

Liqun Chen  
Mark Manulis  
Steve Schneider (Eds.)

LNCS 11060

# Information Security

21st International Conference, ISC 2018  
Guildford, UK, September 9–12, 2018  
Proceedings

 Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, Lancaster, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Friedemann Mattern

*ETH Zurich, Zurich, Switzerland*

John C. Mitchell

*Stanford University, Stanford, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

C. Pandu Rangan

*Indian Institute of Technology Madras, Chennai, India*

Bernhard Steffen

*TU Dortmund University, Dortmund, Germany*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Gerhard Weikum

*Max Planck Institute for Informatics, Saarbrücken, Germany*

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

Liqun Chen · Mark Manulis  
Steve Schneider (Eds.)


# Information Security


21st International Conference, ISC 2018  
Guildford, UK, September 9–12, 2018  
Proceedings



*Editors*

Liqun Chen   
University of Surrey  
Guildford  
UK

Steve Schneider   
University of Surrey  
Guildford  
UK

Mark Manulis   
University of Surrey  
Guildford  
UK

ISSN 0302-9743                      ISSN 1611-3349 (electronic)  
Lecture Notes in Computer Science  
ISBN 978-3-319-99135-1              ISBN 978-3-319-99136-8 (eBook)  
<https://doi.org/10.1007/978-3-319-99136-8>

Library of Congress Control Number: 2018950931

LNCS Sublibrary: SL4 – Security and Cryptology

© Springer Nature Switzerland AG 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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

# Preface

The 21st Information Security Conference, ISC 2018, took place September 9–12, 2018, in Guildford, UK, and was organized by the Surrey Centre for Cyber Security (SCCS) at the University of Surrey.

ISC is an annual conference focusing on original research in cyber security, applied cryptography, and privacy. Both academic research with high relevance to real-world problems and developments in industrial and technical frontiers fall within the scope of the conference.

ISC 2018 received 59 submissions, which were reviewed by the Program Committee. Each of the 46 Program Committee members was assigned an average of four submissions for review. Each paper was assigned to at least three reviewers. The Program Committee was helped by the reports and opinions of 54 external reviewers. The submission process was not anonymous and author names were visible to all reviewers. The review process was organized and managed through EasyChair. The reviewers were asked to declare any conflicts of interest for all submissions in the beginning of the process. The selection process was competitive and after highly interactive discussions and a careful deliberation, 26 papers were selected by the Program Committee for presentation at the conference.

The invited talks at ISC 2018 were given by Jan Camenisch from IBM Research Zurich and Aggelos Kiayias from the University of Edinburgh. Invited speakers were offered the opportunity to publish an invited paper in the conference proceedings. The prize for the Best Paper was awarded to Masahito Ishizaka and Kanta Matsuura for their paper “Strongly Unforgeable Signature Resilient to Polynomially Hard-to-Invert Leakage Under Standard Assumptions.”

ISC 2018 was organized by Liqun Chen and Mark Manulis, who served as program chairs, selected the Program Committee, and led their efforts in selecting papers that you will find in this volume, and by Steve Schneider, who served as general chair and was helped in local organization by Ioana Boureanu and Kaitai Liang. ISC 2018 received generous sponsorship from Springer.

The ISC 2018 chairs would like to thank everyone who contributed to the success of the conference. We are grateful to the Program Committee and external reviewers for their commitment, hard work and enthusiasm, to ensure that each paper received a thorough and fair review. Last but not least, we wish to thank all conference participants for making ISC 2018 an enjoyable experience.

September 2018

Liqun Chen  
Mark Manulis  
Steve Schneider

# Organization

## Program Committee

Jean-Philippe Aumasson	Kudelski Security, Switzerland
Paulo Barreto	University of Washington, USA
Marina Blanton	University at Buffalo, USA
Ioana Boureanu	University of Surrey, UK
Colin Boyd	Norwegian University of Science and Technology, Norway
Liqun Chen	University of Surrey, UK
Sherman S. M. Chow	The Chinese University of Hong Kong, SAR China
Mauro Conti	University of Padua, Italy
James H. Davenport	University of Bath, UK
Lucas Davi	University of Duisburg-Essen, Germany
Alexandra Dmitrienko	University of Würzburg, Germany
Josep Domingo-Ferrer	Universitat Rovira i Virgili, Spain
François Dupressoir	University of Surrey, UK
Thomas Espitau	Sorbonne Université, UPMC LiP6, France
Nils Fleischhacker	Johns Hopkins University/Carnegie Mellon University, USA
Danilo Gligoroski	Norwegian University of Science and Technology, Norway
Tibor Jager	Paderborn University, Germany
Stefan Katzenbeisser	TU Darmstadt, Germany
Franziskus Kiefer	Mozilla, Germany
Xuejia Lai	Shanghai Jiao Tong University, China
Shujun Li	University of Kent, UK
Joseph Liu	Monash University, Australia
Javier Lopez	University of Malaga, Spain
Masahiro Mambo	Kanazawa University, Japan
Mark Manulis	University of Surrey, UK
Catherine Meadows	US Naval Research Laboratory, USA
Florian Mendel	Infineon Technologies, Germany
Chris Mitchell	Royal Holloway, University of London, UK
Atsuko Miyaji	Japan Advanced Institute of Science and Technology, Japan
Maire O'Neill	Queen's University Belfast, UK
Yanbin Pan	AMSS, China
Andreas Peter	University of Twente, The Netherlands
Duong Hieu Phan	Limoges University, France

Bertram Poettering	Royal Holloway, University of London, UK
Jeyavijayan Rajendran	University of Texas at Dallas, USA
Mark Ryan	University of Birmingham, UK
Peter Y. A. Ryan	University of Luxembourg, Luxembourg
Peter Scholl	Aarhus University, Denmark
Joerg Schwenk	Ruhr University Bochum, Germany
Luisa Siniscalchi	University of Salerno, Italy
Willy Susilo	University of Wollongong, Australia
Melanie Volkamer	Karlsruhe Institute of Technology, Germany
Ding Wang	Peking University, China
Lei Wang	Shanghai Jiao Tong University, China
Qian Wang	Wuhan University, China
Jiaying Zhou	Singapore University of Technology and Design, Singapore

### **Additional Reviewers**

Anglès-Tafalla, Carles	Li, Juanru
Bamiloshin, Michael	Losioux, Eleonora
Bemmann, Pascal	Luo, Yiyuan
Bernieri, Giuseppe	Nguyen, Khoa
Cazorla, Lorena	Ning, Jianting
Chen, Rongmao	Omote, Kazumasa
Chow, Yang-Wai	Reinheimer, Benjamin Maximilian
Chvojka, Peter	Ribes-González, Jordi
Ciampi, Michele	Rodler, Michael
Ding, Ning	Rubio, Juan E.
Felsch, Dennis	Shojafar, Mohammad
Fernandez, Carmen	Silva, Javier
Fu, Ximing	Spreitzer, Raphael
Gao, Fei	Surminski, Sebastian
Hagen, Christoph	Takano, Yuuki
Hupperich, Thomas	Yan, Hailun
Kakvi, Saqib A.	Zhao, Yongjun
Kulyk, Oksana	Zhong, Jingli
Lauer, Sebastian	Zollinger, Marie-Laure

# Contents

## Invited Paper

- Relaxed Lattice-Based Signatures with Short Zero-Knowledge Proofs . . . . . 3  
*Cecilia Boschini, Jan Camenisch, and Gregory Neven*

## Software Security

- Secure Code Execution: A Generic PUF-Driven System Architecture . . . . . 25  
*Stephan Kleber, Florian Unterstein, Matthias Hiller, Frank Slomka, Matthias Matousek, Frank Kargl, and Christoph Bösch*

- Lumus*: Dynamically Uncovering Evasive Android Applications . . . . . 47  
*Vitor Afonso, Anatoli Kalysch, Tilo Müller, Daniela Oliveira, André Grégio, and Paulo Lício de Geus*

- ICUFuzzer: Fuzzing ICU Library for Exploitable Bugs  
in Multiple Software . . . . . 67  
*Kun Yang, Yuan Deng, Chao Zhang, Jianwei Zhuge, and Haixin Duan*

- How Safe Is Safety Number? A User Study on SIGNAL’s Fingerprint  
and Safety Number Methods for Public Key Verification . . . . . 85  
*Kemal Bicakci, Enes Altuncu, Muhammet Sakir Sahkulubey, Hakan Ezgi Kiziloz, and Yusuf Uzunay*

## Symmetric Ciphers and Cryptanalysis

- Speeding up MILP Aided Differential Characteristic Search  
with Matsui’s Strategy . . . . . 101  
*Yingjie Zhang, Siwei Sun, Jiahao Cai, and Lei Hu*

- Automatic Search for Related-Key Differential Trails in SIMON-like Block  
Ciphers Based on MILP. . . . . 116  
*Xuzi Wang, Baofeng Wu, Lin Hou, and Dongdai Lin*

- Linear Cryptanalysis of Reduced-Round Speck with a Heuristic Approach:  
Automatic Search for Linear Trails . . . . . 132  
*Daniël Bodden*

- Conditional Cube Searching and Applications on Trivium-Variant Ciphers. . . 151  
*Xiaojuan Zhang, Meicheng Liu, and Dongdai Lin*

**Data Privacy and Anonymization**

Practical Attacks on Relational Databases  
 Protected via Searchable Encryption . . . . . 171  
*Mohamed Ahmed Abdelraheem, Tobias Andersson,  
 Christian Gehrman, and Cornelius Glackin*

A Simple Algorithm for Estimating Distribution Parameters  
 from  $n$ -Dimensional Randomized Binary Responses . . . . . 192  
*Staal A. Vinterbo*

**Outsourcing and Assisted Computing**

Enforcing Access Controls for the Cryptographic Cloud Service Invocation  
 Based on Virtual Machine Introspection. . . . . 213  
*Fangjie Jiang, Quanwei Cai, Le Guan, and Jingqiang Lin*

Multi-authority Fast Data Cloud-Outsourcing for Mobile Devices . . . . . 231  
*Yanting Zhang, Jianwei Liu, Zongyang Zhang, and Yang Hu*

Hide the Modulus: A Secure Non-Interactive Fully Verifiable Delegation  
 Scheme for Modular Exponentiations via CRT . . . . . 250  
*Osmanbey Uzunkol, Jothi Rangasamy, and Lakshmi Kuppusamy*

Offline Assisted Group Key Exchange. . . . . 268  
*Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang*

**Advanced Encryption**

Function-Dependent Commitments for Verifiable  
 Multi-party Computation . . . . . 289  
*Lucas Schabhüser, Denis Butin, Denise Demirel,  
 and Johannes Buchmann*

On Constructing Pairing-Free Identity-Based Encryptions. . . . . 308  
*Xin Wang, Bei Liang, Shimin Li, and Rui Xue*

Multi-key Homomorphic Proxy Re-Encryption . . . . . 328  
*Satoshi Yasuda, Yoshihiro Koseki, Ryo Hiromasa, and Yutaka Kawai*

Verifiable Decryption for Fully Homomorphic Encryption . . . . . 347  
*Fucaí Luo and Kunpeng Wang*

**Privacy-Preserving Applications**

Platform-Independent Secure Blockchain-Based Voting System . . . . . 369  
*Bin Yu, Joseph K. Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld,  
 Paul Rimba, and Man Ho Au*

Privacy in Crowdsourcing: A Systematic Review . . . . . 387  
*Abdulwhab Alkharashi and Karen Renaud*

**Advanced Signatures**

Anonymous yet Traceable Strong Designated Verifier Signature . . . . . 403  
*Veronika Kuchta, Rajeev Anand Sahu, Vishal Saraswat,  
 Gaurav Sharma, Neetu Sharma, and Olivier Markowitch*

Strongly Unforgeable Signature Resilient to Polynomially Hard-to-Invert  
 Leakage Under Standard Assumptions . . . . . 422  
*Masahito Ishizaka and Kanta Matsuura*

A Revocable Group Signature Scheme with Scalability from Simple  
 Assumptions and Its Implementation . . . . . 442  
*Keita Emura and Takuya Hayashi*

**Network Security**

Fast Flux Service Network Detection via Data Mining  
 on Passive DNS Traffic . . . . . 463  
*Pierangelo Lombardo, Salvatore Saeli, Federica Bisio,  
 Davide Bernardi, and Danilo Massa*

Beyond Cookie Monster Amnesia: Real World Persistent Online Tracking. . . 481  
*Nasser Mohammed Al-Fannah, Wanpeng Li, and Chris J. Mitchell*

Cyber-Risks in the Industrial Internet of Things (IIoT): Towards a Method  
 for Continuous Assessment . . . . . 502  
*Carolina Adaros Boye, Paul Kearney, and Mark Josephs*

**Author Index** . . . . . 521

# **Invited Paper**





# Relaxed Lattice-Based Signatures with Short Zero-Knowledge Proofs

Cecilia Boschini<sup>1,2</sup>, Jan Camenisch<sup>1(✉)</sup>, and Gregory Neven<sup>1</sup>

<sup>1</sup> IBM Research - Zurich, Zurich, Switzerland  
{bos,jca,nev}@zurich.ibm.com

<sup>2</sup> Università della Svizzera Italiana, Lugano, Switzerland

**Abstract.** Advanced cryptographic protocols such as anonymous credentials, voting schemes, and e-cash are typically constructed by suitably combining signature, commitment, and encryption schemes with zero-knowledge proofs. Indeed, a large body of protocols have been constructed in that manner from Camenisch-Lysyanskaya signatures and generalized Schnorr proofs. In this paper, we build a similar framework for lattice-based schemes by presenting a signature and commitment scheme that are compatible with Lyubashevsky’s Fiat-Shamir proofs with abort, currently the most efficient zero-knowledge proofs for lattices. The latter proofs provide a weaker, relaxed form of soundness, i.e., the witnesses that the knowledge extractor can obtain are guaranteed to lie only in a domain that is larger than the one from which the inputs of honest provers need to come. To cope with this soundness problem, we define corresponding notions of relaxed signature and commitment schemes. We demonstrate the flexibility and efficiency of our new primitives by constructing a new lattice-based anonymous attribute token scheme and providing concrete parameters to securely instantiate this scheme.

## 1 Introduction

An established and successful way to construct privacy-enhancing cryptographic protocols is to suitably combine various primitives such as signatures, commitments, and encryption schemes with efficient zero-knowledge proofs. Examples of such constructions include blind signatures [2, 33], group signatures [9, 39], direct anonymous attestation [16], electronic cash [26], voting schemes [38], adaptive oblivious transfer [17, 24], and anonymous credentials [8, 21].

A crucial building block for such solutions is a signature scheme enabling efficient zero-knowledge proofs of knowledge of a signature on a hidden message. Commitment schemes are also a common ingredient, either as “glue” to bridge zero-knowledge proofs over different cryptographic primitives [20], or to facilitate zero-knowledge proofs by hiding the message or certain components of the signature [5, 12, 22].

One can of course use generic zero-knowledge techniques [36] to combine cryptographic schemes, but to get truly efficient constructions, one needs schemes

that interact well with each other and allow for efficient zero-knowledge proofs. A well known set of such schemes consist of Camenisch-Lysyanskaya signatures [22], Damgård-Fujisaki commitments [30], and Camenisch-Shoup verifiable encryption [25]. They can be combined using generalized Schnorr proofs [19] and the Fiat-Shamir transform [32] into efficient proofs of relations between their (committed) inputs and (committed) outputs. More recently, an alternative set of primitives has emerged, so-called structure preserving primitives [1, 18], that use Groth-Sahai proofs [37] as a framework to create zero-knowledge proofs.

All of the above schemes, however, are based on hardness assumptions related to factoring large integers and computing discrete logarithms, which are known to succumb to attackers with quantum computers. To guarantee security on the long term, it would be best to switch to quantum-resistant problems such as the difficulty of computing shortest vectors in lattices. Indeed, a number of cryptographic primitives whose security relies on lattice-based assumptions have been proposed, including efficient signature schemes. However, almost all of these schemes do not lend themselves to efficient zero-knowledge proofs, usually because they make use of hash functions or have other properties that do prevent efficient proof protocols. The only exception is the lattice-based signature scheme proposed by Libert et al. [42], who specifically designed this signature scheme to be able to use it as a protocol building block. Unfortunately, their scheme relies on Stern-type zero-knowledge proofs [51] which are proofs with ternary challenges and hence need to be repeated sufficiently many times, resulting in a considerable efficiency penalty.

The issue of small challenges could in principle be overcome by using Lyubashevsky’s “Fiat-Shamir with Aborts” technique [46] which yields much more efficient proofs because it can use large challenges. However, this approach will not work for the scheme by Libert et al., because these proofs have the disadvantage that they are “relaxed,” in the sense that extracted witnesses are only guaranteed to lie in a considerably larger domain than the witnesses used to construct the proof. Thus, if such proofs are for instance used to prove knowledge of a valid message-signature pair, where the message and signature are witnesses, the extracted values will typically *not* be a valid message-signature pair.

## 1.1 Our Results

In this paper, we provide a signature and a commitment scheme that are tailored to deal with the relaxed witnesses extracted from Lyubashevsky’s zero-knowledge proofs. To this end, we first define “relaxed” signature and commitment schemes, in the sense that the respective verification algorithms accept messages, signatures, and openings, respectively, that are never output by the honest signing or committing algorithms. By allowing exactly the increase in the domain of the witnesses induced by the relaxed extraction, and by proving that our schemes remain secure under a suitably adapted notion in spite of that increase, we obtain two lattice-based primitives with efficient and securely composable zero-knowledge proofs.

We demonstrate the use of our signature and commitment schemes in the construction of privacy-enhancing technologies by building an anonymous attribute token (AAT) scheme [23]. An AAT scheme enables users to obtain credentials on multiple attributes, so that they can later selectively disclose these attributes to verifiers in an unlinkable fashion.

We suggest concrete parameter choices for our schemes that yield a secure yet efficient instantiation. We present two sets of parameters, one assuming the hardness of new interactive assumptions and one assuming the hardness of Ring-SIS and Ring-LWE and using a complexity leveraging argument (which blows up parameters). Even in our most conservative analysis, we obtain presentation token sizes less than 20 MB, which is well below the signature sizes or related lattice-based primitives [43]. In our least conservative analysis, assuming the hardness of two new interactive assumptions, we obtain presentation tokens as small as 3.42 MB, which make our scheme the most practical lattice-based AAT scheme.

## 1.2 Related Work

The only known lattice-based anonymous attribute token scheme [23] has presentation token sizes that are linear in the number of group members, and is therefore mainly a proof of concept. Our AAT scheme is the first that could be considered suitable for practical applications in a post-quantum world<sup>1</sup>.

Our proposal of lattice-based signature with protocols is not the first attempt to design efficient cryptographic building blocks. In a concurrent work, Libert et al. [42] presented a signature scheme with proofs based on a Stern-type ZK protocol. Moreover, there exists a line of work on lattice-based group signatures that combines signature schemes (usually variants of Boyen’s signature [14] or Böhl signature [11]) with non-interactive zero-knowledge (NIZK) protocols, usually either Stern-type NIZK protocols (cfr. [41, 42, 44]), or Lyubashevsky proofs [46] with single-bit challenges (cfr. [40, 49])<sup>2</sup>. The advantage of using these protocols is that it is possible to prove knowledge of a witness for the exact relation, thus it is not necessary to relax the verification algorithms. The drawback is that Stern-type protocols have soundness error of  $2/3$  and Lyubashevsky proofs with single-bit challenges of  $1/2$ , thus they require to be repeated a number of times that is linear in the security parameter to have a negligible soundness error. This reflects in parameters choices and sizes: as it was already observed by Libert et al. [43], all the schemes proposed until now output signature of size greater than 61 MB.

---

<sup>1</sup> We do not claim ours to be the first practical AAT. In fact, an AAT scheme based on discrete log is at the core of Microsoft’s U-Prove [50].

<sup>2</sup> We do not consider in our comparison the lattice-based group signature built by Benhamouda et al. [10]. Indeed, it is a special case, as the authors avoided expensive zero-knowledge proofs on lattice signatures by bridging a lattice-based encryption scheme to a *non-lattice-based* signature scheme.

## 2 Preliminaries

If  $A$  is a probabilistic algorithm, then by  $A(x)$  we denote the output distribution of  $A$  on input  $x$  and run with uniformly chosen random coins. Computing  $y$  with  $A$  on input  $x$  amounts to choose  $y$  from the distribution  $A(x)$ , denoted by  $y \stackrel{\$}{\leftarrow} A(x)$ . We write  $y \in A(x)$  if the probability that  $A(x)$  will output  $y$  is non-zero. We use  $A^{\mathcal{H}}$  to denote the fact that  $A$  has oracle access to the function  $\mathcal{H}$ . A function  $\nu(n)$  is said to be *negligible* if  $\nu(n) \leq \frac{1}{p(n)}$  for any polynomial  $p(n)$  and sufficiently large  $n$ . Throughout the paper we denote by  $\lambda$  the security parameter of a scheme.

Let  $\mathcal{R}_q = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$  be a polynomial ring for a prime  $q$ . Operations are the usual addition and multiplication modulo  $q$  and  $\mathbf{x}^n + 1$ . An element of  $\mathcal{R}_q$  is a polynomial  $\mathbf{a} = \sum_{i=0}^{n-1} a_i \mathbf{x}^i$ , where  $a_i \in \{-(q-1)/2, \dots, (q-1)/2\}$ . A matrix in  $\mathcal{R}_q^{n \times m}$  will be denoted by bold upper-case letters. We define the following norms on the set of polynomials:  $\|\mathbf{a}\|_1 = \sum_{i=0}^{n-1} |a_i|$ ,  $\|\mathbf{a}\|_\infty = \max_i |a_i|$  and  $\|\mathbf{a}\| = \sqrt{\sum_{i=0}^{n-1} a_i^2}$ . A *small* element of the ring will be a polynomial in  $\mathcal{R}_q$  with small coefficient w.r.t. one of these norms depending on the context. With  $\mathbf{b} \leftarrow \mathcal{R}_q$  we will mean that the polynomial  $\mathbf{b}$  is sampled uniformly at random from  $\mathcal{R}_q$ . For two matrices (or vectors)  $\mathbf{A}$  and  $\mathbf{B}$ , we will denote by  $[\mathbf{A} \mid \mathbf{B}]$  their horizontal concatenation and with  $[\mathbf{A}; \mathbf{B}]$  their vertical concatenation. With  $\mathbf{1}_m$  we will indicate the vector of length  $m$  whose components are equal to  $\mathbf{1}$ ,  $\mathbf{0}_{m_1 \times m_2}$  (resp.,  $\mathbf{0}_m$ ) will be the zero matrix (resp., vector) of dimension  $m_1 \times m_2$  (resp.,  $m$ ) and  $\mathbb{I}_m$  the identity matrix of dimension  $m$ . The norms of a vector  $\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_k]$  are defined as  $\|\mathbf{V}\|_\infty = \max_i \|\mathbf{v}_i\|_\infty$  and  $\|\mathbf{V}\| = \sqrt{\sum_i \|\mathbf{v}_i\|^2}$ . The norm of the sum or product of small polynomials can be bound as it follows.

**Lemma 1.** *Let  $\mathbf{a}, \mathbf{b} \in \mathcal{R}_q$  be such that  $n\|\mathbf{a}\|_\infty \cdot \|\mathbf{b}\|_\infty \leq (q-1)/2$ . Then we have that  $\|\mathbf{ab}\| \leq \|\mathbf{a}\| \|\mathbf{b}\| \sqrt{n}$  and  $\|\mathbf{ab}\|_\infty \leq \|\mathbf{a}\|_\infty \|\mathbf{b}\|_\infty n \leq \frac{q-1}{2}$ .*

The proof is straightforward and it can be found in the full version [13].

