow \mathsf{Setup}(b_1, \ldots, b_n, \delta, 1^\tau)$.[4]
2. ***Forge.*** *$\mathcal{A}$ is given* $\mathsf{chal} \leftarrow \mathsf{GenChal}(1^\tau)$ *and outputs a proof of accountability $\mathcal{V}$.*

Suppose $\mathcal{L} \leftarrow \mathsf{CheckProof}(\mathsf{pk}, \mathsf{state}, \mathcal{V}, \mathsf{chal})$. *We say that the δ-AS scheme $\mathcal{P}$ is secure if, with probability at least* $1 - \mathsf{neg}(\tau)$: *(i) $\mathcal{L} \subseteq B$; and (ii) there exists a PPT knowledge extractor $\mathcal{E}$ that can extract* all the remaining file blocks *in $B - \mathcal{L}$.*

Note here that if the set $\mathcal{L}$ is empty, then the above definition is equivalent to the original PDP security definition [3]. Also note that the notion of a knowledge extractor is similar to the standard one, introduced in the context of proofs of knowledge [5]. If the adversary can output an accepting proof, then he can execute $\mathsf{GenProof}$ repeatedly until it extracts the selected blocks.

[3] Function $\lambda : \mathbb{N} \rightarrow \mathbb{R}$ is $\mathsf{neg}(\tau)$ iff $\forall$ nonzero polynomials $p(\tau)$ there exists N so that $\forall \tau > N$ it is $\lambda(\tau) < 1/p(\tau)$.

[4] $\mathcal{A}$ could also choose blocks adaptively, after seeing tags for already requested blocks. Our proof of security handles that.

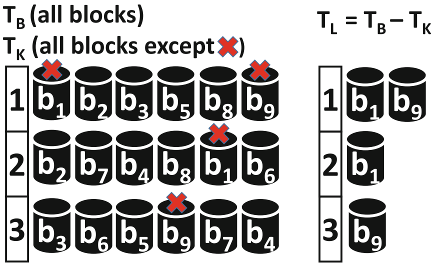
4 Our Basic Construction

We now give an overview of our basic construction: On input blocks $B = \{b_1, \ldots, b_n\}$ in local storage, the client decides on a parameter δ (meaning that he can tolerate up to δ corrupted files) and computes the local state, tags, public and secret key by running $\{\mathsf{pk}, \mathsf{sk}, \mathsf{state}, \mathtt{T}_1, \ldots, \mathtt{T}_n\} \leftarrow \mathsf{Setup}(b_1, \ldots, b_n, \delta, 1^\tau)$. In our construction the tag $\mathtt{T}_i$ is set to $(h(i)g^{b_i})^d \mod N$, as in [3], where $h(.)$ is a collision-resistant hash function, N is an RSA modulus and (e, d) denote an RSA public/private key pair. The client then sends blocks $b_1, \ldots, b_n$ and tags $\mathtt{T}_1, \ldots, \mathtt{T}_n$ to the server and locally stores the state state, which is an IBF of the blocks $b_1, b_2, \ldots, b_n$.

At challenge phase, the client runs $\mathsf{chal} \leftarrow \mathsf{GenChal}(1^\tau)$ that picks a random challenge s and sends it to the server. To generate a proof of accountability (see Fig. 2-left) with $\mathsf{GenProof}$, the server computes an IBF $\mathbf{T}_K$ on the set of blocks that he (believes he) stores, along with a proof of data possession [3] on the same set of blocks. The indices of these blocks are stored in a set Kept. For the computation of the PDP proof, the server uses randomness derived from the challenge s.

To verify the proof, the client takes the difference $\mathbf{T}_L = \mathsf{subtract}(\mathbf{T}_B, \mathbf{T}_K)$ and executes algorithm $\mathsf{recover}$ from Fig. 2-right, which is a modified version of $\mathsf{listDiff}$ from [12]. Algorithm $\mathsf{recover}$ adds blocks whose tags verify to the set of lost blocks $\mathcal{L}$. Then it checks the PDP proof for those block indices corresponding to blocks that were not output by $\mathsf{recover}$. If this PDP proof does not reject, then the client is persuaded that the server stores everything except for blocks in $\mathcal{L}$. To make sure $\mathsf{recover}$ does not fail with a noticeable probability, our construction sets the parameters according to the following corollary. The detailed algorithms of our construction are in Fig. 3.

Corollary 1. *Let τ be the security parameter and B and K be two sets such that $K \subseteq B$ and $|B - K| \leq \delta$. Let $\mathbf{T}_B$ and $\mathbf{T}_K$ be IBFs constructed by algorithm*



T_B (all blocks)
T_K (all blocks except ✖)
1 b_1 b_2 b_3 b_5 b_8 b_9
2 b_2 b_7 b_4 b_8 b_1 b_6
3 b_3 b_6 b_5 b_9 b_7 b_4
$T_L = T_B - T_K$
1 b_1 b_9
2 b_1
3 b_9

```
Algorithm $\mathcal{L} \leftarrow \mathsf{recover}(\mathbf{T}_L)$
$\mathcal{L} = \emptyset$;
while there is a cell $i \leq t$ such that
    $\mathbf{T}_L[i].\mathsf{hashSum}^e / h(ind) = g^{\mathbf{T}_L[i].\mathsf{dataSum}}$
        (where $ind = \mathsf{index}(\mathbf{T}_L[i].\mathsf{dataSum})$)
do
    add $\mathbf{T}_L[i].\mathsf{dataSum}$ to $\mathcal{L}$;
    call $\mathsf{update}(\mathbf{T}_L[i].\mathsf{dataSum}, \mathbf{T}_L)$;
return $\mathcal{L}$;
```

Fig. 2. (**Left**) On input $b_1, b_2, \ldots, b_9$, the client outputs an IBF $\mathbf{T}_B$ of three cells using two hash functions. The server loses blocks b_1 and b_9. $\mathbf{T}_K$ is computed on blocks $b_2, b_3, \ldots, b_8$ and $\mathbf{T}_L$ contains the lost blocks b_1 and b_9. (**Right**) The algorithm for recovering the lost blocks.

Alg. $\{\mathsf{pk}, \mathsf{sk}, \mathsf{state}, \mathtt{T}_1, \ldots, \mathtt{T}_n\} \leftarrow \mathsf{Setup}(b_1, \ldots, b_n, \delta, 1^\tau)$. Let

- $N = pq$ be an RSA modulus of 2τ bits.
- Let e be a random prime and d be a number such that $ed = 1 \mod \phi(N)$.
- $\mathcal{H}$ be a set of k random hash functions $\mathcal{H} = \{h_1, h_2, \ldots, h_k\}$ such that $h_i : \{0,1\}^* \to [t]$, where $t = (k+1)\delta$ and $k = \tau/\log\delta$.
- g be a generator of $\mathbb{QR}_N$.
- $\theta_s : \{0,1\}^\tau \times \{0,1\}^* \to \{0,1\}^\tau$ be a pseudorandom function.
- $h : \{0,1\}^* \to \mathbb{QR}_N$ be a collision-resistant hash function (modeled as random oracle).

Set $\mathsf{pk} = (N, e, \mathcal{H}, g, \theta_s, h)$ and $\mathsf{sk} = d$. For block b_i compute tag

$$\mathtt{T}_i = \left(h(i)g^{b_i}\right)^d \mod N\,.$$

Set state to be the IBF $\mathbf{T}_B$ on $b_1, \ldots, b_n$ computed using Algorithm update from Figure 1.

Alg. $\mathsf{chal} \leftarrow \mathsf{GenChal}(1^k)$. Pick a random $s \in \{0,1\}^\tau$ and output $\mathsf{chal} = s$.

Alg. $\mathcal{V} \leftarrow \mathsf{GenProof}(\mathsf{pk}, b_{i_1}, \ldots, b_{i_m}, \mathtt{T}_{i_1}, \ldots, \mathtt{T}_{i_m}, \mathsf{chal})$. Let $\{b_{i_1}, \ldots, b_{i_m}\} \subseteq \{b_1, \ldots, b_n\}$ be the blocks that the server stores. Set $\mathsf{Kept} = \{i_1, i_2, \ldots, i_m\}$.

1. (PDP proof) Compute a combined tag $\mathtt{T}$ along with a combined sum S as

$$\mathtt{T} = \prod_{i \in \mathsf{Kept}} \mathtt{T}_i^{a_i} \text{ and } S = \sum_{i \in \mathsf{Kept}} a_i b_i, \text{ where } a_i = \theta_{\mathsf{chal}}(i). \tag{2}$$

2. (IBF proof) Compute IBF $\mathbf{T}_K$ on the set of blocks $\{b_i : i \in \mathsf{Kept}\}$.

The proof $\mathcal{V}$ is the tuple $\{\mathtt{T}, S, \mathbf{T}_K\}$.

Alg. $\{\mathsf{reject}, \mathcal{L}\} \leftarrow \mathsf{CheckProof}(\mathsf{pk}, \mathsf{state}, \mathcal{V}, \mathsf{chal})$. Parse $\mathcal{V}$ as $\{\mathtt{T}, S, \mathbf{T}_K\}$, state as $\mathbf{T}_B$.

1. Set $\mathbf{T}_L \leftarrow \mathsf{subtract}(\mathbf{T}_B, \mathbf{T}_K)$
2. Set $\mathcal{L} \leftarrow \mathsf{recover}(\mathbf{T}_L)$ and $\mathsf{Kept} = [n]$.
3. For each $b \in \mathcal{L}$, set $\mathsf{Kept} = \mathsf{Kept} - \{\mathsf{index}(b)\}$.
4. Output reject if $\mathtt{T}^e / \prod_{i \in \mathsf{Kept}} h(i)^{a_i} \neq g^S$, where $a_i = \theta_{\mathsf{chal}}(i)$. Otherwise output $\mathcal{L}$.

Fig. 3. Our δ-AS scheme construction.

update *of Fig. 1 using* $\tau/\log\delta$ *hash functions. The IBFs* $\mathbf{T}_B$ *and* $\mathbf{T}_K$ *have* $t = (\tau/\log\delta + 1)\delta$ *cells and employ tags in the* $\mathsf{hashSum}$ *field that map blocks to* τ *bits. Then with probability at least* $1 - 2^{-\tau}$, *algorithm* $\mathsf{recover}(\mathsf{subtract}(\mathbf{T}_B, \mathbf{T}_K))$ *will output* $\mathcal{L} = B - K$.

Our detailed proof of security is given in the Appendix. The local state that the client must keep is an IBF of $t = (k+1)\delta$ cells, therefore the asymptotic size of the state is $O(\delta)$. For the size of the proof $\mathcal{V}$, the tag $\mathtt{T}$ has size $O(1)$, the sum S has size $O(\log n + \lambda)$ and the IBF $\mathbf{T}_K$ has size $O(\delta)$. Overall, the size of $\mathcal{V}$ is $O(\delta + \log n)$. For the proof computation, note that algorithm $\mathsf{GenProof}$ must first access at least $n - \delta$ blocks in order to compute the PDP proof and then compute an IBF of δ cells over the same blocks, therefore the time is $O(n + \delta)$. Likewise, the verification algorithm needs to verify a PDP proof for a linear number of blocks and to process a proof of size $O(\delta + \log n)$, thus its computation time is again $O(n + \delta)$.

Theorem 1 **(δ-AS scheme).** *Let n be the number of blocks. For all $\delta \leq n$, there exists a δ-AS scheme such that: (1) It is correct according to Definition 2; (2) It is secure in the random oracle model based on the RSA assumption and according to Definition 3; (3) The proof has size $O(\delta + \log n)$ and its computation at the server takes $O(n + \delta)$ time; (4) Verification at the client takes $O(n + \delta)$ time and requires local state of size $O(\delta)$; (5) The space at the server is $O(n)$.*

We now make two observations related to our construction. First, note that the server could potentially launch a DoS attack, by *pretending it does not store some of the blocks* so that the client is forced to spend cycles retrieving these blocks. This is not an issue, since as we will see later, the server will be penalized for that, so it is not in its best interest. Second, note that the tags that the client initially uploads are publicly verifiable so anyone can check their validity—therefore the client cannot upload bogus tags and blame the server later for that.

Streaming and Appending Blocks. Our construction assumes the client has all blocks available in the beginning. This is not necessary. Blocks b_i could come one at a time, and the client could easily update its local state with algorithm $\mathsf{update}(b_i, \mathbf{T}, 1)$, compute the new tag $\mathtt{T}_i$ and send the pair $(b_i, \mathtt{T}_i)$ to the server for storage. This also means that our construction is partially-dynamic, supporting append-only updates. Modifying a block is not so straightforward due to replay attacks. However techniques from various fully-dynamic PDP schemes could be potentially used for this problem (e.g., [13]).

5 Sublinear Construction Using Proofs of Partial Storage

In the previous construction, the server and client run in $O(n + \delta)$ time. In this section we present optimizations that reduce the server and client performance to $O(\delta \log n)$. Recall that the proof generation in Fig. 3 has two distinct, linear-time parts: First, proving that a subset of blocks is kept intact (in particular the blocks with indices in Kept), and second, computing an IBF on this set of blocks. We show here how to execute both these tasks in sublinear time using (i) partial proofs of storage; (ii) a data structure based on segment trees that the client must prepare during preprocessing.

Proofs of Partial Storage. In our original construction, we prove that a subset of blocks is kept intact (in particular the blocks with indices in Kept) using a PDP-style proof, as originally introduced by Ateniese et al. [3]. In our new construction we will replace that part with a new primitive called *proofs of partial storage*. To motivate proofs of partial storage, let us recall how proofs of storage [26] work.

Proofs of storage provide the same guarantees with PDP-style proofs [3] but are much more practical in terms of proof construction time. In particular, one can construct a PoS proof in constant time as follows. Along with the original blocks $b_1, b_2, \ldots, b_n$ the client outsources an additional n redundant blocks $\beta_1, \beta_2, \ldots, \beta_n$ computed with an error-correcting code such as Reed-Solomon, such that any n out of the $2n$ blocks $b_1, b_2, \ldots, b_n, \beta_1, \beta_2, \ldots, \beta_n$ can be used to retrieve the original blocks $b_1, b_2, \ldots, b_n$. Also, the client outsources tags $\mathtt{T}_i$ (as computed in Algorithm Setup in Fig. 3) for all $2n$ blocks. Now, during the challenge phase, the client picks a constant-sized subset of random blocks to challenge (out of the $2n$ blocks), say $\tau = 128$ blocks. Because the subset is chosen at random every time, the server, with probability at least $1 - 2^{-\tau}$, will pass the challenge (i.e., provide verifying tags for the challenged blocks) only if he stores at least half of the blocks $b_1, b_2, \ldots, b_n, \beta_1, \beta_2, \ldots, \beta_n$—which means that the original blocks $b_1, b_2, \ldots, b_n$ are recoverable.

Unfortunately, we cannot use proofs of storage as described above directly, since we want to prove that a *subset of the blocks* is stored intact, and the above construction applies to the whole set of blocks. In the following we describe how to fix this problem using a segment-tree-like data structure.

Our New Construction: Using a Segment Tree. A segment tree T is a binary search tree that stores the set B of n key-value pairs (i, b_i) at the leaves of the tree (ordered by the key). Let v be an internal node of the tree T. Denote with $\mathsf{cover}(v)$ the set of blocks that are included in the leaves of the subtree rooted on node v. Let also $|v| = |\mathsf{cover}(v)|$. Every internal node v of T has a label $\mathsf{label}(v)$ that stores:

1. All blocks $b_1, b_2, \ldots, b_{|v|}$ contained in $\mathsf{cover}(v)$ along with respective tags $\mathtt{T}_i$. The tags are computed as in Algorithm Setup in Fig. 3;
2. Another $|v|$ redundant blocks $\beta_1, \beta_2, \ldots, \beta_{|v|}$ computed using Reed-Solomon codes such that *any* $|v|$ out of the $2|v|$ blocks $b_1, b_2, \ldots, b_{|v|}, \beta_1, \beta_2, \ldots, \beta_{|v|}$ are enough to retrieve the original blocks $b_1, b_2, \ldots, b_{|v|}$. Along with every redundant block β_i, we also store its tag $\mathtt{T}_i$.
3. An IBF T_v on the blocks contained in $\mathsf{cover}(v)$;

By using the segment tree, one can compute functions on *any subset* of $n - \delta$ blocks in $O(\delta \log n)$ time (instead of taking $O(n - \delta)$ time): For example, if $i_1, i_2, \ldots, i_\delta$ are the indices of the omitted δ blocks, the desired IBF T_K can be computed by combining (i.e., XORing the $\mathsf{dataSum}$ and $\mathsf{hashSum}$ fields and adding the count fields):

- The IBF T_1 corresponding to indices from 1 to $i_1 - 1$;
- The IBF T_2 corresponding to indices from $i_1 + 1$ to $i_2 - 1$;
- $\ldots$
- The IBF $T_{\delta+1}$ corresponding to indices from $i_\delta + 1$ to i_n.

Each one of the above IBFs can be computed in $O(\log n)$ time by combining a logarithmic number of IBFs stored at internal nodes of the segment tree and therefore the total complexity of computing the final IBF T_K is $O(\delta \log n)$. Similarly, a partial proof of storage for the lost blocks with indices $i_1, i_2, \ldots, i_\delta$ can be computed by returning.

- A proof of storage corresponding to indices from 1 to $i_1 - 1$;
- A proof of storage corresponding to indices from $i_1 + 1$ to $i_2 - 1$;
- ...
- A proof of storage corresponding to indices from $i_\delta + 1$ to i_n.

Again, each one of the above proofs of storage can be computed by returning $O(\log n)$ partial proofs of storage so in total, one needs to return $O(\delta \log n)$ proofs of storage. Note however that our segment tree increases our space to $O(n \log n)$ and also setting it up requires $O(n \log n)$ time. Therefore we have the following:

Theorem 2 (Sublinear δ-AS scheme). *Let n be the number of blocks. For all $\delta \leq n$, there exists a δ-AS scheme such that: (1) It is correct according to Definition 2; (2) It is secure in the random oracle model based on the RSA assumption and according to Definition 3; (3) The proof has size $O(\delta \log n)$ and its computation at the server takes $O(\delta \log n)$ time; (4) Verification at the client takes $O(\delta \log n)$ time and requires local state of size $O(\delta)$; (5) The space at the server is $O(n \log n)$.*

6 Bitcoin Integration

After the client computes the damage d using the AS protocol described in the previous section, we would like to enable automatic compensation by the server to the client in the amount of d bitcoins. The server initially makes a "security deposit" of A bitcoins by means of a special bitcoin transaction that automatically transfers A bitcoins to the client unless the server transfers d bitcoins to the client before a given deadline. Here, the amount A is a parameter that is contractually established by the client and server and is meant to be larger than the maximum damage that can be incurred by the server.[5]

We have designed a variation of the AS protocol integrated with Bitcoin that, upon termination, achieves one of the following outcomes within an established deadline:

1. If both the server and the client follow the protocol, the client gets exactly d bitcoins from the server and the server gets back his A bitcoins.
2. If the server does not follow the protocol (e.g., he tries to give fewer than d bitcoins to the client, fails to respond in a timely manner, or tries to forge an AS proof), the client gets A bitcoins from the server automatically.
3. If the client requests more than d bitcoins from the server by providing invalid evidence, the server receives all A deposited bitcoins back and the client receives nothing.

[5] Of course, this is just a simple setting, a proof of concept. Clearly other technical and financial instruments can be used to improve this approach if committing such a large amount of A bitcoins is too demanding.

Primitives Used in Our Protocol. Our protocol is using two primitives, which we describe informally in the following.

- A *trusted and tamper-resilient channel* between the client and the server, e.g., a bulletin board. This can be easily implemented by requesting all messages exchanged between the server and the client be posted on the blockchain (note that it is easy for a party P to post arbitrary data D on the blockchain by making a transaction to itself and by including D in the body of the transaction). From now on, we will assume that all messages are posted to the blockchain creating a history *hist*.
- A *trusted bitcoin arbitrator* BA. This is a trusted party that only intervenes in case of disputes. BA will always examine the history of transactions *hist* to assess the situation and determine whether to help the server. In all other cases it can remain offline.

Bitcoin Transactions. Our protocol uses three non-standard bitcoin transactions:

1. $\mathsf{safeGuard}(y)$: This transaction is posted by the server S and it effectively "freezes" A bitcoins to a hash output y. It can be redeemed by a transaction (called $\mathsf{retBtcs}$) posted by the server S which provides the preimage x of $y = \mathsf{H}(x)$ or by a transaction (called $\mathsf{fuse}(t)$) signed by both the client and the server. For the needs of our protocol, $\mathsf{fuse}(t)$ has a locktime t. The $\mathsf{safeGuard}$ transaction is the following:

Prev :	aTransaction
InputsToPrev :	$\mathsf{sig}_S([\mathsf{safeGuard}])$
Conditions :	$\underline{body, \sigma_1, \sigma_2, x :}$ $\mathsf{H}(x) = y \wedge \mathsf{ver}_S(body, \sigma_1)$ $\vee$ $\mathsf{ver}_S(body, \sigma_1) \wedge \mathsf{ver}_C(body, \sigma_2)$
Amount :	A ฿
Locktime :	0

We note here that transaction $\mathsf{safeGuard}(y)$ is based on the timed commitment over Bitcoin by Andrychowicz et al. [2], with an important difference: the committed value x (where $y = \mathsf{H}(x)$) is chosen by the verifier (client) and not by the committer (server). The server just uses y.

2. $\mathsf{retBtcs}$: Once the server gets a hold of value x, it can post the following transaction to redeem $\mathsf{safeGuard}$ and retrieve his A bitcoins.

Prev :	safeGuard
InputsToPrev :	$[\mathsf{retBtcs}], \mathsf{sig}_S([\mathsf{retBtcs}]), \perp, x$
Conditions :	$\underline{body, \sigma :}$ $\mathsf{ver}_S(body, \sigma)$
Amount :	A ฿
Locktime :	0

3. safeGuard can also be redeemed by $\mathsf{fuse}(t)$, as mentioned before:

Prev :	safeGuard
InputsToPrev :	$[\mathsf{fuse}], \mathsf{sig}_S([\mathsf{fuse}]), \mathsf{sig}_C([\mathsf{fuse}]), \perp$
Conditions :	$\underline{body, \sigma :}$ $\mathsf{ver}_C(body, \sigma)$
Amount :	A ฿
Locktime :	t

Protocol Details. We now describe our protocol in detail, as depicted in Fig. 4. Let S denote the server and C the client. For each step $i = 1, \dots, 10$, there is a deadline, t_i, to complete the step, where timelock t of the $\mathsf{fuse}(t)$ transaction is $>> t_{10}$. We recall that, as mentioned in the beginning of this section, all messages exchanged between the client and the server are recorded on the blockchain, creating the history *hist*.

- **Step 1:** C picks a random secret x and sends the following items to S: (i) a hash $hash = \mathsf{H}(\mathbf{T}_B)$ of the IBF of the original blocks he stores; (ii) an encryption $\mathsf{Enc}_{\mathsf{P}}(x)$ of x under BA's public key, P; (iii) a cryptographic hash of x, $y = \mathsf{H}(x)$; and (iv) a zero-knowledge proof, ZKP_1, that $\mathsf{H}(x)$ and $\mathsf{Enc}_{\mathsf{P}}(x)$ encode the same secret x. If ZKP_1 does not verify or is not sent within time t_1, S aborts the protocol.
- **Step 2:** S posts bitcoin transaction $\mathsf{safeGuard}(y)$ for A bitcoins. It also sends to the client a signature of the $\mathsf{Fuse}(t)$ transaction, $\sigma_S = \mathsf{sig}_S([\mathsf{fuse}])$. If this transaction is not posted within time t_2 or the signature is not valid or is not sent within time t_2, C aborts the protocol.
- **Step 3:** S and C run the AS protocol from the previous section. S returns proof $\mathcal{V} = \{\mathsf{T}, S, \mathbf{T}_K\}$ and blocks $b'_1, b'_2, \dots, b'_\delta$ for the current blocks he stores, in the position of the original blocks $b_1, b_2, \dots, b_\delta$ that he lost. The client C computes now the damage d using CheckProof. If CheckProof rejects or S delays it past time t_3, C jumps to Step 9.
- **Step 4:** C notifies S that the damage is d and sends a zero-knowledge proof, ZKP_2, to S for that. If C fails to do so by t_4 or ZKP_2 fails to verify, S jumps to Step 6. We note here that ZKP_2 is for the statement $(hash, \mathcal{V}, b'_1, b'_2, \dots, b'_\delta, d, \mathsf{chal})$: $\exists$ *secret* $\mathbf{T}_B$ *such that*

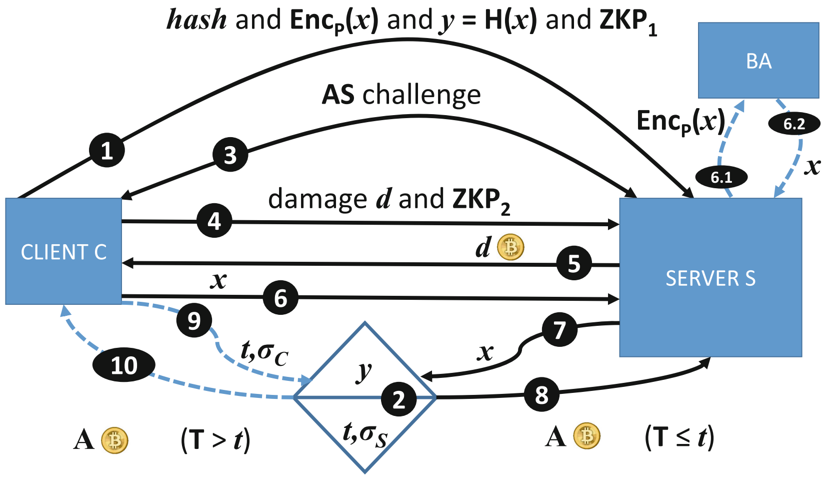


Fig. 4. Integration of the AS protocol with Bitcoin. The dotted lines indicate what happens when a party is trying to cheat. The rhomboid indicates the safeGuard(y) transaction (at the bottom of the rhomboid we show the server signature σ_S on the fuse(t) transaction) that is redeemed either by retBtcs (arrow 7) or by fuse (arrow 9), depending on the flow of the protocol.

$$hash = \mathsf{H}(\mathbf{T}_B) \,\&\, \{b_i\}_{i=1}^{\delta} \leftarrow \mathsf{CheckProof}(\mathsf{pk}, \mathbf{T}_B, \mathcal{V}, \mathsf{chal}) \,\&\, d = \sum_{i=1}^{\delta} ||b_i \oplus b'_i||.$$

The zero-knowledge proof ZKP_2 is needed here since the recovery algorithm takes as input the sensitive state of the client $\mathbf{T}_B$, which we want to hide from the server—otherwise the server can recover the original blocks himself and claim that there was no damage.

- **Step 5:** S sends d bitcoins to C. If S has not done so by time t_5, C jumps to Step 9.
- **Step 6:** C sends secret x to S. If S has not received x by t_6, S contacts BA and asks the BA to examine the history *hist* up to that moment. BA checks *hist* and if it is valid, BA sends x to S.
- **Step 7:** If S has secret x, S posts transaction retBtcs.
- **Step 8:** If transaction retBtcs is valid, S receives A bitcoins before timelock t.
- **Step 9:** C waits until time t, computes $\sigma_C = \mathsf{sig}_S([\mathsf{fuse}])$ and posts transaction fuse(t) using σ_C and σ_S.
- **Step 10:** If transaction fuse is valid, C receives A bitcoins.