With  $\mathcal{R}_3$  we denote the ring of polynomials with coefficients in  $\mathbb{Z}_3 = \{0, \pm 1\}$ . Throughout the paper we will consider these element as also element of the subset of  $\mathcal{R}_q$  of polynomials with coefficients in  $\{\pm 1, 0\}$  using a standard mapping. For any  $K|n$  we can construct a subring  $\mathcal{R}_q^{(K)}$  of  $\mathcal{R}_q$  as the subset of elements  $\mathbf{a} \in \mathcal{R}_q$  such that  $\mathbf{a} = a_0 + a_1 \mathbf{x}^{n/K} + a_2 \mathbf{x}^{2n/K} + \dots + a_{K-1} \mathbf{x}^{(K-1)n/K}$ . For integer  $p$ , we denote by  $\mathcal{R}_p$  (resp.,  $\mathcal{R}_p^{(K)}$ ) the subset of  $\mathcal{R}_q$  (resp.,  $\mathcal{R}_q^{(K)}$ ) that contains polynomials with coefficients in  $[-(p-1)/2, (p-1)/2]$ .

Among others, the choice of  $q$  strongly influences the number of invertible elements that can be found in the ring. In particular, if  $q$  is such that  $q \equiv 5 \pmod{8}$ , so that all the elements with small enough coefficient (i.e., all  $\mathbf{a} \in \mathcal{R}_q$  such that  $\|\mathbf{a}\|_\infty < \sqrt{q/2}$ ) are guaranteed to be invertible [47, Lemma 2.2]. We denote by  $\text{Inv}(\mathcal{R}_q)$  the set of all the invertible polynomials in  $\mathcal{R}_q$ .

## 2.1 Polynomial Lattices

A  $m$ -dimensional polynomial lattice is an additive subgroup of  $\mathcal{R}_q$ , where a basis is a vector  $\mathbf{B} \in \mathcal{R}_q^{1 \times m}$ . Given a vector  $\mathbf{A} \in \mathcal{R}_q^{1 \times m}$  we define the  $m$ -dimensional lattice  $\mathcal{L}^\perp(\mathbf{A})$  as  $\Lambda^\perp = \mathcal{L}^\perp(\mathbf{A}) = \{\mathbf{V} \in (\mathbb{Z}[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle)^m \mid \mathbf{A}\mathbf{V} = \mathbf{0} \pmod{q}\} \subseteq \mathcal{R}_q^m$ . Given  $\mathbf{a} \in \mathcal{R}_q$ , a *coset*  $\Lambda + \mathbf{a}$  of a lattice  $\Lambda$  is the set  $\{\mathbf{a} + \mathbf{v}\}_{\mathbf{v} \in \Lambda}$ . Consider the obvious embedding that maps a polynomial to the vector of its coefficients. Then  $\Lambda^\perp$  can be also seen as a  $nm$ -dimensional integer lattice over  $\mathbb{Z}$ . To generate a discrete Gaussian sample, we can generate a sample over  $\mathbb{Z}^n$  and then map it into  $\mathcal{R}_q$  using the obvious embedding of coordinates into coefficients of the polynomials. With a slight abuse of notation, we will write  $\mathbf{y} \stackrel{\$}{\leftarrow} \mathcal{D}_{\mathcal{R}_q, \mathbf{u}, \sigma}$  to indicate that  $\mathbf{y}$  was sampled from  $\mathcal{D}_{\mathbb{Z}^n, \mathbf{u}, \sigma}$  and then mapped to  $\mathcal{R}_q$ . Similarly, we omit the  $\mathbf{0}$  and write  $(\mathbf{y}_1, \dots, \mathbf{y}_k) \stackrel{\$}{\leftarrow} \mathcal{D}_{\mathcal{R}_q, \sigma}^k$  to mean that a vector  $\mathbf{y}$  is generated according to  $\mathcal{D}_{\mathbb{Z}^{kn}, \mathbf{0}, \sigma}$  and then gets interpreted as  $k$  polynomials  $\mathbf{y}_i$ . Elements sampled from such distribution, have norm bounded by the following lemma.

**Lemma 2 (adaptation of Lemma 1.5 in [7] and Lemma 4.4 in [46]).** *Let  $\mathbf{A} \in \mathcal{R}_q^{1 \times m}$  with  $2^{11} < m$  and  $\mathbf{u} \in \mathcal{R}_q$ . For  $\sigma \geq \tilde{\lambda}(\mathcal{L}^\perp(\mathbf{A}))$  it holds:*

1.  $\Pr_{\mathbf{s} \stackrel{\$}{\leftarrow} \mathcal{D}_{\Lambda^\perp, \mathbf{u}, \sigma}} (\|\mathbf{s}\| > 1.05\sigma\sqrt{mn}) < 2^{-5}$ .
2.  $\Pr_{\mathbf{s} \stackrel{\$}{\leftarrow} \mathcal{D}_{\Lambda^\perp, \mathbf{u}, \sigma}} (\|\mathbf{s}\|_\infty > 8\sigma) < mn2^{-25}$ .

*In particular, the inequalities hold also when  $\mathbf{s} \leftarrow \mathcal{D}_{\mathcal{R}_q^m, \sigma}$ .*

Observe that it is enough that the bound holds with non-negligible probability, as each time we sample we can check the norm of the vector and discard it if the norm is too large.

We define the largest singular value, a quantity that is used to measure the geometric quality of a lattice basis. If  $\mathbf{R} \in \mathcal{R}_q^{k \times m}$ , then  $s_1(\mathbf{R}) = \max_{\mathbf{u} \in \mathcal{R}_q^m} \frac{\|\mathbf{R}\mathbf{u}\|}{\|\mathbf{u}\|}$ . It holds that  $\|\mathbf{R}\mathbf{u}\| \leq s_1(\mathbf{R})\|\mathbf{u}\|$  for every  $\mathbf{R} \in \mathcal{R}_q^{1 \times m}$  and  $\mathbf{u} \in \mathcal{R}_q$ . The following Theorem from [48] shows how a (pseudo-)random vector  $\mathbf{U}$ , for which no trapdoor is known, can be extended into a pseudo-random vector  $[\mathbf{U}|\mathbf{V}]$  for which we will be able to sample from  $\mathcal{D}_{[\mathbf{U}|\mathbf{V}+\mathbf{mG}], \mathbf{u}, \sigma}^\perp$  for any invertible  $\mathbf{m}$  and for some standard deviation  $\sigma$ .

**Theorem 1 (adapted from [48]).** *Let  $\mathbf{A}$  be a vector in  $\mathcal{R}_q^{1 \times \ell}$  and  $\mathbf{X}$  be a matrix in  $\mathcal{R}_q^{\ell \times m}$ . Also define the gadget matrix  $\mathbf{G} = [1 \lceil q^{1/m} \rceil \dots \lceil q^{(m-1)/m} \rceil]$ . Then for any invertible  $\mathbf{m} \in \mathcal{R}_q$ , there is an algorithm that can sample from the distribution  $\mathcal{D}_{[\mathbf{A}|\mathbf{A}\mathbf{X}+\mathbf{mG}], \mathbf{u}, \sigma}^\perp$  for any  $\sigma \sim q^{\frac{1}{m}} s_1(\mathbf{X})$  for any  $\mathbf{u} \in \mathcal{R}_q$ .*

Lemma 3 is a combination of the double-trapdoor idea from Agrawal et al. [3], with the sampling procedure by Brakerski et al. [15].

**Lemma 3.** *Suppose  $\mathbf{U} \in \mathcal{R}_q^{1 \times k}$  and  $\mathbf{V} \in \mathcal{R}_q^{1 \times m}$  are polynomial vectors, and  $\mathbf{B}_U, \mathbf{B}_{(U,V)}$  are bases of  $\Lambda^\perp(\mathbf{U})$  and  $\Lambda^\perp([\mathbf{U}|\mathbf{V}])$  respectively such that  $\|\tilde{\mathbf{B}}_U\|, \|\tilde{\mathbf{B}}_{(U,V)}\| < \sigma\sqrt{\pi/\ln(2n+4)}$ . Then, there exists an algorithm  $\text{SampleD}(\mathbf{U}, \mathbf{V}, \mathbf{B}, \mathbf{u}, \sigma)$ , where  $\mathbf{B}$  is either  $\mathbf{B}_U$  or  $\mathbf{B}_{(U,V)}$ , that can efficiently sample from the distribution  $\mathcal{D}_{[\mathbf{U}|\mathbf{V}], \mathbf{u}, \sigma}^\perp$  for any  $\mathbf{u} \in \mathcal{R}_q$ .*

The security of our construction will be based on two well-studied lattice problems over rings:

**Definition 1** (Ring-SIS $_{m,q\beta}$  problem). *The Ring-SIS $_{m,q\beta}$  problem is given a uniformly distributed vector  $\mathbf{A} \in \mathcal{R}_q^{1 \times (m-1)}$  to find a vector  $\mathbf{S} \in \mathcal{R}_q^m$  such that  $[\mathbf{A}|\mathbf{1}]\mathbf{S} = \mathbf{0}$  and  $\|\mathbf{S}\| \leq \beta$ .*

**Definition 2.** *The Ring-LWE $_{D,s}$  distribution outputs pairs  $(\mathbf{a}, \mathbf{b}) \in \mathcal{R}_q \times \mathcal{R}_q$  such that  $\mathbf{b} = \mathbf{a}\mathbf{s} + \mathbf{e}$  for fixed  $\mathbf{s} \in \mathcal{R}_q$ , a uniformly random  $\mathbf{a}$  from  $\mathcal{R}_q$  and  $\mathbf{e}$  sampled from distribution  $D$ .*

*The Ring-LWE $_{k,D}$  decisional problem on ring  $\mathcal{R}_q$  with distribution  $D$  is to distinguish whether  $k$  pairs  $(\mathbf{a}_1, \mathbf{b}_1), \dots, (\mathbf{a}_k, \mathbf{b}_k)$  were sampled from the Ring-LWE $_D$  distribution or from the uniform distribution over  $\mathcal{R}_q^2$ .*

We use the root Hermite factor  $\delta$  introduced by Gama and Nguyen [34] to estimate the hardness for given parameters of the lattice problems in the security reductions. We will deduct the number of bits of security from it using the worst-case analysis by Akim et al. (cfr. Sect. 6 in [4]).

### 3 Relaxed Zero-Knowledge Proofs over Lattices

The first building blocks we need for our construction are *relaxed  $\Sigma$ -protocols* and *relaxed non-interactive zero-knowledge proofs of knowledge*, where the relaxed soundness definition guarantees the extraction of a witness from a wider language than the one used by an honest prover. Proofs with relaxed extraction notions have been used implicitly in previous work, e.g., for schemes based on discrete logarithms in group of unknown order [19, 22, 25, 52] and some lattice-based schemes [46, 47]. We give a simpler definition here that suffices for the lattice-based protocols that we consider.

#### 3.1 Definition of Relaxed Zero-Knowledge Proofs

Let  $L \subseteq \{0, 1\}^*$  be a language with witness relation  $R$ , meaning  $x \in L \Leftrightarrow \exists w : (x, w) \in R$ . Let  $\bar{L} \supseteq L$  be a relaxed language with witness relation  $\bar{R} \supseteq R$ . We define *relaxed  $\Sigma$ -protocols* inspired by the original definitions by Cramer [28] and Faust et al. [31], but with a relaxed soundness condition that guarantees the extraction of a witness from  $\bar{R}$  rather than  $R$  (similar to Camenisch et al. [19]).

**Definition 3 (Relaxed  $\Sigma$ -protocols).** *A relaxed  $\Sigma$ -protocol  $\Sigma = (\mathcal{P}, \mathcal{V})$  for relations  $(R, \bar{R})$  is a three-round public-coin interactive proof system where  $\mathcal{P} = (\mathcal{P}_0, \mathcal{P}_1)$  and  $\mathcal{V} = (\mathcal{V}_0, \mathcal{V}_1)$  are couples of PPT algorithms that, on top of the standard correctness and honest-verifier zero-knowledge (HVZK) properties, satisfy the following property:*

**Relaxed special soundness.** *There exists an efficient algorithm  $E$ , called special extractor, that given two accepting conversations  $(\alpha, \beta, \gamma)$  and  $(\alpha, \beta', \gamma')$  for language member  $\bar{x} \in \bar{L}$  where  $\beta \neq \beta'$ , computes  $\bar{w} \leftarrow E(\bar{x}, \alpha, \beta, \gamma, \beta', \gamma')$  such that  $(\bar{x}, \bar{w}) \in \bar{R}$ .*

Relaxed  $\Sigma$ -protocols are relaxed proofs of knowledge, as the knowledge extractor extracts from  $\mathcal{P}$  a pair  $(x, w)$  in  $\bar{R}$  (the proof is a straightforward adaptation to relaxed protocols of the proof of Theorem 1 in [29]).

### 3.2 A Relaxed $\Sigma$ -Protocol for Proving Linear Relations

We rephrase Lyubashevsky’s “Fiat-Shamir with aborts” technique [45, 46] as a relaxed  $\Sigma$ -protocol for the languages  $(L, \bar{L})$  associated to the following relations:

$$\begin{aligned} R &= \left\{ ((\mathbf{A}, \mathbf{U}), (\mathbf{S}, \mathbf{1})) \in \mathcal{R}_q^{\ell \times m} \times \mathcal{R}_q^{1 \times \ell} \times \mathcal{R}_q^m \times \{\mathbf{1}\} : \mathbf{A}\mathbf{S} = \mathbf{U}, \|\mathbf{S}\| \leq N, \|\mathbf{S}\|_\infty < (q-1)/(2n) \right\} \\ \bar{R} &= \left\{ ((\mathbf{A}, \mathbf{U}), (\bar{\mathbf{S}}, \bar{\mathbf{c}})) \in \mathcal{R}_q^{\ell \times m} \times \mathcal{R}_q^{1 \times \ell} \times \mathcal{R}_q^m \times \bar{\mathcal{C}} : \mathbf{A}\bar{\mathbf{S}} = \bar{\mathbf{c}}\mathbf{U}, \|\bar{\mathbf{S}}\| \leq \bar{N}, \|\bar{\mathbf{S}}\|_\infty \leq \bar{N}_\infty \right\} \end{aligned}$$

for some positive constants  $N$ ,  $\bar{N}$ , and  $\bar{N}_\infty$ , with  $N \leq \bar{N}$ , and a challenge set  $\mathcal{C} \subseteq \mathcal{R}_3^{(2^{K_c})}$  and a set of relaxed challenges  $\bar{\mathcal{C}} \subseteq \mathcal{R}_5^{(2^{K_c})}$ ,  $K_c > 0$ . Let  $C$  (resp.  $\bar{C}$ ) be a bound on  $\|\mathbf{c}\|$  for  $\mathbf{c} \in \mathcal{C}$  (resp.  $\bar{\mathbf{c}} \in \bar{\mathcal{C}}$ ). Finding a witness  $(\bar{\mathbf{S}}, \bar{\mathbf{c}})$  for an element  $(\mathbf{A}, \mathbf{U})$  of the language  $\bar{L}$  is hard under the computational assumption that Ring-SIS $_\beta$  is hard, where  $\beta = \sqrt{(\bar{N}^2 + \bar{C}^2)}$ .

The relaxed  $\Sigma$ -protocol  $(\mathcal{P}, \mathcal{V})$  for  $(R, \bar{R})$  works as follows.

1. First, the prover  $\mathcal{P}$  samples a masking vector  $\mathbf{Y} \stackrel{\$}{\leftarrow} \mathcal{D}_\sigma^m$  (we will determine the value of  $\sigma$  in a moment), and sends  $\mathbf{T} = \mathbf{A}\mathbf{Y}$  to  $\mathcal{V}$ .
2. The verifier  $\mathcal{V}$  samples a challenge  $\mathbf{c} \in \mathcal{C}$  and sends it back to  $\mathcal{P}$ .
3. The prover then constructs  $\mathbf{Z} = \mathbf{Y} + \mathbf{c}\mathbf{S}$  and, depending on rejection sampling (see Theorem 4.6 in [46]), either aborts or sends it to  $\mathcal{V}$ .
4. The verifier accepts if  $\mathbf{A}\mathbf{Z} - \mathbf{c}\mathbf{U} = \mathbf{T}$  and  $\|\mathbf{Z}\| \leq 1.05\sigma\sqrt{nm} =: N_2$ ,  $\|\mathbf{Z}\|_\infty \leq 8\sigma =: N_\infty$ .

Now, observe that the zero-knowledge property is guaranteed by rejection sampling. A standard deviation  $\sigma = aT$ , where  $T = C \cdot N\sqrt{n}$  is a bound on the norm of  $\mathbf{c}\mathbf{S}$  obtained from Lemma 1, guarantees that the prover does not abort with probability at least  $(1 - 2^{100})/e^{12/a+1/(2a^2)}$  for any  $a > 0$  (cf. [46, Theorem 4.6]). Finally, we set  $\bar{N} = 2N = 2.1\sigma\sqrt{nm}$  and  $\bar{N}_\infty = 2N_\infty = 16\sigma$ .

One can see that our protocol is a relaxed  $\Sigma$ -protocol as follows. Correctness follows from Lemma 1.5 in [7]. Zero-knowledge follows from rejection sampling: a simulator  $\mathcal{S}$  can simply sample  $\mathbf{Z} \stackrel{\$}{\leftarrow} \mathcal{D}_{\mathcal{R}_q, \sigma}^m$ ,  $\mathbf{c} \stackrel{\$}{\leftarrow} \mathcal{C}$  and set  $\mathbf{T} := \mathbf{A}\mathbf{Z} - \mathbf{c}\mathbf{U}$ . Finally, special soundness is proved as usual by defining an extractor that runs  $\mathcal{P}$  twice on different challenges and output as response the difference of the responses. In the full version [13], we show how a relaxed  $\Sigma$ -protocol  $(\mathcal{P}, \mathcal{V})$  can be turned into a relaxed NIZK proof system  $(\mathcal{P}^{\mathcal{H}_c}, \mathcal{V}^{\mathcal{H}_c})$  using the Fiat-Shamir transform [32] and one-time signatures.

### 3.3 Proving Knowledge of Bounded-Degree Secrets in a Subring

In our construction of an anonymous attribute token scheme, we will use the above protocol in a modified form to let a prover prove knowledge of a  $[\mathbf{m}; \mathbf{s}]$

where  $\mathbf{m}$  is a small element in a subring  $\mathcal{R}_q^{(2^{K_m})}$  of  $\mathcal{R}_q$  and with degree  $\deg(\mathbf{m}) < d$  for some constant  $d < n$ . The fact that  $\mathbf{m}$  is in the subring can be proved by simply sampling challenges and the first component of the masking vector  $\mathbf{Y} = [\mathbf{y}_m; \mathbf{y}_s]$  from the subring. The result is that the first component of the response  $[\mathbf{z}_m; \mathbf{z}_s] = [\mathbf{y}_m; \mathbf{y}_s] + \mathbf{c}[\mathbf{m}; \mathbf{s}]$  is in the subring if  $\mathbf{m}$  is. The zero-knowledge property remains guaranteed by rejection sampling.

Proving that  $\mathbf{m}$  is of degree strictly less than  $d < n$  can be done by carefully choosing the challenge set and the domain of the masking vector. In particular, if  $\deg(\mathbf{m}) \leq d_m$  and challenges are chosen to be polynomials of degree  $d_c$  such that  $d_c + d_m < d$ , then  $\deg(\mathbf{m}\mathbf{c}) < d$ . Letting the prover sample the masking vector  $\mathbf{y}_m$  from the polynomials of degree less than  $d$  and applying rejection sampling as usual preserves the zero-knowledge property when computing  $\mathbf{z}_m = \mathbf{m}\mathbf{c} + \mathbf{y}_m$ . By letting the verifier additionally check that  $\deg(\mathbf{z}_m) < d$ , the extractor is guaranteed to be able to extract a witness  $\bar{\mathbf{m}} = \mathbf{z}_{m,1} - \mathbf{z}_{m,2}$  of degree strictly less than  $d$ . Note that sampling a discrete Gaussian distributions of polynomials of degree at most  $d - 1$  from the subring  $\mathcal{R}_q^{(2^K)}$  can be done by sampling from  $\mathcal{D}_{\mathbb{Z}^m, \sigma}$  for  $m = \lfloor (d - 1)n/2^K \rfloor$  and mapping coordinates to coefficients. To have a clearer notation, we define  $\mathcal{Y}_d$  to be the set of elements in the subring  $\mathcal{R}_q^{(2^K)}$  with degree at most  $d - 1$ , so that the distribution of the full masking vector  $\mathbf{Y}$  can be written as  $\mathcal{D}_{\mathcal{Y} \times \mathcal{R}_q, \sigma}$ .

## 4 A Relaxed Lattice-Based Commitment Scheme

Our second building block is a commitment scheme with an efficient proof of knowledge of a committed message that uses the  $\Sigma$ -protocol from Sect. 3. To compensate for relaxed extraction properties of the  $\Sigma$ -protocol, we define *relaxed commitments*, where the opening algorithm accepts messages and opening information that could not be committed to (respectively, produced) by the honest commitment algorithm.

### 4.1 Definition of Relaxed Commitments

A relaxed commitment scheme  $\mathcal{C}$  for message space  $\mathcal{U}$  and relaxed message space  $\bar{\mathcal{U}} \supseteq \mathcal{U}$  consists of a triple of algorithms  $(\text{ParGen}_c, \text{Commit}, \text{OpenVf})$ , where  $cpar \leftarrow \text{ParGen}_c(\mathcal{U}, 1^\lambda)$  generates the parameters on input the message space and the security parameter,  $(c, o) \leftarrow \text{Commit}(cpar, M)$  computes the commitment value  $c$  and the opening information  $o$  on input the parameters and a message in  $\mathcal{U}$ , and  $\{1, 0\} \leftarrow \text{OpenVf}(cpar, c, \bar{M}, \bar{o})$  verifies whether  $\bar{o}$  is an opening of  $\bar{M} \in \bar{\mathcal{U}}$  for the commitment  $c$ .

A commitment scheme must satisfy the standard correctness and hiding properties. The binding property is relaxed by considering message relaxation function  $f : \mathcal{U} \rightarrow 2^{\bar{\mathcal{U}}}$  and by considering only attacks where the adversary can open a commitment to two messages from different components of the partition defined by  $f$ .



**Definition 4 (Relaxed Binding).** *A relaxed commitment scheme  $\mathcal{C}$  is  $f$ -binding for a function  $f : \mathcal{U} \mapsto 2^{\mathcal{U}}$  if for all polynomial-time  $\mathbf{A}$*

$$\Pr \left[ \begin{array}{l} \text{OpenVf}(cpar, c, \bar{M}_0, \bar{o}_0) = 1 \\ \wedge \text{OpenVf}(cpar, c, \bar{M}_1, \bar{o}_1) = 1 \\ \wedge \nexists M \in \mathcal{U} : \{\bar{M}_0, \bar{M}_1\} \subseteq f(M) \end{array} : \begin{array}{l} cpar \leftarrow \text{ParGen}_c(\mathcal{U}, 1^\lambda), \\ (c, \bar{M}_0, \bar{o}_0, \bar{M}_1, \bar{o}_1) \leftarrow \mathbf{A}(cpar) \end{array} \right] \leq \nu(n).$$

## 4.2 Designing Appropriate Message and Challenge Spaces

Our goal is to create a commitment scheme where the relaxed  $\Sigma$ -protocol from Sect. 3 can be used to prove knowledge of a committed message, where the message and opening information are part of the witness. The problem with relaxed  $\Sigma$ -protocols is that they cannot guarantee the extraction of a valid witness for the original relation  $R$ , but only for the relaxed relation  $\bar{R}$ . The witnesses in the latter have larger norms and explicitly admit “small multiples”: if  $(\mathbf{S}, \mathbf{1})$  is a valid witness in  $R$  so that  $\mathbf{AS} = \mathbf{U}$ , then  $(\bar{\mathbf{S}} = \bar{\mathbf{c}}\mathbf{S}, \bar{\mathbf{c}})$  is a valid witness in  $\bar{R}$  so that  $\mathbf{A}\bar{\mathbf{S}} = \bar{\mathbf{c}}\mathbf{U}$ , where  $\bar{\mathbf{c}} \in \bar{\mathcal{C}} = \{\mathbf{c} - \mathbf{c}' : \mathbf{c}, \mathbf{c}' \in \mathcal{C}\}$  and where  $\mathcal{C}$  is the challenge space. By relaxing the opening verification of the commitment scheme to accept extracted messages and opening information, we allow a commitment to be opened to a small multiple  $\bar{\mathbf{c}}\mathbf{m}$  of the originally committed message  $\mathbf{m} \in \mathcal{U}$ . In order to preserve a meaningful notion of relaxed binding, we must choose the message and challenge spaces so that the sets of small multiples of different messages are disjoint, i.e., that there do not exist distinct  $\mathbf{m}, \mathbf{m}' \in \mathcal{U}$  and  $\bar{\mathbf{c}}, \bar{\mathbf{c}}' \in \bar{\mathcal{C}}$  such that  $\mathbf{m}\bar{\mathbf{c}} = \mathbf{m}'\bar{\mathbf{c}}'$ .

For efficiency reasons, we choose messages and challenges from the subring  $\mathcal{R}_3^{(2^K)}$  so that they have at most  $2^K$  nonzero coefficients. By choosing the message and challenge spaces as

$$\begin{aligned} \mathcal{U} &= \{\mathbf{1}\} \cup \{\mathbf{m} \in \mathcal{R}_3^{(2^K)} : \deg(\mathbf{m}) = n/2 \wedge \mathbf{m} \text{ is irreducible in } \mathbb{Z}_q[\mathbf{x}]\} \\ \mathcal{C} &= \{\mathbf{c} \in \mathcal{R}_3^{(2^K)} : \deg(\mathbf{c}) < n/4\} \\ \bar{\mathcal{U}} &= \{\bar{\mathbf{m}} \in \mathcal{R}_{2p+1}^{(2^K)} : \deg(\bar{\mathbf{m}}) < 3n/4\} \\ \bar{\mathcal{C}} &= \{\mathbf{c} - \mathbf{c}' : \mathbf{c}, \mathbf{c}' \in \mathcal{C}\}, \end{aligned} \tag{1}$$

we have that each  $\bar{\mathbf{m}} \in \bar{\mathcal{U}}$  can have at most one irreducible factor of degree  $n/2$  in  $\mathbb{Z}_q[\mathbf{x}]$ . By defining the message relaxation function  $f$  as

$$\begin{aligned} f(\mathbf{m}) &= \{\bar{\mathbf{m}} \in \bar{\mathcal{U}} : \mathbf{m}|\bar{\mathbf{m}} \text{ in } \mathbb{Z}_q[\mathbf{x}]\} \text{ for } \mathbf{m} \neq \mathbf{1} \\ f(\mathbf{1}) &= \{\bar{\mathbf{m}} \in \bar{\mathcal{U}} : \nexists \mathbf{m} \in \mathcal{U} \setminus \{\mathbf{1}\} : \mathbf{m}|\bar{\mathbf{m}} \text{ in } \mathbb{Z}_q[\mathbf{x}]\}, \end{aligned} \tag{2}$$

the unique factorization of polynomials in  $\mathbb{Z}_q[\mathbf{x}]$  guarantees that the partition components  $f(\mathbf{m})$  and  $f(\mathbf{m}')$  are disjoint for any distinct  $\mathbf{m}, \mathbf{m}' \in \mathcal{U}$ .

To generate elements of  $\bar{\mathcal{U}}$ , we suggest to generate random monic polynomials of degree  $n/2$  in  $\mathcal{R}_3^{(2^K)}$  and test them for irreducibility, which can be done efficiently (e.g., using Proposition 3.4.4 in [27]). By the Gauss’ formula, the number of monic polynomials of degree  $n/2$  that are irreducible in  $\mathbb{Z}_q[\mathbf{x}]$

is approximately  $q^{n/2}/(n/2)$ . Assuming that the irreducible polynomials are “spread evenly” across  $\mathbb{Z}_q[\mathbf{x}]$ , one expects to sample an average of  $n/2$  polynomials until finding an irreducible one.

### 4.3 A Lattice-Based Relaxed Commitment Scheme

Our relaxed commitment uses message space  $\mathcal{U}$  and relaxed message space  $\bar{\mathcal{U}}$  defined in Eq. (1). The algorithms of our relaxed commitment scheme  $\text{rC}$  are as follows.

**Parameter generation.**  $\text{ParGen}_c$  selects a uniformly random commitment key

$\mathbf{C} \xleftarrow{\$} \mathcal{R}_q^{1 \times m}$  and the parameters  $\bar{N}_c$  and  $\bar{N}_{c,\infty}$  that will be defined in Sect. 6.

It outputs  $\text{cpar} = (\mathbf{C}, \bar{N}_c, \bar{N}_{c,\infty})$ .

**Commitment generation.** On input  $(\text{cpar}, \mathbf{m})$ , the algorithm  $\text{Commit}$  first checks that  $\mathbf{m} \in \mathcal{R}_3^{(2^K)}$ , that  $\deg(\mathbf{m}) = n/2$ , and that  $\mathbf{m}$  is irreducible. It then selects uniformly random  $\mathbf{E} \xleftarrow{\$} \mathcal{R}_3^{1 \times m}$  and  $\mathbf{b} \xleftarrow{\$} \mathcal{R}_3$ , and constructs the commitment as  $\mathbf{F} = (\mathbf{C} + \mathbf{m}\mathbf{G} + \mathbf{E})\mathbf{b}^{-1}$ . It outputs  $(\mathbf{F}, (\mathbf{1}, \mathbf{E}, \mathbf{b}))$ .

**Opening verification.** On input a message  $\bar{\mathbf{m}}$ , a commitment  $\mathbf{F}$ , and opening values  $(\bar{\mathbf{c}}, \bar{\mathbf{E}}, \bar{\mathbf{b}})$ ,  $\text{OpenVf}$  outputs 1 if  $\mathbf{F} = (\bar{\mathbf{c}}\mathbf{C} + \bar{\mathbf{m}}\mathbf{G} + \bar{\mathbf{E}})\bar{\mathbf{b}}^{-1}$ ,  $\bar{\mathbf{m}} \in \bar{\mathcal{U}}$  and  $(\bar{\mathbf{c}}, \bar{\mathbf{E}}, \bar{\mathbf{b}}) \in \mathcal{OV} = \{(\bar{\mathbf{c}}, \bar{\mathbf{E}}, \bar{\mathbf{b}}) \in \bar{\mathcal{C}} \times \mathcal{R}_q^{1 \times m} \times \mathcal{R}_q : \|\bar{\mathbf{E}}, \bar{\mathbf{b}}\| \leq \bar{N}_c \wedge \|\bar{\mathbf{E}}, \bar{\mathbf{b}}\|_\infty \leq \bar{N}_{c,\infty}\}$ .

Correctness follows easily from the parameters computation (cf. Sect. 6). We show in the full version [13] that it satisfies the relaxed binding property under the Ring-SIS assumption and the hiding property under the following new and, we believe, reasonable assumption. To gain trust in our new assumption, in the full version [13] we give a variant of the assumption that we show to be equivalent to Ring-LWE and that, through a complexity leveraging argument, implies Assumption 1. We therefore obtain two ways to interpret our security result: either one believes Assumption 1 below directly, or one only believes the weaker Ring-LWE assumption and compensates for the tightness loss in complexity leveraging by adjusting the scheme parameters.

**Assumption 1.** *Consider the following game between an adversary  $\mathbf{A}$  and a challenger for fixed  $m \in \mathbb{N}$  and distribution  $D$ :*

1. *The challenger outputs a uniformly random  $\mathbf{C} \xleftarrow{\$} \mathcal{R}_q^{1 \times m}$  to  $\mathbf{A}$ .*
2.  *$\mathbf{A}$  sends back  $\mathbf{m} \in \mathcal{U}$ .*
3. *The challenger samples a uniformly random bit  $b \xleftarrow{\$} \{0, 1\}$ . If  $b = 1$ , it samples an error vector  $\mathbf{E} \xleftarrow{\$} D^m$  and a uniform secret  $\mathbf{s} \xleftarrow{\$} D$ , and sends  $\mathbf{F} = (\mathbf{C} + \mathbf{m}\mathbf{G} - \mathbf{E})\mathbf{s}^{-1}$  to  $\mathbf{A}$ . Otherwise, it sends a uniform  $\mathbf{F} \xleftarrow{\$} \mathcal{R}_q^{1 \times m}$  to  $\mathbf{A}$ .*
4.  *$\mathbf{A}$  sends a bit  $b'$  to the challenger.*

*The advantage of  $\mathbf{A}$  in winning the game is  $|\Pr(b = b') - \frac{1}{2}|$ . The assumption states that no PPT  $\mathbf{A}$  can win the previous game with non-negligible advantage.*

## 5 Relaxed Lattice-Based Signatures

The third building block is a signature scheme for which the  $\text{r}\Sigma$  protocol from Sect. 3 can be used to prove knowledge of a signature on a committed message. Similarly to the relaxed commitments of the previous section, we also define *relaxed signature schemes* to accommodate for the relaxed extraction of the  $\text{r}\Sigma$  protocol. More specifically, the verification algorithm is relaxed to accept messages and signatures that could never be signed, respectively produced, by the honest signing algorithm. At the same time, we also relax the unforgeability notion so that the adversary's forgery cannot be on a message that is within the span, through a function  $f$ , of its previous signing queries.

### 5.1 Definition of Relaxed Signatures

A relaxed signature scheme associated with message space  $\mathcal{M}$  and relaxed messages space  $\bar{\mathcal{M}} \supseteq \mathcal{M}$  consists of a parameter generation algorithm  $\text{SParGen}$  that on input security parameter  $1^\lambda$  outputs system parameters  $\text{spar}$ ; a key generation algorithm  $\text{SKeyGen}$  that on input  $\text{spar}$  outputs a signing key  $sk$  and a verification key  $vk$ ; a signing algorithm  $\text{Sign}$  that on input  $sk$  and a message  $M \in \mathcal{M}$  outputs a signature  $sig$ ; and a verification algorithm  $\text{SVf}$  that on input  $vk$ , a message  $\bar{M} \in \bar{\mathcal{M}}$  and a signature  $\bar{sig}$  returns 1 if the signature is valid or 0 if it is invalid. Correctness requires that  $\text{SVf}(vk, M, sig) = 1$  for all messages  $M \in \mathcal{M}$ , for all security parameters  $\lambda \in \mathbb{N}$ , for all  $(sk, vk) \in \text{SKeyGen}(\text{spar})$ , and for all  $sig \in \text{Sign}(sk, M)$ .

Relaxed unforgeability is parameterized by a message relaxation function  $g : \mathcal{M} \rightarrow 2^{\bar{\mathcal{M}}}$ . The adversary in the  $g$ -relaxed unforgeability below wins the game if it can output a valid signature on a message  $\bar{M} \in \bar{\mathcal{M}}$  that is not in the span through  $g$  of its signature queries.

**Definition 5 (Relaxed Unforgeability).** *A relaxed signature scheme  $(\text{SParGen}, \text{SKeyGen}, \text{Sign}, \text{SVf})$  is  $g$ -relaxed unforgeable if for all PPT  $\mathbf{A}$  the probability*

$$\Pr \left[ \text{SVf}(vk, \bar{M}, \bar{sig}) = 1 \wedge \bar{M} \notin g(\mathcal{Q}) : \begin{array}{l} \text{spar} \leftarrow \text{SParGen}(1^n), (sk, vk) \leftarrow \text{SKeyGen}(\text{spar}), \\ (\bar{M}, \bar{sig}) \leftarrow \mathbf{A}^{\mathcal{O}_S}(n, \text{spar}, vk) \end{array} \right]$$

is negligible, where the oracle  $\mathcal{O}_S(M)$  returns  $\text{Sign}(\text{spar}, sk, vk, M)$  and  $\mathcal{Q}$  is the set of  $\mathbf{A}$ 's queries to  $\mathcal{O}_S$ .

The concept of relaxed signatures is somewhat reminiscent of a technique used for proofs of knowledge of a strong-RSA-based signature in groups of unknown order [22]. Here, one has to prove that the message lies in a certain space, but the correctness of such a proof is only guaranteed when the actual message lies in a smaller interval. The approach was used in several privacy-preserving protocols, but was never formalized and did not require an adapted unforgeability notion.

## 5.2 A Lattice-Based Relaxed Signature Scheme

We describe a relaxed signature scheme with message space  $\mathcal{M} = \{(\mathbf{m}, \alpha) \in \mathcal{U} \times \{0, 1\}^*\}$ , where  $\mathcal{U}$  is as defined in Eq. (1). In a typical use case,  $\mathbf{m}$  is a user identity and  $\alpha$  an attribute value assigned to that user. Our scheme combines a weakly secure version of Boyen signatures [14] to sign user identities and Gentry-Peikert-Vaikuntanathan signatures [35] to sign attribute values.

To use the  $r\Sigma$  protocol from Sect. 3.2 to prove knowledge of a signature for a committed user identity  $\mathbf{m}$ , we relax the verification algorithm so that the (relaxed) witness that can be extracted from a valid  $r\Sigma$  protocol is still considered a valid signature for a message from the relaxed message space  $\bar{\mathcal{M}} = \bar{\mathcal{U}} \times \{0, 1\}^*$ , where  $\bar{\mathcal{U}}$  is as defined in Eq. (1).

Our relaxed signature scheme  $rS$  is described as follows:

**System parameters.** The system parameters *spar* include a uniformly random matrix  $\mathbf{C} \in \mathcal{R}_q^{1 \times m}$ , a gadget vector  $\mathbf{G}$  of length  $m$  as defined in Theorem 1, and a hash function  $\mathcal{H} : \{0, 1\}^* \rightarrow \mathcal{R}_q$ .

It also contains the following parameters:  $\sigma_t$  is the standard deviation of the trapdoor distribution,  $\sigma$  is the standard deviation of the signature distribution,  $p$  is a bound on the norm of user identities  $\bar{\mathbf{m}}$ ,  $N_s$  is a bound on the norm of honestly created signatures;  $\bar{N}_s$ ,  $\bar{N}_{s,\infty}$ , and  $\bar{C}$  are bounds on the norm of components of signatures accepted by the relaxed verification algorithm,  $\mathcal{C}$  and  $\bar{\mathcal{C}}$  are challenge spaces defined in Eq. (1). Concrete values for these parameters will be provided in the correctness discussion and in Table 1. When discussing the correctness of the signature, we give precise formulas for all the previous parameters but  $\bar{N}_s$ ,  $\bar{N}_{s,\infty}$ , and  $\bar{C}$ . These last three depend on the extraction algorithm of the relaxed  $\Sigma$ -protocol in Sect. 6. Their computation is standard, and is described in the full version [13]. For correctness to hold, we only need to impose that  $\bar{N}_s > N_s$  and  $\bar{C} \geq 1$ .

**Key generation.** The signer chooses a uniform polynomial  $\mathbf{a} \in \mathcal{R}_q$  and sets  $\mathbf{A} = [\mathbf{a}|\mathbf{1}]$ . The secret signing key is sampled as  $\mathbf{X} \stackrel{s}{\leftarrow} \mathcal{D}_{\mathcal{R}_q, \sigma_t}^{2 \times m}$ . Letting  $\mathbf{A} = [\mathbf{a}|\mathbf{1}]$ , the public verification key is the vector  $\mathbf{V} = [\mathbf{A}|\mathbf{B}|\mathbf{C}|\mathbf{1}] = [\mathbf{A}|\mathbf{A}\mathbf{X} + \mathbf{G}|\mathbf{C}|\mathbf{1}] \in \mathcal{R}_q^{1 \times (3+2m)}$ .

**Signing.** If  $M = (\mathbf{m}, \alpha) \notin \mathcal{M}$  then abort. Otherwise, the signer calculates  $\mathbf{S} \leftarrow \text{SampleD}([\mathbf{A}|\mathbf{B}|\mathbf{C} + \mathbf{m}\mathbf{G}], \mathcal{H}(\alpha), \sigma)$  (see Lemma 3) and outputs a signature  $sig = (\mathbf{1}, [\mathbf{S}; \mathbf{0}], \mathbf{1})$ . The entry  $(\mathbf{m}, \alpha, sig)$  is stored so that the same signature  $sig$  is returned next time that  $(\mathbf{m}, \alpha)$  is signed.

**Verification.** Verification of a signature  $sig = (\bar{\mathbf{c}}_1, \bar{\mathbf{S}}, \bar{\mathbf{c}}_2)$  on message  $\bar{M} = (\bar{\mathbf{m}}, \alpha)$  returns 1 if  $[\mathbf{A}|\mathbf{B}|\bar{\mathbf{c}}_1\mathbf{C} + \bar{\mathbf{m}}\mathbf{G}|\mathbf{1}]\bar{\mathbf{S}} = \bar{\mathbf{c}}_2\mathcal{H}(\alpha)$ , if  $\bar{M} \in \bar{\mathcal{M}}$ , and if  $sig \in \{(\bar{\mathbf{c}}_1, \bar{\mathbf{S}}, \bar{\mathbf{c}}_2) \in \bar{\mathcal{C}} \times \mathcal{R}_q^{3+2m} \times \mathcal{R}_q : \|\bar{\mathbf{S}}\| \leq \bar{N}_s \wedge \|\bar{\mathbf{S}}\|_\infty \leq \bar{N}_{s,\infty} \wedge \|\bar{\mathbf{c}}_2\| \leq \bar{C}\}$ . Otherwise, it returns 0.

Correctness of the  $rS$  scheme follows from the choices of parameters explained in the full version [13]. We prove the  $g$ -unforgeability of our  $rS$  scheme under the following assumption.

**Assumption 2.** Let  $\bar{\Sigma} = \{(\mathbf{c}_1, \mathbf{S}, \mathbf{c}_2) \in \bar{\mathcal{C}} \times \mathcal{R}_q^{3+2m} \times \mathcal{R}_q : \|\mathbf{S}\| \leq N' \wedge \|\mathbf{c}_2\| \leq C'\}$  for some fixed parameters. Consider the following game between an adversary  $\mathbf{A}$  and a challenger for fixed  $m \in \mathbb{N}$  and distribution  $D$ :

1. The challenger chooses  $\mathbf{a} \xleftarrow{\$} \mathcal{R}_q$ ,  $\mathbf{C} \xleftarrow{\$} \mathcal{R}_q^{1 \times m}$ , and  $\mathbf{X} \xleftarrow{\$} \mathcal{D}_{\mathcal{R}_q, \sigma_t}^{2 \times m}$ . It sets  $\mathbf{A} = [\mathbf{a} | \mathbf{1}]$  and  $\mathbf{B} = \mathbf{A}\mathbf{X} + \mathbf{G}$ , where  $\mathbf{G} = [1 \ [q^{1/m}] \dots [q^{(m-1)/m}]]$ .
2. The challenger runs  $\mathbf{A}$  on input  $[\mathbf{A} \ \mathbf{B} \ \mathbf{C} \ \mathbf{1}]$ , giving it access to a random oracle  $\mathcal{H} : \{0,1\}^* \rightarrow \mathcal{R}_q$  and an oracle  $\mathcal{O}_S$  that on input  $\mathbf{m} \in \mathcal{U}$  and a string  $\alpha \in \{0,1\}^*$  outputs a small vector  $[\mathbf{S} ; \mathbf{0}]$  in the coset  $\mathcal{L}^\perp([\mathbf{A} \ \mathbf{B} \ \mathbf{C} + \mathbf{m}\mathbf{G} \ \mathbf{1}]) + \mathcal{H}(\alpha)$  such that  $\|\mathbf{S}\| \leq N_S$ .
3. Algorithm  $\mathbf{A}$  outputs  $\bar{\mathbf{m}} \in \bar{\mathcal{U}}$ ,  $\bar{\alpha} \in \{0,1\}^*$ ,  $\bar{\mathbf{c}}_1 \in \bar{\mathcal{C}}$ , a ring element  $\bar{\mathbf{c}}_2$  and a vector  $\bar{\mathbf{S}}$ . Algorithm  $\mathbf{A}$  wins the game if  $(\bar{\mathbf{c}}_1, \bar{\mathbf{S}}, \bar{\mathbf{c}}_2) \in \bar{\Sigma}$ ,  $\bar{\mathbf{m}} \in \bar{\mathcal{U}}$ , such that  $\bar{\mathbf{S}}$  is a short vector of the coset  $\mathcal{L}^\perp([\mathbf{A} \ \mathbf{B} \ \mathbf{C} \ \mathbf{1}] + \mathbf{c}_2\mathcal{H}(\bar{\alpha}))$  where  $\bar{\mathbf{C}} = \bar{\mathbf{c}}_1\mathbf{C} - \bar{\mathbf{m}}\mathbf{G}$ , and  $(\bar{\mathbf{m}}\bar{\mathbf{c}}_1^{-1}, \bar{\alpha})$  was not queried to the  $\mathcal{O}_S$  oracle.

The assumption states that no PPT algorithm  $\mathbf{A}$  can win the game with non-negligible probability.

**Theorem 2.** An algorithm  $\mathbf{A}$  that breaks the  $g$ -uf-cma unforgeability of the relaxed signature scheme in time  $t$  and probability  $\epsilon_A$  can break the Assumption 2 in time  $t$  with probability  $\epsilon_A$  in the Random Oracle Model.

A valid forgery can be used to break Assumption 2 because unforgeability is defined w.r.t. a function  $g$ . This guarantees that  $\bar{\mathbf{m}}$  and  $\bar{\mathbf{c}}_1$  output by  $\mathbf{A}$  are such that  $\bar{\mathbf{m}}\bar{\mathbf{c}}_1^{-1}$  was not queried to  $\mathcal{O}_S$  as specified by the assumption.

Assumption 2 is very similar to the  $g$ -unforgeability experiment itself, but, similarly to what we did for the hiding property of the rC scheme, we gain trust in the assumption in the full version [13] by describing a variant of the assumption that is implied by the Ring-LWE and Ring-SIS assumptions, and that through a complexity leveraging argument implies Assumption 2. Again, this yields two possible ways of interpreting our result: either one believes Assumption 2 directly, or one believes the Ring-SIS and Ring-LWE assumption and adjusts the scheme parameters to compensate for the tightness loss due to complexity leveraging.