It is easy to see that when the above protocol terminates, one of the three outcomes described in the beginning of this section is achieved. We emphasize that BA can determine whether C properly followed the protocol by analyzing *hist*. If C has not done so and S reports it, BA will reveal x to S at any point in time. Also, we note here that for the zero-knowledge proofs ZKP_1 and ZKP_2, we can use a SNARK with zero-knowledge [25], that was recently implemented and shown to be practical.

Global safeGuard. The protocol above protects the client at each AS challenge. But the cloud provider could stop interacting, simply disappear, and never be reachable by the client. Instead of the client aborting, we can use a *global* safeguard transaction at the time the client and the server initiate their business relationship (i.e., when the client uploads the original file blocks and they both sign the SLA). This global transaction is meant to protect the client if the server cannot be reached at all or refuses to collaborate, but creates a scalability problem given that the server has to escrow a large amount of bitcoins for every client/customer. We do not address this problem technically but we expect it can be mitigated through financial mechanisms (securities, commodities, credit, etc.) typically deployed for traditional escrow accounts.

Removing the Bitcoin Arbitrator. Even though BA is only involved in case of disputes, it is preferable to remove it completely. Unfortunately, this seems impossible to achieve *efficiently* given the limitations of the Bitcoin scripting language. We sketch in this section two possible approaches to remove the BA. These will be further explored in future work.

The first approach relies on a secure two-party computation protocol. In a secure two-party computation protocol (2PC), party A inputs x and party B inputs y and they want to compute $f_A(x, y)$ and $f_B(x, y)$ respectively, without learning each other's input other than what can be inferred from the output of the two functions. Yao's seminal result [31] showed that oblivious transfer implies 2PC secure against honest-but-curious adversaries. This result can be extended to generically deal with malicious adversaries through zero-knowledge proofs or more efficiently via the *cut-and-choose* method [20] or LEGO and MiniLEGO [14,24] (other efficient solutions were proposed in [18,30]).

To remove the BA, it is enough to create a symmetric version of our original scheme where both parties create a safeGuard transaction and then exchange the secrets of both commitments through a fair exchange protocol embedded into a 2PC. The secrets must be verifiable in the sense that the fair exchange must ensure the secrets open the initial commitments or fail (as in "committed 2PC" by Jarecki and Shmatikov [18]). Unfortunately, generic techniques for 2PC results in quite impractical schemes and this is the reason why we prefer a practical solution with an arbiter. An efficient 2PC protocol with Bitcoin is proposed in [22] but it does not provide fairness since the 2PC protocol can be interrupted at any time by one of the parties. In the end, since this generic approach is too expensive in practice, we will not elaborate on it any further in this paper.

Another promising approach to remove the BA is to adopt smart contracts. Smart contracts are digital contracts that run through a blockchain. Ethereum [1] is a new cryptocurrency system that provides a Turing-complete language to write such contracts, which is expected to enable several decentralized applications without trusted entities. Smart contracts will enable our protocol to be fully automated without any arbitrators or trusted parties in between. To use a smart contract to run our protocol, the contract is expected to receive a deposit from the server, inputs from both parties, and then it will decide the money flow accordingly based on the CheckProof result. The contract in that case will maintain some properties to ensure fair execution, for example both parties should be incentivized to follow the protocol, and if any party does not follow the protocol (by aborting for example), there should be a mechanism to end the protocol properly for the honest party. To make our protocol fit in the smart contract model, we will need to address the fact that the CheckProof computation would be too expensive to be performed by the contract, due to its overhead. In Ethereum for example, the participants must pay for the cost of running the contract, which is run by the miners.

In order to address the points above, zero knowledge SNARKs [6] could be employed to help reduce miners' overhead, and thus reduce the computational cost of the verification algorithm running on the network, while preserving the secrecy of the inputs.

7 Evaluation

We prototyped the proposed Accountable Storage (AS) scheme in Python 2.7.5. Our implementation is open-source[6] and consists of 4 K lines of source code. We use the pycrypto library 2.6.1 [21] and an RSA modulus N of size 1024 bits. We serialize the protocol messages using Google Protocol Buffers [16] and perform all the modulo exponentation operations using GMPY2 [17], which is a C-coded Python extension module that supports fast multiple precision arithmetic (the use of GMPY2 gave us 60% speedup in exponentiations in comparison with the regular python arithmetic library).

We divide the prototype in two major components. The first is responsible for data pre-processing, issuing proof challenges and verifying proofs (including recover process). The second produces proof every time it receives a challenge. Both modules utilize the IBF data structure to produce and verify proofs. Our prototype uses parallel computing via the Python multiprocessing module to carry out many of the heavy, but independent, cryptographic operations simultaneously. We used a *single-producer, many-consumers* approach to divide the available tasks in a pool of [8–12] processes-workers. The workers use message passing to coordinate and update the results of their computations. This approach significantly enhanced the performance of preprocessing as well as the proof generation and checking phase of the protocol. Our parallel implementation provides an approximate 5x speedup over a sequential implementation.

Finally note that since it can be easily estimated, we have not evaluated the Bitcoin part of our protocol which is dominated by the time it takes for transactions to be part of the blockchain. Nowadays this latency is approximately 10 min.

Experimental Setup: Our experimental setup involves two nodes, one implementing the server and another implementing the client functionality. The two

[6] https://github.com/vlekakis/delta-AccountableStorage.

nodes communicate through a Local Area Network (LAN). The two machines are equipped with an Intel 2.3 Ghz Core i7 processor and have 16 GB of RAM.

Our data are randomly generated filesystems. Every file-system includes different number of equally-sized blocks. The number of blocks ranges from 100 to 500000 and the different sizes of blocks used are 1 KB, 2 KB, 4 KB and 8 KB. The total filesystem size varies from 100 KB to 4.1 GB. Our experiments consist of 10 trials of challenge/proof exchanges between the client and the server for different filesystems. Throughout the evaluation we report the average values over these 10 trials. For some of the large filesystems consisting of 100000 and 500000 number of blocks, we have estimated the results based on the experiments in lower filesystems due to computational power limitations.

In our experiments, we select the tolerance parameter δ, which indicates the maximum amount of data blocks that can be lost, to be equal to $\log_2(n)$. One other possible choice of δ is to set it equal to $\sqrt{n}$. We select the logarithm of the number of blocks as δ, because this provides a harder condition on how many blocks can be lost or corrupted from the cloud server.

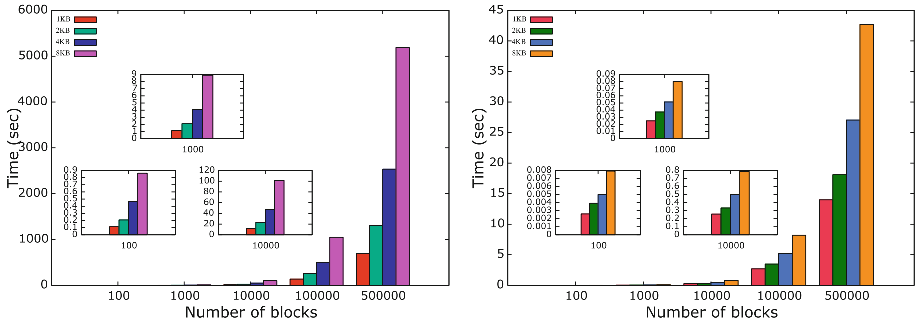
For the IBF construction, we have used the blocks and their generated tags. The selected number of hash functions used for the IBF construction is $k = 6$. This choice of hash functions leads to a very low probability of failure of the recovery algorithm, which depends on the values of k and δ.

Preprocessing Overheads: We first examine the memory overhead of the preprocessing phase, which is shown in Table 1. The first column describes the available number of blocks in a filesystem and the second represents the estimated total size of the tags needed. The preprocessing memory overhead is proportional to the number of blocks in a filesystem.

Figure 5 shows the CPU-time-related overheads of the preprocessing of the protocol. These overheads are divided to tag generation and the creation of the client state represented by the IBF T_B. The tag generation time (Fig. 5a) increases linearly with both the available number of blocks and the size of each block. While this cost is significant for large file systems, it is an operation that client performs only once at the setup phase. On the other hand, the cost of construction of the IBF (Fig. 5b) is estimated to be negligible; the IBF construction of our biggest filesystem is estimated to take around 42 s.

Table 1. Memory footprint of the AS scheme (KB)

n	Tag size (KB)	Proof size (KB)			
		$1KB$	$2KB$	4 KB	8 KB
10^2	32	353	692	1369	2722
10^3	236	528	1035	2048	4073
10^4	2644	762	1493	2593	5875
10^5	24895	937	1898	3526	7226
$5 * 10^5$	118326	1077	2182	4054	8308



(a) Tag Generation

(b) Client IBF Generation

Fig. 5. Preprocessing overheads

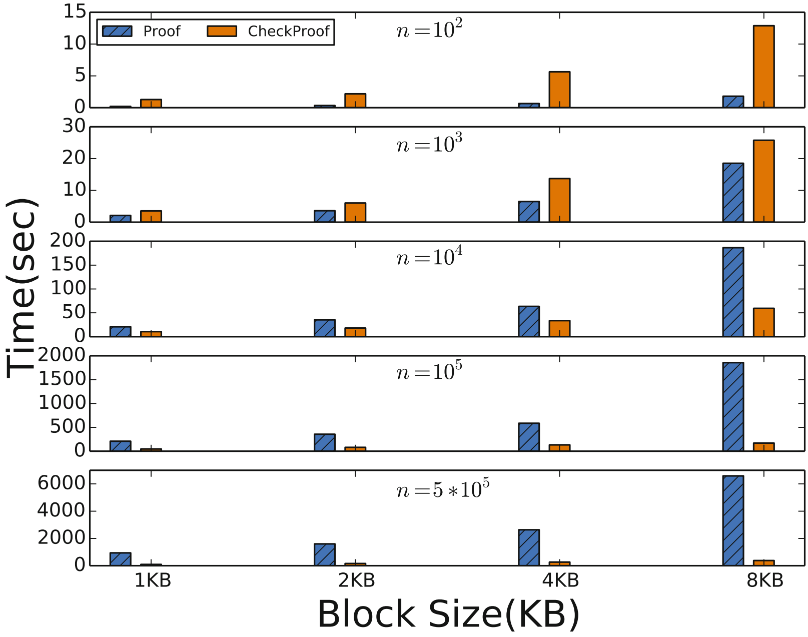


Fig. 6. Proof generation and proof check (including recover) time

Challenge-Proof Overheads: We now examine memory and CPU-related overheads for the challenge-proof exchange and the recovery phase. The last four columns of Table 1 show the proof sizes (in KB) for $\delta = \log_2(n)$, which increase proportionally to the block size.

Every subgraph of Fig. 6 shows how different block sizes affect the performance of the challenge-proof exchange for a given number of blocks. The left bar in the figure shows the proof generation time and the right bar the proof check along that includes the time recover the lost blocks. We notice that larger block sizes are estimated to increase the time-overhead of challenge-proof exchange. We also notice that the proof check and in particular the recover process introduces higher time overhead in comparison to proof generation for small size filesystems (100 and 1000 number of blocks). This is expected, because the tag verification (publicly verifiable) in the recover process (in Fig. 2) introduces significant time overhead compared to proof generation (in Fig. 3) for a small number of blocks. However, in higher size filesystems, we observe from Fig. 6 that the time overhead of proof generation increases exponentially and overcomes the proof check time (including recover process) overhead. This is also expected, because the number of blocks (denoted by Kept set) used in the proof generation process is much higher than the number of blocks used in the recover process.

8 Conclusions

In this paper we put forth the notion of accountability in cloud storage. Unlike existing work such as proof-of-storage schemes and verifiable computation, we design protocols that respond to a verification failure, enabling the client to assess the damage that has occurred in a storage repository. We also present a protocol that enables automatic compensation of the client, based on the amount of damage, and is implemented over Bitcoin. Our implementation shows that our system can be used in practice.

Acknowledgments. Research supported in part by an NSF CAREER award CNS-1652259, NSF grants CNS-1525044, CNS-1526950, CNS-1228639 and CNS-1526631, a NIST award and by the Defense Advanced Research Projects Agency (DARPA) under agreement no. AFRL FA8750-15-2-0092. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

Appendix

RSA Assumption

Definition 4. *Let $N = pq$ be an RSA modulus, where p and q are τ-bit primes. Given N, e and g, where g is randomly chosen from $\mathbb{Z}_N^*$ and e is a prime of $\Theta(\tau)$ bits, there is no PPT algorithm that can output $y^{1/e} \mod N$, except with probability $\mathsf{neg}(\tau)$.*

Proof of Security of Construction in Fig. 3.

Setup. $\mathcal{A}$ chooses parameter $\delta \in [0, n)$, blocks $B = \{b_1, b_2, \ldots, b_n\}$ and is given pk and $\mathtt{T}_1, \ldots, \mathtt{T}_n$ as output by $\{\mathsf{pk}, \mathsf{sk}, \mathsf{state}, \mathtt{T}_1, \ldots, \mathtt{T}_n\} \leftarrow \mathsf{Setup}(b_1, \ldots, b_n, \delta, 1^\tau)$. The random oracle is programmed so that it returns $r_i^e g^{-b_i}$ on input i for some random r_i, i.e., $h(i) = r_i^e g^{-b_i} \mod N$.

Forge. $\mathcal{A}$ is given $\mathsf{chal} \leftarrow \mathsf{GenChal}(1^\tau)$, computes proof of accountability $\mathcal{V}$ and returns $\mathcal{V}$. Suppose $\mathcal{L} \leftarrow \mathsf{CheckProof}(\mathsf{pk}, \mathsf{sk}, \mathsf{state}, \mathcal{V}, \mathsf{chal})$. We must show that with probability $\geq 1 - \mathsf{neg}(\tau)$ it is (i) $\mathcal{L} \subseteq B$; and (ii) there exists a PPT knowledge extractor $\mathcal{E}$ that can extract *all the remaining file blocks* in $B - \mathcal{L}$.

1. **Showing** $\mathcal{L} \subseteq B$. Note that all blocks in $\mathcal{L}$ are output by Algorithm $\mathsf{recover}$ of Fig. 2. In this algorithm a block b'_i can enter $\mathcal{L}$ only if its tag verifies. Suppose now $b'_i \notin B$ (namely $b'_i \neq b_i$) and $\mathtt{tag}^e / h(i) = g^{b'_i}$ for some arbitrary $\mathtt{tag}$ computed by the adversary. But since $h(i) = r_i^e g^{-b_i}$, this can be written as $\mathtt{tag}^e / r_i^e g^{-b_i} = g^{b'_i}$ which gives $g^{b'_i - b_i} = (\mathtt{tag}/r_i)^e$. Since e is a prime and $b_i - b'_i \neq 0$, there exist α and β such that $(b'_i - b_i) \times \alpha + e \times \beta = 1$, giving $g^{1/e} = g^{-\beta}(tag/r_i)^{e\times\alpha}$, breaking the RSA assumption—see Definition 4.
2. **Showing there exists a PPT knowledge extractor $\mathcal{E}$ that can extract *all the remaining file blocks* in** $B - \mathcal{L}$. We now show how to build an extractor that, after $\ell = |B - \mathcal{L}|$ interactions with the adversary, he can extract the blocks $\{b_i : i \in B - \mathcal{L}\}$. The extractor will challenge the adversary exactly ℓ times, each time with different randomness. Let $S_1, S_2, \ldots, S_\ell, \mathtt{T}^{(1)}, \mathtt{T}^{(2)}, \ldots, \mathtt{T}^{(\ell)}$ be the sums and tags he receives by $\mathcal{A}$ during each challenge, as in Eq. (2). We have two cases:
   (a) $S_j = \sum_{i \in B - \mathcal{L}} a_{ij} b_i$, for $j \in B - \mathcal{L}$ (a_{ij} denotes the randomness of the j-th challenge corresponding to the i-th block). In this case, the extractor can solve a system of ℓ linear equations and retrieve the original blocks $\{b_i : i \in B - \mathcal{L}\}$.
   (b) Suppose there exists $j \in B - \mathcal{L}$ such that $S_j \neq \sum_{i \in B - \mathcal{L}} a_{ij} b_i = S$. For simplicity of notation, let's set $\mathtt{T}^{(j)} = \mathtt{T}$ and $S_j = \bar{S}$. Then by the $\mathsf{CheckProof}$ algorithm we have

$$\frac{\mathtt{T}^e}{\prod_{i \in B - \mathcal{L}} h(i)^{a_i}} = g^{\bar{S}}.$$

   But since $h(i) = r_i^e g^{-b_i}$ we have that

$$g^{\bar{S} - S} = \left(\frac{\mathtt{T}}{\prod_{i \in B - \mathcal{L}} r_i^{a_i}} \right)^e = \mathcal{Z}^e.$$

   Therefore we have $\mathcal{Z}^e = g^{\bar{S} - S}$. Again, since e is prime and $S \neq \bar{S}$ we can use the same trick as before, and break the RSA assumption. □

References

1. Ethereum: A platform for decentralized applications. www.ethereum.org/
2. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L.: Secure multiparty computations on bitcoin. In: IEEE SSP (2014)
3. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: ACM CCS (2007)
4. Ateniese, G., Di Pietro, R., Mancini, L.V., Tsudik, G.: Scalable and efficient provable data possession. In: SecureComm (2008)

5. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993). doi:10.1007/3-540-48071-4_28
6. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: verifying program executions succinctly and in zero knowledge. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 90–108. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40084-1_6
7. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Comm. ACM **13**, 422–426 (1970)
8. Boneh, D., Naor, M.: Timed commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000). doi:10.1007/3-540-44598-6_15
9. Carter, I.L., Wegman, M.N.: Universal classes of hash functions. In: ACM STOC (1977)
10. Cash, D., Küpçü, A., Wichs, D.: Dynamic proofs of retrievability via oblivious RAM. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 279–295. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38348-9_17
11. Curtmola, R., Khan, O., Burns, R.C., Ateniese, G.: MR-PDP: multiple-replica provable data possession. In: ICDCS (2008)
12. Eppstein, D., Goodrich, M.T., Uyeda, F., Varghese, G.: What's the difference? Efficient set reconciliation without prior context. In: SIGCOMM (2011)
13. Erway, C., Küpçü, A., Papamanthou, C., Tamassia, R.: Dynamic provable data possession. In: ACM CCS (2009)
14. Frederiksen, T.K., Jakobsen, T.P., Nielsen, J.B., Nordholt, P.S., Orlandi, C.: MiniLEGO: efficient secure two-party computation from general assumptions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 537–556. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38348-9_32
15. Goodrich, M.T., Mitzenmacher, M.: Invertible Bloom Lookup Tables. ArXiv e-prints, January 2011
16. Google. Google protocol buffers. www.developers.google.com/protocol-buffers/
17. Van Horsen, C.: Gmpy2: Mupltiple-precision arithmetic for python. www.gmpy2.readthedocs.org/en/latest/intro.html/
18. Jarecki, S., Shmatikov, V.: Efficient two-party secure computation on committed inputs. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 97–114. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72540-4_6
19. Juels, A., Kaliski Jr. B.S.: PORs: proofs of retrievability for large files. In: ACM CCS (2007)
20. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72540-4_4
21. Litzenberger, D.C.: Pycrypto - the python cryptography toolkit. www.dlitz.net/software/pycrypto/
22. Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L.: Fair two-party computations via the bitcoin deposits. In: FC (2014)
23. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. www.bitcoin.org/bitcoin.pdf
24. Nielsen, J.B., Orlandi, C.: LEGO for two-party secure computation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 368–386. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00457-5_22

25. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: nearly practical verifiable computation. In: IEEE SSP (2013)
26. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008). doi:10.1007/978-3-540-89255-7_7
27. Shi, E., Stefanov, E., Papamanthou, C.: Practical dynamic proofs of retrievability. In: ACM CCS (2013)
28. Stefanov, E., van Dijk, M., Oprea, A., Juels, A.: Iris: a scalable cloud file system with efficient integrity checks. In: ACSAC (2012)
29. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling public verifiability and data dynamics for storage security in cloud computing. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 355–370. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04444-1_22
30. Woodruff, D.P.: Revisiting the efficiency of malicious two-party computation. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 79–96. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72540-4_5
31. Yao, A.C.-C.: How to generate and exchange secrets. In: SFCS (1986)

Maliciously Secure Multi-Client ORAM

Matteo Maffei[1], Giulio Malavolta[2], Manuel Reinert[3](✉),
and Dominique Schröder[2]

[1] TU Wien, Wien, Austria
matteo.maffei@tuwien.ac.at
[2] Friedrich-Alexander Universität Erlangen-Nürnberg, Nürnberg, Germany
{giulio.malavolta,dominique.schroeder}@fau.de
[3] CISPA, Saarland University, Saarbrücken, Germany
reinert@cs.uni-saarland.de

Abstract. Oblivious RAM (ORAM) has emerged as an enabling technology to secure cloud-based storage services. The goal of this cryptographic primitive is to conceal not only the data but also the access patterns from the server. While the early constructions focused on a single client scenario, a few recent works have focused on a setting where multiple clients may access the same data, which is crucial to support data sharing applications. All these works, however, either do not consider malicious clients or they significantly constrain the definition of obliviousness and the system's practicality. It is thus an open question whether a natural definition of obliviousness can be enforced in a malicious multi-client setting and, if so, what the communication and computational lower bounds are.

In this work, we formalize the notion of maliciously secure multi-client ORAM, we prove that the server-side computational complexity of any secure realization has to be $\Omega(n)$, and we present a cryptographic instantiation of this primitive based on private information retrieval techniques, which achieves an $O(\sqrt{N})$ communication complexity. We further devise an efficient access control mechanism, built upon a novel and generally applicable realization of plaintext equivalence proofs for ciphertext vectors. Finally, we demonstrate how our lower bound can be bypassed by leveraging a trusted proxy, obtaining logarithmic communication and server-side computational complexity. We implemented our scheme and conducted an experimental evaluation, demonstrating the feasibility of our approach.

1 Introduction

Oblivious RAM. Cloud storage has rapidly become a central component in the digital society, providing a seamless technology to save large amounts of data, to synchronize them across multiple devices, and to *share* them with other parties. Popular data-sharing, cloud-based applications are e.g., personal health record management systems (PHRs), like those employed in Austria [22] and Estonia [20], collaborative platforms (e.g., Google Docs), and credit score

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 645–664, 2017.
DOI: 10.1007/978-3-319-61204-1_32

systems (e.g., Experian, Equifax, and TransUnion in US) are just a few popular data-sharing, cloud-based applications taking advantage of such features. A stringent and well-understood security requirement is *access control*: read and write access should be granted only to authorized clients.

While access control protects user's data from other clients, encryption on the server's side is needed to obtain privacy guarantees against cloud administrators. Encryption is, however, not enough: as shown in the literature [28,39], the capability to observe which data are accessed by which users allows the cloud administrator to learn sensitive information: for instance, it has been shown that the access patterns to a DNA sequence allow for determining the patient's disease. The property of hiding data accesses is called *obliviousness* and the corresponding cryptographic construction *Oblivious RAM* (ORAM): while the first constructions were highly inefficient [23], recent groundbreaking research paved the way for a tremendous efficiency boost, exploiting ingenious tree-based constructions [2,3,8,14,15,24,32,39,44,45,47], server side computations [26,35], and trusted hardware [5,27,31,42,48].

Except for a few recent noticeable exceptions, discussed below, a fundamental limitation of all these constructions is that they target a single-client architecture, where the data owner is the only party allowed to read outsourced data, which does not make them suitable for data sharing services. The fundamental challenge to solve is to enforce access control and obliviousness *simultaneously*. These properties are seemingly contradictory: can the server check the correctness of data accesses with respect to the access control policy at all, if it is not allowed to learn anything about them?

Multi-Client ORAM. A few recent constructions gave positive answers to this question, devising ORAM constructions in the multi-client setting, which specifically allow the data owner to share data with other clients while imposing fine-grained access control policies. Although, at a first glance, these constructions share the same high-level goal, they actually differ in a number of important aspects. Therefore we find it interesting to draw a systematic comparison among these approaches (cf. Table 1). First of all, obliviousness is normally defined against the server, but in a multi-client setting it is important to consider it against the clients too (**MC**), since they might be curious or, even worse, collude with the server. This latter aspect is important, since depending on the application, the cloud administrator might create fake clients or just have common interests with one of the legitimate clients. Some constructions allow multiple data owners to operate on the same ORAM (**MD**), while others require them to use disjoint ORAMs: the latter are much less efficient, since if the client does not want to reveal the owner of the accessed entry (e.g., to protect her anonymity, think for instance of the doctor accessing the patient's record), then the client has to perform a fake access to each other ORAM, thereby introducing a multiplicative factor of O(m), where m is the number of data owners. Some constructions require the data owner to periodically access the dataset in order to validate previous accesses (**PI**), some others rely on server-side client synchronization, which can be achieved for instance by a shared log on the server, a gossiping protocol among clients, etc. (**CS**), while others assume a trusted proxy (**Pr**). Among these, gossiping is the mildest assumption since it can be realized directly on the server side as described by [30]. Another aspect to consider is the possibility for the data owner to specify fine-grained access control mechanisms (**AC**). Finally, some constructions enable concurrent accesses to the ORAM (**P**). The final three columns compare the asymptotic complexity of server-side and client-side computations as well as communication.

Table 1. Comparison of the related work supporting multiple clients to our constructions. The abbreviations mean: **MC**: oblivious against malicious clients, **MD**: supports multiple data owners sharing their data in one ORAM, **PI**: requires the periodic interaction with the data owner, **CS**: requires synchronization among clients, **AC**: access control, **Pr**: trusted proxy, **P**: parallel accesses, **S comp.**: server computation complexity, **C comp.**: client communication complexity, **Comm.**: communication complexity.

Work	**MC**	**MD**	**PI**	**CS**	**Pr**	**AC**	**P**	**S comp.**	**C comp.**	**Comm.**
Franz *et al.* [21]	✓	✗	✓	✗	✗	✓	✗	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$
GORAM [33]	✗	✗	✗	✓	✗	✓	✗	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$
PIR-MCORAM (this work)	✓	✓	✗	✓	✗	✓	✗	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$
BCP-OPRAM [7]	✗	✗	✓	✓	✗	✗	✓	$\Omega(\log^3(n))$	$\Omega(\log^3(n))$	$\Omega(\log^3(n))$
CLT-OPRAM [10]	✗	✗	✓	✓	✗	✗	✓	$O(\log^2(n))$	$O(\log^2(n))$	$O(\log^2(n))$
PrivateFS [49]	✗	✗	✗	✓	✗	✗	✓	$O(\log^2(n))$	$O(1)$	$O(\log^2(n))$
Shroud [31]	✗	✗	✗	✗	✓	✗	✓	$O(\log^2(n))$	$O(1)$	$O(\log^2(n))$
TaoStore [42]	✗	✗	✗	✗	✓	✗	✓	$O(\log(n))$	$O(1)$	$O(\log(n))$
TAO-MCORAM (this work)	✓	✓	✗	✗	✓	✓	✓	$O(\log(n))$	$O(1)$	$O(\log(n))$

Franz *et al.* pioneered the line of work on multi-client ORAM, introducing the concept of delegated ORAM [21]. The idea of this construction, based on simple symmetric cryptography, is to let clients commit their changes to the server and to let the data owner periodically validate them according to the access control policy, finally transferring the valid entries into the actual database. Assuming periodic accesses from the data owner, however, constrains the applicability of this technique. Furthermore, this construction does not support multiple data owners. Finally, it guarantees the obliviousness of access patterns with respect to the server as well as malicious clients, excluding however the accesses on data readable by the adversary. While excluding write operations is necessary (an adversary can clearly notice that the data has changed), excluding read operations is in principle not necessary and limits the applicability of the obliviousness definition: for instance, we would like to hide the fact that an oncologist accessed the PHR of a certain patient even from parties with read access to the PHR (e.g., the pharmacy, which can read the prescription but not the diagnosis).