## 6 Relaxed Proofs of Signatures on Committed Messages

We now prove how our three primitives can be composed together to prove knowledge of a signature  $\mathbf{S}$  on a secret  $\mathbf{m}$  w.r.t. a public bit-string  $\alpha$ . To prove knowledge of both  $\mathbf{S}$  and  $\mathbf{m}$ , the protocol exploits the relaxed commitment scheme defined in Sect. 4.3. The commitment allows to prove knowledge of the signature and the secret part of the message in two separate equations, and this is needed both for technical and practical reasons. On one hand, this gives a better bound on the extracted message, as rejection sampling can be performed separately on the two equations. On the other hand, this allows to prove knowledge of a set of signatures  $\{\mathbf{S}_i\}_{i=1, \dots, \ell}$  on messages  $(\mathbf{m}, \alpha_i)$  for  $i = 1 \dots, \ell$ , i.e., on message pairs composed by the same secret  $\mathbf{m}$  and by different public bit-strings  $\alpha_i$ . This is important because in the AAT this protocol allows a user with identity  $\mathbf{m}$  to

create a presentation token for an attribute  $\alpha$ , and it is often necessary for the user to disclose multiple attributes at the same time. We show at the end of this section the intuition behind the generalization to the multiple-signatures case, i.e., how the user can prove that she has signatures by the issuer on the pairs  $(\mathbf{m}_i, \alpha_i)_i$ , where in fact  $\mathbf{m} = \mathbf{m}_i = \mathbf{m}_j$  is her id. For sake of simplicity, we start presenting the proof for a single message-signature pair.

Let  $\mathbf{A}$ ,  $\mathbf{B}$ , and  $\mathbf{C}$  be the public vectors of the signature scheme. Given  $\alpha$  and the public parameters of the signature,  $\mathcal{P}$  wants to prove that she owns some “small”  $(\mathbf{m}, (\mathbf{c}_1, \mathbf{S}, \mathbf{c}_2))$  such that  $[\mathbf{A}|\mathbf{B}|\mathbf{c}_1\mathbf{C} + \mathbf{m}\mathbf{G}|\mathbf{1}]\mathbf{S} = \mathbf{c}_2\mathcal{H}(\alpha)$ . To construct a relaxed  $\Sigma$ -protocol (cf. Sect. 3), rewrite the characterizing equation as follows. Let  $(\mathbf{1}, \mathbf{S}, \mathbf{1})$  be a honestly-generated signature on  $(\mathbf{m}, \alpha)$ , i.e.

$$[\mathbf{A}|\mathbf{B}|\mathbf{C} + \mathbf{m}\mathbf{G}|\mathbf{1}]\mathbf{S} = \mathcal{H}(\alpha) \quad (3)$$

Generate a commitment  $\mathbf{F} = \mathbf{b}^{-1}(\mathbf{C} + \mathbf{m}\mathbf{G} + \mathbf{E})$  to  $\mathbf{m}$  and substitute  $\mathbf{C} + \mathbf{m}\mathbf{G} = \mathbf{F}\mathbf{b} - \mathbf{E}$  in Eq. (3). Rearranging the terms,  $\mathcal{P}$  can now use the protocol in Sect. 3 to show she owns some “small”  $(\bar{\mathbf{S}}_c, \bar{\mathbf{c}}_1, \bar{\mathbf{S}}_s, \bar{\mathbf{c}}_2)$  satisfying:

$$(I) \quad \underbrace{[-\mathbf{G}^T \mathbf{F}^T - \mathbb{I}_m]}_{=\mathbf{A}_c} \underbrace{\begin{pmatrix} \bar{\mathbf{m}} \\ \bar{\mathbf{b}} \\ \bar{\mathbf{E}}^T \end{pmatrix}}_{=\bar{\mathbf{S}}_c} = \bar{\mathbf{c}}_1 \mathbf{C}^T, \quad (II) \quad \underbrace{[\mathbf{A}|\mathbf{B}|\mathbf{F}|\mathbf{1}]}_{=\mathbf{A}_s} \underbrace{\begin{pmatrix} \bar{\mathbf{S}}_1 \\ \bar{\mathbf{S}}_2 \\ \bar{\mathbf{S}}_3 \\ \bar{\mathbf{S}}_4 \end{pmatrix}}_{=\bar{\mathbf{S}}_s} = \bar{\mathbf{c}}_2 \mathcal{H}(\alpha). \quad (4)$$

Indeed,  $\mathcal{P}$  samples  $\mathbf{Y}_1 \xleftarrow{\$} \mathcal{D}_{\mathcal{Y}_{3n/4} \times \mathcal{R}_q^{1+m}, \sigma_1}$  and  $\mathbf{Y}_2 \xleftarrow{\$} \mathcal{D}_{\mathcal{R}_q, \sigma_2}^{2m+3}$  and sends as commitments  $(\mathbf{A}_c \mathbf{Y}_1, \mathbf{A}_s \mathbf{Y}_2)$ . Upon receiving the challenge  $\mathbf{c}$  from the verifier, the prover sets  $\mathbf{Z}_1 = \mathbf{Y}_1 + \mathbf{c}\mathbf{S}_c$  and  $\mathbf{Z}_2 = \mathbf{Y}_2 + \mathbf{c}\mathbf{S}_s$ , and does rejection sampling separately on them. The verifier accepts if the response satisfies the bounds deriving from the use of rejection sampling, i.e.  $\|\mathbf{Z}_1\| \leq 1.05\sigma_1\sqrt{n(2+m)}$ ,  $\|\mathbf{Z}_2\| \leq 1.05\sigma_2\sqrt{n(2+2m)}$ ,  $\|\mathbf{Z}_1\|_\infty \leq 8\sigma_1$ ,  $\|\mathbf{Z}_2\|_\infty \leq 8\sigma_2$ , and the first component of  $\mathbf{Z}_1$  is in the subring. More discussion on the protocol can be found in the full version [13].

In Theorem 3 are defined the relations for the relaxed  $\Sigma$ -protocol. We only highlight that the extractor is standard, so it rewinds the prover to obtain two responses to different challenges. Hence, for it to extract a valid opening information, it is enough to set  $\bar{N}_c = 2.1\sigma_2\sqrt{n(2+2m)}$  and  $\bar{N}_{c,\infty} = 16\sigma_1$ . The algorithm and the bounds on the norm of the extracted witness  $N_s, \bar{N}_s, \bar{N}_{s,\infty}, \bar{C}$  can be found in the full version [13].

**Theorem 3.** *Given  $N_s$  as in Sect. 5.2 and  $\bar{N}_s = 8.82(3 + 2m)\sigma_1\sigma_2n\sqrt{nm}$ ,  $\bar{N}_{s,\infty} = 512\sigma_1\sigma_2n$ ,  $\bar{C} = 4.2\sigma_1n\sqrt{2^{K-2}-1}$  and  $p = 16\sigma_1$ , the protocol  $(\mathcal{P}, \mathcal{V})$  is a relaxed  $\Sigma$ -protocol for the following pair of relations:*

$$\begin{aligned} R &= \{ ((\mathbf{A}_s, \mathcal{H}(\alpha)), (\mathbf{m}, (\mathbf{1}, \mathbf{S}, \mathbf{1}))) : \mathbf{m} \in \mathcal{U}, [\mathbf{A}|\mathbf{B}|\mathbf{1}\mathbf{C} + \mathbf{m}\mathbf{G}|\mathbf{1}]\mathbf{S} = \mathbf{1}\mathcal{H}(\alpha) \\ &\quad \text{and } \|\mathbf{S}\| \leq N_s \} \\ \bar{R} &= \{ ((\mathbf{A}_s, \mathcal{H}(\alpha)), (\bar{\mathbf{m}}, (\bar{\mathbf{c}}_1, \bar{\mathbf{S}}, \bar{\mathbf{c}}_2))) : \bar{\mathbf{m}} \in \bar{\mathcal{U}}, \bar{\mathbf{c}}_1 \in \bar{\mathcal{C}}, \|\bar{\mathbf{c}}_2\| \leq \bar{C} \\ &\quad [\mathbf{A}|\mathbf{B}|\bar{\mathbf{c}}_1\mathbf{C} + \bar{\mathbf{m}}\mathbf{G}|\mathbf{1}]\bar{\mathbf{S}} = \bar{\mathbf{c}}_2\mathcal{H}(\alpha) \text{ and } \|\bar{\mathbf{S}}\| \leq \bar{N}_s, \|\bar{\mathbf{S}}\|_\infty \leq \bar{N}_{s,\infty} \} \end{aligned}$$

under Ring-LWE $_{\mathcal{R}_3}$  with the uniform distribution.

Setting when  $K = 6$  as in Table 1 the cardinality of  $\mathcal{C}$  is  $|\mathcal{C}| = 3^{2^{K-2}-1} = 3^{15}$ , hence the proof has to be repeated 6 times to have negligible soundness error.

The relaxed  $\Sigma$ -protocol can be transformed into a NIZK proof using the Fiat-Shamir transform combined with an OTS (cf. Sect. 3.2).

A proof of knowledge of  $\ell$  signatures  $\mathbf{S}_i$  generated by signer  $i$  on  $\ell$  messages  $(\mathbf{m}, \alpha_i)$  is constructed by combining  $\ell$  of the previous proofs in parallel. Assume that the parameters of the rC and rS schemes are shared among all signers. This means that the verification key of signer  $j$  is  $[\mathbf{A}_j | \mathbf{B}_j | \mathbf{C}]$  for the same  $\mathbf{C}$ . Hence, the prover can generate a commitment  $\mathbf{F}$  to  $\mathbf{m}$  using  $\mathbf{C}$  as public matrix, and generate a proof  $\Pi_i$  that she knows a secret  $\bar{\mathbf{S}}_c$  that satisfies relation (I) in (4) and  $\bar{\mathbf{S}}_{s,i}, \bar{\mathbf{c}}$  that satisfy  $[\mathbf{A}_i | \mathbf{B}_i | \mathbf{F} | \mathbf{1}] \bar{\mathbf{S}}_{s,i} = \bar{\mathbf{c}} \mathcal{H}(\alpha_i)$  for  $i = 1, \dots, \ell$ . The relaxed binding property of the commitment guarantees that the hidden part of the message  $\mathbf{m}$  is the same in all proofs.

## 7 Compact Anonymous Attribute Tokens from Lattices

Anonymous attribute tokens [23] can be seen as simplified anonymous credentials, allowing users to obtain a credential from an issuer that contains a list of attributes. Users can selectively disclose subsets of these attributes to verifiers in such a way that not even the verifier and the issuer together can link different presentations by the same user. In this section, we focus on AAT schemes without opening (AAT-O), i.e., without a trusted opener who can de-anonymize presentation tokens. Formal definitions of the scheme can be found in the full version [13].

### 7.1 Compact AATs from Lattices

From the relaxed primitives that we introduced, it is possible to construct an AAT-O scheme with compact presentation tokens. Parameters for the commitment scheme are generated from the signature parameters  $spar$  using  $\text{DerivePar}_c$  that, on input  $spar$ , sets the commitment public matrix  $\mathbf{C}$  to be the third block of the signature public key  $vk = [\mathbf{A} | \mathbf{B} | \mathbf{C} | \mathbf{1}]$  and computes  $N'_{c,2}, N'_{c,\infty}$ .

**System Parameter Generation.** The system parameters are the signature parameters  $spar$  from Sect. 5.2. Then it runs  $cpar \leftarrow \text{DerivePar}_c(spar)$  and outputs  $par = (spar, cpar)$ .

**Issuer Key Generation.** The issuer runs the signing key generation  $\text{SKeyGen}$  to obtain  $isk = \mathbf{X}$  and the public matrix  $ipk = [\mathbf{A} | \mathbf{B} | \mathbf{C} | \mathbf{1}]$ .

**Issuance.** To issue a credential to a user for attributes  $(\alpha_i)_{i=1}^\ell$ , the issuer chooses an  $id = \mathbf{m} \in \mathcal{U}$ , checks that  $\mathbf{m} \notin \mathcal{S}$  and computes signatures on  $(\mathbf{m}, i || att_i)$  using the  $\text{Sign}$  algorithm. The credential consists of  $\mathbf{m}$ ,  $(\alpha_i)_{i=1}^\ell$  together with the resulting signatures  $(\mathbf{1}, [\mathbf{S}_i; \mathbf{1}], \mathbf{1})$ . The issuer adds  $\mathbf{m}$  to  $\mathcal{S}$ .

**Presentation.** To create a presentation token for attributes  $(\alpha_i)_{i \in R}$  and message  $M$ , the user creates a relaxed commitment  $\mathbf{F}$  to  $\mathbf{m}$  and generates NIZK proofs  $\Pi_i$  that he knows signatures on the committed message and  $i || \alpha_i$  for  $i \in R$ , whereby he includes the message  $M$  in the Fiat-Shamir hash. The presentation token  $pt$  consists of the commitment  $\mathbf{F}$  and the transcripts  $(\Pi_i)_{i \in R}$ .

**Verification.** The verifier checks the validity of  $(II)_{i \in R}$  w.r.t.  $\mathbf{F}$  and the message  $M$ . If the tests pass, he outputs *accept*, otherwise *reject*.

Parameters for this scheme are presented in Table 1. Details on how they were computed can be found in the full version [13].

**Table 1.** Table of parameters for the AAT scheme without opening for  $\lambda = 80$  bits of security and  $K = 6$ . All the values are rounded up.

Compl. Lev.	$\delta$	Parameters				Sizes		
		$n$	$q$	$m$	$\sigma_i$	<i>ipk</i> (KB)	<i>usk</i> (KB)	<i>token</i> (MB)
NO	<1.002926	$2^{12}$	$\sim 2^{110}$	5	4	620	271	3.42
YES	<1.0003788	$2^{13}$	$\sim 2^{98}$	18	4	3714	863	17.77

**Security.** The security of this AAT-O follows from the security guarantees of its building blocks. Unforgeability relies on the relaxed unforgeability of the  $\mathbf{rS}$  scheme, on the relaxed binding property of the  $\mathbf{rC}$  scheme and on the relaxed simulation soundness of the  $\mathbf{r\Sigma}$  scheme. The proof strategy is to run the adversary and extract from the forged presentation token using the Generalized Forking Lemma [6]. The proof can be found in the full version of the paper [13].

**Theorem 4 (Unforgeability).** *Assume  $A$  is an adversary that runs in time  $t_A$ , makes  $q_D$  random-oracle queries for credentials issued to dishonest users (if  $A$  queries for a credential on  $(id, (\alpha_i)_{i=1, \dots, m})$ , we count it as  $m$  queries) and  $q_H$  queries for credential issued and presentation tokens of honest users and breaks the unforgeability of the AAT with probability  $\epsilon_A$ , then there exists an algorithm that breaks the unforgeability of the signature in time  $t_B = 32t_a(q_D + q_H)/\epsilon_A \cdot \ln(16/\epsilon_A)$  with probability  $\epsilon_B = \epsilon_A/8$  after asking  $q_D$  queries to the signing oracle in the Random Oracle Model.*

Anonymity is guaranteed by the zero-knowledge property of  $\mathbf{r\Sigma}$  and by the hiding property of  $\mathbf{rC}$ . The proof of the theorem can be found in the full version [13].

**Theorem 5 (Anonymity).** *If an adversary  $A$  running in time  $t$  breaks the anonymity of the AAT with probability at most  $\epsilon$ , then there is an adversary running in time  $t$  who breaks the hiding property of the commitment scheme with advantage at most  $\epsilon$  in the Random Oracle Model.*

**Acknowledgements.** Working on this paper, we have enjoyed many discussions with Vadim Lyubashevsky. Thank you! This work was supported by the ERC under grant #321310 PERCY) and the SNF under grant #200021\_157080 (Efficient Lattice-Based Cryptographic Protocols).



## References

1. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-14623-7\\_12](https://doi.org/10.1007/978-3-642-14623-7_12)
2. Abe, M., Ohkubo, M.: A framework for universally composable non-committing blind signatures. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 435–450. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-10366-7\\_26](https://doi.org/10.1007/978-3-642-10366-7_26)
3. 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). [https://doi.org/10.1007/978-3-642-13190-5\\_28](https://doi.org/10.1007/978-3-642-13190-5_28)
4. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange: a new hope. In: USENIX Security Symposium (2016)
5. 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). [https://doi.org/10.1007/3-540-44598-6\\_16](https://doi.org/10.1007/3-540-44598-6_16)
6. Bagherzandi, A., Cheon, J.H., Jarecki, S.: Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In: ACM CCS (2008)
7. Banaszczyk, W.: New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen* **296**(1), 625–635 (1993)
8. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and non-interactive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-78524-8\\_20](https://doi.org/10.1007/978-3-540-78524-8_20)
9. 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). [https://doi.org/10.1007/3-540-39200-9\\_38](https://doi.org/10.1007/3-540-39200-9_38)
10. Benhamouda, F., Camenisch, J., Krenn, S., Lyubashevsky, V., Neven, G.: Better zero-knowledge proofs for lattice encryption and their application to group signatures. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 551–572. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-45611-8\\_29](https://doi.org/10.1007/978-3-662-45611-8_29)
11. Böhl, F., Hofheinz, D., Jäger, T., Koch, J., Striecks, C.: Confined guessing: new signatures from standard assumptions. *J. Cryptol.* **28**(1), 176–208 (2015)
12. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-28628-8\\_3](https://doi.org/10.1007/978-3-540-28628-8_3)
13. Boschini, C., Camenisch, J., Neven, G.: Relaxed lattice-based signatures with short zero-knowledge proofs. *Cryptology ePrint Archive*, Report 2017/1123 (2017)
14. 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). [https://doi.org/10.1007/978-3-642-13013-7\\_29](https://doi.org/10.1007/978-3-642-13013-7_29)
15. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical hardness of learning with errors. In: 45th ACM STOC (2013)

16. Brickell, E.F., Camenisch, J., Chen, L.: Direct anonymous attestation. In: ACM CCS (2004)
17. Camenisch, J., Dubovitskaya, M., Neven, G., Zaverucha, G.M.: Oblivious transfer with hidden access control policies. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 192–209. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-19379-8\\_12](https://doi.org/10.1007/978-3-642-19379-8_12)
18. Camenisch, J., Haralambiev, K., Kohlweiss, M., Lapon, J., Naessens, V.: Structure preserving CCA secure encryption and applications. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 89–106. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-25385-0\\_5](https://doi.org/10.1007/978-3-642-25385-0_5)
19. Camenisch, J., Kiayias, A., Yung, M.: On the portability of generalized Schnorr proofs. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 425–442. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-01001-9\\_25](https://doi.org/10.1007/978-3-642-01001-9_25)
20. Camenisch, J., Krenn, S., Lehmann, A., Mikkelsen, G.L., Neven, G., Pedersen, M.Ø.: Formal treatment of privacy-enhancing credential systems. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015. LNCS, vol. 9566, pp. 3–24. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-31301-6\\_1](https://doi.org/10.1007/978-3-319-31301-6_1)
21. 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). [https://doi.org/10.1007/3-540-44987-6\\_7](https://doi.org/10.1007/3-540-44987-6_7)
22. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Persiano, G., Galdi, C. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-36413-7\\_20](https://doi.org/10.1007/3-540-36413-7_20)
23. Camenisch, J., Neven, G., Rückert, M.: Fully anonymous attribute tokens from lattices. In: Visconti, I., De Prisco, R. (eds.) SCN 2012. LNCS, vol. 7485, pp. 57–75. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-32928-9\\_4](https://doi.org/10.1007/978-3-642-32928-9_4)
24. Camenisch, J., Neven, G., Shelat, A.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-72540-4\\_33](https://doi.org/10.1007/978-3-540-72540-4_33)
25. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-45146-4\\_8](https://doi.org/10.1007/978-3-540-45146-4_8)
26. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 319–327. Springer, New York (1990). [https://doi.org/10.1007/0-387-34799-2\\_25](https://doi.org/10.1007/0-387-34799-2_25)
27. Cohen, H.: A Course in Computational Algebraic Number Theory. Springer, Heidelberg (2013). <https://doi.org/10.1007/978-3-662-02945-9>
28. Cramer, R.: Modular design of secure, yet practical cryptographic protocols. Ph.D. thesis, University of Amsterdam (1996)
29. Damgård, I.: On  $\sigma$ -protocols. Lecture Notes, Department for Computer Science, University of Aarhus (2002)
30. Damgård, I., Fujisaki, E.: A statistically-hiding integer commitment scheme based on groups with hidden order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-36178-2\\_8](https://doi.org/10.1007/3-540-36178-2_8)
31. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 60–79. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-34931-7\\_5](https://doi.org/10.1007/978-3-642-34931-7_5)

32. 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). [https://doi.org/10.1007/3-540-47721-7\\_12](https://doi.org/10.1007/3-540-47721-7_12)
33. Fischlin, M.: Round-optimal composable blind signatures in the common reference string model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (2006). [https://doi.org/10.1007/11818175\\_4](https://doi.org/10.1007/11818175_4)
34. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-78967-3\\_3](https://doi.org/10.1007/978-3-540-78967-3_3)
35. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: 40th ACM STOC (2008)
36. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In: 27th FOCS (1986)
37. 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). [https://doi.org/10.1007/978-3-540-78967-3\\_24](https://doi.org/10.1007/978-3-540-78967-3_24)
38. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-45539-6\\_38](https://doi.org/10.1007/3-540-45539-6_38)
39. Kiayias, A., Yung, M.: Group signatures with efficient concurrent join. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 198–214. Springer, Heidelberg (2005). [https://doi.org/10.1007/11426639\\_12](https://doi.org/10.1007/11426639_12)
40. 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). [https://doi.org/10.1007/978-3-642-42045-0\\_3](https://doi.org/10.1007/978-3-642-42045-0_3)
41. 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). [https://doi.org/10.1007/978-3-642-54631-0\\_20](https://doi.org/10.1007/978-3-642-54631-0_20)
42. 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). [https://doi.org/10.1007/978-3-662-53890-6\\_13](https://doi.org/10.1007/978-3-662-53890-6_13)
43. 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). [https://doi.org/10.1007/978-3-662-49896-5\\_1](https://doi.org/10.1007/978-3-662-49896-5_1)
44. 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). [https://doi.org/10.1007/978-3-662-46447-2\\_19](https://doi.org/10.1007/978-3-662-46447-2_19)
45. 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). [https://doi.org/10.1007/978-3-642-10366-7\\_35](https://doi.org/10.1007/978-3-642-10366-7_35)
46. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-29011-4\\_43](https://doi.org/10.1007/978-3-642-29011-4_43)

47. Lyubashevsky, V., Neven, G.: One-shot verifiable encryption from lattices. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 293–323. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56620-7\\_11](https://doi.org/10.1007/978-3-319-56620-7_11)
48. 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). [https://doi.org/10.1007/978-3-642-29011-4\\_41](https://doi.org/10.1007/978-3-642-29011-4_41)
49. 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). [https://doi.org/10.1007/978-3-662-46447-2\\_18](https://doi.org/10.1007/978-3-662-46447-2_18)
50. Paquin, C., Zaverucha, G.: U-prove cryptographic specification v1. 1, revision 3, December 2013
51. Stern, J.: A new identification scheme based on syndrome decoding. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 13–21. Springer, Heidelberg (1994). [https://doi.org/10.1007/3-540-48329-2\\_2](https://doi.org/10.1007/3-540-48329-2_2)
52. Xue, R., Li, N., Li, J.: Algebraic construction for zero-knowledge sets. *J. Comput. Sci. Technol.* **23**(2), 166–175 (2008)

# **Software Security**



# Secure Code Execution: A Generic PUF-Driven System Architecture

Stephan Kleber<sup>1</sup>(✉), Florian Unterstein<sup>2</sup>, Matthias Hiller<sup>2</sup>, Frank Slomka<sup>3</sup>,  
Matthias Matousek<sup>1</sup>, Frank Kargl<sup>1</sup>, and Christoph Bösch<sup>1</sup>

<sup>1</sup> Institute of Distributed Systems, Ulm University, Ulm, Germany  
{stephan.kleber,matthias.matousek,frank.kargl,christoph.bosch}@uni-ulm.de

<sup>2</sup> Fraunhofer Institute for Applied and Integrated Security (AISEC),  
Munich, Germany

{florian.unterstein,matthias.hiller}@aisec.fraunhofer.de

<sup>3</sup> Institute of Embedded Systems/Real-Time Systems,  
Ulm University, Ulm, Germany  
frank.slomka@uni-ulm.de

**Abstract.** In his invited talk, joint between CHES 2016 and CRYPTO 2016 on the Future of Embedded Security, Paul Kocher suggested to move the security into chips because hardware is the lowest level and thus security can not be compromised by a lower layer. In this paper, we propose a generic PUF-driven secure code execution architecture that employs instruction-level code encryption. Our design foresees a tight integration of a Physically Unclonable Function (PUF) and the decryption of encrypted program code directly inside the processor's instruction pipeline to avert revealing keys or decrypted code in externally accessible registers or memory. The architecture prevents code-injection by executing only code encrypted for individual target CPUs, has an adaptable impact on performance, and requires only minor changes to the software development process. Our PUF-based code encryption defends also from reverse engineering attempts and enforces IP protection. A proof-of-concept implementation demonstrates the feasibility of our proposed architecture.

**Keywords:** Secure Execution Environment · PUF · FPGA

## 1 Introduction

Secure and reliable execution of code is of high importance for classical networked systems in the Internet but even more for networked control systems for critical processes. Such Cyber-Physical Systems can be found in industrial plants, in home-automation and in connected cars. In order to avoid negative consequences like misguided process control, one needs to ensure that only valid and unaltered code is executed on such embedded devices. At the same time, such code often contains intellectual property (IP) which makes it necessary to protect the code from extraction and reverse engineering to avoid leaking of IP.

Code injection, especially when performed remotely, is one of the most effective strategies for malicious attackers. Stuxnet, for example, exploited such a remote code execution vulnerability (CVE-2008-4250) to infect remote machines [18]. Since the popular phrack article “Smashing the Stack for Fun and Profit” [24] by Aleph One in 1996, which described simple stack buffer overflows, new detection and prevention techniques like stack canaries or non-executable stacks have been proposed just to soon thereafter be circumvented by more sophisticated attack techniques like Return-Oriented Programming (ROP). Now, more than two decades later, it is still an open security challenge to effectively prevent injection of unauthorized code into an execution environment. A *Secure Execution Environment* (SEE) protects against such *code injection* to prevent malicious actions to be executed inside the environment (ensuring *code integrity*) and additionally, often prevents genuine code from getting extracted out of its execution environment to prevent reverse engineering (ensuring *code confidentiality*). Thereby, an SEE not only contributes to enhanced resilience of an otherwise vulnerable platform against malicious manipulation, but also helps to protect the IP of software and hardware manufacturers. Based on our threat model (Sect. 2), we present our generic PUF-driven secure code execution architecture which allows secure execution of encrypted programs where programs are encrypted per processor instance (Sect. 3). Code that is not properly encrypted for a specific device will not execute correctly, triggering faults. Thus, an attacker will not be able to produce and inject valid code that will exhibit a desired malicious behavior. As an additional effect, relying on a PUF for re-generating decryption keys implements a strict hardware-binding, where code cannot be transferred between devices [11, 26]. At the same time, the architecture will effectively prevent reverse engineering of programs stored in such an embedded system as all such code is encrypted and will only be decrypted directly inside the execution pipeline on a per-instruction basis. Thereby, our design minimizes exposure of decryption keys or decrypted instructions to isolated dedicated registers. The proposed design is comparatively lightweight with a well-defined minimum trusted computing base (TCB) that does not include external or internal memory or caches. The design can be implemented by a standard production process. While implementing the architecture, use-case-specific parameters of freely exchangeable building blocks allow for customization which we discuss in Sect. 4. Based on a proof-of-concept implementation (Sect. 5) we investigate the feasibility of our architecture and its overhead in Sect. 6. The analysis of the state-of-the-art in Sect. 7 shows where our architecture outperforms comparable earlier approaches. In Sect. 8, we conclude that our design constitutes a major step towards code integrity and code confidentiality in embedded devices. Based on the lessons learned, we present a list of extensions and enhancements that might be of interest in future work.

## 2 Threat Model

Our main goal is to protect from code injection attacks, starting from simple execution of untrusted code but also extending to injection attacks like stack

buffer overflows and even advanced attacks like ROP. As additional goal, we aim for code confidentiality, and thus, reliably prevent reverse engineering of code and IP infringements.

For protection from malicious *code injection*, we prevent any non-authorized code to be executed – or rather lead to any behavior that the attacker can control or predict. Beyond, manipulations of the control flow of a process needs to be prevented, as some advanced attacks like ROP can achieve their malicious goals without actually injecting additional instructions.

We consider three types of attacks under this term:

1. Attacks where unauthorized code should be executed in a new process or thread of its own using whatever execution procedure is foreseen on the target platform.
2. We consider attacks where an attacker tries to exploit software implementation flaws like susceptibility to buffer overflows in order to inject self-crafted code into an existing process or thread and manipulate the program flow to execute this code.
3. There are attacks that do not involve any additional instructions but only change the program flow in a way to exhibit new and unintended behavior. ROP is a prime example of this.

Attacks two and three may be performed remotely or locally via the intended input methods of legitimate software running on the target processor. All these attacks are software-centric, however, we do not exclude hardware attackers in our discussion. Hardware attackers have physical access to the processor, its memory and peripherals, and might try to inject code by direct manipulation of memory and any data and instruction cache off- and on-chip. Fault-injection attacks also fall in this category.

In the case of *code confidentiality*, an attacker wants to gain additional knowledge on the application code deployed on a system, e.g., as part of a reverse engineering effort. The system may be under physical control of the attacker, like often the case in an embedded or mobile system running a supplier’s confidential firmware. We assume that all off-chip bus lines can be probed via openly accessible chip pins and that all external memory can be read out.

A more sophisticated attacker may even be able to use hardware probes to read-out from or inject bits into on-chip buses [17]. Beyond, the chip design and layout may be known to the attacker. This results in an attacker capable to obtain memory contents via methods like linear code extraction and direct memory probing. In addition, invasive tampering like glitching or fault injection on bus lines may also be possible. While an attacker with such advanced capabilities may be extremely hard to defend from, the goal of our architecture is to minimize the trusted computing base, i.e., the area of the chip and the duration where such confidential information is exposed.

We note that further attacks like denial-of-service (DoS) and hardware side-channels are not specifically considered within our approach. Complementary research exists, proposing countermeasures to side-channel attacks.



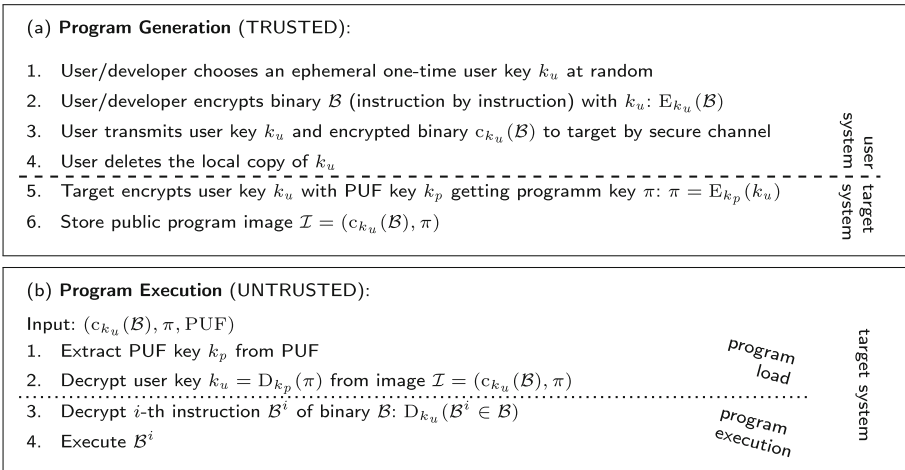
Ascend [10], for example, is specifically designed to hamper side-channel attackers by obfuscating memory access patterns, power consumption and temperature analyses.

### 3 Our Architecture for Secure Code Execution

Execution of program code typically happens detached from its development. Therefore, our architecture distinguishes between a development machine (*user system*) and the execution environment (*target system*). A central advantage of our architecture is that only encrypted program binaries can be executed which are bound to a specific processor and which can be decrypted and run only by that specific processor. To be able to use the benefits of the PUF at execution time without the need to have it available at development time, we separated the encryption of the program binary from binding this binary to the PUF. Therefore, two cryptographic processes have to be discerned: The encryption and decryption of the binary itself is performed by a random access cipher (e.g., XTS [1] or CTR mode), without the use of the PUF (Fig. 1(a), Step 2 and (b), Step 3). In a separate process, the key used for this operation is encrypted and decrypted involving the PUF (Fig. 1(a), Step 5 and (b), Step 2).

The architecture of the target system consists of a generic RISC CPU, a instruction decryption module, and a PUF module. Our CPU uses fixed-length instructions of 32 bit each. The instruction decryption module and the PUF module are closely coupled with the CPU’s execution pipeline.

During program execution, the cryptographic **instruction decryption module** reads the encrypted binary and decrypts it, (Fig. 1(b), Step 3) instruction by instruction, during the instruction fetch (IF) stage. This is performed by



**Fig. 1.** Program encryption and execution process.

a random access block cipher. Decrypted instructions are directly forwarded to the instruction decode stage (Fig. 1(b), Step 4) and continue their normal walk through the pipeline. The binary-specific key  $k_u$  for the block cipher is secured through the use of the **PUF module**, ensuring that any encrypted program can be decrypted only on the device instance it was created for.

**Notation:** Encryption and decryption operations of plaintext  $p$  and ciphertext  $c$  with key  $k$ :  $E_k(p)$  and  $D_k(c)$ ; Encrypted entity or ciphertext  $c$  of plaintext  $p$  with key  $k$ :  $c_k(p)$  is output of  $E_k(p)$ ; Cryptographic hash function:  $H(\cdot)$ .

We call the development machine *user system* and consider it trusted. To generate a program to be executable on the target system, the user compiles confidential source code into a binary using a default compiler, e.g., `gcc`. The user arbitrarily chooses an ephemeral one-time key  $k_u$  (Fig. 1(a), Step 1) and encrypts this binary, instruction by instruction (Fig. 1(a), Step 2). The encrypted binary is then transferred alongside with  $k_u$  to the target system through a secure channel (Fig. 1(a), Step 3). Afterwards, the users local copy of  $k_u$  can be deleted (Fig. 1(a), Step 4). On the target system, the user key  $k_u$  to encrypt the binary gets itself encrypted by the PUF-generated key  $k_p$  (Fig. 1(a), Step 5) yielding the  $\pi = c_{k_p}(k_u)$ . The PUF is available on the target system and the PUF's output can therefore be used as key for this second cipher step.

Each time the program is loaded, this second encryption process is reversed by decrypting the program-specific key into  $k_u$  (Fig. 1(b), Step 1 and 2). This  $k_u$  is used during execution of the according binary to decrypt the individual instructions by the instruction decryption module and is kept directly inside the decryption module. The choice of a suitable cipher mode has to take into account the requirements induced by the program flow. Program flow must generally allow for jumps to enable branching and loops, so random access within the instruction stream is mandatory. Only specific random access modes like CTR mode of a symmetric cipher fulfill the requirement of random access. We discuss security implications, in Sect. 4, and Sect. 6.1.

The roles of the components of the architecture during program generation and program execution are explained in the following sections. Generation and execution processes are dependent on several entities, besides  $k_u$ . Foremost, this is the binary  $\mathcal{B}$ , resulting from the user's compilation of source code.  $\mathcal{B}$  is generic and needs not to be recompiled for use with another processor instance. The binary, encrypted with the key  $k_u$ , is denoted  $c_{k_u}(\mathcal{B})$ .  $k_u$  and  $\mathcal{B}$  have to remain confidential, whereas  $c_{k_u}(\mathcal{B})$  may be public.

### 3.1 Program Generation

Program generation takes place in a trusted environment (Fig. 2a). The user chooses an ephemeral one-time key  $k_u$  for the block cipher and encrypts the compiled  $\mathcal{B}$  with a random access cipher. A program flow is structured in basic blocks. A basic block is a sequence of instructions with exactly one entry point and one exit point, and no branches in between. Each basic block is encrypted to later be decrypted by the hardware instruction decryption module.

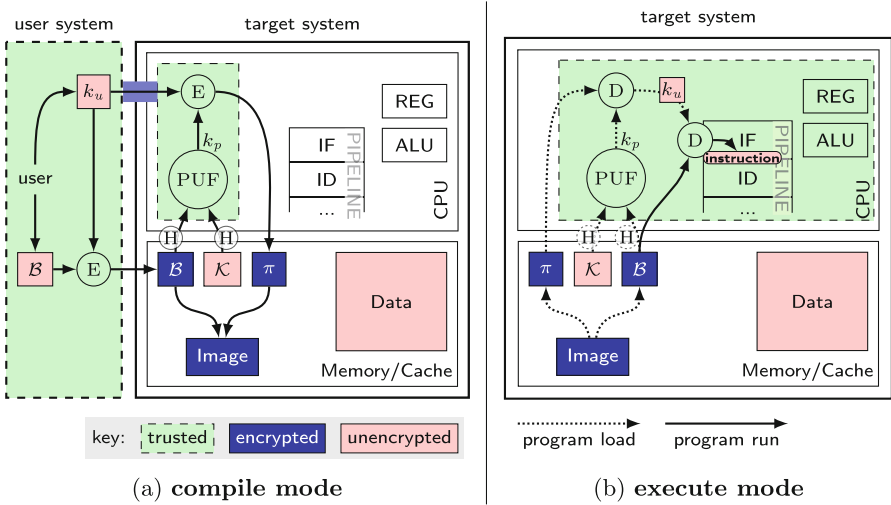


Fig. 2. Our proposed architecture

The program encryption itself is not critical for runtime performance and requires no hardware support. Particularly, it does not need the PUF or any key material from it. We currently implement it as standalone tool, processing the compiled binary in software, but it may be included directly into the compiler.

To finally bind  $\mathcal{B}$  to a hardware instance, inherent properties of the PUF are utilized. This part of the process of program generation is shown in Fig. 2a. The encrypted binary  $c_{k_u}(\mathcal{B})$  (depicted in blue) and  $k_u$  are transmitted to the target system. Since encrypted,  $c_{k_u}(\mathcal{B})$  may be public, but  $k_u$  must remain confidential. To transmit  $k_u$  securely to the PUF module (depicted as blue “tunnel”), a dedicated cable connection can be used during deployment in a secure production environment. For over-the-air (OTA) deployment, a secure channel (e.g., TLS, SSH) is required. The PUF module then generates a cryptographic key  $k_p$  taking the encrypted binary  $c_{k_u}(\mathcal{B})$  and the security kernel  $\mathcal{K}$  as additional input to prevent their modification. A security kernel  $\mathcal{K}$  comprises all basic software components required for secure boot of the system (see also Sect. 3.3).

To protect the user key  $k_u$ —required for program execution—it is encrypted on the target system using  $k_p$ . The encrypted representation  $\pi = c_{k_p}(k_u)$  can be disclosed publicly. It is then stored together with the encrypted binary to form the *program image*  $(c_{k_u}(\mathcal{B}), \pi)$  in publicly shared memory. This scheme has the advantage that the user does not require access to the target hardware for preparation of a binary to be executed on it while retaining the security properties of the PUF.

### 3.2 Program Decryption and Execution

To minimize the TCB, i.e., the parts of the processor required to be trusted, the decryption module is entangled with the instruction fetch (IF) stage of the processor’s pipeline as shown in Fig. 2b. Encrypted instructions enter the stage and are decrypted immediately before proceeding to the instruction decode (ID) stage. Thus, it is possible to encrypt  $\mathcal{B}$  on a per instruction level.

The *nonce and counter generator* prepares the counter as input to the block cipher. It concatenates the base address of each basic block concatenated with an instruction counter inside the block. Whenever there is a jump, a new base address is set and the counter is reset to 0. That way, jumps can only target a start of a basic block, otherwise, the decryption result will be invalid. The module pre-computes keystream blocks by incrementing the counter value and allows modularity in the control flow of the executed code. The end of a basic block is reached when either a jump occurs or the execution linearly continues into the next basic block. In either case, the processor detects the new block and reinitializes the decryption accordingly. The implementation has to take care of resetting the encryption and stalling the processor whenever necessary.

### 3.3 The Role of the PUF

Since encrypted under  $k_u$ , any program image  $\mathcal{I} = (c_{k_u}(\mathcal{B}), \pi = c_{k_p}(k_u))$  for a processor instance may be public. However, the secret key  $k_u$  is required in the target system’s instruction-fetch (IF) logic during execution to generate the decrypted keystream. This is depicted in Fig. 2b, showing how the PUF module decrypts  $k_u$  from the image  $\mathcal{I}$  with help of the PUF key  $k_p$  and then forwards  $k_u$  directly to the decryption. However, a processor instance should not need to store any information about an image; thus,  $\mathcal{I}$  has to be self-contained. This is accomplished by recovering  $k_u$  from the encrypted  $\pi$  inside the program image and  $k_p$  each time a binary is loaded for execution. Only the very same processor instance can generate the correct  $k_p$  to restore  $k_u = c_{k_p}(\pi)$ .  $k_p$  has to be derived from the target system’s PUF, using hashes of the encrypted binary and the security kernel ( $H(c_{k_u}(\mathcal{B})), H(\mathcal{K})$ ) as parameters. This way  $k_p$  is not only bound to the device but also to the integrity of the program binary  $\mathcal{B}$  and the security kernel  $\mathcal{K}$ . If  $\mathcal{B}$  or  $\mathcal{K}$  is modified, the derived key  $k_p$  changes and cannot be used to decrypt the user key  $k_u$  anymore. A security kernel  $\mathcal{K}$ , like the boot loader or the operating-system core, guarantees secure access to system resources.

To securely combine all parameters of the PUF module into one key, we propose a key derivation function (KDF) like presented by Krawczyk [16]. From the PUF output  $s_p$  and the key-derivation parameters, the PUF module calculates:

$$k_p = \text{KDF}(s_p, H(c_{k_u}(\mathcal{B})), H(\mathcal{K}))$$

During **loading** of  $\mathcal{B}$ ,  $k_p$  is derived once and used to recover  $k_u = c_{k_p}(\pi)$ . In the IF stage while **executing**, the  $i$ th instruction  $\mathcal{B}^i$  of  $\mathcal{B}$  gets decrypted by:

$$\text{Cipher}_{k_u}(\text{nonce}||i) \oplus c_{k_u}(\mathcal{B}^i) = \mathcal{B}^i$$

Using a PUF, we do not need any secure storage across all memory hierarchies in our system and thereby minimize the secure and trusted computing base.

## 4 Customization for an Implementation

A number of security-relevant building blocks exist in our architecture, for which concrete crypto-primitives have to be chosen during an implementation. The building blocks that an implementation has to define are: (1) the PUF-type, (2) the KDF in the PUF-module, (3) the block-cipher for the user-key en- and decryption, (4) the cipher-mode for the instruction decryption, and (5) the block-cipher for the instruction decryption.

The selection of crypto primitives is depending on the desired level of security and performance for the use case of an implementation. A large number of options exist for this customization since almost any combination of block-ciphers, random-access cipher modes, KDFs, and PUFs is possible. Without loss of generality, we provide examples for a suitable selection of crypto-primitive sets for each building block in Table 1. All alternatives are a trade-off between the level of security and performance or chip-area. Hardware implementations of ciphers with high throughput, typically, need more area. Low latency ciphers reduce the performance impact, while they, typically, are more susceptible to attacks [21].

**Table 1.** Examples of crypto-primitives for an implementation.

Building block	Example 1	Example 2	Example 3
PUF-type	Ring-Oscillator	SRAM	Bistable-Ring
Key derivation (KDF)	SHA-256 of concatenated keys	HKDF	HMAC-SHA-256
Block-cipher (user-key)	AES-128	PRINCE	AES-256
Cipher-mode (instr. decr.)	CTR	XTS	LRW
Block-cipher (instr. decr.)	AES-128	PRINCE	SIMON

The throughput of block-ciphers in hardware can be increased by parallelizing cores, but occupying more area. Alternative low-latency block ciphers, like PRINCE [6] or SIMON [4], may improve performance and reduce area overhead, at the cost of reduced security as evaluated by Maene et al. [21]. Alternative random access cipher modes are understood from disc encryption, where an encryption block is not chained with the previous plain-text block. Candidates for the usage in our architecture are the block-cipher modes devised by *Liskov, Rivest, and Wagner* (LRW) [20], being an improved security-complexity trade-off compared to CTR, or the *XEX-based tweaked-codebook mode with ciphertext stealing* (XTS) [1].

## 5 Proof-of-Concept Implementation

To demonstrate the feasibility of our architecture, we implemented a proof-of-concept (PoC) of our design capable of creating and executing encrypted standalone program images. We call our PoC implementation the *Secure Execution PUF-based Processor* (SEPP). For the cryptographic primitives for the customizable parts, we decided for a conservative selection of thoroughly tested implementations of well-known algorithms (Table 1, Example 1).

SEPP is based on an OR1200<sup>1</sup> processor, an implementation of the OpenRISC OR1000 architecture. It is a popular open-source RISC architecture with a 32 bit wide fixed-length instruction set and a five stage, single issue pipeline. We chose an architecture with fixed-length instructions in order to be able to always map whole instructions to cipher blocks without overlap and alignment issues. The key parameters of our system are: Architecture: 32 bit RISC; clock frequency: 50 MHz; instruction cache: 8 Kbyte, 1-way direct-mapped; data cache: 8 Kbyte, 1-way direct-mapped; Memory: 128 MB DDR2 SDRAM.

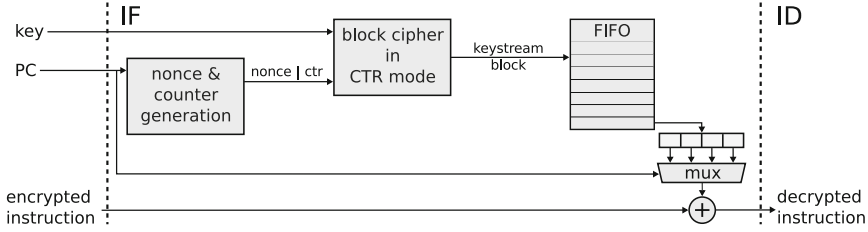
We extended the OpenRISC Reference Platform (ORPSoC) by two modules according to our architecture: the PUF module and the instruction decryption module. We deployed our design on a Digilent Atlys Board powered by a Xilinx Spartan-6 LX45 FPGA. As development environment we used Xilinx ISE 14.7. For synthesis, placement, and routing parameters, as well as FPGA-dependent settings, like operating frequency, we used the defaults distributed with the ORP-SoC make files. This enables us to directly compare SEPP to ORPSoC.

The RO PUF [23] implementation we utilized generates a single fixed response by comparing the oscillating frequencies of ring oscillator pairs. To counter the noise of the PUF output, the C-IBS fuzzy extractor [12] (see Sect. 7) takes the PUF output and creates helper data to reliably recreate a response. The corresponding helper data can be stored in memory since an attacker gains no advantage from it. The output of the C-IBS fuzzy extractor we call  $s_p$ , the reliable, embedded *PUF secret*. The PUF module encapsulates the RO PUF with the C-IBS fuzzy extractor and a low area AES core in ECB mode for encryption and decryption of  $k_u$  on load of the program image. Our PoC implements hashing and combining of the encrypted binary  $H(c_{k_u}(\mathcal{B}))$  and the security kernel  $H(\mathcal{K})$  in software. The hardware PUF module then combines this result with the PUF secret  $s_p$  to create the PUF key  $k_p$ . For  $s_p$  to remain confidential, there is no external interface to access the outputs of the PUF or the fuzzy extractor.