More recently, Maffei *et al.* [33] proposed the notion of group ORAM, in which the server performs access control by verifying client-provided zero-knowledge proofs: this approach enables direct client accesses without any interaction with the data owner and more generic access control policies. The scheme relies on a gossiping protocol, but malicious clients are considered only in the context of access control and, indeed, obliviousness does not hold against them.

Another line of work, summarized in the lower part of Table 1, focuses on the parallelization of client accesses, which is crucial to scale to a large number of clients, while retaining obliviousness guarantees. Most of them [5,31,42,48] assume a trusted proxy performing accesses on behalf of users, with TaoStore [42] being the most efficient and secure among them. These constructions do not formally consider obliviousness against malicious clients nor access control, although a contribution of this work is to prove that a simple variant of TaoStore [42] guarantees both. Finally, instead of a trusted proxy, BCP-OPRAM [7] and CLT-OPRAM [10] rely on a gossiping protocol while PrivateFS [49] assumes a client-maintained log on the server-side, but they do not achieve obliviousness against malicious clients nor access control. Moreover, PrivateFS guarantees concurrent client accesses only if the underlying ORAM already does so.

To summarize, the progress in the field does not answer a few foundational questions, which touch the core of the application of ORAM technologies in cloud-based data-sharing applications. First, is it possible at all to enforce the obliviousness of data accesses without constraining the security definition or placing severe system assumptions? If the answer is positive, it would be interesting to know at what computational cost.

Our Contributions. This work answers the questions above, providing a foundational framework for multi-client ORAM. In particular,

- We give for the first time a formal definition of *obliviousness against malicious clients in the multi-client setting*. Intuitively, none should be able to determine which entry is read by which client. However, write operations are oblivious only with respect to the server and to those clients who cannot read the modified entry, since clients with read access can obviously notice that the entry has changed.
- We establish an insightful *computational lower bound*: in a multi-client setting where clients have *direct access* to the database, the number of operations *on the server side* has to be linear in the database size. Intuitively, the reason is that if a client does not want to access all entries in a read operation, then it must know where the required entry is located in the database. Since malicious clients can share this information with the server, the server can determine for each read operation performed by an honest client, which among the entries the adversary has access to might be the subject of the read, and which certainly not.
- We present PIR-MCORAM, the first *cryptographic construction* that ensures the obliviousness of data accesses as well as access control in a malicious multi-client setting. Our construction relies on Private Information Retrieval (PIR) [11] to achieve obliviousness and uses new accumulation technique

based on an oblivious gossiping protocol to reduce the communication bandwidth in an amortized fashion. Moreover, it combines public-key cryptography and zero-knowledge proofs for access control.

- We present a novel technique based on universal pair-wise hash functions [9] in order to speed up the efficiency of Plaintext Equivalence Proofs, a computationally demanding cryptographic building block of PIR-MCORAM. This construction is generally applicable and we show that it improves solutions recently adopted in the multi-client ORAM literature [33] by one order of magnitude.
- To bypass the aforementioned lower bound, we consider the recently proposed *proxy-based Setting* [5,31,42,48,49], which assumes the presence of a trusted party mediating the accesses between clients and server. We prove, in particular, that a simple variant of TaoStore [42] guarantees obliviousness in the malicious setting as well as access control.
- We implement PIR-MCORAM and conduct an *experimental evaluation* of our schemes. PIR-MCORAM constitutes a practical solution for databases of modest size: for instance, DNA encoded in VCF files requires approximately 125MB [40]. Thus, an extended personal health record fits without problems in a 256MB database, for which a read or write operation in PIR-MCORAM takes approximately 14 seconds amortized. TaoStore offers much better performance as well as support for parallel accesses, but it assumes a trusted proxy.

2 A Lower Bound for Maliciously Secure Multi-Client ORAM

In this section, we study how much computational effort is necessary to securely realize ORAM in the malicious multi-client setting. Our result shows that *any* construction, regardless of the underlying computational assumptions, must access the entire memory (up to a constant factor) in every operation. Our lower bound can be seen as a generalization of the result on history independence of Roche et al. [41], in the sense that they consider a "catastrophic attack" where the complete state of the client is leaked to the adversary, whereas we allow only the corruption of a certain subset of clients. Note that, while the bound in [41] concerns the *communication* complexity, our result only bounds the *computation* complexity on the server side.

Before stating our lower bound, we formalize the notion of Multi-Client ORAM in the malicious setting. We follow the definitional framework introduced by Maffei *et al.* [33], refining the obliviousness definition in order to consider malicious clients possibly colluding with the server.

2.1 Multi-Client Oblivious RAM

In a Multi-Client ORAM scheme the parties consist of the data owner $\mathcal{O}$, several clients $\mathcal{C}_1, \ldots, \mathcal{C}_k$, and the server $\mathcal{S}$. The data owner outsources its database $\mathcal{DB}$

to $\mathcal{S}$ while granting access to the clients $\mathcal{C}_1, \ldots, \mathcal{C}_k$ in a selective manner. This is expressed by an access control matrix ACM which has an entry $\mathsf{ACM}(i, \mathsf{idx})$ for every client $\mathcal{C}_i$ and every entry idx in the database, characterizing which access right $\mathcal{C}_i$ has for entry idx: either no access ($\perp$), read-only access (R), or read-write access (RW). We treat ACM as a global variable for the data owner so as to ease the presentation. Moreover, ACM is only accessible to the data owner and not to any client. We write $o \leftarrow A(\ldots)$ to denote that algorithm A on some input generates output o. Likewise, we write $\langle o_{\mathcal{C}}, o_{\mathcal{S}} \rangle \leftarrow \langle A(\ldots), \mathcal{S}_A(\ldots) \rangle$ to denote that the protocol A executed between the client and the server yields client output $o_{\mathcal{C}}$ and server output $o_{\mathcal{S}}$.

Definition 1 (Multi-Client ORAM [33]). *A* Multi-Client ORAM *scheme Θ is composed of the following (interactive)* PPT *algorithms:*

$(cap_{\mathcal{O}}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda, n)$. *The generation algorithm initializes a database $\mathcal{DB}$ of size n and an empty access control matrix* ACM. *Finally, the algorithm returns the data owner's capability $cap_{\mathcal{O}}$.*

$cap_i \leftarrow \mathsf{addCl}(cap_{\mathcal{O}}, i)$. *The input of the add client algorithm is the data owner's capability $cap_{\mathcal{O}}$ and a client identifier i. It appends a row corresponding to i in* ACM *such that for all $j \in \{1, \ldots, n\} : \mathsf{ACM}(i, j) = \perp$. The algorithm outputs the capability for client $\mathcal{C}_i$.*

$\langle \perp, \mathcal{DB}' \rangle \leftarrow \langle \mathsf{addE}(cap_{\mathcal{O}}, \mathsf{idx}, \mathsf{data}), \mathcal{S}_{\mathsf{addE}}(\mathcal{DB}) \rangle$. *The add entry algorithm takes as input the data owner's capability $cap_{\mathcal{O}}$, an index* idx*, and a data* data *in interaction with $\mathcal{S}$ that takes $\mathcal{DB}$ as input. It appends a column corresponding to* idx *in* ACM *such that for all $i \in \{1, \ldots, k\} : \mathsf{ACM}(i, \mathsf{idx}) = \perp$, writes* data *at position* idx *in $\mathcal{DB}$, and outputs the modified database $\mathcal{DB}'$ on $\mathcal{S}$.*

$\langle \perp, \mathcal{DB}' \rangle \leftarrow \langle \mathsf{chMode}(cap_{\mathcal{O}}, \mathsf{idx}, i, p), \mathcal{S}_{\mathsf{chMode}}(\mathcal{DB}) \rangle$. *The change mode algorithmtakes as input the data owner's capability $cap_{\mathcal{O}}$, some index* idx*, a client identifier i, and a permission $p \in \{\mathsf{R}, \mathsf{RW}, \perp\}$ in interaction with $\mathcal{S}$ that takes $\mathcal{DB}$ as input. It updates the entry* $\mathsf{ACM}(i, \mathsf{idx})$ *to p and returns the modified database $\mathcal{DB}'$ on $\mathcal{S}$.*

$\langle \mathsf{data}, \perp \rangle \leftarrow \langle \mathsf{read}(\mathsf{idx}, cap_i), \mathcal{S}_{\mathsf{read}}(\mathcal{DB}) \rangle$. *The read algorithm takes as input an index* idx *and a client capability cap_i on the client side and the database $\mathcal{DB}$ on $\mathcal{S}$ and returns a data* data *on the client and generates no output on the server.*

$\langle \mathsf{data}', \mathcal{DB}' \rangle \leftarrow \langle \mathsf{write}(\mathsf{idx}, cap_i, \mathsf{data}), \mathcal{S}_{\mathsf{write}}(\mathcal{DB}) \rangle$. *The write algorithm takes as input an index* idx*, a client capability cap_i, and a data* data *on the client side and the database $\mathcal{DB}$ on $\mathcal{S}$. Let* data' *be the data stored at* idx *in $\mathcal{DB}$. The protocol modifies $\mathcal{DB}$ at index* idx *to* data. *Finally, it returns* data' *on the client side as well as the modified database $\mathcal{DB}'$ on $\mathcal{S}$.*

Attacker Model. The data owner is assumed to be trusted, since she is interested to protect her data. We allow the server to be fully compromised and to corrupt an arbitrary subset of clients. As explained below, this attacker model is relaxed when it comes to the integrity of outsourced data, which can only be achieved by assuming an honest-but-curious server (while still allowing for client compromise), as discussed below.

Security. A Multi-Client ORAM has four fundamental security properties. The first three concern access control and are intuitively described below.

Secrecy: only users with at least read permissions on an entry can learn its content.

Integrity: only users with write permissions on an entry can change its content.

Tamper Resistance: only users with write permissions on an entry can change its content in a way that the updated entry is considered valid by honest clients.

The difference between integrity and tamper-resistance is that integrity prevents unauthorized changes and thus requires an honest-but-curious server to perform access control, while tamper resistance is a weaker property that allows clients to detect unauthorized changes a-posteriori and thus can in principle be achieved even if the server is malicious.

Obliviousness Against Malicious Clients. Intuitively, a Multi-Client ORAM is secure if the server and an arbitrary subset of clients cannot get any information about the access patterns of honest clients, other than what is trivially leaked by the entries that the corrupted clients have read access to. The original obliviousness definition [33] does not allow the server to corrupt honest clients: here we extend it to handle static corruption of the clients and, in order to avoid trivial attacks, we restrict the queries of the adversary to the write oracle to indices that the set of corrupted clients cannot read.

Definition 2 (Obliviousness against Malicious Clients). *A Multi-Client ORAM Θ is* secure against malicious clients*, if for all* PPT *adversaries $\mathcal{A}$ the success probability of $\mathcal{A}$ in the following experiment is negligibly close to* $1/2$.

1. *$\mathcal{A}$ commits to a set of client identifiers* ID.
2. *The challenger samples $b \in \{0,1\}$, executes $(\mathsf{ACM}, \mathcal{DB}) \leftarrow \mathsf{gen}(1^\lambda, n)$ and forwards $\mathcal{DB}$ to $\mathcal{A}$ and hands over the capabilities of all the clients $\in \mathsf{ID}$ to $\mathcal{A}$.*
3. *The adversary has access to the following interfaces that he can query adaptively and in any order.*

   $\mathsf{addCl}_{cap_{\mathcal{O}}}(i)$: *The challenger adds an empty row entry to* ACM *corresponding to i.*

   $\mathsf{addE}_{cap_{\mathcal{O}}}(\mathsf{idx}, \mathsf{data})$: *The challenger runs $\langle \mathsf{addE}(\mathsf{idx}, \mathsf{data}, cap_{\mathcal{O}}), \mathcal{A}\rangle$ in interaction with $\mathcal{A}$.*

   $\mathsf{chMode}_{cap_{\mathcal{O}}}(\mathsf{idx}, i, \{\mathsf{R}, \mathsf{RW}, \bot\})$: *The challenger runs $\langle \mathsf{chMode}(cap_{\mathcal{O}}, \mathsf{idx}, i, \{\mathsf{R}, \mathsf{RW}, \bot\}), \mathcal{A}\rangle$ in interaction with $\mathcal{A}$.*

   $\mathsf{read}(\mathsf{idx}, i)$: *The challenger runs $\langle \mathsf{read}(\mathsf{idx}, cap_i), \mathcal{A}\rangle$ in interaction with $\mathcal{A}$.*

   $\mathsf{write}(\mathsf{idx}, i, \mathsf{data})$: *The challenger runs $\langle \mathsf{write}(\mathsf{idx}, cap_i, \mathsf{data}), \mathcal{A}\rangle$ in interaction with $\mathcal{A}$.*

   $\mathsf{query}((\mathsf{op}_0, \mathsf{op}_1), (\mathsf{idx}_0, \mathsf{idx}_1), (i_0, i_1), (\mathsf{data}_0, \mathsf{data}_1))$: *The challenger checks in case that $\mathsf{op}_0 = \mathsf{write}$ or $\mathsf{op}_1 = \mathsf{write}$ if there is an $i \in \mathsf{ID}$ such that $\mathsf{ACM}(i, \mathsf{idx}_0) \neq \bot$ and $\mathsf{ACM}(i, \mathsf{idx}_1) \neq \bot$, if this is the case the challenger aborts. Otherwise it executes $\langle \mathsf{read}(\mathsf{idx}_b, cap_{i_b}), \mathcal{A}\rangle$ (or $\langle \mathsf{write}(\mathsf{idx}_b, cap_{i_b},$*

$\mathsf{data}_b), \mathcal{A}\rangle$*, depending on the operation) in interaction with* $\mathcal{A}$*. In case* $\mathsf{op}_0 = \mathsf{write}$ *or* $\mathsf{op}_1 = \mathsf{write}$*, from this moment on, the queries of* $\mathcal{A}$ *to the interface* chMode *on any* $i \in \mathsf{ID}$ *and* idx_0 *or* idx_1 *are forbidden.*

4. $\mathcal{A}$ *outputs a bit* b'*, the challenger returns 1 if* $b' = b$*.*

2.2 Formal Result

In the following we state a formal lower bound on the computational complexity of any ORAM secure against malicious clients. We denote by physical addresses of a database the memory addresses associated with each storage cell of the memory. Intuitively, the lower bound says that the server has to access a constant fraction of the dataset for any read and write operation.

Theorem 1. *Let* n *be the number of entries in the database and* Θ *be a multi-client ORAM scheme. If* Θ *accesses on average* $o(n)$ *physical addresses for each read and write operation (over the random coins of the read or write operation, respectively),* Θ *is not secure against malicious clients (see Definition 2).*

We formally prove this theorem in our full version [34].

2.3 Discussion

Given the lower bound established in the previous section, we know that any multi-client ORAM scheme that is secure against malicious clients *must* read and write a constant fraction of the database on every access. However, the bound does not impose any restriction on the required communication bandwidth. In fact, it does not exclude constructions with sublinear communication complexity, where the server performs a significant amount of computation. In particular, the aforementioned lower bound calls for the deployment of private information retrieval (PIR) [11] technologies, which allow a client to read an entry from a database without the server learning which entry has been read.

The problem of private database modification is harder. A naïve approach would be to let the client change each entry in the database $\mathcal{DB}$ upon every access, which is however too expensive. Homomorphic encryption might be a natural candidate to outsource the computation to the server and to reduce the required bandwidth: unfortunately, Ostrovsky and Skeith III [37] showed that no private database modification (or PIR writing) scheme with sublinear communication (in the worst case) can be implemented using algebraic cryptographic constructions, such as linearly homomorphic encryption schemes. This result does not apply to schemes based on fully-homomorphic encryption, which is however hardly usable in practice due to the high computation cost associated with the currently known schemes.

The following sections describe our approach to bypass these two lower bounds. First we show how to integrate non-algebraic techniques, specifically out-of-band communication among clients, in order to achieve sublinear *amortized* communication complexity (Sect. 3). Second, we show how to leverage a trusted proxy performing the access to the server on behalf of clients in order to reach a logarithmic overhead in communication and server-side computation, with constant client-slide computation (Sect. 5).

3 PIR-MCORAM

In this section, we present a high-level description of $\mathsf{PIR\text{-}MCORAM}$, a Multi-Client ORAM scheme based on PIR. The full details can be found in our full version [34]. Our construction is inspired by Franz *et al.* [21], who proposed to augment the database with a stack of modified entries, which is periodically flushed into the database by the data owner. In our construction, we let each client $\mathcal{C}_i$ maintain its own temporary stack of entries S_i that is stored on the server side in addition to the regular database $\mathcal{DB}$. These stacks contain recent changes to entries in $\mathcal{DB}$ and to entries in other clients' stacks, which are not yet propagated to $\mathcal{DB}$. In contrast to the approach by Franz *et al.* [21], clients themselves are responsible to flush their stack once it is filled (i.e., after $|\mathsf{S}_i|$ many operations), *without requiring any intervention of the data owner*. An *oblivious gossiping protocol*, which can be realized using standard techniques [16, 30], allows clients to find the most up-to-date entry in the database, thereby obtaining a sublinear communication bandwith even for write operations and thus bypassing the impossibility result by Ostrovsky and Skeith III [37].

More precisely, when operating on index j, the client performs a PIR read on $\mathcal{DB}$ and on all stacks S_i, which can easily be realized since all stacks are stored on the server. Thanks to the oblivious gossiping protocol, the client knows which index is the most current one. At this point, the client appends either a dummy entry (read) or a real entry (write) to its personal stack. If the stack is full, the client flushes it. Flushing means to apply all changes in the personal stack to the database. To be oblivious, the client has to ensure that *all* entries in $\mathcal{DB}$ change. Moreover, for guaranteeing correctness, the client has to ensure that it does not overwrite entries which are more recent than those in its stack.

After explaining how to achieve obliviousness, we also need to discuss how to realize access control and how to protect the clients against the server. Data secrecy (i.e., read access control) is obtained via public-key encryption. Tamper-resistance (i.e., a-posteriori detection of illegal changes) is achieved by letting each client sign the modified entry so that others can check that this entry was produced by a client with write access. Data integrity (i.e., write access control) is achieved by further letting each client prove to the server that it is eligible to write the entry. As previously mentioned, data integrity is stronger than tamper-resistance, but assumes an honest-but-curious server: a malicious server may collude with malicious clients and thus store arbitrary information without checking integrity proofs.

3.1 Analysis

We elaborate on the communication complexity of our solution. We assume that $|\mathsf{DB}| = N$, that there are M clients, and we set the stack length $len_{\mathsf{S}} = \sqrt{N}$

for every client. The worst case for an operation, hence, happens every $\sqrt{N}$-th operation for a client $\mathcal{C}_i$, meaning that besides extracting the data from the database and adding an entry to the personal stack, $\mathcal{C}_i$ has also to flush the stack. We analyze the four algorithms independently: extracting data requires two PIR reads, one on DB and the other on the concatenation of all stacks. Thus, the overall cost is $\mathsf{PIR}(N) + \mathsf{PIR}(M\sqrt{N})$. Adding an entry to the personal stack always requires to upload one entry, independently of whether this replacement is real or dummy.

Our flushing algorithm assumes that $\mathcal{C}_i$ holds $\sqrt{N}$ entries and then down-and-uploads every entry of DB. Thus, the overall complexity is $2N + \sqrt{N}$. A similar analysis shows that if the client holds only $O(1)$ many entries, then $\mathcal{C}_i$ down-and-uploads DB but additionally performs a PIR step for every downloaded entry in its own stack to retrieve a potential replacement, resulting in a complexity of $2N + N \cdot \mathsf{PIR}(\sqrt{N})$.

To conclude, the construction achieves a worst-case complexity of $O(\mathsf{PIR}(N) + \mathsf{PIR}(M\sqrt{N}) + N)$ and $O(\mathsf{PIR}(N) + \mathsf{PIR}(M\sqrt{N}) + N\mathsf{PIR}(\sqrt{N}))$ for $O(\sqrt{N})$ and $O(1)$ client-side memory, respectively. By amortizing the flush step over $\sqrt{N}$ many operations, we achieve an amortized complexity of $O(\mathsf{PIR}(N) + \mathsf{PIR}(M\sqrt{N}) + \sqrt{N})$ or $O(\mathsf{PIR}(N) + \mathsf{PIR}(M\sqrt{N}) + \sqrt{N}\mathsf{PIR}(\sqrt{N}))$, respectively. Since our construction is parametric over the specific PIR protocol, we can leverage the progress in this field: at present, the best $\mathsf{PIR}(N)$ is $O(\log\log(N))$ [17] and, hence, the amortized cost becomes $O(\log\log(M\sqrt{N}) + \sqrt{N})$ or $O(\log\log(M\sqrt{N}) + \sqrt{N}\log\log(N))$, respectively. Since, in most scenarios, $M\sqrt{N} < 2^{2^{N/2}}$, we get $O(\sqrt{N})$ and $O(\sqrt{N}\log\log(N))$.

3.2 Discussion

The construction presented in this section leverages PIR for reading entries and an accumulated PIR writing technique to replace old entries with newer ones. Due to the nature of PIR, one advantage of the construction is its possibility to allow multiple clients to concurrently read from the database and to append single entries to their stacks. This is no longer possible when a client flushes her personal stack since the database is entirely updated, which might lead to inconsistent results when reading from the database. To overcome this drawback, we present a fully concurrent, maliciously secure Multi-Client ORAM in Sect. 5. Another drawback of the flush algorithm is the cost of the integrity (zero-knowledge) proofs. Since we have to use public-key encryption as the top-layer encryption scheme for every entry to allow for proving properties about the underlying plaintexts, the number of proofs to be computed, naïvely implemented, is proportional to the block size. Varying block sizes require us to split an entry into chunks and encrypt every chunk separately since the message space of public-key encryption is a constant amount of bits. The zero-knowledge proof has then to be computed on every of these encrypted chunks. To overcome this linear dependency, we present a new proof paradigm to make the number of computed zero-knowledge proofs independent of the block size in Sect. 4.

4 Integrity Proof Revised: The Hash-and-Proof Paradigm

In this section, we focus on the integrity proofs employed in our construction, presenting a novel and generally applicable cryptographic technique to boost their efficiency.

Plaintext Equivalence Proofs. To guarantee the integrity of the database, our construction requires extensive use of proofs showing that the ciphertexts were correctly rerandomized. In the literature, these proofs are called plaintext-equivalence-proofs (PEPs) and are the main efficiency bottleneck of our writing algorithm. Since the block size of an entry is in general *much larger* than the message space of the encryption scheme, we have to compute zero-knowledge proofs over vectors of ciphertexts. In this case, the integrity proof shows *for each* of these ciphertext vectors that they have been correctly rerandomized. The computational cost for these proofs scales linearly with the block size, which is clearly an undesirable dependency. In fact, this problem is not unique to our setting but affects any system deploying PEPs over long entries, among others verifiable secret shuffling [4,25,36] and mix networks [29].

In the following, we put forward a general technique to improve the computational efficiency of PEPs over ciphertext vectors. Our approach is fully black-box, non-interactive, and its proof size is *independent* of the number of ciphertexts of each entry. Thus, our technique can be used to boost the efficiency of not only $\mathsf{PIR\text{-}MCORAM}$, but also any system based on PEPs. The basic idea behind our solution is to homomorphically compute a pairwise independent hash function [9] over the plaintexts of the two vectors and a PEP over the two resulting ciphertexts. Intuitively, a pairwise independent hash function is a collection of compressing functions such that the probability of two inputs to yield the same output is negligibly small in the size of the output domain (over the random choice of the function). This property ensures that the soundness of the proof is preserved.

General Problem Description. Let $\Pi_{\mathsf{PKE}} = (\mathsf{Gen}_{\mathsf{PKE}}, \mathsf{E}, \mathsf{D}, \mathsf{Rnd})$ be a randomizable, additively homomorphic public-key encryption scheme and $(\mathcal{P}, \mathcal{V})$ be a zero-knowledge proof system (ZKP) that takes as input two instances of ciphertexts $(c, b) \in \mathsf{E}(ek, m)^2$ for some $m \in \mathcal{M}$ and outputs a proof π for the statement $\exists r : b = \mathsf{Rnd}(ek, c, r)$. Construct a zero-knowledge proof system $(\mathcal{P}^*, \mathcal{V}^*)$ that takes as input two vectors of ciphertexts of length n, $(\mathbf{c}, \mathbf{b}) \in \mathsf{E}(ek, \mathbf{m})^{n \times 2}$ for some vector $\mathbf{m}$ and a vector $\mathbf{r}$ of randomnesses of the same length and outputs a proof π^* of the following statement: for all $i \in \{1, \dots, n\}$ there exists a value r_i such that $b_i = \mathsf{Rnd}(ek, c_i, r_i)$. The efficiency goal is to make the size of the proof as well as the invocations of $(\mathcal{P}, \mathcal{V})$ independent of n. Knowing the decryption key dk, this statement is equivalent to the following one: for all $i \in \{1, \dots, n\}$ we have $\mathsf{D}(dk, b_i) = \mathsf{D}(dk, c_i)$.

Our Solution. Let $\mathcal{M} = \mathbb{F}_p$ be the message space of Π_{PKE} for some field $\mathbb{F}_p$, such as the ElGamal or the Paillier encryption scheme [18,38]. We describe our solution as an honest-verifier Σ-protocol which can be made non-interactive and resilient against any malicious verifier by applying the Fiat-Shamir heuristic [19]. In the following, $\mathsf{E}(ek, z_0; z_1)$ denotes the encryption of z_0 with key ek and randomness z_1.

(1) $\mathcal{P}^*$ sends the vectors $(\mathbf{c}, \mathbf{b})$ to $\mathcal{V}^*$.
(2) $\mathcal{V}^*$ samples a vector $\mathbf{z} \in \mathbb{F}_p^{n+2}$ uniformly at random and sends it to $\mathcal{P}^*$.
(3) $\mathcal{P}^*$ computes $c' \leftarrow \mathsf{E}(ek, z_0; z_1) \bigotimes_{i=1}^n z_{i+1} \cdot c_i$ and $b' \leftarrow \mathsf{E}(ek, z_0; z_1) \bigotimes_{i=1}^n z_{i+1} \cdot b_i$ and runs $\mathcal{P}$ on inputs (c', b') to obtain π; $\mathcal{P}^*$ sends π to $\mathcal{V}^*$, who can recompute (c', b') and run $\mathcal{V}$ on $((c', b'), \pi)$. $\mathcal{V}^*$ returns the output of $\mathcal{V}$.

Security Analysis. In the following we state the formal guarantees of our techniques.

Theorem 2 (Hash-and-Proof). *Let Π_{PKE} be an additively homomorphic CPA-secure public-key encryption scheme and let $(\mathcal{P}, \mathcal{V})$ be a* ZKP *for PEPs over Π_{PKE}. Then $(\mathcal{P}^*, \mathcal{V}^*)$ is a* ZKP *for PEPs over Π_{PKE}.*

Proof. The correctness of Π_{PKE} and of the ZKP $(\mathcal{P}, \mathcal{V})$ imply the *correctness* of the protocol described above. The *zero-knowledge* of the protocol follows from the zero-knowledge of $(\mathcal{P}, \mathcal{V})$. Arguing about the *soundness* requires a more accurate analysis: we define as $\mathsf{cheat}_{(\mathcal{P}^*, \mathcal{V}^*)}$ the event where a malicious $\mathcal{P}^*$ fools $\mathcal{V}^*$ into accepting a proof over a false statement. This event happens with probability

$$\begin{aligned}\Pr\left[\mathsf{cheat}_{(\mathcal{P}^*, \mathcal{V}^*)}\right] = & \Pr\left[\mathsf{cheat}_{(\mathcal{P}, \mathcal{V})} \mid \mathsf{D}(dk, c') = \mathsf{D}(dk, b')\right] \\ & \cdot \Pr\left[\mathsf{D}(dk, c') = \mathsf{D}(dk, b')\right] + \\ & \Pr\left[\mathsf{cheat}_{(\mathcal{P}, \mathcal{V})} \mid \mathsf{D}(dk, c') \neq \mathsf{D}(dk, b')\right] \\ & \cdot \Pr\left[\mathsf{D}(dk, c') \neq \mathsf{D}(dk, b')\right]\end{aligned}$$

where the probabilities are taken over the random coins of $\mathcal{P}^*$ and $\mathcal{V}^*$. By the soundness of $(\mathcal{P}, \mathcal{V})$ we get

$$\begin{aligned}\Pr\left[\mathsf{cheat}_{(\mathcal{P}^*, \mathcal{V}^*)}\right] \leq \; & 1 \cdot \Pr\left[\mathsf{D}(dk, c') = \mathsf{D}(dk, b')\right] + \mu \cdot \Pr\left[\mathsf{D}(dk, c') \neq \mathsf{D}(dk, b')\right] \\ \leq \; & \mu + \Pr\left[\mathsf{D}(dk, c') = \mathsf{D}(dk, b')\right]\end{aligned}$$

where μ is a negligible function in the security parameter. Therefore, to prove soundness, it is sufficient to show that when $\mathsf{cheat}_{(\mathcal{P}^*, \mathcal{V}^*)}$ happens, then the probability $\Pr\left[\mathsf{D}(dk, c') = \mathsf{D}(dk, b')\right]$ is a negligible function in the security parameter. We shall note that, due to the homomorphic properties of Π_{PKE}, the resulting plaintext of c' and b' are $z_0 + \sum_{i=1}^n z_{i+1} \mathsf{D}(dk, c_i) \in \mathbb{F}_p$, and $z_0 + \sum_{i=1}^n z_{i+1} \mathsf{D}(dk, b_i) \in \mathbb{F}_p$, respectively. It is easy to see that this corresponds to the computation of the *universal pair-wise hash function* $\mathsf{h}_{(\mathbf{z})}$ as described by Carter and Wegman in [9] (Proposition 8). It follows that for all $\mathbf{c} \neq \mathbf{b}$ the resulting plaintexts of c' and b' are uniformly distributed over $\mathbb{F}_p$, thus $\Pr\left[\mathsf{D}(dk, c') = \mathsf{D}(dk, b')\right] = p^{-2}$, which is a negligible function in the security parameter. This concludes our proof. □

5 Proxy-Based Realization

Driven by the goal of building an efficient and scaleable Multi-Client ORAM that is secure against malicious users, we explore the usage of a trusted proxy mediating accesses between clients and the server, an approach advocated in recent parallel ORAM constructions [5,42,48]. In contrast to previous works, we are not only interested in parallel accesses, but also in handling access control and providing obliviousness against multiple, possibly malicious, clients.

TaoStore [42]**.** In a nutshell, trusted proxy-based ORAM constructions implement a single-client ORAM which is run by the trusted entity on behalf of clients, which connect to it with read and write requests in a parallel fashion. We leverage the state of the art, TaoStore [42], which implements a variant of a Path-ORAM [46] client on the proxy and allows for retrieving multiple paths from the server concurrently. More specifically, the proxy consists of the *processor* and the *sequencer*. The processor performs read and write requests to the untrusted server: this is the most complex part of TaoStore and we leave it untouched. The sequencer is triggered by client requests and forwards them to the processor which executes them in a concurrent fashion.

Our Modifications. Since the proxy is trusted, it can enforce access control. In particular, we can change the sequencer so as to let it know the access control matrix and check for every client's read and write requests whether they are eligible or not. As already envisioned by Sahin *et al.* [42], the underlying ORAM construction can be further refined in order to make it secure against a malicious server, either by following the approach based on Merkle-trees proposed by Stefanov *et al.* [46] or by using authenticated encryption as suggested by Sahin *et al.* [42]. In the rest of the paper, we call the system TAO-MCORAM.

6 Security and Privacy Results

In the following we report the security results for PIR-MCORAM: those for TAO-MCORAM follow along the same lines. For the formal definition of the security properties we refer to [33]. Note that in our proofs we consider the adaptive version of each definition where the attacker is allowed to spawn and corrupt clients without restrictions. As a consequence, our instantiation requires us to fix in advance the number of clients M supported by the construction. Alternatively, one could consider the selective versions of the security definitions where the attacker is required to commit in advance to the client subset that he wants to corrupt. We postpone full proofs to our full version [34].

Theorem 3 (Secrecy). *Let Π_{PKE} be a CPA-secure encryption scheme, then PIR-MCORAM achieves secrecy.*

Theorem 4 (Integrity). *Let Π_{DS} be an existentially unforgeable digital signature scheme, ZKP be a zero-knowledge proof of knowledge protocol, and Π_{PKE} be a CCA-secure encryption scheme, then PIR-MCORAM achieves integrity.*

Theorem 5 (Tamper Resistance). *Let Π_{DS} be an existentially unforgeable digital signature scheme and let Π_{PKE} be a CCA-secure encryption scheme, then PIR-MCORAM achieves tamper resistance.*

Theorem 6 (Obliviousness against mal. clients). *Let* PIR *be a private information retrieval scheme, let Π_{PKE} be a CPA-secure encryption scheme, let Π_{DS} be an existentially unforgeable digital signature scheme, and let* ZKP *be a zero-knowledge proof of knowledge protocol, then PIR-MCORAM is secure against malicious clients.*

7 Evaluation

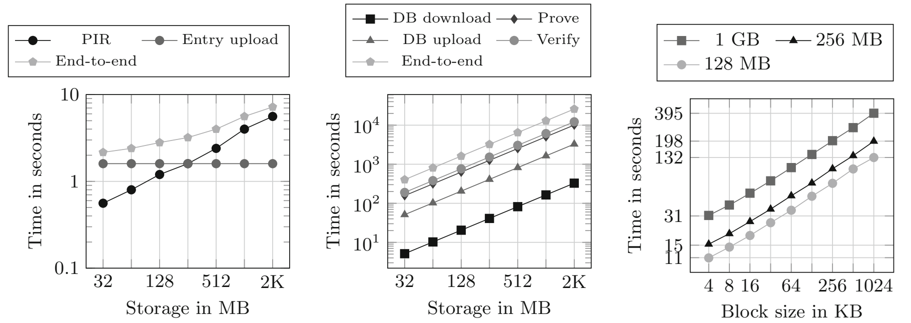
In this section, we describe our implementation and report on the experimental results. We start by reviewing the cryptographic schemes that we deploy: all of them are instantiated with a security parameter of 128 bits [6].

Cryptographic Instantiations. We deploy ElGamal encryption [18] in a hybrid fashion to construct an entry in the database. Using the hybrid technique, we decrease the entry size from $O(MB)$ to $O(M + B)$ since the data is encrypted only once and the corresponding secret key is encrypted for all clients with read access. In contrast, we encrypt the signing keys of the Schnorr signature scheme [43] using the Cramer-Shoup encryption scheme [13]. We use XPIR [1], the state of the art in computational PIR.

Finally, in order to construct integrity proofs, we use an OR-proof [12] over a conjunction of plaintext-equivalence proofs [29] (PEP) on the ElGamal ciphertexts forming one entry and a standard discrete logarithm proof [43] showing that the client knows the signing key corresponding to the authenticated verification key. In the homomorphic hash version, the conjunction of PEPs reduces to the computation of the homomorphic hash plus one PEP. As a matter of fact, since the public components necessary to verify a proof (the new and old ciphertexts and the verification key) and the secret components necessary to compute the proof (the randomness used for rerandomization or the signing key) are independent of the number of clients, all deployed proofs solely depend on the block size.

Implementation and Experiments. We implemented the cryptographic components of PIR-MCORAM in Java and we use a wrapper to GMP to speed up computations.

We used an Intel Xeon E5-4650L with 2.60 GHz, 8 cores, and 20 MB cache for the client and server experiments. We performed micro-benchmarks for PIR-MCORAM while varying the storage size from 32 MB to 2 GB and the block size from 4 KB to 1 MB, both for the solution with and without the homomorphic hash computation. We measured partial computation times as well as the end-to-end access time where we assume a network with 100 Mbit/s downstream and 10 Mbit/s upstream. In order to show the efficiency of our homomorphic hash construction and demonstrate its generic applicability, we also compare GORAM [33] with batched shuffle proofs, as originally presented, with a variant thereof where we replace the batched shuffle proofs with our homomorphic hash plus one shuffle proof.



(a) Access without flush, 1 MB block size. (b) Flush, 1 MB block size. (c) Amortized time, varying block size.

Fig. 1. The end-to-end running time of an operation in PIR-MCORAM.

Discussion. Figures 1 and 2 report the results for PIR-MCORAM. Figure 1a shows the end-to-end and partial running times of an access to the ORAM when the flush algorithm is not executed, whereas Fig. 1b depicts the worst case running time (i.e., with flush operation). For the example of the medical record which usually fits into 128 MB (resp. 256 MB for additional files such as X-ray images), the amortized times per access range from 11 (resp. 15) seconds for 4 KB up to 131 (resp. 198) seconds for 1 MB sized entries (see Fig. 1c).

Figure 2 shows the improvement as we compare the combined proof computation and proof verification time in the flush algorithm of PIR-MCORAM, first as described in Sect. 3 and then with the integrity proof based on the universal homomorphic hash (see Sect. 4). We observe that our expectations are fulfilled: the larger the block size, the more effect has the universal hash computation since the number of proofs to compute decreases. Concretely, with 1 MB block size we gain a speed-up of about 4% for flush operations with respect to the construction without homomorphic hash.

To demonstrate its general applicability, we instantiate our proof technique into GORAM [33], which uses so-called batched shuffle proofs, achieving much better results. In GORAM, clients have to compute integrity proofs, which are proofs of shuffle correctness–a much more expensive primitive than PEPs. To overcome the efficiency problem, the authors have developed batched shuffle proofs: the idea is to homomorphically sum up half of the columns of the database matrix at random and to perform a shuffle proof on the resulting list of ciphertexts. To achieve soundness, the protocol has to be repeated $k = 128$ times. We observe that we can replace batched shuffle proofs by our protocol: clients compute the homomorphic hash on the old and the new ciphertexts and then one shuffle proof on the resulting lists of ciphertexts. As shown in Table 2, this modification speeds up GORAM by one order of magnitude (14x on the client and 10.8x on the server).

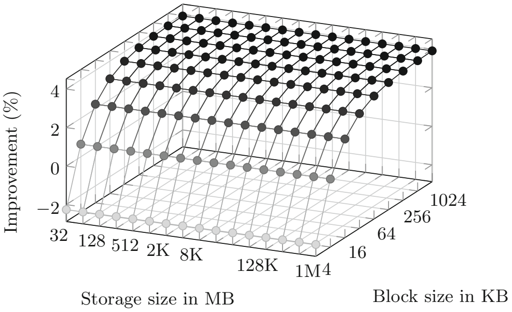


Fig. 2. The improvement in percent when comparing the combined proof computation time on the client and proof verification time on the server for varying storage and block sizes, once without and once with the universal homomorphic hash.

Table 2. Comparison of GORAM [33] with batched shuffle proofs and GORAM instantiated with our homomorphic hash (HH) variant for 10 users, 1 GB storage, and 8 KB block size.

Construction	Client time	Server time
GORAM with $k = 128$	91.315 s	39.213 s
GORAM with HH	5.980 s	3.384 s
Improvement	**14x**	**10.8x**

Finally, our solution TAO-MCORAM only adds access control to the actual computation of TaoStore's trusted proxy [42]. Interestingly enough, TaoStore's bottleneck is not computation, but communication. Hence, our modifications do not cause any noticeable slowdown on the throughput of TaoStore. Consequently, we end up with a throughput of about 40 operations per second when considering an actual deployment of TAO-MCORAM in a cloud-based setting [42].

8 Conclusion

This work studies the problem of obliviousness in multi-client outsourced storage. We establish a lower bound on the server-side computational complexity, showing that any secure realization has to involve at least $\Omega(n)$ computation steps. We further present a novel cryptographic instantiation, which achieves an amortized communication overhead of $O(\sqrt{n})$ by combining private information retrieval technologies, a new accumulation technique, and an oblivious gossiping protocol. Access control is enforced by efficient integrity proofs, which leverage a new construction for Plaintext Equivalence Proofs based on a homomorphic universal pair-wise hash function. Finally, we showed how to bypass our lower bound by leveraging a trusted proxy [42], thereby achieving logarithmic communication and server side computational complexity.

This work opens up a number of interesting research directions. Among those, it would be interesting to prove a lower bound on the communication complexity. Furthermore, we would like to relax the obliviousness property in order to bypass the lower bound established in this paper, coming up with more efficient constructions and quantifying the associated privacy loss.

Acknowledgements. This research is based upon work supported by the German research foundation (DFG) through the collaborative research center 1223, by the German Federal Ministry of Education and Research (BMBF) through the Center for IT-Security, Privacy and Accountability (CISPA), and by the state of Bavaria at the Nuremberg Campus of Technology (NCT). NCT is a research cooperation between the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Technische Hochschule Nürnberg Georg Simon Ohm (THN). Dominique Schröder is supported by the German Federal Ministry of Education and Research (BMBF) through funding for the project PROMISE. Finally, we thank the reviewers for their helpful comments.

References

1. Aguilar-Melchor, C., Barrier, J., Fousse, L., Killijian, M.O.: XPIR : private information retrieval for everyone. In: Proceedings of the Privacy Enhancing Technologies Symposium (PETS 2016), pp. 155–174. De Gruyter (2016)
2. Ajtai, M.: Oblivious RAMs without cryptographic assumptions. In: Proceedings of ACM Symposium on Theory of Computing (STOC 2010), pp. 181–190. ACM (2010)
3. Apon, D., Katz, J., Shi, E., Thiruvengadam, A.: Verifiable oblivious storage. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 131–148. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54631-0_8
4. Bayer, S., Groth, J.: Efficient zero-knowledge argument for correctness of a shuffle. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 263–280. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29011-4_17
5. Bindschaedler, V., Naveed, M., Pan, X., Wang, X., Huang, Y.: Practicing oblivious access on cloud storage: the gap, the fallacy, and the new way forward. In: Proceedings of the Conference on Computer and Communications Security (CCS 2015), pp. 837–849. ACM (2015)
6. BlueKrypt: Cryptograhpic Key Length Recommendation. www.keylength.com
7. Boyle, E., Chung, K.-M., Pass, R.: Oblivious parallel RAM and applications. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016 Part II. LNCS, vol. 9563, pp. 175–204. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49099-0_7
8. Carbunar, B., Sion, R.: Regulatory compliant oblivious RAM. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 456–474. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13708-2_27
9. Carter, J.L., Wegman, M.N.: Universal classes of hash functions (extended abstract). In: Proceedings of the ACM Symposium on Theory of Computing (STOC 1977), pp. 106–112. ACM (1977)
10. Chen, B., Lin, H., Tessaro, S.: Oblivious parallel RAM: improved efficiency and generic constructions. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 205–234. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49099-0_8

11. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM **45**(6), 965–981 (1998)
12. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994). doi:10.1007/3-540-48658-5_19
13. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998). doi:10.1007/BFb0055717
14. Damgård, I., Meldgaard, S., Nielsen, J.B.: Perfectly secure oblivious RAM without random oracles. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 144–163. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19571-6_10
15. Dautrich, J., Stefanov, E., Shi, E.: Burst ORAM: minimizing ORAM response times for bursty access patterns. In: Proceedings of the USENIX Security Symposium (USENIX 2014), pp. 749–764. USENIX Association (2014)
16. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., Terry, D.: Epidemic algorithms for replicated database maintenance. In: Proceedings of the Symposium on Principles of Distributed Computing (PODC 1987), pp. 1–12. ACM (1987)
17. Dong, C., Chen, L.: A fast single server private information retrieval protocol with low communication cost. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014 Part I. LNCS, vol. 8712, pp. 380–399. Springer, Cham (2014). doi:10.1007/978-3-319-11203-9_22
18. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985). doi:10.1007/3-540-39568-7_2
19. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). doi:10.1007/3-540-47721-7_12
20. The EH Foundation. http://www.e-tervis.ee
21. Franz, M., Williams, P., Carbunar, B., Katzenbeisser, S., Peter, A., Sion, R., Sotakova, M.: Oblivious outsourced storage with delegation. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 127–140. Springer, Heidelberg (2012). doi:10.1007/978-3-642-27576-0_11
22. GmbH, E.: ELGA. https://www.elga.gv.at
23. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. J. ACM **43**(3), 431–473 (1996)
24. Goodrich, M.T., Mitzenmacher, M.: Privacy-preserving access of outsourced data via oblivious RAM simulation. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011. LNCS, vol. 6756, pp. 576–587. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22012-8_46
25. Groth, J.: A verifiable secret shuffe of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2003). doi:10.1007/3-540-36288-6_11
26. Huang, Y., Goldberg, I.: Outsourced private information retrieval with pricing and access control. In: Proceedings of the Annual ACM Workshop on Privacy in the Electronic Society (WPES 2013). ACM (2013)
27. Iliev, A., Smith, S.W.: Protecting client privacy with trusted computing at the server. IEEE Secur. Priv. **3**(2), 20–28 (2005)

28. Islam, M., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In: Proceedings of the Annual Network & Distributed System Security Symposium (NDSS 2012). Internet Society (2012)
29. Jakobsson, M., Juels, A.: Millimix: mixing in small batches. Technical report, pp. 99–33. DIMACS (1999)
30. Kim, B.H., Lie, D.: Caelus: verifying the consistency of cloud services with battery-powered devices. In: Proceedings of the IEEE Symposium on Security & Privacy (S&P 2015), pp. 880–896. IEEE Press (2015)
31. Lorch, J.R., Parno, B., Mickens, J., Raykova, M., Schiffman, J.: Shroud: ensuring private access to large-scale data in the data center. In: Proceedings of the USENIX Conference on File and Storage Technologies (FAST 2013), pp. 199–214. USENIX Association (2013)
32. Maas, M., Love, E., Stefanov, E., Tiwari, M., Shi, E., Asanovic, K., Kubiatowicz, J., Song, D.: PHANTOM: practical oblivious computation in a secure processor. In: Proceedings of the Conference on Computer and Communications Security (CCS 2013), pp. 311–324. ACM (2013)
33. Maffei, M., Malavolta, G., Reinert, M., Schröder, D.: Privacy and access control for outsourced personal records. In: Proceedings of the IEEE Symposium on Security & Privacy (S&P 2015). IEEE Press (2015)
34. Maffei, M., Malavolta, G., Reinert, M., Schröder, D.: Maliciously Secure Multi-Client ORAM. Cryptology ePrint Archive, Report 2017/329 (2017). eprint.iacr.org
35. Mayberry, T., Blass, E.O., Chan, A.H.: Efficient private file retrieval by combining ORAM and PIR. In: Proceedings of the Annual Network & Distributed System Security Symposium (NDSS 2014). Internet Society (2013)
36. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Proceedings of Conference on Computer and Communications Security (CCS 2001), pp. 116–125. ACM (2001)
37. Ostrovsky, R., III, W.E.S.: Algebraic Lower Bounds for Computing on Encrypted Data. In: Electronic Colloquium on Computational Complexity (ECCC), vol. 14, no. 022 (2007)
38. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). doi:10.1007/3-540-48910-X_16
39. Pinkas, B., Reinman, T.: Oblivious RAM revisited. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 502–519. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14623-7_27
40. Robinson, R.J.: How big is the human genome? https://medium.com/precision-medicine/how-big-is-the-human-genome-e90caa3409b0
41. Roche, D.S., Aviv, A., Choi, S.G.: A practical oblivious map data structure with secure deletion and history independence. In: Proceedings of the IEEE Symposium on Security & Privacy (S&P 2016). IEEE Press (2016)
42. Sahin, C., Zakhary, V., Abbadi, A.E., Lin, H.R., Tessaro, S.: TaoStore: overcoming asynchronicity in oblivious data storage. In: Proceedings of the IEEE Symposium on Security & Privacy (S&P 2016). IEEE Press (2016)
43. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 688–689. Springer, Heidelberg (1990). doi:10.1007/3-540-46885-4_68
44. Shi, E., Chan, T.-H.H., Stefanov, E., Li, M.: Oblivious RAM with $O((\log N)^3)$ worst-case cost. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 197–214. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25385-0_11

45. Stefanov, E., Shi, E., Song, D.: Towards practical oblivious RAM. In: Proceedings of the Annual Network & Distributed System Security Symposium (NDSS 2012). Internet Society (2012)
46. Stefanov, E., van Dijk, M., Shi, E., Fletcher, C., Ren, L., Yu, X., Devadas, S.: Path ORAM: an extremely simple oblivious RAM protocol. In: Proceedings of the Conference on Computer and Communications Security (CCS 2013). ACM (2013)
47. Stefanov, E., Shi, E.: Multi-cloud oblivious storage. In: Proceedings of the Conference on Computer and Communications Security (CCS 2013), pp. 247–258. ACM (2013)
48. Stefanov, E., Shi, E.: ObliviStore: high performance oblivious cloud storage. In: Proceedings of the IEEE Symposium on Security & Privacy (S&P 2013), pp. 253–267. IEEE Press (2013)
49. Williams, P., Sion, R., Tomescu, A.: PrivateFS: a parallel oblivious file system. In: Proceedings of the Conference on Computer and Communications Security (CCS 2012), pp. 977–988. ACM (2012)

Legacy-Compliant Data Authentication for Industrial Control System Traffic

John Henry Castellanos(✉), Daniele Antonioli, Nils Ole Tippenhauer, and Martín Ochoa

Singapore University of Technology and Design, Singapore, Singapore
{john_castellanos,daniele_antonioli}@mymail.sutd.edu.sg,
{nils_tippenhauer,martin_ochoa}@sutd.edu.sg

Abstract. Industrial Control Systems (ICS) commonly rely on unencrypted and unauthenticated communication between devices such as Programmable Logic Controllers, Human-Machine-Interfaces, sensors, and actuators. In this work, we discuss solutions to extend such environments with established cryptographic authentication schemes. In particular, we consider schemes that are legacy compliant in the sense that authentication data is embedded as additional payload for domain specific protocols, for example the industrial EtherNet/IP protocol. To that end, we propose a selective protocol (that signs every critical packet sent) and a protocol that aggregates groups of packets based on real-time requirements and the available throughput, for various realistic hardware configurations. We evaluate our analysis by implementing an authenticated channel in a realistic Water Treatment testbed.

Keywords: Industrial Control Systems · Authentication · Network security

1 Introduction

Industrial Control Systems (ICS) commonly rely on unencrypted and unauthenticated communication between the industrial devices such as Programmable Logic Controllers (PLC), Human-Machine-Interfaces (HMI), sensors, and actuators. The use of cryptographic schemes in such devices is often hindered by their long lifetime, compatibility issues, low processing power of the embedded devices, and real-time requirements in the communication [9]. Most common industrial communication protocols do not feature any built-in capabilities for authentication (e.g., Modbus/TCP, EtherNet/IP). In the past, critical infrastructure control networks' were isolated from the office and corporate networks, and thus malware and other advanced attacks did not pose a realistic threat to such control networks.

Nowadays, with the increased connectivity to general IT infrastructures such as a LAN, and the Internet itself, attacks such as the well-known Man-in-the-Middle between ICS devices are more realistic [29]. For ICS communications,

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 665–685, 2017.
DOI: 10.1007/978-3-319-61204-1_33

message *integrity* is paramount, while *confidentiality* of messages exchanged is less important. In particular, if an attacker can alter the values of the sensors or the commands being sent to the actuators, he could effectively alter the control of the ICS and potentially cause the malfunctioning of the physical system.

A number of works (e.g., [2,3,5,23,26,31] to name a few) highlight the importance of securing the networks of ICS. Most of them also remark that by using cryptography a non negligible overhead can be introduced, and that such systems usually have strict timing constraints. However, to the best of our knowledge, no detailed analysis on cryptography-enabled authentication has been reported so far for ICS under realistic constraints. In particular, data in ICS is also often passing through intermediate gateways and other industrial network appliances. For that reason, the solution must be legacy-compliant (which can prevent solutions such as TLS encapsulation).

In this paper, we explore the application of well-known cryptographic primitives to verify authenticity of communication between devices in modern ICS, embedded as payload in legacy industrial protocols. In particular, we are interested in solutions that would either be feasible to implement in constrained legacy devices, or could be provided by low-cost additional devices. The focus of the paper is *to detect* manipulations of legitimate traffic by the attacker, and the presence of new messages introduced by the attacker. If confidentiality is required, an additional suitable encryption scheme should be used. In addition, the discussion of consequences of successful detection of message manipulation attacks is out of scope of this work. The mitigation to such an attack is a subject on its own, since in general one cannot simply drop incorrectly signed packets: on one hand, even a single missing packet could have unforeseen consequences in controlling physical processes, on the other hand if packets are dropped attackers could easily implement denial of service attacks [15].

In order to maximize the efficiency of legacy-compliant authentication while preserving security, we propose a basic authentication protocol that signs a subset of *critical* packets. This is in contrast to closely related work, that usually focuses on authenticating *all* of the communication between nodes in ICS [18,25,28,33]. In addition, we propose a second protocol that aggregates groups of packets based on the real-time requirements, network throughput, and the processor capacity for various realistic hardware configurations. We evaluate our analysis experimentally in an industrial Water Treatment testbed [11] (Secure Water Treatment, SWaT) by means of additional network components. However our protocols can be deployed by controller manufacturers as a firmware modification of existing Ethernet modules.

We summarize our contributions as follows. In this work we: **(a)** discuss design options for legacy-compliant basic authentication protocols in the context of ICS networks; **(b)** perform an experimental performance evaluation of a number of cryptographic primitives on several hardware platforms; **(c)** propose a novel aggregated authentication protocol, adapted to requirements of ICS; **(d)** empirically evaluate the proposed protocol in a Water Treatment testbed.

2 Preliminaries

In the following we introduce some fundamental notions on Industrial Control Systems, and introduce the attacker model and the expected security guarantees of our solution.