The CPU is interfaced with the PUF module over a dedicated class of special purpose registers. Those include configuration and status registers as well as the input for the combined hashes ( $H(c_{k_u}(\mathcal{B})), H(\mathcal{K})$ ). It is important to note that in program execution mode,  $k_u$  is directly forwarded to the decryption module over a separate connection and is not visible on any system bus.

The instruction decryption module is integrated in the processor's instruction fetch stage and has no dedicated interface other than the input of  $k_u$  from the PUF module. As shown in Fig. 3, the decryption module evaluates the Program

<sup>1</sup> <http://opencores.org/or1k/OR1200.OpenRISC.Processor>, accessed 03/13/2017.



**Fig. 3.** Simplified structure of the decryption module

Counter (PC) to detect branches and sets the nonce and counter accordingly. The program flow from one basic block to the other is marked by a custom instruction, if the block has no preceding branch instruction. These marker instructions are added prior to the encryption on the user system by a small compiler add-on. Some instructions take more than one processor cycle to execute. Thus, a first-in, first-out (FIFO) buffer is added to store the produced blocks of the keystream until they are needed by the processor. Since a keystream block produced by the AES-128 cipher can cover four 32 bit instructions of the CPU, an additional multiplexer following the FIFO is used. It selects the correct part of the keystream block depending on the current PC. This portion of the keystream block is XORed with the incoming encrypted instruction and the decrypted instruction is forwarded to the instruction decode stage.

The AES decryption core, for instruction decryption in CTR, should provide a throughput sufficiently high to allow sequential code to be decrypted and executed without stalling. We evaluated several freely available 128 bit-key AES cores and selected one<sup>2</sup> with enough throughput (in *keystream blocks per cycle*) for uninterrupted execution.

For an optimal trade-off between FPGA resources, lowest latency, and throughput, we designed a parallel combination of four 13-cycle encrypt-only AES cores. In CTR mode, the AES core has to support only encryption. Using this design, the processor needs to be stalled only once at the beginning of each basic block for 13 cycles until the first portion of keystream for a basic block is computed.

The whole SEPP implementation on the Spartan-6 FPGA requires 20,643 Slice LUTs overall and 11,321 Registers, compared to 12,542 LTUs and 6,752 Registers of the baseline ORPSoC. Most of the additional Slices are required for the PUF and a significant amount for the AES cores. The control logic and interfaces require only few additional resources. The required resources seem to be a significant overhead, but recall that all building blocks are exchangeable and nothing except for SEPP has to run on the device.

The Universal Bootloader *Das U-Boot* (or U-Boot)<sup>3</sup> is used as our software platform. It was modified to implement the functionality of the target system

<sup>2</sup> [http://opencores.org/project,aes\\_core](http://opencores.org/project,aes_core), accessed on 03/13/2017.

<sup>3</sup> <http://www.denx.de/wiki/U-Boot>, accessed on 03/13/2017.

security kernel, i.e., the generation and execution of encrypted program images in interaction with the PUF module. It provides a simple command line interface to the user, for which we added custom commands. RSA public key cryptography is used by U-Boot to establish a secure communication channel between the user system and the target system. The KDF calculation for the PUF is implemented as part of our security kernel.

Program encryption is independent from the PoC hardware and is handled at the user system. In our case, the user system is a common Linux system for which we developed helper tools that analyze the compiled binary, encrypt the code, and package it together with the RSA-encrypted  $k_u$  in an U-Boot image. The user transfers such an encrypted program image to the PoC system over an Ethernet connection. There  $\pi$  is generated and replaces the  $k_u$  embedded in the image. This image can now be securely made public as it only contains encrypted code and the encrypted key  $k_u$  in form of the public  $\pi$ . The encrypted image is tied to this exact hardware instance and only this processor with its unique PUF is able to decrypt and execute it.

## 6 Discussion

### 6.1 Security

Through the use of a PUF, there is no unsecured key material that needs to be stored persistently. Moreover, our architecture is designed to minimize the TCB, so only the PUF, registers, pipeline, and the ALU need to be trusted. In contrast, previous approaches needed to trust all on- and off-chip bus lines and all memory and caches, as presented in Sect. 7. However, invasive hardware attacks against our architecture, during program execution, could allow for access to bus lines between PUF, registers, pipeline and ALU or those components themselves. The most prevalent hardware attacks on processors currently target off- and on-chip memory bus lines [17]. Although sophisticated and very costly, these attacks allow for read-out and injection—mostly fault injection—of data bit by bit. This class of attacks is countered with meshes and other sophisticated hardware countermeasures, e.g., in smartcard processors. The authors of XOM, AEGIS, OASIS, and Ascend note that a variety of methods exist to prevent or impede hardware tampering, e.g., probing or fault-injection. We agree that such means will become necessary as soon as this class of attacks should become feasible for processors in CMOS technique. In such a case, costly hardware attack countermeasures can be applied to our architecture much easier, compared to securing the whole SoC, including caches and any kind of memory. This follows from the small chip area to be trusted, compared with previous designs. Figure 2a and 2b show the trusted area in the design in green color. This provides an unprecedented option to reduce production cost for this kind of hardware security measure. To confirm the security of our design, we consider both parts of the threat model (Sect. 2) *code confidentiality* and *injection prevention* separately.



**Code Confidentiality.** To achieve code confidentiality, the binary image is encrypted. The immediate key necessary to decrypt the binary is  $k_u$ . Knowing a specific  $\pi = E_{k_p}(k_u)$  and having available the encrypted binary  $c_{k_u}(\mathcal{B})$  and the security kernel  $\mathcal{K}$  as context input for key derivation [16],  $k_u$  can be recovered using the correct PUF instance to derive  $k_p$ . For key derivation, we require the KDF to be constructed using keyed cryptographic hash functions  $H(\cdot)$ . Their second pre-image resistance prevents finding another binary that produces the same output of the KDF. The KDF also breaks the direct link between user inputs and  $k_p$  which could otherwise be exploited. Thereby, it prevents known-plaintext-attacks on any of the inputs and possibly the output of the en- or decryption. In addition, a KDF is resilient against length extension attacks. Only  $k_u$  and  $k_p$  need to remain confidential.  $k_p$  never leaves the PUF module, whereas  $k_u$  originates externally. Therefore, the confidentiality of a program is determined by the confidentiality of  $k_u$  outside of the processor’s instruction decode module.

The only time where  $k_u$  is required outside of the processor is for the user to encrypt the binary after compilation. Provided the adversary model and scope of our approach, the only attack vector arises during the necessary transfer of  $k_u$  from the user machine to the target machine. Therefore, we require a secure channel between those two endpoints. This can be ensured during deployment. Only a low bandwidth is necessary for this channel, since the key is small compared to the binary. We assume that an update mechanism implemented in trusted software is possible and propose an approach to be explored in follow-up work in Appendix A. The user machine itself must be secured and trusted. This can be addressed by known means for host security and is out of the scope of this work.

**Injection Prevention.** The prevention of code injection is accomplished by executing only correctly encrypted code.  $k_p$  is required to recover  $k_u$  necessary to en- and decrypt code of a specific program. Thus, security relies only on the confidentiality of  $k_p$ . Provided the threat model, no other kind of trust needs to be assumed.  $k_p$  can only be generated by the correct PUF instance with an unmodified binary  $\mathcal{B}$  and an unmodified security kernel  $\mathcal{K}$  as input. Blocking the key generation data path in the processor—for ultimate security demands by a hard-wired switch, otherwise by a privileged instruction—the generation of a new valid  $\pi = E_{k_p}(k_u)$  for any program can be prevented. Since  $k_p$  is dependent on  $c_{k_u}(\mathcal{B})$ , no new code using a valid  $\pi$  can be generated. This way, arbitrary code injections, like buffer overflows, are effectively prevented. Because of the per-basic-block encryption, the architecture severely limits an attacker’s ability to construct useful return oriented programming (ROP) gadgets. Therefore, it effectively prevents malicious control flow manipulation, even for code that is vulnerable on a conventional architecture.

Our SEPP implementation does not use any signatures of individual instructions in  $\mathcal{B}$  to check code integrity. Instead it relies on the failure to decrypt a *valid* CPU-instruction at runtime, when an attacker injects a guessed encryption of an

instruction. In the event of an attack, this exception can gracefully be dealt with by defining an exception handler in the processor that returns the control flow back to the trusted  $\mathcal{B}$  or even  $\mathcal{K}$ . In the context of instruction set randomization, Barrantes et al. [3] established that the guessing of a valid instruction is a sufficiently large obstacle for an attacker: For the PowerPC architecture, Barrantes et al. conclude that escapes are successful in less than 5% of all cases. Since OpenRISC has a smaller instruction set than the PowerPC, this makes it even less likely for random bytes to be interpreted as a valid instruction. We estimate that a typical binary contains 10% branching targets to which the execution can jump. Combined with the findings of Barrantes et al., the probability for a successful instruction escape drops to below  $10\% \cdot 5\% = 0.5\%$ . Thus, to targetedly inject 5 instructions in a row would succeed in only  $3 \cdot 10^{-12}$  of all cases. To run a meaningful attack, the attacker needs to control more than one instruction in sequence. Therefore, only relying on the decryption of invalid instructions is superior to validated encryption, under most *practical* circumstances, when taking runtime performance into account, due to the high probability that executing random bytes results in an exception [3]. For high security applications, the used cipher can be replaced by a lightweight authenticated and verifiable encryption scheme, e.g., ALE [5].

To prevent the recovery of the true  $k_u$  from  $\pi$  for a tampered-with binary, the PUF-based key derivation requires  $c_{k_u}(\mathcal{B})$  as input to decrypt  $\pi$  into  $k_u$ . Moreover, we rely on the failure to correctly recover  $k_u$  at the *time of use* if there had been any tampering with the binary since the *time of check*. It is possible to targetedly change a selected portion of a binary in memory, after it has been read to generate the decryption key. This, however, does not lead to a practically exploitable time-of-check time-of-use (TOCTOU) attack: An attacker is not able to discern which instruction actually failed during the manipulated binary’s execution and for what exact reason. So (s)he can just resort to blind guessing attacks with the success probability of breaking the encryption scheme itself. The attack, therefore, does not scale since each single instruction needs to be reverse engineered again by guessing. The encryption scheme ensures that the same instruction at another memory location is not discernible for the attacker this way, either.

In our PoC implementation, we use the well-known CTR mode for random access to instructions during execution. We are aware of a weakness in CTR that may be exploited for a theoretical attack under rare circumstances. With considerable effort of brute-forcing opcodes and in combination with the mentioned tedious TOCTOU, an attacker could substitute single instructions. Even if successful, the usage of the PUF prevents the predictability of the keystream for other binaries and other devices. Therefore, this attack does not scale to multiple target machines. If even stricter security properties are desired, CTR readily may be exchanged for other random access modes in an own implementation. As pointed out in Sect. 4, CTR is only one of the options as a suitable crypto-primitive for an implementation.

## 6.2 Performance

The PoC shows that our enhanced architecture can be implemented with identical clock rate as the baseline processor. Therefore, the performance of SEPP is identical to the baseline system with the exception of the decryption. We therefore concentrate our discussion on the performance impact inflicted by decryption. Decryption latency and bandwidth potentially impact our architecture’s performance twofold:

First, in a sequential stream of instructions, the processor would be stalled if the bandwidth of the decryption module would be smaller than the processor’s bandwidth; we call this *execution latency*. As described in Sect. 5, our PoC demonstrates that a hardware-implementation is possible with a decryption-throughput high enough to completely prevent inflicting any execution latency. Second, upon each jump, the processor has to be stalled until the newly fetched instruction is decrypted, causing the so-called *warm-up latency*  $lat_w$ . Block cipher modes of operation, like the AES-CTR we used in our PoC, requires a couple of cycles to warm-up. The number of cycles, before the cipher yields the first block of output, depends on the implementation of the block cipher. Consequently, the only remaining occurring performance impact during execution is the warm-up latency at the beginning of basic blocks.

**Calculating the Performance Penalty:** Since the warm-up latency occurs at branching instructions, the overall performance penalty is dependent on the control flow of the program. We can calculate the runtime penalty by normalizing the number of clock cycles required to execute an encrypted program on the SEPP platform to the number of clock cycles required to execute an unencrypted program on the baseline platform:

$$runtime\ penalty = \frac{IC \cdot CPI + BIC \cdot lat_w}{IC \cdot CPI},$$

where  $IC$  denotes the total number of executed instructions of a program and  $CPI$  the average *clock cycles per instruction*. The values of  $IC$  and  $CPI$  are identical for SEPP and the baseline processor. Therefore, the overhead can be calculated by  $BIC \cdot lat_w$  as the product of the number of branching instructions ( $BIC$ ) and the warm-up latency in clock cycles ( $lat_w$ ).

The AES implementation in our PoC system requires a warm-up latency of 13 clock cycles. Moreover, every processor has a latency induced by each memory access, for OR1200 and SEPP this is 5 cycles on average. The average memory access latency and  $CPI$  are rough estimations based on the OR1200’s data sheet and system simulations. For a hypothetical program with 1 mio. instructions, 10% branching instructions and a  $CPI$  of 1.5, the runtime penalty calculates to 59% of SEPP compared to the baseline.

We validated this theoretical calculations by reducing the number of jump instructions in a actual binary from 14.3% to 5.3%. Compared to the baseline system, the performance penalty decreases from 84% to only 34% for our PoC measurements. This clearly demonstrates the impact of branch and jump

instructions on the execution time of encrypted programs. Appendix B contains details about the practical performance measurements which we conducted.

**Remark:** Recent advances in the design of hardware block ciphers promise a significant reduction of the warm-up latency. Since our architecture is not restricted to AES-CTR, but allows for arbitrary random access modes of suitable block ciphers, other ciphers can easily be substituted for AES. Low-latency block ciphers like PRINCE [6] can perform encryption in a single cycle at 14 to 15 times less area cost than AES-128. With an estimated average of 5 cycles per memory access, the performance penalty inflicted by the warm-up latency drops below the memory access latency. It needs to be determined at which length of the critical path the yielded throughput allows for uninterrupted execution. However, we argue, that such an implementation of the processor would come without any performance penalty whatsoever, compared to the baseline, but at the cost of a lower security level [21].

## 7 Related Work

**Secure Code Execution.** To our knowledge, there is no previous description of a system design that embeds security as deep into a chip as our architecture. This kind of architectures have been termed Isolated Execution Environment (IEE) [25]. Several recent works on IEEs [10, 25, 28, 29, 31] focus on confidentiality of processed user data and in addition, aim to minimize the side channel attack surface of processors. However, comparatively few papers address the confidentiality of application code which is the goal of our design. For these approaches, the assumed attacker capabilities vary substantially.

The *eXecute-Only Memory* (XOM) architecture [19], for example, considers main memory to be insecure but on-chip memory like caches to be secure. All data including code is encrypted when it leaves the cache and decrypted when it is brought back from main memory. It has been shown that this leads to substantial latency issues of memory accesses which is enhanced by Yang et al. [29]. In contrast, our approach considers all memory and caches as untrusted and thus stores only encrypted code.

*Ascend* [10] is a secure CPU architecture designed to obfuscate input and output signals on the processor’s pins. The architecture has been extended into Stream-Ascend [31] to overcome Ascend’s rather harsh limitations on the processors interactions with the outside world. Ascend’s main goal is to obfuscate and minimize side channels due to access timings of off-chip data transfers. The capabilities of Ascend can be considered orthogonal to our architecture.

The *AEGIS* secure processor [27, 28] is the first attempt to utilize the challenge-response behavior of a PUF for an IEE. Like the XOM processor, AEGIS encrypts only main memory using a similar OTP encryption scheme. The encryption keys are derived from the embedded PUF. Applications can switch the processor to different secure execution modes to match the current security demands. This is very flexible and improves performance compared to full

program encryption but requires careful consideration by the application’s programmer, which makes porting existing software to AEGIS a non-trivial task. In addition, AEGIS requires extensive compiler and OS support, as well as modified hardware like a custom memory controller. Compared to AEGIS, our approach aims for a smaller TCB where code is only decrypted directly in the execution pipeline. As we show, we provide better compatibility with existing code.

*OASIS* is an instruction set extension for secure CPUs which provides an IEE for secure execution and remote attestation [25]. All cryptographic keys are bootstrapped from a PUF secret generated by an SRAM PUF [11]. Data confidentiality and integrity is established by encrypting program data with keys bound to the program code. However, unlike our approach, *OASIS* does not encrypt the code itself.

*Intel SGX* provides security enclaves for isolated and secure execution of programs [9]. To provide and manage enclaves, a large TCB is necessary, encompassing not only the processor logic itself. *SGX* assumes that everything on-chip is trusted, including caches and memory management. To provide confidentiality and integrity for data residing outside of the processor, it requires a complex Memory Encryption Engine. Thus, *SGX* does encrypt program code on transfer to the external memory, but it is decrypted transparently on access such that data and code is available in clear-text within the processor. The separation between enclaves, to prevent them from reading data from each other, is hardware-supported but managed completely by the security kernel in software. So the TCB is further enlarged by the necessity of a complex security kernel.

In summary, most presented approaches are data-centric and attempt to secure communication to and from the processor by introducing an additional encryption or obfuscation layer. They essentially localize security at the memory interface. Once an attacker gains access to CPU internal caches or pins, instructions are unencrypted and can be read out or modified.

In our architecture, we propose a deeper embedding into the execution pipeline where code remains encrypted in all memory and caches—even the instruction cache. Previous architectures come at the price of a relatively large TCB, including most of the processor and memory attachment, in some cases even large portions of software with parts of the operating system. This requires modified compilers, with significant customizations. Thus, the programming models of such concepts differ significantly from conventional ones.

Our design aims at minimizing such obstacles to the adoption of a secure IEE, so that it can be integrated much easier into existing environments.

**Physical Unclonable Functions and Fuzzy Extractors.** Storing encryption keys in memory inside or – even worse – outside of the CPU is a source of potential vulnerabilities, as attackers may succeed in extracting them, e.g., by the use of cold-boot attacks. In our design, we utilize the physical diversity of chip hardware to deduce a device-unique key that does not need to be stored in expensive secure non-volatile memory.

*Physical unclonable functions* (PUFs) evaluate manufacturing variations in integrated circuits to derive unique secrets inside a device to generate

cryptographic keys or authenticate a device in a challenge-response protocol [2, 15]. Instead of storing secrets permanently, PUFs reveal their secret only during runtime. Popular PUF types for key generation are the SRAM PUF [11] and the Ring-Oscillator (RO) PUF [23].

The secrets derived from PUFs are noisy and affected by environmental conditions such that they require additional error correction in form of a so-called Fuzzy Extractor or Helper Data Key Extractor. Helper data is generated to map the random PUF responses to codewords of an Error-Correcting Code (ECC), thereby eliminating the variation in the PUF responses. Implementations of related approaches are based on the Code-Offset construction [7], the Syndrome construction [22] or Differential Sequence Coding [13, 14]. In this work, we use a Complementary Index-Based Syndrome coding (C-IBS) RO implementation [12] which is an extension of IBS [30]. The implementation contains a small Reed-Muller code with GMC soft-decision decoding [8] as ECC.

## 8 Conclusions and Future Work

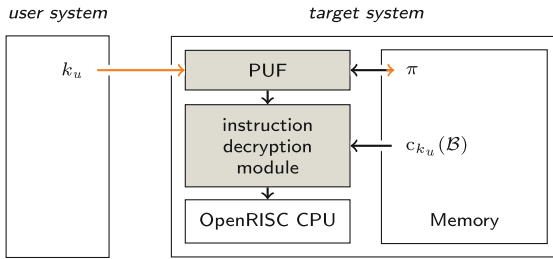
In this paper, we presented a PUF-based secure code execution architecture to prevent reverse engineering of programs and to counter injection of malicious code into memory and cache to enhance the security of embedded devices. Deploying confidential code on our platform provides for additional application security and an enhanced protection of intellectual properties of the software developer. Novelties of this architecture are to embed a PUF-based decryption module, as deep as into the instruction fetch stage of the CPU's pipeline. This allows for a significantly smaller trusted computing base than previous designs of a secure processor. By employing a PUF we devise a new cryptographic protocol to allow binding of decryption to a target hardware instance. Encryption remains possible at a trusted development machine without access to the target. Despite these benefits, the architecture's minimal impact on the software development process simplifies application migration.

For our proof-of-concept implementation we used AES in CTR mode as a widely accepted mode of operation. Like all employed crypto-primitives, AES-CTR mode can be exchanged for a number of other block cipher modes suited for any individual security demands and addressing practical requirements. Depending on the acceptable area overhead, branching penalties may be removed almost completely by doubling the block cipher cores and compute two keystreams for both control flow branches in parallel. The flexible architecture we proposed allows for the customization of any implementation to meet the individual requirements of the specific use-case, for which our PoC implementation (SEPP) is one example. We leave it to future work to evaluate the differences of multiple incarnations of our architecture besides SEPP, regarding security impact, performance penalty, and chip-area requirement. Ultimately, this ought to result in an ASIC implementation for productive use.

Our SEPP showed, that direct and immediate placement of the PUF-driven instruction decryption, right into the instruction fetch stage, can be realized

and allows reasonable performance. Compared to related work evaluated on an FPGA, our SEPP has a similar overall performance penalty. At the same time, our architecture allows SEPP to reach the envisioned stricter security goals by moving the security deeper into the hardware, as proposed by Paul Kocher at his CRYPTO/CHES 2016 talk on the Future of Embedded Security. The reduced attack surface prevents the success of any kind of practical attack for reverse engineering and injecting code on memory and caches – even the instruction cache – which related secure architectures do not address in particular. SEPP’s architecture achieves these properties with minimal impact on the development of software.

## Appendices



**Fig. 4.** SEPP architecture building blocks overview (orange arrows: key generation, black arrows: execution) (Color figure online)

### A Deployment of Software Updates

Programs are initially flashed to a SEPP-powered System-on-a-Chip (SoC) at deployment time by the user via a secure connection to the device, e.g., a dedicated cable. To enable software updates in the field, an encrypted binary  $c_{k_u}(\mathcal{B})$ —initially deployed in a secure environment as stated above—may contain an *update function*. This update function needs to be able to take a new user-key  $k_u'$  and a new encrypted binary  $c_{k_u'}(\mathcal{B}')$  and feed it into the PUF-driven executable-image generation process. SEPP will return  $\pi' = c_{k_p'}(k_u')$  to be packaged into a new image  $(c_{k_u'}(\mathcal{B}'), \pi')$  for its current instance. A user sends  $c_{k_u'}(\mathcal{B}')$  and  $k_u'$  to this function; the encrypted  $c_{k_u'}(\mathcal{B}')$  can be transmitted via an untrusted network connection, while  $k_u'$  needs to be transmitted securely. To provide this secure channel between user and  $\mathcal{B}$  running on the SEPP device, the update function already deployed in the trusted binary  $\mathcal{B}$  must contain a suitable encryption scheme, e.g., RSA-based such as TLS or SSH. Due to the small key size, the secure channel for  $k_u'$  requires only a low data rate.