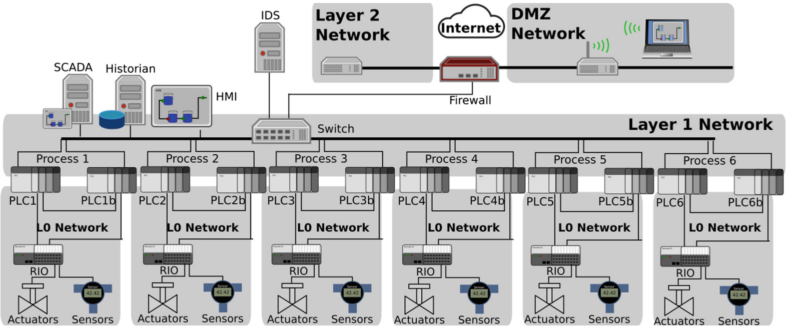


Fig. 1. SWaT network architecture.

2.1 Industrial Control Systems

ICSs are a subset of Cyber-Physical Systems that monitor industrial infrastructures such as Water Treatment systems, nuclear power plants, smart grids, and electric power distribution systems [22]. Securing a safety-critical ICS against malicious attackers is crucial to avoid catastrophic events that may result in natural disasters, economical crises, and loss of human life. The threat model of ICS often assumes a strong attacker, the system under attack provides a large attack surface because an ICS may be vulnerable both to *cyber-attacks* and *physical attacks*. Unfortunately, a combination of cyber and physical attacks can result in a severe damage of the system even without physical access to the system, e.g., [2,32].

Figure 1 shows an example of the architecture of an ICS network. The network is layered to logically separate devices and monitored processes. Compared to a standard corporate network, an ICS network includes a wider range of devices. The ICS has to connect several older legacy hardware and new hardware, and it has to manage software with different capabilities and interfaces. In addition, a traditional ICS network is expected to have a long life time (e.g., twenty years), and many of its components are unlikely to change or be upgraded over the years. That is why the protocols we propose in the remainder of the paper target *both* high-end ICS devices (able to tolerate the computation overhead), and low-end ICS devices (with the introduction of an external module that is able to tap into the ICS network).

Different industrial protocols have been used in ICS. They evolved from serial communication networks (e.g., RS-485, RS-232) to bus systems (e.g., Fieldbus), and then to Ethernet-based communications such as EtherNet/IP (ENIP) [6]. ENIP is a modern, object-oriented application layer industrial protocol, that implements the Common Industrial Protocol (CIP) specifications [24] over the TCP/IP protocol stack. ENIP can be extended to support custom commands and device profiles, and it provides a native compatibility with traditional TCP/IP based IT corporate network. It is important to notice that our authentication scheme *does not* depend on the underlying industrial protocol. We use ENIP as an example protocol as we have a local Water Treatment testbed that uses ENIP. However, the same scheme should easily translate to other modern industrial protocols.

2.2 Security Guarantees

In this work, we discuss solutions to provide authenticity of network traffic for ICS. The solution is not designed to provide confidentiality, and does not prevent denial of service attacks. The attacker is assumed to be able to eavesdrop, insert, drop, and manipulate messages on the network. The attacker does not have access to pre-shared keys, and he is constrained by polynomial computational power on the size of the key, where the computational hardness assumption are simular to the ones proposed in [27].

Moreover, one key observation of this work is that not all packets being transmitted by nodes in ICS need to be authenticated: in the following we will discuss (using ENIP as an example) how to select a subset of critical packets that need to be protected, whereas we argue that other packets are less critical and do not necessarily need to be authenticated since their manipulation does not pose a threat to ICS.

3 Traffic Authentication for ICS: The SPA Protocol

In this section we introduce a first efficient protocol to guarantee authenticity of critical communication in ICS. In addition to providing authenticity, the solution also needs to integrate well into existing systems. In particular, the solution needs to fulfill *real-time* requirements and *legacy-compliance* integration. Note that as stated in the previous section, we will use ENIP as a running example to illustrate our protocol, since it is the protocol used in our evaluation setting. Our idea can be applied to other industrial network protocols (i.e. Modbus TCP [19] and PROFINET [10]), however the implementation details will vary, essentially because a signature must be appended to certain packets, which is easily accomplished in ENIP as we will discuss later.

Our solution must be computationally efficient enough to allow resource-constrained devices (such as PLCs) to sign and verify packets fast enough. In particular, the solution should be able to handle high volume traffic loads, without introducing high queuing, and processing delay. We start by proposing a first solution we called SPA (Selective Packet Authentication). The SPA protocol relies on a simple algorithm: we assume a setting where two devices, that want to communicate, possess a common pre-shared key, and we propose to sign only selected outgoing packets using well-known cryptographic algorithms, such as authenticated signatures schemes [12,14].

This solution is conceptually simple, however it potentially conflicts with the real-time constraints outlined earlier. In Sect. 4, we discuss our refinement of this idea, we present a mathematical analysis able to capture the constraints, in order to guarantee the normal operation of the ICS.

3.1 Legacy-Compliance

ICS often integrate devices that cannot easily be replaced or updated. As a result, a number of legacy industrial protocols are established that are widely supported, but do not feature any security capabilities by design. In general, such protocols allow the reading of distinct memory locations (e.g., in Modbus/TCP) or *tags* (in EtherNet/IP) that represent sensor values or similar. We argue that an upgrade of all such devices in an ICS is costly; therefore an authentication system needs to impact the existing system as little as possible. In particular, the use of TLS tunnels to transmit data would not only incur computational overhead, but could also fail to pass through industrial network appliances or intermediate gateways. Therefore, we propose to embed the authentication data as additional payload in the existing industrial protocols. This will ensure that receivers that are not aware of the authentication scheme can at least process the normal payload without benefiting from the authentication data. Intermediate devices that are unaware of the authentication scheme could also just pass along the authentication data as normal payload. As result, our authentication solution can be integrated in legacy systems, e.g., by introducing external modules to data sources, or through firmware modification of Network modules.

3.2 SPA Protocol Description

The intuition behind the SPA protocol is that (a) only a subset of transmitted messages is security relevant, and (b) selective signing of that subset is more efficient than signing all messages in the stream. For simplicity, we will refer to sign as the process of generating a Message Authentication Code using a pre-shared key for the case of symmetric key cryptography or, a signature using the counterpart's public key for the asymmetric case.

Let p be a critical data packet (we discuss how to identify them later in the context of ENIP). The SPA protocol, shown in Fig. 2, calculates a signature $Sig_k\{p\}$ of the packet p, generates a new packet $p' = (p, Sig_k\{p\})$, and sends it to B. By using the inverse function Ver_K (where $K = k$ in the symmetric case or the corresponding public key otherwise), B verifies the authenticity of the message as $p = Ver_k\{Sig_k\{p\}\}$. Note that the signature scheme Sig guarantees that a computationally constrained adversary cannot forge a signature (with high probability).

We note that to prevent replay attacks, the payload should contain a timestamp or a counter to identify the ENIP session or packet. Therefore, if an active adversary replays the message in a future ENIP session, the application layer will check the nonce and mark it as invalid. On the other hand, since we do not sign non-critical packets in the data stream (nor e.g., TCP synchronisation packets) an attacker could manipulate the content of those messages. Such an attack is easily detected by orthogonal means [7]. By design, ICS systems will trigger alarms in case of problems in network connectivity. In particular, we cannot prevent against denial of service attacks by means of authenticity and such attacks are out of scope.

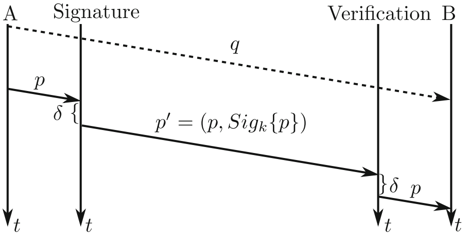


Fig. 2. SPA protocol overview: p is a critical message that is signed by the signature module. q is a non-critical message, which is simply forwarded; δ is the delay introduced by authentication and verification.

Real-Time Requirements and Backlogs. ICS operate under very strict real-time operation conditions, with maximal critical response time, very high availability requirements, and low tolerance for high delay or jitter conditions. As described before, in order to guarantee the authenticity of messages, we authenticate outgoing packets using a *Signature* module, and verify incoming packets using a *Verification* module respectively. We define $g(\varDelta)$ as the number of packets generated by a device in an interval of time $\varDelta$, and $s(\varDelta)$ as the rate at which packets are being signed. As an example, $g(\varDelta)$ can be 1000 packets per second if $\varDelta = 1$ second, while $s(\varDelta)$ generally depends on the processing speed, e.g., 300 packets per second. Similarly we denote $r(\varDelta)$ the number of packets received by a node and $v(\varDelta)$ the number of verifications performed.

In order to be compliant with the real-time requirements of a given system, essentially one needs to authenticate/verify packets at least as fast at the rate they are being produced/received, that is $g \leq s$ and $r \leq v$ (see Appendix A).

3.3 Application to ENIP-CIP

We now discuss the application of the SPA protocol to Common Industrial Protocol (CIP) traffic to ensure legacy compliance, leveraging its extension possibilities. SPA payload is given by $p = (payload, ENIPsequencenr, ENIPsessionid)$ based on a critical ENIP packet from A to B. The ENIP session ID is a randomly generated 32-bit integer and will serve as a counter as discussed in Sect. 3.2. Note that although 32-bit is a relatively small search space, in this context we do not rely on it for security, but merely to prevent replay attacks. An attacker still needs to forge a secure MAC or cryptographic signature to bypass verification, as we will illustrate in Sect. 5.

When replacing the original p with p' of SPA, we increase the length of any packet p that we sign. The packet extension must conform to the ENIP standard. The structure is defined as follows: *Type ID*: $0x00c1$. *Length*: size of the signature specified in Bytes. *Data*: the signature $Sig_k\{p\}$.

The device receiving p' will search for the Type ID $0x00c1$, verify the content of the payload, and then remove the signature. In the case of a mismatch, i.e., $p \neq Ver_k\{Sig_k\{p\}\}$ the device will raise an alarm.

Identification of Critical Data. The integrity of messages exchanged in an ICS system can be protected in different layers of the OSI network stack. We identified critical data from the pool of CIP services observed in the traffic captures. In particular, protection is required for data that can affect the normal operation of the control of a physical process.

The identified critical services are: *Read Data* (Service 0x4C), *Write Data* (Service 0x4D), and *Read Tag Fragmented Data* (Service 0x52). In Sect. 5, we discuss in more detail this choice in the context of other CIP services observed in our Water Treatment testbed. By authenticating critical packets only, we increase security and minimise the computation overhead.

3.4 Ad-Hoc Protocols vs TLS

We have chosen to focus on ad-hoc protocols at the application layer, rather than of using TLS [4], for various reasons. Firstly, we want to reduce the computation overhead, by only authenticating packets that contain critical payloads (such as commands to actuators and values from sensors). In comparison, while TLS would sign every packet in a ENIP connection, SPA protocol would only sign and verify a comparatively small amount of those ignoring packets such as TCP handshake, EtherNet/IP communication control messages and CIP non critical service messages, as shown in Fig. 3. We believe that integrity attacks on TCP handshake and ACK messages can be detected by orthogonal methods. On the other hand, we want to be backwards compatible with devices that do not support message authentication, a feature that we achieve by using the extension capabilities of ENIP. Such a feature would not be achievable by using TLS.

We now discuss performance of SPA vs TLS. Let v the speed in packets per second that a given cryptographic signature can provide for a given average packet size. For simplicity we assume that both TCP packets and ENIP packets are about the same size, although in practice TCP will be slightly bigger. Then if the number of critical packets is c, then the actual throughput on the total of generated packets g will be higher, since we only need to sign $c \cdot g$. Thus, effectively we will increase the tolerance on the generated packets g by a factor of c, allowing a maximum of $\frac{v}{c}$ packets per second. This is illustrated in Fig. 3.

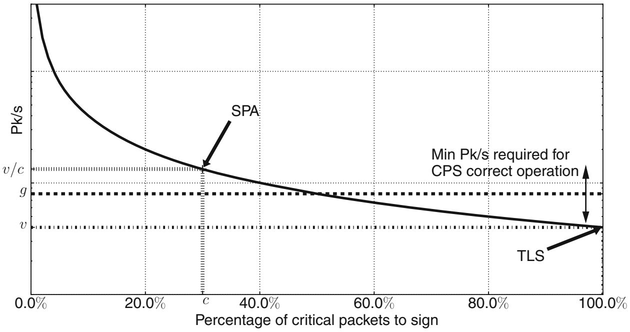


Fig. 3. The y-axis represents the tolerance to a network constraint's in packets/second. The x-axis represents the percentage of critical packets authenticated.

Example. In practice, the amount of sent and received packets is symmetric, as we will discuss in the following sections. In addition, the signature and verification time are similar. Let $g = r$ be 10^3 packets per second, and let $c = 0.4$. Let $s = v$ be 10^5 packets per second. In this case, cryptography-enabled authentication does not create backlogs neither for TLS (signing every packet) nor for SPA since $g + r = 2 \cdot 10^3 < s + v = 2 \cdot 10^5$. This example is based in the real-time communication constraints of SWaT and the signature and verification rates for SHA-256 based HMAC on a virtualized ARM similar to the one used in the SWaT PLCs. A deeper analysis and discussion is presented in Sect. 5.

In sum, the proposed SPA protocol has both advantages and drawbacks in terms of the authentication goals. We briefly summarized them as:

Advantages. The protocol is conceptually simple and easy to implement and to integrate into legacy systems by means of an external component as we will discuss in Sect. 5. Moreover, it gives fine-granularity detection capabilities, since it can be pointed out which packet was altered in transit with almost instantaneous detection times. Additionally, by using the extension capabilities of ENIP, the protocol is backwards compatible with devices that do not implement a verification module.

Disadvantages. The main disadvantage of the SPA protocol is that although it is designed to sign critical packets only, the overheads might still be unacceptable for devices and/or algorithms with low signature/verification rates. This is the main motivation for proposing an extended protocol, as we will discuss in the following section.

4 Extensions: The ASPA Protocol

The SPA protocol presented in the previous section and the constraint analysis, give us guidelines to determine whether the scheme is feasible, depending on the packet generation rate, packet size, and signature algorithm performance on links that have a reliable connectivity. However, there exist two limitations to the practical application of this protocol: first, we would like the protocol to be used even in scenarios where (for legacy or cost reasons) the underlying hardware is not as fast as necessary (i.e. in the sense of the signature rate s vs generation rate g as discussed in the previous section), and second, even faster hardware might be insufficient for stronger signature algorithms, that are computationally more expensive.

In particular, strong signature algorithms would enable the use of asymmetric cryptography, that has several advantages in terms of key management. For example, the use of public/private keys in asymmetric cryptography allows dynamic addition of new devices to the system if centrally signed certificates are used. In addition, the compromise of a single device will only expose a single private key, instead of exposing a secret key shared between many devices.

Our protocol can be extended to deal with more expensive cryptographic algorithms and slower CPUs. The intuition behind this extension, called *ASPA* (Aggregated Selective Packet Authentication) is the following: typically, authenticated signature schemes are relatively inefficient for short amount of data, but they get more efficient for large amounts of data. Thus, on average, it is usually faster to perform an aggregate signature over multiple packets instead of signing them individually, and as a result the aggregate signature increases the signing rate s.

4.1 The ASPA Protocol

Let $P(T)$ be the sequence of outgoing packets in the time interval T. Let n be the expected number of packets in this time $\mathbf{E}[|P(T)|] = n$ such that the typical set is $P(T) = \{p_1, \ldots, p_n\}$. In the ASPA protocol, shown in Fig. 4, the *Signature* module simply forwards packets p_i to B, and in addition accumulates their payload in a queue. After time T has passed, it signs the accumulated queue, and sends the *aggregated signature* together with the sequence number of the first and the last element of the queue. The *Verification* module will forward all received messages to the original destination (without immediately validating a signature), and in addition store the received messages in a queue

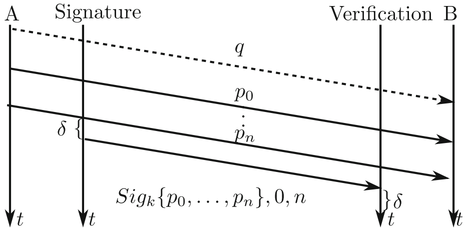


Fig. 4. ASPA overview. p_i are critical messages which are aggregated. q is a non-critical message, thus not authenticated. δ is the delay introduced by authentication.

until the aggregated signature arrives. Then, the verification module can check whether the signature matches the content of the respective packets received. If the Verification module cannot validate the signature, an attack was detected. For example values of T and $P(T)$, we refer to Sect. 5. We note that the ASPA protocol relies on lower layers (i.e., TCP) to ensure that no message gets lost during transmission. That will ensure that the verification and signature modules always have the same view of exchanged messages.

Security Trade-Off. In the first proposed protocol, an attack can be detected as soon as a spoofed packet with an invalid signature is received. In the ASPA protocol, we can only detect an attack after T time has passed, and an invalid signature is received. Depending on the value of T, and the particular ICS system, this could be problematic (or not). This is related with the problem of ICS resilience and reaction time in case of failure, and therefore we see it as orthogonal. For most practical applications (see Sect. 5), the values of T will be small and thus the reaction time will not differ much from the reaction time of the first protocol.

4.2 Performance Advantage

Symmetric Authentication. Let $\delta(\mathit{size}, \mathit{cpu}, \mathit{alg})$ be the time needed to sign a packet of size *size* with algorithm *alg* on CPU *cpu*. The signature scheme is typically based on HMAC and an underlying hash function (such as SHA-256). In that case, there is a constant b such that $\delta(b', \mathit{cpu}, \mathit{alg}) \approx \delta(b, \mathit{cpu}, \mathit{alg})$ for $b' \leq b$. However, for $B > b$: $\delta(B, \mathit{cpu}, \mathit{alg}) = \delta(b, \mathit{cpu}, \mathit{alg}) + \left\lceil \frac{B-b}{b} \right\rceil \cdot c$ where $c < \delta(b, \mathit{cpu}, \mathit{alg})$. Therefore, given an expected packet number of a certain expected size, it is more efficient to sign multiple packets instead of just one, in terms of the rate per interval s. As we will discuss in the next section, the optimization value converges after a certain number of packets, as is to be expected.

Asymmetric Authentication. In the case of signatures based on asymmetric cryptography (such as ECDSA [12]), typically the payload is first hashed and then an operation involving the private key is performed on the digest (such as decryption). Since the cryptographic operation is orders of magnitude slower than the hashing operation (i.e., hundreds of ms vs. μs in some cases), signing multiple packets takes almost as long as signing a single packet.

The above described effects on the signing rate s can be observed in the plotted values for different algorithms and values of n of Fig. 6. Note that a similar discussion about the signing rate s and the packet generation rate g, presented in the previous section, also applies for the ASPA protocol. In practise, a device might have active connections (TCP streams) to m devices at the same time. As result, there will be m queues $Q^S_{\Delta,i}$, possibly signed with different keys. In that case, the constraint becomes $\forall_i \ g_i < s_i$, where $g = \sum_{i=1}^{m} g_i$. In the case of ICS such as SWaT, devices typically communicate only with one or two devices (e.g., $m = 2$). For the sake of simplicity we assume a single queue in the following.

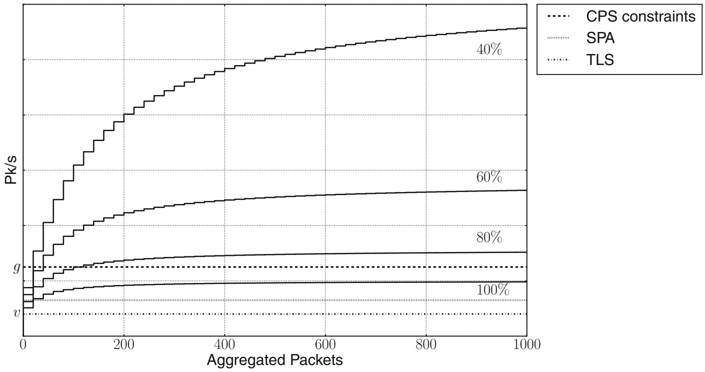


Fig. 5. The x-axis represents chunks of packets, y-axis tolerance in terms of packets per second. The step functions represent different percentages of critical packets, segmented lines various CPS communication constraints. CPU power and packet size are constant.

ASPA vs TLS. In Fig. 5, we illustrate the general case comparison against TLS. Let $v = \frac{1}{\delta}$ the number of packets per second an ideal implementation of TLS could sign. By using ASPA to aggregate n packets we can tolerate approximately: $v' = \frac{n}{\delta+(n-1)\cdot\delta'}$ packets per second, which will be faster than v and will converge to a constant since: $\lim_{n\to\infty} \frac{n}{\delta+(n-1)\cdot\delta'} = \frac{1}{\delta'}$. We will discuss empirically this phenomenon in the evaluation section for various hardware configurations.

Example. Let $p = r$ be 10^3 packets/second as in the previous example (as observed by us in SWaT). Let $s = v$ be 300 packets/second using the SPA protocol of the previous section and ECDSA and a Raspberry Pi3 (for more details see Sect. 5). In that case, cryptography-enabled authentication clearly causes backlogs since $g + r = 2 \cdot 10^3 > s + v = 600$. As discussed above, signing one packet takes almost as much time as signing multiple packets, and then we can implement the ASPA protocol by accumulating chunks of 50 packets (for $T \approx 30\,\text{ms}$), thus augmenting s to about $1,5 \cdot 10^4\,\text{s}$ packets per second. This implies a minimum reaction time to attacks of about 60 ms.

In sum, the ASPA protocol improves the signing and verification rates per packet (s and v), while at the same time providing good reaction time.

Advantages. The ASPA protocol is useful in situations where signing each packet individually is not feasible due to slow hardware or constraints of the ICS network. In particular, it offers a significant advantage when signing multiple packets with ECC based authentication. The ASPA protocol can be used selectively for critical messages (as in the SPA protocol). Alternatively, it could be used to provide delayed authentication for all messages, as the amount of data included in the aggregated signature has a negligible impact on the overall time of creating the signature.

Disadvantages. The coarse granularity of the authentication potentially can delay reaction times to attacks, depending on the size T of the signing window. However in our evaluation we have found that the ideal window for authentication is a few dozens of packets for most signing algorithms/hardware, which allows for a fast reaction time in practice. In our experience, actual attacks require a relatively long interaction with the system in order to influence its physical state. However, we stress that the subject of reaction to attacks is out of the scope of this paper and is left for future work.

5 Validation

In this section, we use a real SWaT [11] to obtain realistic examples of real-time constraints. We investigate the amount of critical traffic shared between devices in the network, in order to approximate the expected number of packets per time interval, and the expected size of those packets. Furthermore, we benchmark symmetric and asymmetric signature algorithms for a variety of hardware platforms, and discuss the feasibility of their implementation with respect to the constraints derived in the previous sections. We use standard algorithms such as SHA (instead of light-weight algorithms specifically designed for embedded systems) to allow better comparison against TLS. When using light-weight algorithms, the presented numbers are expected to improve.

5.1 SWaT

SWaT [11] (see Fig. 1) consists of an ICS with a process made up of six stages. In a nutshell, the process begins by collecting the incoming water in a tank, then it performs a chemical treatment stage, it filters the treated water through an Ultrafiltration (UF) system. Afterwards, the water is de-chlorinated using a combination of chemicals and Ultraviolet lamps, and then fed to a Reverse Osmosis (RO) stage. A backwash process cleans the membranes in UF using the water produced by RO. The cyber portion of SWaT consists of a layered communications network, PLCs, HMIs, SCADA, and a Historian.

The validation focuses on benchmarking integrity, and authenticity controls in the plant network, that connects the PLCs, the HMI and the SCADA system. The network is using the EtherNet/IP industrial protocols on top of an (Ethernet based) TCP/IP network. We performed several network capture by setting up a mirroring port on the plant network industrial switch. From those captures we identified ENIP-CIP communications among 21 hosts, through implicit and explicit messages. Implicit messages (UDP/2222) are used in our plant for keep-alive signals, while explicit messages (TCP/44818) are used for configuring, monitoring and controlling the plant stages. The plant performs its communication to a rate of 16.000 ENIP-CIP messages per second on average over all stages. About 14,3% of ENIP-CIP connections belong to UDP/2222, and the rest 85,7% to TCP/44818. We can split TCP connections between TCP session traffic (42,7%), and CIP explicit messages (42,9%).

We focused on CIP explicit messages, and we tried to extract a subset of CIP services that deals with critical data. Manipulation of those services could affect the state of the controlled physical process. We selected the following services as critical: **Read Data** (Service 0x4C); **Write Data** (Service 0x4D); **Read Tag Fragmented Data** (Service 0x52).

Read Data and Read Tag Fragmented Data. CIP services are classified as critical data since an attacker might rise a fake alarm in the SCADA system or he might hide a safety-related event modifying its data on the fly. The *Write Data* CIP service is classified as critical because an attacker might directly modify the behavior of actuators pushing data into PLCs. By selecting CIP services with critical data, our proposal only signs around 42% of SWaT's traffic (including TCP and UDP), avoiding processing and bandwidth overheads for non-critical data CIP services such as the *Get all attributes* service.

We performed an in-depth analysis of the capture file in order to estimate the frequency, and the size of the packets received per second, by an arbitrary testbed's device. Table 1 shows a summary of the results targeting PLC2. We decided to use PLC2 as an upper bound because we estimated that it is the "busiest" device in our testbed. As you can see from the table, PLC2 sends 1127 packets per second on average, and it receives 1168 packets per second on average.

Table 1. Frequency and size of critical packets shared by host PLC2 to others.

		Sent	Received
ENIP message	Request	561 Pk/s $\mu = 63$B, $\sigma = 3.36$	607 Pk/s $\mu = 69$B, $\sigma = 5.32$
	Response	566 Pk/s $\mu = 75$B, $\sigma = 58.16$	561 Pk/s $\mu = 86$B, $\sigma = 9.42$
Total Pk/s		1127	1168

5.2 Hardware Benchmark

In order to evaluate the efficiency of the underlying primitives, and therefore the packet signing rate $s(t, cpu, alg)$, we used different types of hardware platforms.

Controllino is an open source Hardware PLC, based on Arduino Mega 2560 board, with an ATmega2560 CPU (16 MHz), 256 KB of flash memory, an Ethernet connector, and two serial interfaces. For our experiments, we used the spaniakos cryptographic library [8]. **ARM** We used the QEMU emulator with the following settings: ARM926EJ-S rev 5 processors family at 530MHz, 256MB of RAM, Debian 3.2.51 32 bit Operating System, and the libgcrypt-1.6.5 cryptographic library. **Raspberry Pi** is a single-board computer of credit card-size. It was initially developed for educational purposes, but because of its low energy consumption and low cost, it has become into a popular multipurpose hardware. We choose it as a possible hardware to implement the authentication and integrity mechanism. The characteristics of the Raspberry Pi model 2 (RPi2) are: Quad-core ARM Cortex-A7 processor at 900 MHz, 1 GB of RAM, 4 USB ports, 40 GPIO pins and an Ethernet port. The cryptographic library used was again libgcrypt-1.6.5 [17]. **PC** We used a workstation with the following specifications: Intel Core i5-5300U processor at 2.30GHz, 3 GB of RAM, Xubuntu 15.10 64 bit OS, and libgcrypt-1.6.5 as cryptographic library.

Table 2. Benchmark of HMAC-SHA256 and ECDSA signature process for different packet sizes over 5 types of hardware (rounded values). Times are in μs.

	HMAC - Time				
Size	Controllino	ARM	RPi2	RPi3	PC
64 B	$2.2 \cdot 10^4$	76	53	15	2
128 B	$3.3 \cdot 10^4$	78	58	16	2
256 B	$5.5 \cdot 10^4$	84	69	18	3
512 B	$1 \cdot 10^5$	117	89	32	4
1 KB	$1.8 \cdot 10^5$	171	130	35	6
2 KB	$3.6 \cdot 10^5$	252	211	58	10
4 KB	$7 \cdot 10^5$	474	374	104	18
	ECDSA - Time				
4 KB	N/A	$1.5 \cdot 10^5$	$1 \cdot 10^5$	$3.2 \cdot 10^4$	$3.1 \cdot 10^3$

Table 3. Benchmark of performance of HMAC-SHA256 and ECDSA authentication for different packet sizes and hardware. Average size per packet 73 Bytes from Table 1.

	HMAC - Average Pkt/s				
Size	Contr.	ARM	RPi2	RPi3	PC
64 B	40	$1.1 \cdot 10^4$	$1.6 \cdot 10^4$	$5.8 \cdot 10^4$	$4.4 \cdot 10^5$
128 B	53	$2.2 \cdot 10^4$	$3 \cdot 10^4$	$1.1 \cdot 10^5$	$8.8 \cdot 10^5$
256 B	64	$4.2 \cdot 10^4$	$5 \cdot 10^4$	$1.9 \cdot 10^5$	$1.2 \cdot 10^6$
512 B	70	$6 \cdot 10^4$	$7.9 \cdot 10^4$	$2.2 \cdot 10^5$	$1.7 \cdot 10^6$
1 KB	78	$8.2 \cdot 10^4$	$1.1 \cdot 10^5$	$4 \cdot 10^5$	$2.3 \cdot 10^6$
2 KB	78	$1.1 \cdot 10^5$	$1.3 \cdot 10^5$	$4.8 \cdot 10^5$	$2.8 \cdot 10^6$
4 KB	80	$1.2 \cdot 10^5$	$1.5 \cdot 10^5$	$5.4 \cdot 10^5$	$3 \cdot 10^6$
	ECDSA - Average Pkt/s				
4 KB	N/A	$3.7 \cdot 10^2$	$5.6 \cdot 10^2$	$1.7 \cdot 10^3$	$1.8 \cdot 10^4$

We also benchmarked the elliptic curve based ECDSA signing standard. As discussed previously, the cryptographic operation dominates the execution time of the hashing component of the algorithm. This makes the times for signing payloads of up to 4KB almost identical to small payloads of 32B. The results are reported in the last row of Table 2. The resulting s for ASPA for aggregated signatures of about 58 packets (4KB) is reported in Table 3.

5.3 Discussion

We now provide a summary of our empirical results using different hardware platforms and cryptographic algorithms combination over our SWaT. In Table 4, we compare the SPA protocol (first two columns) with the ASPA protocol for $n = 58$ packets. As we can see, with a processor such as the one in the Controllino (16MHz), it is infeasible to cope with the real-time requirements of the SWaT networks for all algorithms considered. As shown in Fig. 6, and taking as reference our plant constraints, a symmetric signature is supported by most hardware. On the other hand, ECC signatures are possible in Raspberry Pi2, Pi3 and PCs thanks to our extension.

From a communications cost perspective, a signature in HMAC would add an overhead of 28% in size for an average ENIP packet, while ECDSA would add about 57%. Since we are only signing critical data, which corresponds to 42% of total traffic, our overhead in bandwidth will be 12% and 24% for HMAC and ECDSA respectively.

In sum, we have shown that in some cases, there is an advantage on using aggregation and selectiveness over traditional authenticated tunnels in terms of efficiency. Although we have shown an example for a concrete testbed and some hardware configurations, the possible configurations in practice could vary

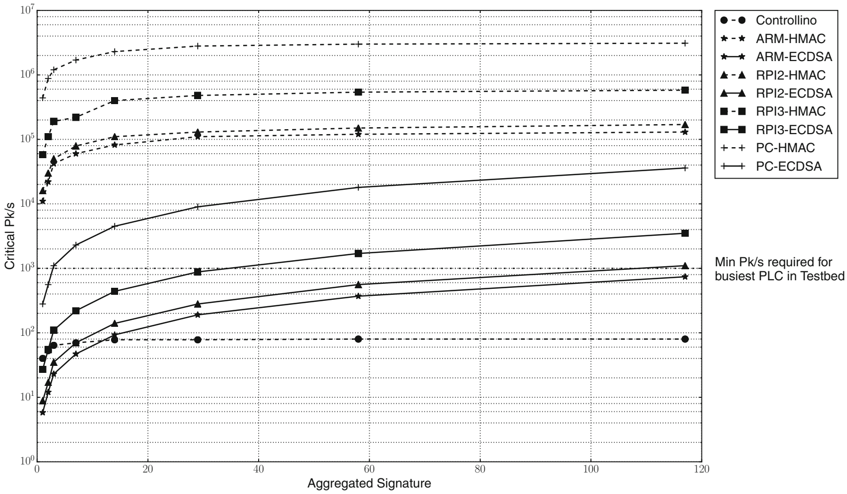


Fig. 6. ASPA performance on various hardware. PLC2, as the busiest, requires 1000 Pk/s for implementing a multi-link authentication process, y-axis is in log scale.

Table 4. Feasibility of algorithms vs. hardware in the SWaT. Slow hardware does not tolerate the minimum required signing rate even for ASPA. However, thanks to ASPA we can use RPi2, RPi3 and PCs to implement ECC based authentication.

	Sequential				Parallel			
	HMAC	ECC	HMAC-ASPA	ECC-ASPA	HMAC	ECC	HMAC-ASPA	ECC-ASPA
Controllino	×	×	×	×	×	×	×	×
ARM	✓	×	✓	×	✓	×	✓	×
RPi2	✓	×	✓	✓	✓	×	✓	✓
RPi3	✓	×	✓	✓	✓	×	✓	✓
PC	✓	×	✓	✓	✓	×	✓	✓

highly. For instance, in a full scale water plant the number of sensors and actuators could be one order of magnitude higher ($g = 10^4$), whereas the available hardware could range between the Controllino (16 MhZ) and the ARM processor (500 MhZ). Moreover, the underlying cryptographic algorithm could also vary, thus affecting the concrete performance.

Implementation in SWaT. We have successfully tested the proposed approach in a link between two PLCs of SWaT using Raspberry PIs and summarize our implementation efforts in Appendix B.

6 Related Work

In [23], the authors analyze different attacks on PLCs and several protection techniques. The authors state that the bulk of the security issues in ICS is the lack of security in the network communications. A public key based authentication protocol is proposed. However, the authors mention that their techniques could degrade the performance of the system, and they do not provide any further analysis.

A detailed analysis on cryptography-enabled authentication is reported in [31]. The authors analyze the security of electric power grids and identify authenticity and integrity as the most important security properties after availability. Some constraints on the message delays are considered during the analysis based on the expected message frequency as discussed in industrial standards.

In [1], the authors discuss integrity, authenticity, and authorization policies of ENIP and CIP. Two security profiles (providing integrity and confidentiality) for the ENIP are described and two for CIP (providing authorization and integrity) are proposed as a future extension. Authors define security profile as a set of well-defined capabilities to facilitate device interoperability, and end-user selection of devices with the appropriate security capability. In particular, the

CIP authorization profile will provide secure communications between CIP endpoints, to ensure device and user authenticity. Authors describe ENIP over TLS (for TCP-based communication) and DTLS (for UDP).

Authors propose to implement confidentiality and integrity checks between paired devices called SCADA Cryptographic Modules (SCM) [33]. They are featured with two ports (plaintext and ciphertext). While it receives plaintext messages from a device through the plaintext port, it encrypts the message using AES and send it to its remote SCM through the ciphertext port. The message includes a MAC code (HMAC-SHA-1 or CBC-MAC) for integrity verification. In [28], authors propose a bump-in-the-wire solution to add security features such as Data privacy and authenticity to ICS communication. The solution suggests adding two devices (each per peer) into the communication channel (YASIR transmitter and YASIR receiver). YASIR Transmitter encrypts the message using AES-CTR, and attach an HMAC-SHA-1 signature. A YASIR Receiver decrypts and checks the message's integrity. In the case of integrity violation, YASIR Receiver adds an error byte to the frame; it guarantees that destination device discards the final message.

In IP-based communications, authors at [18] introduce a DNP protocol modification. They propose to assure confidentiality using DES cryptographic primitive and Integrity-Authenticity with HMAC-MD5-96 or HMAC- SHA-1-96. Their work suggests a change in the DNP's data structure. It could thwart their adoption by industry. Recently, researchers create a new MODBUS security development module (NMSDM) [25] that replaces MODBUS message. They propose to use cryptographic algorithms such as RSA, AES and SHA-2 to guarantee security properties as confidentiality, integrity, authentication and non-repudiation. Their proposal is to add a *cryptographic buffer* on top of PDU, as part of the TCP payload. Like ours, it allows adding security properties to the communication without modifying the protocol by itself. Both works [18,25] search to protect every message in an ICS, and computing overhead of cryptographic implementation are not taken into account.

In automotive applications, authors at [21,30] present authentication techniques called CANAuth and LeiA for CAN bus protocol. Similarly to ICS, CAN bus systems relies on hard real-time constraints. The authors shows that state of the art authentication mechanisms for broadcast networks cannot be used due to time (and storage) constraints. LeiA and CANAuth, however, are specifically designed for CAN bus systems. A low encryption overhead is also of paramount importance in Wireless Sensor Network (WSN), e.g., [13,20]. This is mainly due to bandwidth, latency and energy cost introduced by encryption. Solutions such as [13] provide authentication, but are not analyzed in the context of real-time constraints.

To the best of our knowledge, we are the first to propose the aggregated selective packet authentication protocol in the context of ICS. The idea is inspired by a delayed authentication scheme for GPS proposed in [16].

7 Conclusions

In this work, we discussed the introduction of efficient legacy-compliant authentication in ICS networks. By design, our protocols are backward compatible with devices not implementing them, as they transmit authentication data as payload in legacy industrial protocols.

We have shown that with the advent of inexpensive and fast hardware (such as the Raspberry Pi), it is feasible to enhance legacy plants with constraints similar to the ones of SWaT with authentic channels for strong signature algorithms with simple protocols. However, introducing state of the art asymmetric algorithms (which are advantageous from the key management point of view), or implementing our solution in slower hardware requires careful performance trade-offs. In future work, we plan to compare the real-time constraints of SWaT with constraints found in other ICS test-beds (i.e. smartgrid).

A Appendix: Real-Time Requirements and Backlogs

The delay produced by the signature module can be represented as a queue of packets pending to be signed after a time interval Δ: $Q^S_\Delta = g(\Delta) - s(\Delta)$. We assume an empty queue at the beginning of the time interval Δ. To avoid an avalanche effect on the packet queue Q^S_Δ the queue must remain below a certain constant threshold C: $\forall\ \Delta.\ Q^S_\Delta < C$. In particular, since Δ is arbitrary $C \approx 0$ (i.e., $\forall\ \Delta.\ Q^S_\Delta < 0$) and thus: $\forall\ \Delta.\ g(\Delta) < s(\Delta)$. Similarly, if $r(\Delta)$ is the number of expected packets in a time interval Δ, then the number v of verified packets in this interval must be $v(\Delta, cpu, alg) > r(\Delta)$ to avoid backlogs of the verification queue Q^V_Δ.

$s(\Delta)$ depends not only on the time but also on the algorithm used, the CPU capacity, and the size of the packets. In a fraction of time Δ, the device will produce not only packets at different rates, but will also nondeterministically produce packets of different sizes. To simplify our analysis, and without loss of generality, we thus set $\Delta = 1\,\text{s}$, and $s(cpu, alg)$ the rate at which packets of a certain constant expected size can be signed using a certain *cpu* and a given signature algorithm *alg*. We use s, v, g and r to abbreviate the rates per second of the signing, verification, outgoing (or *generated*) and incoming packets respectively.

Parallel versus Sequential Signature and Verification. We are assuming an architecture where network communications are handled by dedicated hardware. This avoids sharing the main device CPU (such as the main PLC computing unit), which is busy computing other control and communication related tasks. In practice this is usually the case, since communications are often handled by a dedicated network module. For a parallel implementation of the protocol, where the signature and verification components have at least a single core each, then we have effectively two queues that we can consider separate, Q^S_Δ and Q^V_Δ. Otherwise, for a sequential implementation of the protocol, an incoming packet that needs to be verified has to potentially wait until an outgoing packet is signed; and vice versa, an outgoing packet has to wait until a verification is finished. We have thus the constraints: $\forall\, \Delta.\ Q^S_\Delta < C \wedge Q^V_\Delta < C'$ which we can simplify by considering a single queue $Q_\Delta = Q^S_\Delta \cup Q^V_\Delta = (g(\Delta) - s(\Delta)) + (r(\Delta) - v(\Delta)) < C + C'$. As above, $C + C' \approx 0$ and thus: $g(\Delta) + r(\Delta) < s(\Delta) + v(\Delta)$. In other words, the rate of signature and verification in a time interval has to be faster than the amount of packets sent and received in that same time interval.

B Appendix: Authenticated Link Using Raspberry Pi

We used two (paired) Raspberry Pi3, (as shown in Fig. 7) each with two Ethernet adapters. We choose a link between two PLCs in our SWaT to perform the test. The devices were configured as Ethernet bridges and placed as physical Man-in-the-Middle over the link. This choice was motivated because the source code of the Network Adapter in the SWaT was unavailable to us (as such proprietary code is confidential), and we could not directly implement our solution without additional hardware. However, a vendor could easily deploy our solution.

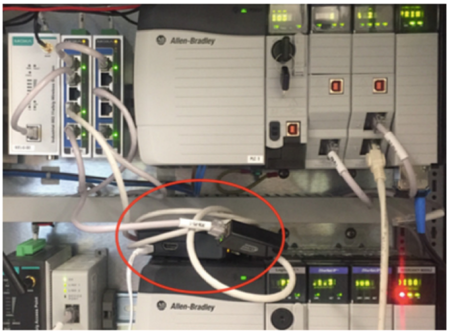


Fig. 7. Raspberry Pi3 connected between PLCs.

Once connected, the devices passively listen to packets from and to the PLCs. When a critical-data packet is identified, it is captured and the ENIP payload is signed with HMAC-SHA256 algorithm using a pre-shared key. The concatenation of the captured packet and its signature is injected back into the communication channel.

Similarly, the remote Raspberry Pi3 is placed as a verification module in front of the destination PLC. Once the verification module identifies a packet coming from its counterpart, the packet is analysed looking for an attached signature, the signature is extracted and verified against the ENIP payload using HMAC-SHA256 algorithm with the pre-shared key. The packet is converted back to its original version and it is delivered to its destination.

We configured four additional variables in the PLCs to store information about the authentication processes: *Signed messages*: Number of signed messages by its cryptographic module; *Checked messages*: Number of signed message correctly verified by its cryptographic module; *Wrong-signature messages*: Number of signed messages which signature does not correspond to its payload detected by its cryptographic module; *No-signed messages*: Messages with critical data from a peered host with no-attached signature detected by its cryptographic module.

References

1. Batke, B., Wiberg, J., Dubè, D.: CIP security phase 1 secure transport for Ethernet/IP. In: ODVA Industry Conference (2015)
2. Cárdenas, A.A., Amin, S.M., Sinopoli, B., Giani, A., Perrig, A., Sastry, S.S.: Challenges for securing cyber physical systems. In: Workshop on Future Directions in Cyber-physical Systems Security, DHS, July 2009
3. Cárdenas, A.A., Baras, J.S.: Evaluation of classifiers: practical considerations for security applications. In: AAAI Workshop on Evaluation Methods for Machine Learning (2006)
4. Dierks, T.: The transport layer security (TLS) protocol version 1.2 (2008). https://www.ietf.org/rfc/rfc5246.txt
5. Fletcher, K.K., Liu, X.: Security requirements analysis, specification, prioritization and policy development in cyber-physical systems. In: Secure Software Integration Reliability Improvement Companion (SSIRI-C), pp. 106–113 (2011)
6. Galloway, B., Hancke, G.: Introduction to industrial control networks. Commun. Surv. Tutor. **15**(2), 860–880 (2013). IEEE
7. Gomes, N., Mattos, L.: Attacks detection based on IP and TCP protocols violation. Int. J. Forensic Comput. Sci. **1**, 49–56 (2006)
8. Hash libraries for arduino. http://spaniakos.github.io/Cryptosuite/
9. Igure, V.M., Laughter, S.A., Williams, R.D.: Security issues in scada networks. Comput. Secur. **25**(7), 498–506 (2006)
10. P. Inc.: Profinet and it. Technical report, PROFIBUS Nutzerorganisation e.V. (2008)
11. iTrust: Center for Research in Cyber Security. Secure water treatment test-bed. http://itrust.sutd.edu.sg/research/testbeds/secure-water-treatment-swat/
12. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ECDSA). Int. J. Inf. Secur. **1**(1), 36–63 (2001)
13. Karlof, C., Sastry, N., Wagner, D.: TinySec: a link layer security architecture for wireless sensor networks. In: Proceedings of the International Conference on Embedded Networked Sensor Systems, SenSys 04, pp. 162–175. ACM (2004)
14. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: keyed-hashing for message authentication (1997). https://www.ietf.org/rfc/rfc2104.txt
15. Krotofil, M., Cárdenas, A.A., Manning, B., Larsen, J.: CPS: driving cyber-physical systems to unsafe operating conditions by timing DoS attacks on sensor signals. In: Proceedings of the Computer Security Applications Conference (ACSAC), pp. 146–155. ACM (2014)
16. Kuhn, M.G.: An asymmetric security mechanism for navigation signals. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 239–252. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30114-1_17
17. Gnu cryptographic library. https://www.gnu.org/software/libgcrypt/
18. Majdalawieh, M., Parisi-Presicce, F., Wijesekera, D.: DNPSec: distributed network protocol version 3 (DNP3) security framework. In: Elleithy, K., Sobh, T., Mahmood, A., Iskander, M., Karim, M. (eds.) Advances in Computer, Information, and Systems Sciences, and Engineering, vol. 3, pp. 227–234. Springer, Dordrecht (2007). doi:10.1007/1-4020-5261-8_36
19. Modbus-IDA. Modbus messaging on tcp/ip implementation guide v1.0b. Technical report, Modbus Organization (2006)

20. Nie, P., Vähä-Herttua, J., Aura, T., Gurtov, A.: Performance analysis of HIP diet exchange for wsn security establishment. In: Proceedings of the ACM Symposium on QoS and Security for Wireless and Mobile Networks, Q2SWinet 11, pp. 51–56. ACM (2011)
21. Radu, A.I., Garcia, F.D.: LeiA: a lightweight authentication protocol for CAN. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9879, pp. 283–300. Springer, Cham (2016). doi:10.1007/978-3-319-45741-3_15
22. Rajkumar, R., Lee, I., Sha, L., Stankovic, J.: Cyber-physical systems: the next computing revolution. In: 2010 47th ACM/IEEE on Design Automation Conference (DAC), pp. 731–736, June 2010
23. Sandaruwan, G., Ranaweera, P., Oleshchuk, V.A.: PLC security and critical infrastructure protection. In: Industrial and Information Systems (ICIIS), pp. 81–85. IEEE (2013)
24. Schiffer, V., Vangompel, D., Voss, R.: The common industrial protocol (CIP) and the family of CIP networks. ODVA, Ann Arbor (2006)
25. Shahzad, A., Lee, M., Lee, Y.-K.K., Kim, S., Xiong, N., Choi, J.-Y.Y., Cho, Y.: Real time MODBUS transmissions and cryptography security designs and enhancements of protocol sensitive information. Symmetry **7**(3), 1176–1210 (2015)
26. Slay, J., Miller, M.: Lessons learned from the maroochy water breach. In: Goetz, E., Shenoi, S. (eds.) ICCIP 2007. IIFIP, vol. 253, pp. 73–82. Springer, Boston, MA (2008). doi:10.1007/978-0-387-75462-8_6
27. Smart, N., Babbage, S., Catalano, D., Cid, C., Weger, B. d., Dunkelman, O., Ward, M.: Ecrypt ii yearly report on algorithms and keysizes (2011–2012). In: European Network of Excellence in Cryptology (ECRYPT II) (2012)
28. Tsang, P.P., Smith, S.W.: YASIR: a low-latency, high-integrity security retrofit for legacy SCADA systems. In: Jajodia, S., Samarati, P., Cimato, S. (eds.) SEC 2008. ITIFIP, vol. 278, pp. 445–459. Springer, Boston, MA (2008). doi:10.1007/978-0-387-09699-5_29
29. Urbina, D., Giraldo, J., Tippenhauer, N.O., Cárdenas, A.: Attacking fieldbus communications in ICS: applications to the SWaT testbed. In: Proceedings of Singapore Cyber Security Conference (SG-CRC), January 2016
30. Van Herrewege, A., Singelee, D., Verbauwhede, I.: CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus. In: ECRYPT Workshop on Lightweight Cryptography, vol. 2011 (2011)
31. Wang, W., Lu, Z.: Cyber security in the smart grid: survey and challenges. Comput. Netw. **57**(5), 1344–1371 (2013)
32. Weinberger, S.: Computer security: is this the start of cyberwarfare? Nature **174**, 142–145 (2011)
33. Wright, A.K., Kinast, J.A., McCarty, J.: Low-latency cryptographic protection for SCADA communications. Acns **3089**, 263–277 (2004)

Multi-client Oblivious RAM Secure Against Malicious Servers

Erik-Oliver Blass[1], Travis Mayberry[2](✉), and Guevara Noubir[3]

[1] Airbus Group Innovations, Munich, Germany
erik-oliver.blass@airbus.com
[2] US Naval Academy, Annapolis, MD, USA
mayberry@usna.edu
[3] Northeastern University, Boston, MA, USA
noubir@ccs.neu.edu

Abstract. This paper tackles the open problem whether an Oblivious RAM can be shared among multiple clients in the presence of a fully malicious server. Current ORAM constructions rely on clients knowing the ORAM state to not reveal information about their access pattern. With multiple clients, a straightforward approach requires clients exchanging updated state to maintain security. However, clients on the internet usually cannot directly communicate with each other due to NAT and firewall settings. Storing state on the server is the only option, but a malicious server can arbitrarily tamper with that information.

We first extend the classical square-root ORAM by Goldreich and the hierarchical one by Goldreich and Ostrovsky to add multi-client security. We accomplish this by separating the *critical* portions of the access, which depend on the state of the ORAM, from the non-critical parts (cache access) that can be executed securely in any state. Our second contribution is a secure multi-client variant of Path ORAM. To enable secure meta-data update during evictions in Path ORAM, we employ our first result, small multi-client secure classical ORAMs, as a building block. Depending on the block size, the communication complexity of our multi-client secure construction reaches a low $O(\log N)$ communication complexity per client, similar to state-of-the-art single-client ORAMs.

1 Introduction

The main metric of an ORAM's performance, communication overhead, has improved by orders of magnitude over the last few years. However, at least one significant hurdle to actual adoption remains: security of modern ORAMs relies on there being only a single client at all times. While there already exist multi-client ORAMs secure in the face of a *semi-honest* server [18], existence of an ORAM secure against a fully *malicious* server is still an open question. The main challenge here stems from the fact that today's ORAMs modify some of the data on the server after every access. If a malicious server "rewinds" the data and presents an old version to a client, further interactions may reveal details about the access pattern. In single client scenarios, this is typically solved by

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 686–707, 2017.
DOI: 10.1007/978-3-319-61204-1_34

storing a small token on the client, such as the root of a hash tree [22]. This token authenticates and verifies freshness of all data retrieved from the server, ensuring that no such rewind attack is possible.

In this paper, we address the fundamental problem how multiple clients can share data stored in a single ORAM. With multiple clients, an authentication token is not sufficient. Data may not pass one client's authentication, simply because it has been modified by one of the other clients. If clients could communicate with each other using a secure out-of-band channel, then it becomes trivial to continually exchange and update each other with the most recent token. However, existence of secure out-of-band communication is often not a reasonable assumption for modern devices. As we will see, it is the absence of out-of-band-communication which makes multi-client ORAM technically challenging.

Current solutions for multi-client ORAM work only in the presence of semi-honest (honest-but-curious) adversaries, which cannot perform rewind attacks on the clients. Often, this is not a very satisfying model, since rewind attacks are very easy to execute for real-world adversaries and would be difficult to detect. Consequently, we only address fully malicious servers. Goodrich et al. [12], in their paper examining multi-client ORAM, recently proposed as an open question whether one could be secure for multiple clients against a malicious server.

Technical Highlights. We introduce the first construction for a multi-client ORAM and prove access pattern indistinguishability, even if the server is fully malicious. Our contribution is twofold, specifically:

- We start by focusing on two ORAM constructions that follow a "classical" approach, the *square-root* ORAM by Goldreich [8] and the *hierarchical* ORAM by Goldreich and Ostrovsky [9]. We adapt these ORAMs for multi-client security. Our approach is to separate client accesses into two parts: non-critical portions, which can be performed securely in the presence of a malicious server and critical portions, which cannot but contains efficient integrity checks which will reveal any malicious behavior and allow the client to terminate the protocol.
- The "classical" ORAM constructions have been largely overshadowed by more recent tree-based ORAMs [23,25]. Consequently, we go on to demonstrate how a multi-client secure Path ORAM [25] can be constructed. We solve the key challenge of realizing a multi-client secure version of the *read* protocol by storing Path ORAM's metadata using small "classical" ORAMs as building blocks. For block sizes in $\Omega(\log^4 N)$, this results in a multi-client ORAM which has overall communication complexity of $O(\phi \cdot \log N)$.

Table 1 summarizes asymptotic behavior for our new multi-client ORAMs and compares them to their corresponding single-client ORAMs.

2 Motivation: Multi-client ORAM

Instead of a single ORAM accessed by a single client, we envision multiple clients securely exchanging or sharing data stored in a single ORAM. For example, imag-

Table 1. Communication and storage worst-case complexity for existing single-client ORAMs and our new multi-client versions. ϕ is the number of different clients supported by the ORAM. $\hat{O}$ denotes amortized complexity.

	Communication		Storage	
	Single client	Multi client	Single client	Multi client
Square-root [8]	$\hat{O}(\sqrt{N})$	$\hat{O}(\phi \cdot \sqrt{N})$	$O(N)$	$O(\phi \cdot N)$
(Deamortized) Hierarchical [9, 15]	$O(\log^3 N)$	$O(\phi \cdot \log^3 N)$	$O(N \cdot \log N)$	$O(\phi \cdot N \cdot \log N)$
Tree-based [25]	$O(\log N)$	$O(\phi \cdot \log N)$	$O(N \cdot \log N)$	$O(\phi \cdot N \cdot \log N)$

ine multiple employees of a company that read from and write into the same database stored at an untrusted server. Similar to standard ORAM security, sharing data and jointly working on the database should not leak the employees' access patterns to the server. Alternatively, we can also envision a single person with multiple different devices (laptop, tablet, smartphone) accessing the same data hosted at an untrusted server (e.g., Dropbox). Again, working on the same data should not reveal access patterns. Throughout this paper, we consider the terms "multi-client" and "multi-user" to be equivalent. As suggested by Goodrich et al. [12], we assume that clients all trust each other and leave expansion of our results for more fine-grained access control as future work. In the multi-device scenarios above, it is reasonable that clients trust each other since they belong to a single user.