The PUF-driven executable-image generation process of SEPP, generating  $\pi = c_{k_p}(k_u)$ , has to be deactivated physically after deployment of the binaries for *ultimate* security demands. This completely prohibits the generation of any new executable sequence of instructions for this instance of SEPP and therefore

completely prevents injections. For the over-the-air update, this security feature has to be deactivated as trade-off for an update path. The executable-image generation process of SEPP alternatively can only be triggered by a privileged instruction. This restricts its usage to the security kernel, however shifting the responsibility into software to decide what trusted code is allowed to generate new code.

## B Practical Performance Measurements

Benchmarks and tests were conducted on our prototype implementation. We compare the test results with our prototype’s base platform, the OR1200 CPU, running on the same FPGA board as SEPP’s prototype implementation. In order to demonstrate the performance impact, we developed a number of custom tests with known parameters such as the number of branching instructions. These custom tests were all compiled without any compiler optimization in order to ensure deterministic outcomes. The results of our tests are summarized in Table 2.

**Table 2.** Results of custom tests executed on prototype implementation

		$\frac{2}{3}$ of jumps removed	manual loop unroll
number of branching instructions	unmodified	14.3%	9.8%
	modified	5.3%	3.8%
baseline system			
iterations per second	unmodified	909,091	138,122
	modified	1,063,830	183,824
speed-up factor		1.17	1.33
SEPP			
iterations per second	unmodified	495,050	85,690
	modified	793,651	157,233
speed-up factor		1.6	1.83
runtime penalty compared to baseline system	unmodified	1.84	1.61
	modified	1.34	1.17

By reducing the number of jump instructions from 14.3% to 5.3% through the removal of a function call within a loop, compared to the baseline system, the performance penalty decreases from 84% to 34%. This clearly demonstrates the impact of branch and jump instructions on the execution time of encrypted programs.



Loops generally introduce a substantial number of jumps into program execution. To verify this assumption, we manually unrolled a loop of a short example program. The number of branch instructions thereby decreased from 9.8% to 3.8%, reducing performance penalty from 61% to 17% for our custom test cases.

Furthermore, we performed CoreMark<sup>4</sup> benchmarks, to show the influence of unrolling loops and to enable comparison with future developments and other platforms. We conducted the benchmark both on our system in encrypted form as well as on the baseline architecture in unencrypted form. We compare the benchmark compiled with GCC's optimization level `-O3` with another version that is compiled with additional loop unrolling optimization, enabled with the flag `-funroll-all-loops`. Compilation with `-O3` results in a 48.8% penalty of SEPP, and enabling unrolling reduces this to 43.5%. While the baseline processor only speeds up by 9% with the additional `-funroll-all-loops`, SEPP's execution speed gains 21%. This confirms that the reduction of branching instruction benefits SEPP greatly. These benchmark comparisons prove the expected effects of our instruction-level decryption, specifically the warm-up latency on branching, for actual calculations.

Unfortunately, the comparison of our prototype with related implementations is not straightforward, as authors in this field use a variety of methods for performance assessment. The AEGIS developers used the SPEC2000 CPU<sup>5</sup> benchmark suite [27], which is not freely available. Lie et al. [19] provide a theoretical performance analysis for their XOM architecture [19]. The OASIS instruction set extension is not directly comparable to SEPP, as the authors only give absolute time overheads for specific platform operations [25].

The AEGIS developers state that degradation can be as high as 60%; while it mostly stays below 40%. XOM's slow down is less than 50% according to the authors' calculations. Given this comparison, we conclude that the runtime penalty of SEPP lies well within worst case, best case, and average for comparable systems. We expect a significant performance advantage of SEPP over the compared platforms when the described improvements are implemented in future work.

We are not able to compare SEPP to previous work in respect to FPGA resources overhead for the lack of information about properties of those approaches: XOM is only an architectural design without a hardware implementation [19]. OASIS was only simulated in software [25]. Although, an FPGA implementation of AEGIS was evaluated [28] no information about their FPGA resource requirements were given. Its resource overhead was only determined by an ASIC synthesis. The ASIC area overhead of AEGIS was given as being roughly 90% larger than their baseline. We consider this value not to be directly comparable to the roughly 68% Slices overhead of SEPP.

---

<sup>4</sup> <http://www.eembc.org/coremark> accessed on 27/02/2016.

<sup>5</sup> <https://www.spec.org/cpu2000/>, accessed on 27/02/2016.

## References

1. Ahmed, S., Samsudin, K., Ramli, A.R., Rokhani, F.Z.: Effective implementation of AES-XTS on FPGA. In: 2011 IEEE Region 10 Conference (TENCON) (2011)
2. Armknecht, F., Maes, R., Sadeghi, A.R., Standaert, F.X., Wachsmann, C.: A formal foundation for the security features of physical functions. In: S&P. IEEE (2011)
3. Barrantes, E.G., Ackley, D.H., Forrest, S., Stefanović, D.: Randomized instruction set emulation. *ACM Trans. Inf. Syst. Secur.* **8**(1), 3–40 (2005)
4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: DAC (2015)
5. Bogdanov, A., Mendel, F., Regazzoni, F., Rijmen, V., Tischhauser, E.: ALE: AES-based lightweight authenticated encryption. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 447–466. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-43933-3\\_23](https://doi.org/10.1007/978-3-662-43933-3_23)
6. 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). [https://doi.org/10.1007/978-3-642-34961-4\\_14](https://doi.org/10.1007/978-3-642-34961-4_14)
7. Bösch, C., Guajardo, J., Sadeghi, A.-R., Shokrollahi, J., Tuyls, P.: Efficient helper data key extractor on FPGAs. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 181–197. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-85053-3\\_12](https://doi.org/10.1007/978-3-540-85053-3_12)
8. Bossert, M.: Channel Coding for Telecommunications. Wiley, Hoboken (1999)
9. Costan, V., Devadas, S.: Intel SGX Explained. *IACR* **2016**(86), 1–118 (2016)
10. Fletcher, C.W., Dijk, M.V., Devadas, S.: A secure processor architecture for encrypted computation on untrusted programs. In: STC. ACM (2012)
11. Guajardo, J., Kumar, S.S., Schrijen, G.-J., Tuyls, P.: FPGA Intrinsic PUFs and Their Use for IP Protection. In: Paillier, P., Verbaudhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 63–80. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-74735-2\\_5](https://doi.org/10.1007/978-3-540-74735-2_5)
12. Hiller, M., Merli, D., Stumpf, F., Sigl, G.: Complementary IBS: application specific error correction for PUFs. In: HOST. IEEE (2012)
13. Hiller, M., Sigl, G.: Increasing the efficiency of syndrome coding for PUFs with helper data compression. In: DATE. ACM/IEEE (2014)
14. Hiller, M., Weiner, M., et al.: Breaking through fixed PUF block limitations with differential sequence coding and convolutional codes. In: TrustED. ACM (2013)
15. Katzenbeisser, S., Kocabaş, Ü., Rožić, V., Sadeghi, A.-R., Verbaudhede, I., Wachsmann, C.: PUFs: myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 283–301. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-33027-8\\_17](https://doi.org/10.1007/978-3-642-33027-8_17)
16. Krawczyk, H.: Cryptographic extraction and key derivation: the HKDF scheme. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 631–648. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-14623-7\\_34](https://doi.org/10.1007/978-3-642-14623-7_34)
17. Kömmerling, O., Kuhn, M.G.: Design principles for tamper-resistant smartcard processors. In: Proceedings of the 1st Workshop on Smartcard Technology (1999)
18. Langner, R.: To kill a centrifuge - a technical analysis of what stuxnet’s creators tried to achieve. Technical report. The Langner Group, November 2013
19. Lie, D., Thekkath, C., Mitchell, M., et al.: Architectural support for copy and tamper resistant software. *SIGOPS Oper. Syst. Rev.* **34**(5), 168–177 (2000)

20. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. *J. Cryptol.* **24**(3), 588–613 (2010)
21. Maene, P., Verbauwhede, I.: Single-cycle implementations of block ciphers. In: Güneysu, T., Leander, G., Moradi, A. (eds.) *LightSec 2015*. LNCS, vol. 9542, pp. 131–147. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-29078-2\\_8](https://doi.org/10.1007/978-3-319-29078-2_8)
22. Maes, R., Van Herrewege, A., Verbauwhede, I.: PUFKY: a fully functional PUF-based cryptographic key generator. In: *CHES (2012)*
23. Maiti, A., Schaumont, P.: Improved ring oscillator PUF: an FPGA-friendly secure primitive. *J. Cryptol.* **24**(2), 375–397 (2011)
24. One, A.: Smashing the stack for fun and profit. *Phrack* **7**(49) (1996)
25. Owusu, E., Guajardo, J., et al.: OASIS: on achieving a sanctuary for integrity and secrecy on untrusted platforms. In: *SIGSAC*. ACM (2013)
26. Simpson, E., Schaumont, P.: Offline hardware/software authentication for reconfigurable platforms. In: Goubin, L., Matsui, M. (eds.) *CHES 2006*. LNCS, vol. 4249, pp. 311–323. Springer, Heidelberg (2006). [https://doi.org/10.1007/11894063\\_25](https://doi.org/10.1007/11894063_25)
27. Suh, E.G., Clarke, D., Gassend, B., van Dijk, M., Devadas, S.: AEGIS: architecture for tamper-evident and tamper-resistant processing. In: *ICS*. ACM (2003)
28. Suh, E.G., et al.: Design and implementation of the AEGIS single-chip secure processor using PUFs. *SIGARCH Comput. Archit. News* **33**(2), 25–36 (2005)
29. Yang, J., Zhang, Y., Gao, L.: Fast secure processor for inhibiting software piracy and tampering. In: *International Symposium on Microarchitecture*. IEEE/ACM (2003)
30. Yu, M.D., Devadas, S.: Secure and robust error correction for physical unclonable functions. *IEEE Des.Test Comput.* **27**(1), 48–65 (2010)
31. Yu, X., Fletcher, C.W., et al.: Generalized external interaction with tamper-resistant hardware with bounded information leakage. In: *CCSW*. ACM (2013)



# *Lumus*: Dynamically Uncovering Evasive Android Applications

Vitor Afonso<sup>1,5</sup>, Anatoli Kalysch<sup>4</sup>, Tilo Müller<sup>4</sup>, Daniela Oliveira<sup>3</sup>,  
André Grégio<sup>2(✉)</sup>, and Paulo Lício de Geus<sup>1</sup>

<sup>1</sup> Institute of Computing, University of Campinas, Campinas, Brazil  
pgeus@unicamp.br

<sup>2</sup> Department of Informatics, Federal University of Paraná, Curitiba, Brazil  
gregio@inf.ufpr.br

<sup>3</sup> Florida Institute for Cybersecurity Research,  
University of Florida, Gainesville, USA  
daniela@ece.ufl.edu

<sup>4</sup> Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany  
anatoli.kalysch@fau.de,tilo.mueller@cs.fau.de

<sup>5</sup> Content Keeper, Sydney, Australia  
vitor.afonso@contentkeeper.com

**Abstract.** Dynamic analysis of Android malware suffers from techniques that identify the analysis environment and prevent the malicious behavior from being observed. While there are many analysis solutions that can thwart evasive malware on Windows, the application of similar techniques for Android has not been studied in-depth. In this paper, we present *Lumus*, a novel technique to uncover evasive malware on Android. *Lumus* compares the execution traces of malware on bare metal and emulated environments. We used *Lumus* to analyze 1,470 Android malware samples and were able to uncover 192 evasive samples. Comparing our approach with other solutions yields better results in terms of accuracy and false positives. We discuss which information are typically used by evasive malware for detecting emulated environments, and conclude on how analysis sandboxes can be strengthened in the future.

## 1 Introduction

Malicious applications are a major threat to Android users, as they may steal sensitive data, send SMS messages to premium numbers, and manipulate mobile banking transactions [8]. The analysis of an apps behavior is crucial for protecting mobile devices, e.g., to analyze applications before they are published in app stores. Several approaches have been proposed for Android app analysis [6, 7, 16, 24, 27], including the classification into malicious or benign [1, 5, 25, 28, 30, 31, 33].

Analysis techniques are typically divided into static and dynamic approaches. Static approaches become less effective when dealing with highly obfuscated samples [3, 10, 20], or with samples that obtain and execute code at runtime [17, 23]. Dynamic approaches are able to run obfuscated samples and samples that

fetch code dynamically, but can be evaded by malware that employs anti-analysis techniques. Anti-analysis techniques are used to identify emulated environments, and eventually to change an apps behavior to evade detection.

Researchers have identified several anti-analysis techniques that are used by Android apps to distinguish real from analysis environments [9, 18, 22, 26, 29]. Emulation is used by most analysis systems due to scalability. Hence, an alternative for analyzing apps without being evaded by common anti-analysis techniques is using real devices, as, for example, done by BareDroid [21] and Bolt [7]. However, by studying evasive malware samples in-depth, researchers can also identify ways to make emulated systems resilient to evasive malware.

In this paper, we present *Lumus*, a technique to identify Android malware that exhibits evasive behavior. We analyze the behavior of malware samples by comparing their execution traces on actual devices and emulators. While systems proposed in the literature also identify evasive malware on Windows [12, 15], those techniques cannot directly be applied to Android given the differences between the two operating systems. To demonstrate this, we created detectors based on the Windows techniques proposed in Disarm [15] and Barecloud [12], and compared their results with *Lumus*, which obtained better results.

We analyzed 1,470 Android malware samples selected from different families and identified 192 samples that exhibit evasive behavior. We manually inspected a subset of the detected malware to identify how they evade dynamic analysis. To compare our technique with other solutions, we randomly selected 50 samples from different families and manually analyzed them to validate our results.

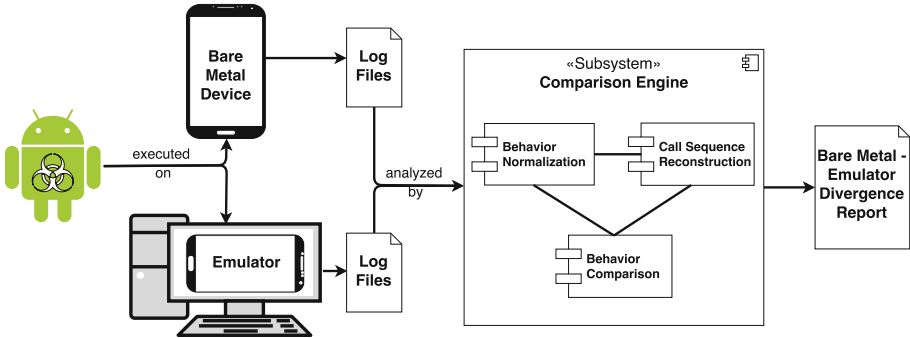
## 2 Approach

To identify evasive malware, systems proposed in non-mobile literature [12, 15] usually compare behavior profiles using distance equations or hierarchically structuring them to compute their similarity—they use system call traces as input data and focus on Windows malware. However, Android malware in general executes far fewer actions than Windows malware and many of the proposed approaches for detecting evasive malware require a minimum number of actions from malware for the detection process (e.g., Disarm [15] requires at least 150 actions). Furthermore, many Android malware are repackaged apps that also perform benign behavior. Therefore, depending on the malware family, infectious actions that make use of anti-analysis features in Android malware can be a very small subset of all possible behaviors, but might just as well comprise most of the app’s behavior.

Our approach to identify actual differences of behaviors in Android malware running in bare metal and emulated environments that are not related to idiosyncrasies between these two environments is as follows. First we try to identify the cause of each observed different action through information that is easily obtained in Android, but not for Windows programs. More precisely, we track: (i) executed methods of the app under analysis; (ii) methods from the framework called by the methods identified in (i); (iii) system calls invoked by

the app; (iv) interaction of the apps with functionalities that create threads or indirectly change their execution flow; (v) information provided by the system regarding events that stop the execution of apps; and (vi) information about external stimuli. With all this information, we can trace back the call sequence that led to the behavior observed only in bare metal, identifying the entry point that originated this sequence and possibly the external stimulus that caused it. By comparing the sequence obtained from the bare metal system to the behavior observed in the emulator we can identify the reason for the divergence (e.g., some event not being generated, a difference in some method’s execution, the analysis time ending in one of the environments, or the system stopping the app for some reason).

*Lumus*’ work flow is detailed in Fig. 1 in form of a process diagram. The events from (i)–(vi) are collected in form of log files for the bare metal and emulator executions. While the bare metal device is brought back to a consistent state the comparison engine uses the collected information to create the divergence report by comparing the information in (i)–(vi) for the emulator and bare metal executions.



**Fig. 1.** Abstract process diagram of our approach. For simplicity reasons the snapshot reconstruction subsystem for the bare metal device has been left out of the loop.

### 3 Behavior Representation

We represent the behavior of an app in a given analysis environment as a set of actions observed during its execution. Each action is a tuple  $a$  represented as  $a = (action\_type, operation, argument)$ , where  $action\_type$  is one of  $\{Network, File, Intent, Exec, Phone, Dex, Billing, Multimedia\}$ . Details about the action types and its operations and arguments are presented in the Appendix.

### 3.1 Behavior Normalization

Filenames written by apps may be randomly generated, which makes multiple executions of the same application produce different behavior profiles. To overcome this problem we adopt Disarm’s approach [15]: for each sandbox—emulated or bare metal—we identify files that were written in only one instance of this sandbox and consider these as possibly random files. Possibly random files in multiple instances of a sandbox that have the same directory and extension are considered as random. We keep the directory name and extension of these actions but replace the file name by  $\langle RANDOM \rangle$ . We also normalize file paths related to the SD card, as it can be accessed in different ways. Malware may also randomly select contacts registered in the system as destinations of SMS messages.

Therefore, we inspect actions related to sending SMS messages and, if some destination is a contact registered in the system, we replace it by  $\langle CONTACT \rangle$ . Additionally, we filter simple actions that are common to most apps, such as writing to the shared memory device or to the logging device. Another group of actions we filter are related to Androids’ Webview, an in-app solution to displaying web pages. Since it relies on libraries only available on bare metal devices the behavior will differ in the emulator, leading to false positives in applications using WebViews. Another challenge of the behavioral analysis problem, especially when the app needs to be executed multiple times, is that certain network behavior might only be observed in some of the runs. To address this challenge, during our analysis when some host is accessed in bare metal and the same DNS name is requested in the emulated context, but this request fails, we add the same failed request to the emulated analysis.

## 4 Evasive Behavior Identification

To identify whether an app is evasive or not, *Lumus* analyzes its execution in a bare metal environment and in an emulated environment, and then compares the monitored behavior for differences. If the behavior in the two environments is different, *Lumus* identifies the root cause for the divergence, which can be: (i) a variation in the code path executed or (ii) some event that prevented the app from continuing executing in the emulated environment. To increase the app code coverage during dynamic analysis, *Lumus* generates stimuli in the form of GUI interactions and Intents, which can be used to start activities or receivers. *Lumus* provides the same stimuli for both bare metal and emulated environments. Moreover, *Lumus* takes into account non-determinism during app execution, *Lumus* executes each sample three times in each environment.

Let  $B_i$  and  $E_j$  be the set of actions monitored in the bare metal environment for the  $i$ th run and in the emulated environment for the  $j$ th run, respectively, with  $1 \leq i \leq 3$  and  $1 \leq j \leq 3$ . Also, let  $B = \bigcup_{i=1}^3 B_i$  and  $E = \bigcup_{j=1}^3 E_j$  be the set of all actions executed in bare metal and in emulated environments, respectively.

Since we are interested in finding apps that hide their actions during analysis in the emulated environment, *Lumus* first selects the set  $A$  of actions that were only executed in a real device. Thus,  $A = B - B \cap E$ .

For each action  $a_k$  in  $A$ , *Lumus* constructs  $R_k$ , a set with the Android malware traces obtained from bare metal analysis that contain this action. *Lumus* compares each  $B_l$  in  $R_k$  to every  $E_j$  to identify why  $a_k$  was not executed in the emulated analyses. Since *Lumus* tracks when methods begin and end, it can identify the app's method that executed the action we are interested in. At this point, *Lumus* knows the main entry point for the execution of an app obtained through bare metal analysis ( $B_l$ ), which will be called from now on  $M_k$ . This entry point  $M_k$  led to the execution of action  $a_k$ , and possibly the external stimulus that caused this action. *Lumus* finds the occurrences of  $M_k$  in  $E_j$  and, if it knows the stimulus that originated it, *Lumus* compares each  $M_k$  in  $E_j$  with the  $B_l$  call sequence that led to  $a_k$ . With this comparison, *Lumus* identifies where is the point of divergence of the execution path, i.e., when the emulated system should also execute the action, but instead chose to follow another path. More precisely, *Lumus* identifies which of the following is the cause of the divergence: (i) difference in execution path; (ii) app not responding; (iii) end of analysis; (iv) fatal exception; or (v) entry point not reached. If the reason for the divergence is a different code path executed, *Lumus* considers the app evasive, otherwise, an execution problem.

#### 4.1 Call Sequence Reconstruction

*Lumus* records when each thread of the analyzed app enters and leaves its methods, identifying what method performed a given action. It also logs the methods called, so that the analysis trace can be looked back, starting from the method that executed the action, and sequence of method calls that led to this action is identified. This allows the tracking of actions to an entry point interfacing the Android framework, where the capability to observe direct calls to methods no longer exists. Android classes may have several entry points, which may be executed because of commonly creating activities (e.g., *onCreate*), starting services (e.g., *onStart*), starting receivers (e.g., *onReceive*), running tasks (e.g., *run*), and handling of received messages (e.g., *handleMessage*). One possibility would be to compare all executions of the entry point method in the bare metal and emulated systems, but this could lead to wrong results, because of uncertainty. The phenomenon of uncertainty can happen, for instance, if an activity handles different functionalities, all executed through the same entry point. In that case, our approach is to try to identify the source methods from which the execution changed to the entry points, to perform a more precise comparison of the executions. To accomplish this, *Lumus* investigates Intents sent by the app, the use of several methods that cause indirect changes in the execution flow and the use of external stimuli.

To identify Intents that may have resulted in a specific entry point method being executed, we look for Intents sent by the app that match that method. For example, if the method we are analyzing is *ClassA.onStartCommand(Intent,*



*int, int*), we assume *ClassA* is a service, since *onStartCommand(Intent, int, int)* is one of the entry points of the class *Service* that can be overwritten. As the Android documentation states, this method is called by the system when a client explicitly starts the service by calling *startService(Intent)*. Thus, to find the source that directed the execution to this method we look for actions that start services using *ClassA* as an argument. Finding sources of entry points to activities is similar to finding services. To find the sources that led to the execution of receivers, however, we need to inspect the intent filters used by the class and find out which Intents sent match these filters (see Sect. 5).

Another source of control flow changes performed by *Lumus* is the use of the following groups of methods: (i) methods that schedule a class to be invoked after some delay or periodically (e.g., *Timer.schedule* and *ScheduledThreadPoolExecutor.schedule*), which result in the execution of the methods *run()* or *call()* of the destination class, (ii) methods that send messages to its UI thread (e.g., *Handler.sendMessageDelayed*), which result in the execution of *handleMessage(Message)*, and (iii) methods that start a new thread, e.g., *Thread.start()*, which result in the execution of *run()* and *AsyncTask.execute(Params...)*, which in turn may result in the execution of different methods (e.g., *doInBackground(Params...)*). To track these control flow changes, *Lumus* employs instrumentation of the Android framework to assign labels to the messages/tasks sent or to the threads created. To do so, we log the control flow changes when the source method is executed and also when the destination methods are executed. With this information, *Lumus* can track the source call of any of these methods.