To provide security, ORAM protocols are *stateful*. Hiding client accesses to a certain data block is typically achieved by performing shuffling or reordering of blocks, such that two accesses are not recognizable as being the same. An obvious attack that a malicious server can do is to undo or "rewind" that shuffling after the first access and present the same, original view (state) of the data to the client when they make the second access. If the client was to blindly execute their access, and it was the same block of data as the first access, it would result in the same pattern of interactions with the server that the first access did. The server would immediately have broken the security of the ORAM scheme. This is a straightforward attack that can be easily defeated in case of a single client: as an internal state, the client stores and updates a token for authentication and freshness, see Ren et al. [22].

However, with two or more clients sharing data in an ORAM, this attack becomes a new challenge. After watching one client retrieve some data, the adversary rewinds the ORAM's state and present the original view to the second client. If the second client accesses the same data (or not) that the first client did, the server will recognize it, therefore violating security. Without having some secure side-channel to exchange authentication tokens after every access, it is difficult for clients to detect such an attack.

2.1 Technical Challenges

A multi-client ORAM has to overcome a new technical challenge. Roughly speaking, the server is fully malicious and can present different ORAM states to different clients, i.e., different devices of the same user. As different clients do not have a direct communication channel to synchronize *their* state, it is difficult for them to synchronize on an ORAM state. We expand on this challenge below.

Adversary Model: This paper tackles the scenario of ϕ trusted clients sharing storage on a fully malicious server (the adversary). Other works such as Maffei et al. [18] have addressed the problem against a semi-honest server, but in many scenarios that may not be sufficient. Real-world attacks that clients need to defend against include, e.g., insider attacks from a cloud provider hosting the server and outside hackers compromising the server. Such attacks would allow for malicious adversarial behavior. In general, there is no strong line between the two adversarial models that suggests that one is more reasonable to defend against. To cope with all possible adversaries, it is therefore important to protect against malicious adversaries, too.

No Out-of-Band Communication: We assume that beyond a single cryptographic key (possibly derived from a password) the clients do not share any long-term secrets and cannot communicate with each other except through the malicious server. This matches with existing cloud settings, since most consumer devices are behind NAT and cannot be directly contacted from the Internet. Major real-world cryptographic applications, for instance WhatsApp [26] and Semaphor [24], have all messages between clients relayed through the server for this reason.

We emphasize that in the malicious setting the server always has the option to simply stop responding or send purposefully wrong data as a denial-of-service attack. This cannot be avoided, but is also not a significant problem since it will be easily detected by clients. In contrast, the attacks we focus on in this paper are those where the server tries to compromise security without being detected.

2.2 Other Applications

A major application of this work is in supporting private cloud storage that is accessible from multiple devices. In reality, this is one of the most compelling use cases for cloud storage a la Dropbox, Google Drive, iCloud, etc. and is a good target for a privacy-preserving solution. Beyond simple storage, Oblivious RAM is used as a subroutine in other constructions, such as dynamic proofs of retrievability [3]. Any construction which uses ORAM and wishes to support multiple clients must rely on an ORAM that is secure for multiple clients.

Finally, an interesting application comes from the release of Intel's new SGX-enabled processors. SGX enables a trusted enclave to run protected code which cannot be examined from the outside, even by the operating system or hypervisor running on the machine. The major remaining channel for leakage in this system is in the pattern of accesses that the enclave code makes to the untrusted RAM lying outside the processor. It has already been noted that Oblivious RAM may be a valuable tool to eliminate that leakage [5]. Furthermore, it is expected that systems may have multiple enclaves running at the same time which wish to share information through the untrusted RAM. This scenario corresponds exactly with our multi-client ORAM setting, so the solution here could be used to securely offer that functionality.

2.3 Related Work

Most existing work on Oblivious RAM assumes only a single client. Franz et al. [7] proposed an solution for multiple clients using the original square-root ORAM, but relies on a semi-honest server. Goodrich et al. [12] extend that work to more modern tree-based ORAMs, still relying on a semi-honest server. Recent work by Maffei et al. [18] supports multiple clients with read/write access control, but again requires a semi-honest server. Other efficient solutions are possible with an even weaker security model by using trusted hardware on the server side [13,17].

There is also a concurrent line of work in Parallel ORAMs which target ORAMs running on multi-core or multi-processor systems [2,4,20]. These schemes either do not target malicious adversaries or require constant and continuous communication between clients to synchronize their state. As stated above, this is not a viable solution for clients which are not always on or may be across different networks.

Currently, no solution exists that allows for multiple clients interacting with a malicious server and without direct client-to-client communication, or constant polling to create the same effect.

3 Security Definition

We briefly recall standard ORAM concepts. An ORAM provides an interface to read from and write to blocks of a RAM (an array of storage blocks). It supports $\mathsf{Read}(x)$, to read from the block at address x, and $\mathsf{Write}(x, v)$ to write value v to block x. The ORAM allows storage of N blocks, each of size B.

To securely realize this functionality, an ORAM interacts with a malicious storage device. Below, we use Σ to represent the *interface* between a client and the actual storage device. A client with access to Σ issues Read and Write requests as needed. We stress that we make no assumptions about how the storage device responds to these requests, and allow for arbitrary malicious behavior. For instance, an adversary could respond to a Read request with old or corrupt data, refuse to actually perform a Write correctly etc. As related work, the (untrusted) storage device is part of a *server* as in envisioned applications for a (multi-client) ORAM such as outsourced cloud storage.

Definition 1 (ORAM Operation OP**).** *An operation* OP *is defined as* $\mathrm{OP} = (o, x, v)$*, where* $o = \{\mathsf{Read}, \mathsf{Write}\}$*,* x *is the virtual address of the block to be accessed and* v *is the value to write to that block.* $v = \perp$ *when* $o = \mathsf{Read}$*.*

We now present our multi-client ORAM security definition which slightly augments the standard, single-client ORAM definition. We emphasize that clients only interact with the server by read or write operations to a memory location; there are no other messages sent. Therefore the "protocol" is fully defined by these patterns of accesses.

Definition 2 (Multi-client ORAM Π**).** *A multi-client ORAM* $\Pi = (\mathsf{Init}, \mathsf{ClientInit}, \mathsf{Access})$ *comprises the following three algorithms.*

1. $\mathsf{Init}(\lambda, N, B, \phi)$ *initializes* Π*. It takes as input security parameter* λ*, total number of blocks* N*, block size* B*, and number of clients* ϕ*.* Init *initializes the storage device represented by* Σ *and outputs a key* κ*.*
2. $\mathsf{ClientInit}(\lambda, N, B, j, \kappa)$ *uses security parameter* λ*, number of blocks* N*, block size* B*, and a secret key* κ *to initialize client* u_j*. It outputs a client state* st_{u_j}*.*
3. $\mathsf{Access}(\mathrm{OP}, \Sigma, st_{u_j})$ *performs operation* OP *on the ORAM using client* u_j*'s state* st_{u_j} *and interface* Σ*.* Access *outputs a new state* st_{u_j} *for client* u_j*.*

In contrast to single-client ORAM, a multi-client ORAM introduces the notion of clients. This is modeled by different per-client states, st_{u_i} for client u_i. After initializing the multi-client ORAM with Init, Algorithm $\mathsf{ClientInit}$ is run by each client (with no communication between them) separately and outputs their initial local states st_{u_i}. The $\mathsf{ClientInit}$ function only requires that each client have a shared key κ, which could be derived from a password. Whenever client u_i executes Access on the multi-client ORAM, they can attempt to update the multi-client ORAM represented by Σ, and update their own local state st_{u_i}, but not the other clients' local states.

Finally, we define the security of a multi-client ORAM against malicious servers. Consider the game-based experiment $\mathsf{Sec}^{\mathsf{ORAM}}_{\mathcal{A},\Pi}(\lambda)$ below. In this game, $\mathcal{A}$ has complete control over the storage device and how it responds to client requests. For ease of exposition, we model this as $\mathcal{A}$ outputting their own compromised version $\Sigma_{\mathcal{A}}$ of an interface to the storage device. It is this malicious interface $\Sigma_{\mathcal{A}}$ that clients will subsequently use for their Access operations. Interface $\Sigma_{\mathcal{A}}$ is controlled by the adversary and updates state $st_{\mathcal{A}}$. That is, $\mathcal{A}$ learns all clients' calls to the interface, therewith the clients' requested access pattern, and can adaptively react to clients' interface requests. To initialize the ORAM, ϕ time steps are used to do the setup for each client, then the game continues for $\mathrm{poly}(\lambda)$ additional steps where the adversary interactively specifies operations and maliciously modifies the storage.

Experiment 1 (Experiment $\mathsf{Sec}^{\mathsf{ORAM}}_{\mathcal{A},\Pi}(\lambda)$)

```
$b \xleftarrow{\$} \{0,1\}$
$(\kappa, \Sigma) \leftarrow \mathsf{Init}(\lambda, B, N, \phi)$
for $i = 1$ to $\phi$ do
    $st_{u_i} \leftarrow \mathsf{ClientInit}(\lambda, N, B, \kappa, i)$
end
$(\mathrm{OP}_{\phi+1,0}, \mathrm{OP}_{\phi+1,1}, i, st_{\mathcal{A}}, \Sigma_{\mathcal{A}}) \leftarrow \mathcal{A}(\lambda, N, B, \phi, \Sigma)$
for $j = \phi + 1$ to $poly(\lambda)$ do
    $st_{u_i} \leftarrow \mathsf{Access}(\mathrm{OP}_{j,b}, \Sigma_{\mathcal{A}}, st_{u_i})$
    $(\mathrm{OP}_{j+1,0}, \mathrm{OP}_{j+1,1}, i, st_{\mathcal{A}}) \leftarrow \mathcal{A}(P, st_{\mathcal{A}})$
end
$b' \leftarrow \mathcal{A}(st_{\mathcal{A}})$
output $1$ iff $b = b'$
```

In summary, $\mathcal{A}$ gets oracle access to the ORAM and can adaptively query it during $poly(\lambda)$ rounds. In each round, $\mathcal{A}$ selects a client u_i and determines two operations $\mathrm{OP}_{j,0}$ and $\mathrm{OP}_{j,1}$. The oracle performs operation OP_b as client u_i with state st_{u_i}, interacting with the adversary-controlled $\Sigma_{\mathcal{A}}$ using protocol Π. Eventually, $\mathcal{A}$ guesses b.

Definition 3 (Multi-client ORAM security). *An ORAM $\Pi = (\mathsf{Init}, \mathsf{Access})$ is multi-client secure* iff *for all PPT adversaries $\mathcal{A}$, there exists a function $\epsilon(\lambda)$ negligible in security parameter λ such that*

$$Pr[\mathsf{Sec}^{\mathsf{ORAM}}_{\mathcal{A},\Pi}(\lambda) = 1] < \frac{1}{2} + \epsilon(\lambda).$$

Our game-based definition is equivalent to ORAM's standard security definition with two exceptions: we allow the adversary to arbitrarily change the state of the on-server storage Σ, and we split the ORAM algorithm into ϕ different "pieces" which cannot share state among themselves.

As discussed above, this work assumes that all clients trust each other and do not conspire. For ease of exposition, we assume that all client share key κ used for encryption and MAC computations that we will introduce later.

Consistency: An orthogonal concern to security for multi-client schemes is *consistency*, whether the clients each see the same version of the database when they access it. Because the clients in our model do not have any way of communicating except through the malicious adversary, it is possible for $\mathcal{A}$ to "desynchronize" the clients so that their updates are not propagated to each other. Our multi-client ORAM guarantees that in this case the clients still have complete security and access pattern privacy, but consistency cannot be guaranteed. This is a well known problem with the only solution being *fork* consistency [16], which we achieve.

4 Multi-client Security for Classical ORAMs

We start by transforming two classical ORAM constructions, the original square-root solution by Goldreich [8] and the hierarchical one by Goldreich and

Ostrovsky [9], into multi-client secure versions, retaining the same communication complexity per client. Our exposition focuses in the beginning on details for the multi-client square-root ORAM, as the hierarchical ORAM is borrowing from the same ideas.

Recall that the square-root ORAM algorithm works by dividing the server storage into two parts: the main memory and the cache, which is of size $O(\sqrt{N}))$. The main memory is shuffled by a pseudo-random permutation π. Every access reads the entire cache, plus one block in the main memory. If the block the client wants is found in the cache, a "dummy" location is read in the main memory, otherwise the actual location of the target block is read and it is inserted into the cache for later accesses. After $\sqrt{N}$ accesses, the cache is full and the client must download the entire ORAM and reshuffle it into a fresh state.

Specific Challenge: When considering a multi-client scenario, it becomes easy for a malicious server to break security of the square-root ORAM. For example, client u_1 can access a block x that is not in the cache, requiring u_1 to read $\pi(x)$ from main memory and insert it into the cache. The malicious server now restores the cache to the state it was in before u_1's access added block x. If a second client u_2 also attempts to access block x, the server will now observe that both clients read from the same location $\pi(x)$ in main memory and know that u_1 and u_2 have accessed the same block (or not). Without the clients having a way to communicate directly with each other and pass information that allows them to verify the changes to the cache, the server can always "rewind" the cache back to a previous state. This will eventually force one client to leak information about their accesses.

Rationale: Our approach for multi-client security is based on the observation that the *cache update* part of the square-root solution is secure by itself. Updating the cache only involves downloading the cache, changing one element in it, re-encrypting, and finally storing it back on the server. Downloading and later uploading the cache implies always "touching" the same $\sqrt{N}$ blocks. This is independent of what the malicious server presents to a client as Σ and also independent of the block being updated by the client. Changing values inside the cache cannot leak any information to the server, as its content is always newly IND-CPA encrypted. Succinctly, being similar to a trivial ORAM, updating a cache is automatically multi-client secure.

However, reading can leak information. Reading from the main ORAM is conditional on what the client finds in the cache. We call this part the *critical* part of the access, and the cache update correspondingly *non-critical.* To counteract this leakage, we implement the following changes to enable multiple clients for the square-root ORAM:

1. **Separate ORAMs:** Instead of a single ORAM, we use a sequence of ORAMs, $\Sigma = \mathsf{ORAM}_1, \mathsf{ORAM}_2, \ldots, \mathsf{ORAM}_\phi$, one for each client. Client u_i will perform the critical part of their access only on ORAM_i's main memory

```
Input: Storage interface $\Sigma$, Security parameter $\lambda$, number of blocks in each
       ORAM $N$, block size $B$, client number $i$, key $\kappa$
Output: Client state $st_{u_i}$
Generate permutation $\pi_{i,0}$ from key $\kappa$;
Initialize $\sqrt{N} + N$ main memory blocks (size $B$, shuffled with $\pi_{i,0}$) and $\sqrt{N}$
  cache blocks;
Set cache counter $\chi_i = 0$; Set epoch counter $\gamma_i = 0$;
$data_i = \mathsf{Enc}_\kappa(\text{main memory})||\mathsf{Enc}_\kappa(\text{cache}||\chi_i||\gamma_i)$;
$mac_i = \mathsf{MAC}_\kappa(\mathsf{Enc}_\kappa(\text{cache}||\chi_i||\gamma_i))$;
$\mathsf{ORAM}_i = data_i||mac_i$;
On $\Sigma$, replace the $i^{\text{th}}$ ORAM by $\mathsf{ORAM}_i$;
output $st_{u_i} = \{\kappa, \chi_i\}$;
```

Algorithm 1. ClientInit$(\Sigma, \lambda, N, B, i, \kappa)$, initialize client square-root ORAM

and cache. Thus, each client can guarantee they will not read the same address from their ORAM's main memory twice. However, any change to the cache as part of ORAM Read(x) or Write(x, v) operations will be written to every ORAM's cache. Updating the cache on any ORAM is already guaranteed to be multi-client secure and does not leak information.

2. **Authenticated Caches:** For each client u_i to guarantee that they will not repeat access to the main memory of ORAM_i, the cache is stored together with an encrypted *access counter* χ on the server. Each client stores locally a MAC over both the cache and the encrypted access counter χ of their own ORAM. Every access to their own cache increments the counter and updates the MAC. Since clients read only from their own ORAMs, and they can always verify the counter value for the last time that they performed a read, the server cannot roll back beyond that point. Two reads will never be performed with the cache in the same state.

4.1 Details

We detail the above ideas in two algorithms: Algorithm 1 shows the per client initialization procedure ClientInit, and Algorithm 2 describes the way a client performs an Access with our multi-client secure square-root ORAM. The Init algorithm is trivial in our case, as it initializes Σ to ϕ empty arrays ORAM_j. Each array is of size $N + 2 \cdot \sqrt{N}$ blocks, each block has size B bits.

Before explaining ClientAccess, we first introduce the notion of an *epoch.* In general, after $\sqrt{N}$ accesses to a square-root ORAM, its cache is "full", and the whole ORAM needs to be re-shuffled. Re-shuffling requires computing a new permutation π. Per ORAM, a permutation can be used for $\sqrt{N}$ operations, i.e., one *epoch.* The next $\sqrt{N}$ operations, i.e., the next epoch, will use another permutation and so on. In the two algorithms, we use an epoch counter γ_i. Therewith, π_{i,γ_i} denotes the permutation of client u_i in ORAM_i's epoch γ_i. For

```
Input: Mult-client ORAM $\Sigma$, address $x$, new value $v$, client $u_i$, $st_{u_i} = \{\kappa, \chi_i\}$
Output: Value of block $x$, new state $st_{u_i}$
From $\mathsf{ORAM}_i$ in $\Sigma$: read $c_i = \mathsf{Enc}_\kappa(\text{cache}||\chi_i||\gamma_i)$ and $mac_i$;
$mac'_i = \mathsf{MAC}_\kappa(c_i)$;
if $mac'_i \neq mac_i$ then output Abort;
Decrypt $c_i$ to get cache and counter $\chi'_i$;
if $\chi'_i < \chi_i$ then output Abort;
if $block\ x \notin cache$ then
    Read and decrypt block $\pi_{i,\gamma_i}(x)$ from $\mathsf{ORAM}_i$'s main memory;
else
    Read next dummy block from $\mathsf{ORAM}_i$'s main memory;
if $v = \perp$ then // operation is a Read
    $\nu \leftarrow$ existing value of block $x$;
else // operation is a Write
    $\nu \leftarrow v$ ;
Append block $(x, \nu)$ to cache;
if $cache\ is\ full$ then
    $\gamma_i = \gamma_i + 1$; Compute new permutation $\pi_{i,\gamma_i}$;
    Read and decrypt $\mathsf{ORAM}_i$'s main memory;
    Shuffle cache and main memory using $\pi_{i,\gamma}$;
    Update $\mathsf{ORAM}_i$ with $\mathsf{Enc}_\kappa$(main memory);
$\chi_i = \chi'_i + 1$; $mac_i = \mathsf{MAC}_\kappa(\mathsf{Enc}_\kappa(\text{cache}||\chi_i||\gamma_i))$;
Update $\mathsf{ORAM}_i$ with $\mathsf{Enc}_\kappa(\text{cache}||\chi_i||\gamma_i)$ and $mac_i$;
for $j \neq i$ do // for all $\mathsf{ORAM}_j \neq \mathsf{ORAM}_i$
    Read and decrypt cache and $\chi_j$ from $\mathsf{ORAM}_j$; Read and verify $mac_i$ from
    $\mathsf{ORAM}_j$;
    Append block $(x, \nu)$ to cache;
    if $cache\ is\ full$ then
        $\gamma_j = \gamma_j + 1$; Compute new permutation $\pi_{j,\gamma_j}$;
        Read and decrypt $\mathsf{ORAM}_j$'s main memory;
        Shuffle cache and main memory using $\pi_{j,\gamma_j}$;
        Update $\mathsf{ORAM}_j$ with $\mathsf{Enc}_\kappa$(main memory);
    $mac_j = \mathsf{MAC}_\kappa(\mathsf{Enc}_\kappa(\text{cache}||\chi_j||\gamma_j))$;
    Update $\mathsf{ORAM}_j$ with $\mathsf{Enc}_\kappa(\text{cache}||\chi_j||\gamma_j)$ and $mac_j$;
end
output $(\nu, st_{u_i} = \{\kappa, \chi_i\})$;
```

Algorithm 2. Access(OP, Σ, st_{u_i}), Read, Write for multi-client square-root ORAM

any client, to be able to know the current epoch of ORAM_i, we store γ_i together with the ORAM's cache on the server.

On a side note, we point out that there are various ways to generate pseudorandom permutations π_{i,γ_i} on N elements in a deterministic fashion. For example, in the a cloud context, one can use $\text{PRF}_\kappa(i||\gamma_i)$ as the seed in a PRG and therewith perform Knuth's Algorithm P (Fisher-Yates shuffle) [14]. Alternatively, one can use the idea of random tags followed by oblivious sorting by Goldreich and Ostrovsky [9].

In addition to the epoch counter, we also introduce a per client *cache counter* χ_i. Using χ_i, client u_i counts the number of accesses of u_i to the main memory and cache of their own ORAM_i. After each access to ORAM_i by client u_i, χ_i is incremented. Each client u_i keeps a local copy of χ_i and therewith verifies freshness of data presented by the server. As we will see below, this method ensures multi-client ORAM security. Note in Algorithm 2 that a client u_j never increases χ_i of another client u_i. Only u_i ever updates χ_i.

In our algorithms, Enc_κ is an IND-CPA encryption such as AES-CBC. For convenience, we only write Enc_κ(main memory), although the main memory needs to be encrypted block by block to allow for the retrieval of specific blocks. Also, for the encryption of main memory blocks, Enc_κ offers authenticated encryption such as encrypt-then-MAC.

A client can determine whether a *cache is full* in Algorithm 2 by the convention that empty blocks in the cache decrypt to $\perp$. As long as there are blocks in the cache remaining with value $\perp$, the cache is not full.

ClientInit: Each client runs the ClientInit algorithm to initialize their ORAM. The server stores the ORAMs (with MACs) computed with a single key κ. Each client receives their state from the ClientInit algorithm, comprising the cache counter. Note that although not captured in the security definition, our scheme also allows for dynamic adding and removing of clients. Removing is as simple as just asking the server to delete one of the ORAMs, and adding could be done by running ClientInit, but instead of initializing the blocks to be empty, the client first downloads a copy of another client's ORAM to get the most recent version of the database.

Access: After verifying the MAC for ORAM_i and whether its cache is not from before u_i's last access, u_i performs a standard Read or Write operation for block x on ORAM_i. If the cache is full, u_i re-shuffles ORAM_i updating π. In addition, u_i also adds block x to all other clients' ORAMs. Note that for this, u_i does not read from the other ORAMs, but only completely downloads and re-encrypts their cache.

Our scheme is effectively running ϕ traditional square-root ORAMs in parallel, making the overall complexity $O(\phi\sqrt{N})$. Due to limited space, see the full version of this paper [1] for a detailed security analysis.

Fork Consistency: When a client makes an access, they add an element to the cache for all ϕ clients. Therefore, at any given timestep, if the server is not maliciously changing caches, all caches will have the same number of elements in them. Since each cache is verified by a MAC, the server cannot remove individual elements from a cache. The only viable attack is to present an old view of a cache which was at one point valid, but does not contain new updates that have been added by other clients. If the server chooses to do this, he creates a *fork* between the views of the clients which have seen the update and those that have not. Since the server can never "merge" caches together, but only present entire caches that have been verified with a MAC by a legitimate client, there is no way to reconcile two forks that were created without a client finding out. This achieves fork consistency for our scheme.

Complexity: Making the square-root solution multi-client secure does not induce any additional asymptotic complexity, per client. Each access requires downloading the cache of size $\sqrt{N}$ and accessing one block from the main memory. Every $\sqrt{N}$ accesses, the main memory and cache must be shuffled, requiring N communication if the client has enough storage to temporarily hold the database locally. If not, then Goldreich and Ostrovsky [9] noticed that one can use a Batcher sorting network to obliviously shuffle the database with complexity $O(N \log^2 N)$, or the AKS algorithm with complexity $O(N \log N)$. One can also reduce the hidden constant using a more efficient Zig-Zag sort [10]. In the first scenario, the amortized overall complexity is then $O(\phi\sqrt{N})$, while the second is $O(\phi\sqrt{N} \log N)$.

Goodrich et al. [11] also propose a way to deamortize the classical square-root ORAM such that it obtains a worst-case overhead factor of $\sqrt{N} \cdot \log^2(N)$. Their method involves dividing the work of shuffling over the $\sqrt{N}$ operations during an epoch such that when the cache is full there is a newly shuffled main memory to swap in right away. Since the shuffling is completely oblivious (does not depend on any pattern of data accesses) and memoryless (the clients only need to know what step of the shuffle they are on in order to continue the shuffle), it can be considered a "non-critical" portion of the algorithm and no special protections need to be added for malicious security.

Note on Computational Complexity: While Algorithm 1 returns the whole updated state Σ, in practice a client only needs to update the other clients' caches (up to $\sqrt{N}$ times). In addition to the communication complexity involved, there is also computation the client must perform in our scheme. Fortunately, the computation is exactly proportional to the communication and easily quantifiable. Every block of data retrieved from the server has a MAC that must be verified and a layer of encryption that must be removed. Since modern ciphers and hash functions are very efficient, and can even be done in hardware on many computers, communication is the clear bottleneck. For comparison, encryption and MACs are common on almost every secure network protocol, so we consider only the communication overhead in our analysis.

Unified Cache: A natural optimization to this scheme is to have one single shared cache instead of a separate one for each user. If the server behaves honestly, then all caches will contain the same blocks and be in the same state anyway, so a single cache can save some communication and storage. To still protect against a malicious server, one must be careful in this case to store ϕ different counters with the cache and have each client only increment their counter when they do an access. This ensures that if a client inserts a block into the cache it cannot be "rolled back" past the point that they inserted that block without them noticing. Since the cost to reshuffle the ORAMs dominates complexity of our scheme, this optimization does not change asymptotic performance. Resulting in only a small constant improvement and making the presentation and proof unnecessary difficult, we omit full discussion of this technique.

4.2 Hierarchical Construction

In addition to the square-root ORAM, Goldreich and Ostrovsky [9] also propose a generalization which achieves poly-log overhead. In order to do this, it has a hierarchical series of caches instead of a single cache. Each cache has 2^j slots in it, for j from 1 to $\log N$, where each slot is a bucket holding $O(\log N)$ blocks. At the bottom of the hierarchy is the main memory which has $2 \cdot N$ buckets.

The reader is encouraged to refer to the original paper [9] for full details, but the main idea is that each level of the cache is structured as a hash table. Up to 2^{j-1} blocks can be stored in cache level j, half the space is reserved for dummies like in the previous construction. After accessing 2^{j-1} blocks, the entire level is retrieved and shuffled into the next level. Shuffling involves generating a new hash function and rehashing all the blocks into their new locations in level $j+1$, until the shuffling percolates all the way to the bottom, and the client must shuffle main memory to start again. Level j must be shuffled after 2^{j-1} accesses, resulting in an amortized poly-logarithmic cost.

To actually access a block, a client queries the caches in order using the unique hash function at each level. When the block is found, the remainder of the queries will be on dummy blocks to hide that the block was already found. After reading, and potentially changing the value of the block, it is added back into the first level of the cache and the cache is shuffled as necessary.

Multi-client Security: As this scheme is a generalization of the square-root one, our modifications extend naturally to provide multi-client security. Again, each client should have their own ORAM which they read from. Writing to other clients' ORAMs is done by inserting the block into the top level of their cache and then shuffling as necessary. The only difference this time is that each level of the cache must be independently authenticated. Since the cache levels are now hash tables, and computing a MAC over every level for each access would require downloading the whole data structure, we can instead use a Merkle tree [19]. This allows for efficient verification and updating of pieces of the cache without having access to the entire thing, and it maintains poly-logarithmic communication complexity. The root of the Merkle tree will contain the counter that is incremented by the ORAM owner when they perform an access.

Deamortizing: Other authors have proposed deamortized versions of the hierarchical construction that achieve worst-case poly-logarithmic complexity, such as Kushilevitz et al. [15] and Ostrovsky and Shoup [21]. We will use as an example the "warm-up" construction from Kushilevitz et al. [15], Sect. 6.1. This is a direct deamortization of the original hierarchical scheme described above. They deamortize by using three separate hash tables at each level of the ORAM, labelled "active", "inactive", and "output". Instead of shuffling all at one time after 2^{j-1} accesses (which would lead to worst case $O(N)$ complexity), their approach is now different. When the cache fills up at level j, it is marked "inactive", and the old "inactive" buffer is cleared and marked "active". The idea will be that the "inactive" buffer is shuffled over time with each ORAM access, so that no worst-case $O(N)$ operations are required. As it is shuffled, the contents are copied into the "output" buffer. Accesses can continue while the "inactive" buffer is being shuffled, as long as a read operation searches both the "active" and "inactive" buffers (since a block could be in either one).

When the shuffle completes, the "output" buffer will contain the newly shuffled contents that go into level $j+1$. This buffer is marked as "active" for level $j+1$, the "active" buffer on level j is marked "inactive" and the "inactive" buffer is cleared and marked "active", restarting the whole process. Since the shuffle is spread out over 2^{j-1} accesses, and the shuffling was the only part that was worst-case $O(N)$, this makes a full construction that now has worst-case $O(\log^3 N)$ communication complexity.

In terms of multi-client security, the only important aspects of this process is that no elements be removed from "active" or "inactive" buffers that the owner of the ORAM has put there – until a shuffle is complete, starting a new epoch. The shuffling itself is automatically data oblivious and therewith "non-critical", in the terms we have established in this paper. Using a Merkle tree and counters, as described in the amortized version, will assure that the server cannot roll back the cache to any state prior to the last access by the owner, guaranteeing security.

Kushilevitz et al. [15] also propose an improved hierarchical scheme that achieves $O(\log^2 N/\log\log N)$ complexity, which is substantially more involving. As the deamortized hierarchical ORAM as described above is sufficient for our main contribution in Sect. 5, we leave it to future research to adapt Kushilevitz et al. [15]'s scheme for multi-client security.

5 Tree-Based Construction

While pioneering the research, classical ORAMs have been outperformed by newer tree-based ORAMs which achieve better average and worst-case complexity and low constants in practice. We now proceed to show how these constructions can be modified to also support multiple clients. Our strategy will be similar to before, but with one major twist: in order to avoid linear worst case complexity, tree-based ORAMs do only small local "shuffling," which turns out to make separating a client access into critical and non-critical parts much more difficult. When writing, one must not only add a new version of the block to the ORAM, but also explicitly mark the old version as obsolete, requiring a conditional access. This is in contrast with our previous construction where old versions of a block would simply be discarded during the shuffle.

5.1 Overview

For this section, we will use Path ORAM [25] as the basis for our multi-client scheme, but the concepts apply similarly to other tree-based schemes.

Although the interface exposed to the client by Path ORAM is the same as other ORAM protocols, it is easiest to understand the Access operation as being broken down into three parts: ReadAndRemove, Add, and Evict [23]. ReadAndRemove, as the name suggests, reads a block from the ORAM and removes it, while Add adds it back to the ORAM, potentially with a different value. These two operations used together form the basis of the Access operation, but it begins to illustrate the difficulty we have making this scheme multi-client secure: changing the value of a block implicitly requires reading it, meaning that both reading and writing are equally *critical* and not easily separated as our previous construction. The third operation, Evict, is a partial shuffling that is done after each access in order to maintain the integrity of the tree.

The RAM in Path ORAM is structured as a tree with N leaf nodes. Each node in the tree holds up to Z blocks, where Z is a small constant. Each block is tagged with a value uniform in the range $[0, N)$. As an invariant, blocks will always be located on the path from the root of the tree to the leaf node corresponding to their tag. Over the lifecycle of the tree, blocks will enter at the root and filter their way down toward the leaves, making room for new blocks to in turn enter at the root. The client has a map storing for every block which leaf node the block is tagged for.

ReadAndRemove: To retrieve block x, the client looks up in the map which leaf node it is tagged for and retrieves all nodes from the root to that leaf node, denoted $\mathcal{P}(x)$. By the tree invariant, block x will be found somewhere on the path $\mathcal{P}(x)$. The client then removes block x from the node it was found in, reencrypts all the nodes and puts them back in the RAM.

Add: To put a block back in the ORAM, the client simply retrieves the root node and inserts the block into one of its free slots, reencrypting and writing the node back afterwards. The map is updated with a new random tag for this block in the interval $[0, N)$. If there is not enough room in the root node, the client keeps the block locally in a "stash" of size $Y = O(\log N)$, waiting for a later opportunity to insert the block into the tree.

Evict: So that the stash does not become too large, after every operation the client also performs an eviction which moves blocks down the tree to free up space. Eviction consists of picking a path in the tree (using reverse lexicographic order [6]) and moving blocks on that path as far down the tree as they can go, without violating the invariant. Additionally, the client inserts any matching block from the stash into the path.

Recursive Map: Typically, the client's map, which stores the tag for each block, has size $O(N \cdot \log N)$ bit and is often too large to store locally. Yet, if block size B is at least $2 \cdot \log N$ bit, the map can itself be stored *recursively* in an ORAM on the server, inducing a total communication complexity of $O(\log^2 N)$ blocks.

Additionally, Stefanov et al. [25] show that if $B = \Omega(\log^2 N)$ bit, communication complexity can be reduced to $O(\log N)$ blocks.

Integrity: Because of its tree structure, it is straightforward to ensure integrity in Path ORAM. Similar to a Merkle tree, the client can store a MAC in every node of the tree that is computed over the contents of that node and the respective MACs of its two children. Since the client accesses entire paths in the tree at once, verifying and updating the MAC values when an access is done incurs minimal overhead. This is a common strategy with tree-based ORAMs, which we will make integral use of in our scheme. We will also include client u_i's counter χ_u in the root MAC as before, to prevent rollback attacks (see below).

Challenge. Looking at Path ORAM, there exist several additional challenges when trying to add multi-client capabilities with our previous strategy. First, recursively storing the map into ORAMs imposes a problem. To resolve a tag, each path accessed in the recursive ORAMs has to be different for each client. If we separate the map into ϕ separate ORAMs (which we will do), the standard recursive lookup results in a large blowup in communication costs. At the top level of the recursion, we would have ϕ ORAMs, one for each client. Yet, each of those will fan out to ϕ ORAMs to obliviously support the next level of recursion, each of which will have ϕ more, going down $\log n$ levels. The overall communication complexity for the tag lookup would be $\phi^{\log N} \in \Omega(N)$.

Second, an Add in Path ORAM cannot be performed without ReadAndRemove, so we cannot easily split the access into *critical* and *non-critical* parts like before.

Rationale. To remedy these problems, we institute major changes to Path ORAM:

1. **Unified Tagging:** Instead of separately tagging blocks in each of the ORAMs and storing the tags recursively, we will have a unified tagging system where the tag for a block can be computed for any of the separate ORAMs from a common "base tag." This is crucial to avoiding the $O(N)$ communication overhead that would otherwise be induced by the recursive map as described above. For a block x, the map will resolve to a base tag value t. This same tag value is stored in every client's recursive ORAM. Let h be a PRF mapping from $[0, 2^\lambda) \times [1, \phi]$ to $[0, N)$. The idea is now that the leaf that block x will be percolating to in the recursive ORAM tree differs for every ORAM of every client u_i and is pseudo-randomly determined by value $h(t, i)$. This way, (1) the paths accessed in all recursive map ORAMs for all clients differ for the same block x, and (2) only one lookup is necessary at each level of the recursive map to get the leaf node tag for all ϕ ORAMs.
2. **Secure Block Removal:** The central problem with ReadAndRemove is that it is required before every Add so that the tree will not fill up with old, obsolete blocks which cannot be removed. Unlike the square-root ORAM, the shuffling process (eviction) happens locally and cannot know about other versions of

a block which exist on different paths. We solve this problem by including metadata on each bucket. For every node in the tree, we include an encrypted array which indicates the ID of every block in that node. Removing a block from the tree can then be performed by simply changing the metadata to indicate that the slot is empty. It will be overwritten by the eviction routine with a real block if that slot is ever needed. If B is large, this metadata is substantially smaller than the real blocks. We can then store it in a less efficient classical ORAM described above which is itself multi-client secure. This allows us to take advantage of the better complexity provided by tree-based ORAMs for the majority of the data, while falling back on a simpler ORAM for the metadata which is independent of B.

We also note that Path ORAM's stash concept cannot be used in a multi-client setting. Since the clients do not have a way of communicating with each other out of band, all shared state (which includes the stash) must be stored in the RAM. This has already been noted by Goodrich et al. [12], and since the size of the stash does not exceed $\log N$, storing it in the RAM (encrypted and integrity protected) does not affect the overall complexity. As before, we also introduce an eviction counter e for each ORAM. Client u_i will verify whether, for each of their recursive ORAMs, this eviction counter is fresh.

5.2 Details

To initialize the multi-client ORAM (Algorithm 3), ϕ separate ORAMs are created and the initial states (containing the shared key) are distributed to each client. For each client u_i, the ORAM takes the form of a series of trees $T_{j,i}$. The first tree stores the data blocks, while the remaining trees recursively store the map which relates block addresses to leaf nodes. In addition to this, as described above, each tree has its own sub-ORAM to keep track of block metadata. The stash of each (sub-)ORAM is called $S_{0,i}$, and the metadata (classical) ORAM $M_{j,i}$.

To avoid confusion between different ORAM initialization functions, MInit is a reference to Algorithm 1, i.e., initialization of a multi-client secure classical ORAM.

For simplicity, we assume that Enc_κ encrypts each node of a tree separately, therewith allowing individual node access. Also, we assume authenticated encryption, using the per node integrity protection previously mentioned.

As noted above, the functions ($\mathsf{ReadAndRemove}, \mathsf{Add}$) can be used to implement ($\mathsf{Read}, \mathsf{Write}$), which in turn can implement a simple interface (Access). Because our construction introduces dependencies between $\mathsf{ReadAndRemove}$ and Add, in Algorithm 4 we illustrate a unified Access function for our scheme. The client starts with the root block and traverses the recursive map upwards, finds the address of block x, and finally retrieves it from the main tree. For each recursive tree, it retrieves a tag value t allowing to locate the correct block in the next tree. After retrieving a block in each tree, the client marks that block as free in the metadata ORAM so that it can be overwritten during a future

```
Input: Security parameter $\lambda$, number of blocks in each ORAM $N$, block size $B$,
       number of clients $\phi$, initialization sub-routine for multi-client classical
       ORAM $\mathsf{MInit}$
Output: Initial ORAM state $\Sigma_{\text{init}}$, initial per client states $\{st_{u_1}, \ldots, st_{u_\phi}\}$
$\kappa \xleftarrow{\$} \{0,1\}^\lambda$;
for $j = 1$ to $\phi$ do
    $i = 0$;
    $N_0 = N$;
    while $N_i > 1$ do
        Initialize a tree $T_{j,i}$ with $N_i$ leaf nodes;
        Set eviction counter $e_{j,i} = 0$;
        // The stash must also be stored on the server
        Create array $S_{j,i}$ with $Y$ blocks;
        // Use a sub-ORAM to hold block metadata
        $M_{j,i} = \mathsf{MInit}(\lambda, 2n_i \cdot Z, Z \cdot \log n_i, \phi)$;
        $N_{i+1} = N_i \cdot \lceil \log N_i / B \rceil$;
        $i = i + 1$;
    end
    // Let $m$ be number of recursive trees made in previous loop
    $m = i$;
    Create a root block $\mathcal{R}_j$;
    Set ORAM counter $\chi_j = 0$;
    $\mathsf{ORAM}_j = \mathsf{Enc}_\kappa((T_{j,0}, M_{j,0}, S_{j,0}, e_{j,0}) || \ldots \ ||(T_{j,m}, M_{j,m}, S_{j,m}, e_{j,m})||\chi_j||\mathcal{R}_j)$;
    Send $st_{u_j} = \{\kappa, \chi_j, e_{j,0}, \ldots, e_{j_m}\}$ to client $u_j$;
end
Initialize $\Sigma$ to hold $\{\mathsf{ORAM}_1, \ldots, \mathsf{ORAM}_\phi\}$;
```

Algorithm 3. $\mathsf{Init}(\lambda, N, B, \phi)$, initialize multi-client tree-based ORAM

eviction. This is necessary to maintain the integrity of the tree and ensure that it does not overflow. At the same time, the client also marks that block free in the metadata of each other client and inserts the new block value into the root of their trees. This is analogous to the previous scheme where a client reads from their own ORAM and writes back to the ORAMs of the other clients. We use a straightforward MAC technique for paths $\mathsf{MACPath}$, which we present in the extended version of this paper [1].

Again, we avoid confusion between different ORAM access operations by referring to the multi-client secure classical ORAM access operation of Algorithm 2 as $\mathsf{MAccess}$.

Algorithm 5 illustrates the eviction procedure. Since eviction does not take as input any client access, it is non-critical. The client simply downloads a path in the tree which is specified by eviction counter e and retrieves it in its entirety. The only modification that we make from the original Path ORAM scheme is that we read block metadata from the sub-ORAM that indicates which blocks in the path are free and can be overwritten by new blocks being pushed down the tree.

```
Input: Address $x$, client $u_i$, $st_{u_i} = \{\kappa, \chi_i\}$, sub-routine for multi-client classical
       ORAM $\mathsf{MAccess}$
Output: The value of block $x$
// Let $m$ be recursion depth, $n_j$ the number of blocks in tree $j$
Retrieve root block $\mathcal{R}$;
$pos = x/n$; $x_m = \lfloor pos \cdot (B/\lambda) \rfloor$; $t_m = \mathcal{R}[x_m]$// Find tag $t_m$ of address $x$;
$t'_m \xleftarrow{\$} [0, 2^\lambda)$// Compute new tag $t'_m$ for $x$;
for $j = m$ to $0$ do
    $\text{leaf}_j = h(t_j, u_i)$// Compute leaf of client $u_i$'s $\mathsf{ORAM}_i$;
    Read path $\mathcal{P}(\text{leaf}_j)$ and $S_{i,j}$ from $T_{i,j}$, locating block $x_j$;
    Retrieve MAC values for $\mathcal{P}(\text{leaf}_j)$ as $V$ and the stored counter as $\chi'_i$;
    if $V \neq \mathsf{MACPath}(\Sigma, st_{u_i}, \mathcal{P}(\text{leaf}_j), S, \chi'_i) \vee \chi'_i \neq \chi_i$ then Abort ;
    Re-encrypt and write back $\mathcal{P}(\text{leaf}_j)$ and $S_{i,j}$ to $T_{i,j}$;
    // Let $(a, b)$ be the node and slot that $x_j$ was found at
    $\mathsf{MAccess}(M_j, (\mathsf{write}, a \cdot Z + b, \perp), u_i)$;
    if $j \neq 0$ then
        $t'_j \xleftarrow{\$} [0, 2^\lambda)$ // Sample a new value for $t$;
        // Block $x_j$ contains multiple $t$ values
        Extract $t_{j-1}$ from block $x_j$;
        Update block $x_j$ with new value $t'_{j-1}$ and new leaf tag $t'_j$;
    else
        Set $v$ to the value of block $x_j$;
        If OP is a write, update $x_j$ with new value;
    Insert block $x_j$ into the stash $S_{i,j}$;
    $\chi_i = \chi_i + 1$;
    Update MAC of stash to $\mathsf{MAC}_\kappa(S_{i,j}$, $\mathsf{MAC}$ of root bucket, $\chi_i$, $e_{i,j})$;
    // Update the block in other client's ORAMs
    for $p \neq i$ do
        Retrieve path $\mathcal{P}(h(t_j, u_p))$ from $T_{p,j}$ and update metadata so block $x_j$ is
        removed;
        Insert block $x_j$ into the stash $S_{p,j}$ of $T_{p,j}$;
        Update MAC of root bucket in $T_{p,j}$;
    end
    output $(v, st_{u_i} = \{\kappa, \chi_i, e_{i,0}, \ldots, e_{i,m}\})$;
end
```

Algorithm 4. $\mathsf{Access}(\text{OP}, \Sigma, st_{u_i})$, Read or Write on multi-client tree-based ORAM

The overhead from the additional metadata ORAMs that we have in our construction is fortunately not dependent on the block size B. Therefore, if B is large enough, we can achieve as low as the overhead of single-client Path ORAM, $O(\log N)$, or a total complexity of $O(\phi \log N)$ for ϕ users. However, this only applies if the block size B is sufficiently large, at least $\Omega(\log^4 N)$. Otherwise, for smaller B, the complexity can be up to $O(\phi \log^5 N)$. Due to limited space, we refer to the extended version of this paper [1] for a detailed security analysis.

```
Input: Address $x$, new value $v$, client $u_i$, $st_{u_i} = \{k, \chi_i\}$, the number of
       recursive trees $m$
Output: The value of block $x$
for $j = 1$ to $\phi$ do
    for $r = 1$ to $m$ do
        Retrieve eviction counter $e_{j,r}$ for $T_{j,r}$;
        Retrieve path $\mathcal{P}(e_{j,r})$, $S_{j,r}$ and MAC chain $V$;
        // Verify integrity of the path and eviction counter
        if $V \neq \mathsf{MACPath}(\Sigma, st_{u_i}, \mathcal{P}(\text{leaf}_j), S_{j,r}, \chi_i', e_{j,r})$ then Abort ;
        Read metadata for path from $M_{j,r}$;
        Move blocks out of the stash and down the path as far as possible;
        Reencrypt $\mathcal{P}(e_{j,r})$ and $S_{j,r}$ and write back to server;
        Update metadata for path $M_{j,r}$;
        $e_{j,r} = e_{j,r} + 1$;
    end
end
```

Algorithm 5. $\mathsf{Evict}(\Sigma, st_{u_i})$ – Perform Evict on multi-client tree-based ORAM

Complexity: The complexity of our scheme is dominated by the cost of an eviction. For a client to read a path in each of $O(\log N)$ recursive trees, for each of the ϕ different ORAMs, it takes $O(\phi \cdot B \cdot \log^2 N)$ bits of communication. Additionally, the client must make $O(\phi \cdot \log^2 N)$ accesses to a metadata ORAM. If $\mu(N, B)$ denotes the cost of a single access in such a sub-ORAM, the overall communication complexity is then $O(\phi \cdot \log^2 N \cdot [B + \mu(N, \log N)])$ bit. The deamortized hierarchical ORAM by Kushilevitz et al. [15] has $O(\log^3 N)$ blocks communication complexity, where each block is of size $\log N$ bit (the meta-data we need for our construction). Taking this hierarchical ORAM as a sub-ORAM, the total communication complexity computes to $O(\phi \cdot \log^2 N[B + \log^4 N])$ bits. If $B \in \Omega(\log^4 N)$ then the communication complexity, in terms of blocks, is $O(\phi \log^2 N)$, otherwise it is at most $O(\phi \log^5 N)$, i.e., with the assumption $B \in \Omega(\log N)$ (the minimal possible block size for Path ORAM to work).

Additionally, if we use the recursive optimization trick from Stefanov et al. [25] to reduce the overhead from the Path ORAM part of the construction from $O(\log^2 N)$ to $O(\log N)$, we can achieve a total complexity of $O(\log N)$ for blocks of size $\Omega(\log^4 N)$.

Although a complexity linear in ϕ may seem at first to be expensive, we stress that this is a substantial improvement over naive solutions which achieve the same level of security. The only straightforward way to have multi-client security against malicious servers is for each client to append their updates to a master list, and for clients to scan this list to find the most updated version of a block during reads. This is not only linear in the size of the database, but in the number of operations performed over the entire life of the ORAM.

One notable difference in parameters from basic Path ORAM is that we require a block size of at least $c \cdot \lambda$, where $c \geq 2$. Path ORAM only needs $c \cdot \log n$, and for security parameter λ, $\lambda > \log N$ holds. In our scheme, the map trees do not directly hold addresses, but t values which are of size λ. In order for the map recursion to terminate in $O(\log N)$ steps, blocks must be big enough to hold at least two t values of size λ. If the block size is $\Omega(\lambda^2)$, we can also take advantage of the asymmetric block optimization from Stefanov et al. [25] to reduce the complexity to $O(\phi \cdot (\log^6 n + B \cdot \log N)$. Then, if additionally $B \in \Omega(\log^5 N)$, the total complexity is reduced to $O(\log N)$ per client.

6 Conclusion

We have presented the first techniques that allow multi-client ORAM, specifically secure against *fully malicious* servers. Our multi-client ORAMs are reasonably efficient with communication complexities as low as $O(\log N)$ per client. Future work will focus on efficiency improvements, including reducing worst-case complexity to sublinear in ϕ. Additionally, the question of whether tree-based constructions are more efficient than classical ones is not as clear in the multi-client setting as it is for a single client. Although tree ORAMs are more efficient for a number of parameter choices, they incur substantial overhead from using sub-ORAMs to hold tree metadata. This is not required for the classical constructions. Future research may focus on achieving a "pure" tree-based construction which does not depend on another ORAM as a subroutine. Finally, it may be interesting to investigate whether multiple clients can be supported with a more fine-grained access control, secure against fully malicious servers.

References

1. Blass, E.-O., Mayberry, T., Noubir, G.: Multi-client oblivious ram secure against malicious servers. Cryptology ePrint Archive, Report 2015/121 (2015). http://eprint.iacr.org/2015/121
2. Boyle, E., Chung, K.-M., Pass, R.: Oblivious parallel RAM and applications. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 175–204. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49099-0_7
3. Cash, D., Küpçü, A., Wichs, D.: Dynamic proofs of retrievability via oblivious RAM. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 279–295. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38348-9_17
4. Chan, T.-H.H., Shi, E.: Circuit OPRAM: a (somewhat) tight oblivious parallel RAM. Cryptology ePrint Archive, Report 2016/1084 (2016). http://eprint.iacr.org/2016/1084
5. Costan, V., Devadas, S.: Intel SGX explained. Technical report, Cryptology ePrint Archive, Report 2016/086 (2016). https://eprint.iacr.org/2016/086
6. Fletcher, C.W., Ren, L., Kwon, A., Van Dijk, M., Stefanov, E., Devadas, S., Tiny, O.: A Low-Latency, Low-Area Hardware ORAM Controller. Cryptology ePrint Archive, Report 2014/431 (2014). http://eprint.iacr.org/
7. Franz, M., Williams, P., Carbunar, B., Katzenbeisser, S., Peter, A., Sion, R., Sotakova, M.: Oblivious outsourced storage with delegation. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 127–140. Springer, Heidelberg (2012). doi:10.1007/978-3-642-27576-0_11
8. Goldreich, O.: Towards a theory of software protection and simulation by oblivious RAMs. In: Symposium on Theory of Computing, pp. 182–194, New York, USA (1987)

9. Goldreich, O., Ostrovsky, R.: Software protection, simulation on oblivious RAMs. J. ACM **43**(3), 431–473 (1996). ISSN 0004–5411
10. Goodrich, M.T.: Zig-zag sort: a simple deterministic data-oblivious sorting algorithm running in o(n log n) time. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing, STOC 2014, pp. 684–693 (2014). ISBN 978-1-4503-2710-7
11. Goodrich, M.T., Mitzenmacher, M., Ohrimenko, O., Tamassia, R.: Oblivious RAM simulation with efficient worst-case access overhead. In: Proceedings of Workshop on Cloud Computing Security Workshop, pp. 95–100, Chicago, USA (2011)
12. Goodrich, M.T., Mitzenmacher, M., Ohrimenko, O., Tamassia, R.: Privacy-preserving group data access via stateless oblivious RAM simulation. In: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 157–167 (2012)
13. Iliev, A., Smith, S.W.: Protecting client privacy with trusted computing at the server. IEEE Secur. Priv. **3**(2), 20–28 (2005)
14. Knuth, D.E.: The Art of Computer Programming, Seminumerical Algorithms, 2nd edn., vol. 2, chap. 3.4.2, pp. 139–140. Addison Wesley 1981. ISBN 978-0201896848
15. Kushilevitz, E., Lu, S., Ostrovsky, R.: On the (in) security of hash-based oblivious RAM and a new balancing scheme. In: Proceedings of Symposium on Discrete Algorithms, pp. 143–156, Kyoto, Japan (2012)
16. Li, J., Krohn, M.N., Mazières, D., Shasha, D.: Secure untrusted data repository (SUNDR). In: Proceedings of Operating System Design and Implementation, pp. 121–136, San Francisco, USA (2004)
17. Lorch, J.R., Parno, B., Mickens, J., Raykova, M., Schiffman, J.: Shroud: ensuring private access to large-scale data in the data center. In: USENIX Conference on File and Storage Technologies, pp. 199–213 (2013)
18. Maffei, M., Malavolta, G., Reinert, M., Schröder, D.: Privacy and access control for outsourced personal records. In: IEEE Symposium on Security and Privacy, pp. 341–358. IEEE (2015)
19. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988). doi:10.1007/3-540-48184-2_32
20. Nayak, K., Katz, J.: An oblivious parallel RAM with $o(\log^2 n)$ parallel runtime blowup. Cryptology ePrint Archive, Report 2016/1141 (2016). http://eprint.iacr.org/2016/1141
21. Ostrovsky, R., Shoup, V.: Private information storage. In: Proceedings of Symposium on Theory of Computing, pp. 294–303. ACM (1997)
22. Ren, L., Fletcher, C.W., Yu, X., van Dijk, M., Devadas, S.: Integrity verification for path oblivious-RAM. In: Proceedings of High Performance Extreme Computing Conference, pp. 1–6, Waltham, USA (2013)
23. Shi, E., Chan, T.-H.H., Stefanov, E., Li, M.: Oblivious RAM with $O((\log N)^3)$ worst-case cost. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 197–214. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25385-0_11
24. SpiderOak. Semaphor (2016). https://spideroak.com/solutions/semaphor
25. Stefanov, E., van Dijk, M., Shi, E., Fletcher, C., Ren, L., Yu, X., Devadas, S., Path, O.: An extremely simple oblivious RAM protocol. In: Proceedings of Conference on Computer & Communications Security, pp. 299–310, Berlin, Germany (2013). ISBN 978-1-4503-2477-9
26. WhatsApp. Whatsapp encryption overview (2016). https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf

Author Index