The last type of interaction that can cause the execution of entry points is external stimuli. *Lumus* also employs instrumentation of the tools used to create the stimuli, identifying when Intents are sent, keys are pressed, and GUI interactions are performed. These Intents are identified as the source of some entry point, similarly to the approach adopted for Intents sent by the app, explained above. Examples of entry points executed by key strokes are *onKey* and *onKeyDown*. For GUI interactions, examples of common entry points executed are *onClick*, *onTouchEvent* and *onItemClick*.

When tracing the sequence of calls that led to some action, a list of subsequences is created. Along with each subsequence the time of the call that created the next subsequence or that executed the action is kept.

## 4.2 Comparing Sequences

After identifying the list of subsequences of method calls that led to the execution of an action in the bare metal environment, *Lumus* needs to compare this list with the list obtained from the analysis within the emulated system to identify the cause of divergence. We hereafter refer to this list of subsequences as *BareSeq* and to the resulting list of subsequences obtained from the emulated environment as *ResEmu*. *Lumus* iterates over each subsequence *SubBare<sub>i</sub>* of *BareSeq*, comparing it to its counterpart in *ResEmu*. For each iteration, *action.time* is the time when the call that created the next subsequence was executed or the time when the action was performed. Also, let *EP<sub>i</sub>* be the entry point of *SubBare<sub>i</sub>*.

```

[78] BARE: 'com.adobe.flashplayer...AdobeFlashCore.onCreate()' -> '%com.adobe.flashplayer...
      AdobeFlashCore.writeConfig(java.lang.String, java.lang.String)'
      EMU: 'com.adobe.flashplayer...AdobeFlashCore.onCreate()' -> '%com.adobe.flashplayer...
      AdobeFlashCore.writeConfig(java.lang.String, java.lang.String)'
...
[85] BARE: 'com.adobe.flashplayer...AdobeFlashCore.onCreate()' -> 'java.lang.String.indexOf(
      java.lang.String)'
      EMU: 'com.adobe.flashplayer...AdobeFlashCore.onCreate()' -> 'java.lang.String.indexOf(
      java.lang.String)'
[86] BARE: 'None'
      EMU: 'com.adobe.flashplayer...AdobeFlashCore.onCreate()' -> 'java.lang.System.exit(int)'
[87] BARE: 'com.adobe.flashplayer...AdobeFlashCore.onCreate()' -> 'com.adobe.flashplayer...
      AdobeFlashCore.isOnline()'
      EMU: 'None'
...
[102] BARE: 'com.adobe.flashplayer...AdobeFlashCore.onCreate()' -> 'com.adobe.flashplayer...
      FlashVars.<init>()'
      EMU: 'None'

```

**Listing 1.1.** Excerpt of the alignment of an evasive sample. The comparison extends to methods and method values to mitigate the downsides of signature only comparisons.

*Lumus* finds all occurrences of  $EP_i$  in ResEmu that have the same origin as in BareSeq. *Lumus* then proceeds to compare  $EP_i$  from BareSeq with each instance of  $EP_i$  identified in the emulated results.

Given two entry point methods, *Lumus* finds where they begin and end, obtaining the call sequence in this interval. *Lumus* aligns these two sequences, one from BareSeq and other from ResEmu, using a global alignment algorithm. If  $SubBare_i$  is the last subsequence of BareSeq, *Lumus* compares the aligned sequences to determine the divergence that prevented ResEmu from reaching the call at *action.time*. Otherwise, let  $CallNext$  be the method call in  $SubBare_i$  that created the next subsequence of BareSeq. If the app did not reach  $CallNext$  in ResEmu, *Lumus* compares the aligned sequences to determine the divergence that prevented ResEmu from reaching  $CallNext$ . However, if the app reached  $CallNext$  in ResEmu, *Lumus* obtains the next subsequence of BareSeq, with entry point  $EP_{i+1}$ , and finds this entry point in ResEmu by checking for possible destinations of  $CallNext$ . If *Lumus* is not able to find an equivalent of  $EP_{i+1}$  in ResEmu, it is likely that the execution was interrupted before the call performed at  $CallNext$  could take effect, so *Lumus* does not consider it an evasion.

When comparing two aligned sequences, we want to identify the reason for their divergence regarding some action executed at time  $t_i$  (either a behavior only observed in BareSeq or some call that created the next subsequence of BareSeq and that was not executed in ResEmu). To do so, *Lumus* iterates over the calls in the aligned sequences and when  $t_i$  passes, considering the time of the bare metal calls, *Lumus* checks what was the last call in the emulated sequence. There are three possibilities for this last call: (i) a tag indicating the end of the analysis, (ii) a tag indicating that the system killed the app for not responding or for some other error, (iii) a call to some app's or *Lumus* method.

If *Lumus* identifies case (iii), we assume a divergence in code path taken—an indication of evasive behavior. *Lumus* prints the aligned sequences to help an analyst who needs to manually identify what caused the executions to follow different code paths. Conversely, if the last identified call matches cases (i) or (ii), we assume that an execution error occurred, not an evasion. For illustration, we present an excerpt of the output generated for one sample that is evasive in Listing 1.1. To perform sequence alignment, *Lumus* uses the global alignment algorithm provided by the *swalign* library. We chose a global alignment algorithm because we need to have a global understanding of the sequences, as our analysis depends on the alignment reaching the point in the bare metal sequence where the target action happened. If the aligned sequence does not reach this point, *Lumus* is unable to identify the cause of divergence.

Arguments selection is an important step when using alignment algorithms. The arguments *Lumus* needs to define to calculate the similarity score between two sequences are: (i)  $m$  for matches, with  $m > 0$ ; (ii)  $mi$  for mismatches, with  $mi < 0$ ; (iii)  $g_o$  for opening gaps, with  $g_o < 0$ ; (iv)  $g_e$  for extending gaps, with  $g_e < 0$ . To prevent mismatches during the analysis, we used a high value for  $|mismatch|$  in *Lumus*. We also believe that beginnings and endings of methods of the analyzed app are more important in the alignment than other types of calls, so we assign  $2*m$  for matches of this type. Furthermore, since *Lumus* aims at investigating evasive behavior, we want to prioritize gap extensions over gap openings, so  $|g_o| > |g_e|$ . In the end, the inequality  $|mismatch| > m > |g_o| > |g_e|$  guides the definition of arguments.

## 5 Monitoring System

To track the behavior of the analyzed apps, *Lumus* monitors which apps' methods were executed, which methods were called from them, and which system calls were executed. To monitor system calls, *Lumus* uses a kernel driver that intercepts them. When a system call is executed, the driver registers its arguments and calls the original system call. To obtain information related to the use of Intents, the driver inspects *ioctl* calls that target the binder device. If the operation performed is a binder transaction (BC\_TRANSACTION), *Lumus* logs the destination class, method id and arguments passed. To identify which actual method is represented by the method id, *Lumus* examines the corresponding AIDL file in the Android source code.

To monitor the executed methods, *Lumus* leverages the “method trace” functionality of the Android runtime (ART) and instrument *libart*. Every time the execution goes in and out of a method, *Lumus* registers it. Also, when some method is called, *Lumus* logs the source and destination of such action. This allows *Lumus* to also identify Java methods called from native code. To avoid registering too much information, *Lumus* focuses on new UIDs, so it does not track apps that are already installed in the system when it is in a clean state.

When trying to identify the method call that resulted in the execution of some receiver, *Lumus* needs to identify which intent filters are used by this receiver

and look for broadcasts sent that match them. Parsing the app’s manifest is not enough to obtain all intent filters that were used, since the app can register others at runtime. To overcome this limitation, *Lumus* also tracks all calls to *Context.registerReceiver*.

The use of threads, tasks and message passing between them introduces a level of indirection that prevents us from tracking the execution flow just by looking at method invocations (Sect. 4). To be able to reconstruct the call sequence in these cases, *Lumus* needs to track the use of threads, tasks, and messages. To do so, *Lumus* generates a random number that is assigned to a thread (when it is created), to a task (when it is scheduled), and to a message (when it is sent). *Lumus* also logs this identifier when they are actually used or executed, allowing a parser to match each use or execution of these types to their creation. This matching allows us to track the call sequence in these cases. So, for instance, when the method *Timer.schedule*, which schedules a task for repeated fixed-delay execution, is executed, the system generates a random number, logs it and assigns it to the task. Every time this task is executed, the identification number is logged. To correlate the external stimuli with the app behavior (e.g., clicks and broadcasts), *Lumus* needs to know the time at which each stimulus was provided. To achieve this, *Lumus* employs the instrumentation of the `am` and `input` tools used to create these actions.

**Analysis Environments.** When analyzing malware, it is important to make sure that the environment is not infected before the start of the analysis. Performing an analysis in an infected system may result in wrong results, as one piece of malware can influence the behavior of others. To analyze samples in the emulator, we take advantage of the snapshot functionality, which allows us to restore the system to a clean state after every analysis without incurring any boot time. Analyzing malware in real devices is more challenging as we cannot take advantage of snapshots. One possible way to overcome this problem is to restore the state of the device’s partitions after every analysis, as performed by Baredroid [21]. However, this approach is time consuming because the system needs to reboot every time it is restored.

The approach adopted in *Lumus* is to maintain the system clean after each analysis as follows. In Android, apps can only write to a very limited set of directories, which includes mainly the app’s dir (`/data/data/<PACKAGE-NAME>/`) and the SD card. When the app is uninstalled after analysis, its directory is deleted by *Lumus*. Files written to the SD card can affect the behavior of other apps that might interact with these files. Thus, all files belonging to the SD card are deleted after every analysis.

Many malware target vulnerabilities in the kernel or privileged processes to operate with root privileges. As the `/system` partition, corresponding to kernel code, is mounted as read-only by default, even apps that are able to obtain root privilege first need to remount this partition. To prevent it, our kernel driver blocks all system calls that attempt to remount the `/system` partition in writing mode. This protection can be bypassed if the malware manages to

access the original `mount` system call. However, we only used the system to test our proposed technique to identify evasive malware. If one wants to use a similar system to receive submissions or analyze apps that could potentially target the system, a better restoration process would be necessary. Furthermore, during our experiments our driver did not actually have to block any calls to `mount`, so we believe that bypass through `remount` was not a problem.

To increase code coverage during dynamic analysis, it is important to provide GUI interactions and to cause activities, services and receivers to execute. However, as we are comparing multiple executions, it is also important that we provide exactly the same interactions, so that the same code paths are exercised, at least until evasive code is reached or some problem stops the app execution. To accomplish this, we use the Droidbot [14] tool to interact with apps. Droidbot generates random events, including GUI interactions, broadcasts and specific activities. It also registers the exact events generated and is able to replay them from a file instead of randomly generating them. Thus, in our first bare metal execution of each malware, we randomly generate events and save them into a file. In the following bare metal executions and in the emulated analyses, we make Droidbot read the events from the saved file and replay them.

One way that malware can identify analysis environments is by checking which apps are installed in the system. The lack of the Google Play app, for instance, is a strong indication that the device is not used by an actual user. Hence, we installed in the bare metal environment Open GAPPs, a set of basic apps present in all Android systems. Further, we also installed a few very popular apps and created fake contact information. These apps and contact information make the bare metal and emulated systems different, which could, therefore, cause some apps to behave differently, but not because they intend to evade analysis systems. On the one hand, this could possibly lead our technique to identify such samples as evasive, increasing the number of false-positives. On the other hand, preventing such false-positives may result in false-negatives since the bare metal system could also be detected as an analysis system. We chose to use these techniques and risk increasing false-positives instead of risking increasing false-negatives.

## 6 Evaluation

For our experiments we used Google’s QEMU-based Android emulator, the SDKs’ Android Virtual Device (AVD) and an LG G2Mini device. Both the bare metal and the emulated Android systems had our modified version of Android 5.1. Each analysis was executed for at most three minutes in the bare metal environment and at most 10 min in the emulated environment—our experiments showed that the emulator environment incurred a 3X performance penalty on the analysis. Since we can identify when a divergence in behavior is caused by one analysis system finishing before the other, this difference in execution time has no negative effects on our technique.

To evaluate *Lumus*, we dynamically analyzed a subset of the samples in our malware dataset obtained from VirusShare, Malgenome [32], contagio mobile,

AndroMalShare and Drebin [1]. To select this subset we first obtained their detection label by antivirus software, using Virustotal. We separated them by families, using the results of the ESET-NOD32 anti-virus, and selected at most five samples from each family, resulting in a set of 1,470 samples. We analyzed these samples to obtain their behavior and used *Lumus* to identify which ones have evasive behavior. Since we did not have a ground truth with information about all these samples, we randomly selected 50 samples, all from different families, and manually inspected their results to identify possible false-negatives and false-positives. This manually analyzed samples became the ground-truth for our subsequent analysis.

Our technique detected 7 out of 50 samples (14%) in the subset as evasive. In the following, we provide a detailed analysis of each of the 7 cases. We consider as false-negatives the samples that did evade analysis but *Lumus* did not identify as evasive, and we consider as false-positives those whose behavior exhibited in the bare metal was different from the emulated without trying to identify if the execution was inside an analysis system. Note that *Lumus* considers evasive those samples that execute some action only in the bare metal system, without executing some similar action in the emulated system, even if this divergence is not caused by a clear identification of the analysis system. For instance, if a sample tries to send SMS messages to contacts stored in the phone and it only shows this behavior in the bare metal because there is no contact registered in the emulated environment, *Lumus* considers it as evasive. We do this because, despite not being a clear sign of anti-analysis behavior, it is successful in preventing some action from being observed in the emulator and so could be employed as an anti-analysis technique. To test our intuition that the existing techniques to identify evasive Windows malware would not present as good results if applied to Android malware, we implemented detectors based on the techniques proposed by Disarm [15] and Barecloud [12]. Since the behavior of Android and Windows malware are different in various aspects, we used our behavior model when implementing these techniques. To make the comparison fairer, we used the threshold that would yield the best results to each of these techniques, instead of the threshold they found for Windows malware. Table 1 presents this comparisons’ results, showing that our technique is far more suitable to the Android environment.

**Table 1.** *Lumus* vs. other Windows-based approaches to uncover evasive malware reimplemented for Android.

Approach	TP	TN	FP	FN	A
Lumus	100%	93.5%	6.5%	0.0%	96.7%
Disarm (t = 0.12)	100%	78.3%	21.7%	0.0%	89.1%
Barecloud (t = 0.36)	100%	67.4%	32.6%	0.0%	83.7%

## 6.1 Discussion

We discuss below the samples that *Lumus* identified as evasive, explaining why we consider them as a true-positive (TP) or false-positive (FP). For the TP, we discuss which extra behavior was observed due to the divergence and what difference between the environments was the cause of divergence.

**Sample 1:** It changes its behavior if `/system/xbin/busybox`, `/system/bin/busybox` or `/bin/busybox` is present in the system. This deviation resulted in the malware writing to the file `shared_prefs/config.xml` and many files in the dir `/SD card/LuckyPatcher/`. This may not have been intended as an anti-analysis technique, since most user systems do not have these files. However, because it does prevent some of the malware behavior from being observed in the emulated environment, we considered the behavior as TP;

**Sample 2:** It identifies if the phone number starts with “1555”, whether the IMEI starts with “00000000” or if the IMSI starts with “31026”. Upon detection, it calls `System.exit(0)`. This is a clear case of evasive malware and a TP. The behavior resulting from the divergence is composed of starting a service and starting two alarms that send Intents;

**Sample 3:** It copies the icons of the apps installed in the system to the directory `/data/data/com.pintudog/files/icons/`. Since the list of apps installed in the emulator and in the bare metal environments is not the same, the monitored actions ended up being different. However, at a higher level it is still the same behavior, so we consider this as FP;

**Sample 4:** It verifies if the IMEI contains the string “0000000000000000”. If so, the malware calls `System.exit(0)`. Similarly to Sample 2, this is a clear example of anti-analysis and a TP. The actions resulting from the divergence are the following: starting a service, creating a wake lock and connecting to the `dnsproxyd` device to make a DNS request;

**Sample 5:** The different actions in this sample’s behavior are related to a file associated with the graphical interface, as the graphical libraries used in the bare metal and emulated systems are different. Since this is not actually related to the behavior of the malware, we considered this sample as a FP;

**Sample 6:** During its execution it verified which Wifi networks are available. In the emulated system it does not identify any Wifi network, so it takes a different execution path. The behavior that is executed only in bare metal, as a result of this difference, is writing a file in the SD card. Since this behavior is related to the malware execution and cannot be observed in the emulator unless some update is made to it, we considered this sample as a TP;

**Sample 7:** This sample randomly chose the domain name to access from a list of predefined names. This resulted in one domain used in bare metal not being used in the emulated analysis. Except for the domain difference, their behavior is the same, so we considered this as FP.

## 6.2 Employed Emulator Detection Techniques

Out of the complete dataset of 1,470 samples, our technique identified 192 as evasive. A manual inspection of these evasive samples yielded several techniques that were used to discern emulators from bare metal devices. Most of these techniques use static or dynamic artifacts to detect differences from a bare metal.

Static artifacts usually constitute environmental values and configurations that differ between emulators and bare metal devices, and do not change between many complex states. Most of these values in fact remain constant at runtime and do not react much or not at all to environmental stimuli. Dynamic artifacts on the other hand usually result from the emulated interfaces that either show insufficient behavior compared to a bare metal device or are not emulated at all and can thus a variety of states can occur at runtime. This results in inconsistencies that shouldn't occur during normal bare metal execution [21].

We manually inspected the results of some of these samples to understand how they evade analysis. Below we describe the anti-analysis techniques we identified that are different from the ones explained before:

**Static artifacts** often can be queried through simple value lookup mechanisms. This allows for an easy to implement branching control flow, where depending on the lookups' result the behavior can be benign or invasive. The previously introduced Sample 2 and Sample 4 also exhibited this behavior. Specifically, they leverage telephony related values to detect an emulated environment. Androids' `TelephonyManager` class allows to query specific information about the phones' identifiers, such as the IMEI and the IMSI, and the phones' line1 number. Vidas et al. pointed out that the AVD has the hard coded values of `155552155**` (wildcards in the line1 number are replaced by the system at boot time with the last two digits of the Android debug bridge (ADB) port) as its line1 number, a zeroed IMEI and an IMSI of `310260000000000`. Hence, checking for these values enables an application to detect the AVD as several samples in our dataset did. The downside of using telephony related values is necessary `Phone` permission group, specifically the `READ_PHONE_STATE` permission. Requesting this permission as an application that does not necessarily need it, e.g., a flashlight app, might raise some red flags and speed up discovery by malware analysts.

A permissionless technique is given through Androids' build properties. Applications can use the `android.os.Build` class or query the properties through one of the API methods `ProcessBuilder.start` and `Runtime.exec`. The tell-tale build properties of an emulator the malware probed for were the presence of the string "google\_sdk" in the `PRODUCT` or `MODEL` properties, "generic" in the `BRAND` or `DEVICE` properties and "goldfish" ("Goldfish" is the previous canonical name for the QEMU-based AVD; the current canonical name is "Ranchu") in the `HARDWARE` property. If the device `FINGERPRINT` contained any of the strings "qemu", "sdk" or "generic" an emulator was detected as well. These build properties tests have in common that a value from a bare metal device is compared to its changed but present equivalent on an emulator. Another possibility is the examination of values that are only present on one of the two, e.g., the



QEMU properties which are only present in Google’s QEMU-based emulator. A query for “qemu” being present in the properties is enough to detect an analysis environment in this case, e.g., “qemu.sf.fake\_camera”, “ro.kernel.qemu” or “ro.kernel.android.qemud”. Similar is to the build properties the existence of specific files can be queried. Exemplary the existence of QEMUD, Androids’ QEMU Multiplexing Daemon can be assumed if the file `/system/bin/qemud` exists or as with sample 1 the existence of `/system/bin/busybox` helps detect specific environments.

Lastly, the presence of specific installed packages can be leveraged to make assumptions about the runtime environment. The absence of the Google Service Framework (`com.google.android.gsf`) constitutes an exception to the rule on a bare metal device. However, the AVD does not have its own version preinstalled making the query for this framework a valid choice. In addition the overall number of installed packages is revealing as well. An application can query installed packages through the `PackageManager`. Finding only packages from the domains `com.android` and `com.google` on a device reveals a bogus environment as a normal user would install packages from several other domains.

The **Dynamic artifacts** we encountered often leveraged the reactions to certain stimuli given by the application at runtime to ascertain whether the underlying device is an emulator. Although this sometimes can be handled through a value lookup mechanism a more stable solution to test dynamic artifacts at runtime is given through error handling. To hide malicious logic through error handling an application could either try to access interfaces or runtime resources unavailable in an emulator or try to trigger an exception only on bare metal devices. Accessing resources unavailable inside an AVD will trigger an exception preventing code after the access request from being executed. Exceptions created only on bare metal devices allow malware authors to place exploitation logic inside the exception handling routine, leading to the same result as above. This blurs the line between regular and evasion logic and is harder to detect, especially through static analysis techniques.

Amongst the analyzed samples the connectivity interfaces `WifiManager` and `BluetoothManager` were frequented to detect a bogus environment dynamically. By default the emulator does not emulate the connectivity interfaces, meaning no other Wifi networks can be encountered around the device and the number of saved Wifi networks is zero. Samples used this to detect emulation if no saved Wifi networks were present and a scan request for Wifi networks came back empty or raised an exception. The `BluetoothManager` was used to retrieve the Handle for a bluetooth adapter used to interact with other bluetooth devices. Google’s AVD does not emulate a bluetooth interface, hence the bluetooth adapter returned is a `null` Object and raises a `NullPointerException` if used. Applications can also abuse the emulated network connection itself. Specifically the emulators inability to forward ICMP packets can easily be used for detection.

The dynamic counterpart to the static query whether certain packages are installed is the interaction with those packages. Specifically the interaction with Google’s own PlayStore allows the detection of an emulator since by default

only the bare minimum of applications are installed, excluding Google Play and as previously mentioned even the Google Services Framework. Interaction with unavailable services and applications, such as an intent trying to start the Play-Store, causes an exception which some of the analyzed samples used to display a benign behavior should an exception occur. Additionally, *any action requiring a fully set up Google Account will fail* as emulators are not set up with an account by default. Google Cloud Messaging is an exemplary service that requires a registered Google account to set up a Cloud Messaging id, also failing with an exception if no account has been set up. Root usage can prove helpful as well in distinguishing emulators from real devices. Specifically the “su” binary was used by some samples to try and gain superuser privileges. If no additional privileges were acquired by the application the *execution path* did not change and the additional malicious actions were not executed.

The last category we will present are sensor-related evasion techniques. The average Android device comes outfitted with several sensor types fitting the broad categories “Motion Sensors”, “Environmental Sensors” and “Position Sensors”. Similar to the connectivity interfaces the AVD does not emulate all sensors. We compared a LG Nexus 5X emulator to its bare metal counterpart and found 16 sensors emulated on the AVD while the actual model possesses 25 sensors, excluding location sensors. A query about the sensor name reveals “Goldfish-\*” (the wildcard represents the sensor name, e.g., Goldfish-Accelerometer for an accelerometer) as opposed to real devices that offer the vendors canonic name, e.g., “BMI160 accelerometer”. Another conspicuous detail is that the emulators’ sensors all have one of two values as the sensors’ vendor, either “The Android Open Source Project” or “AOSP” while bare metal devices feature an existing company, for example “Bosch”. Trying to interact with sensors that are not available will lead to exceptions as well as using services that depend on the existence of certain sensors. We discovered malware in our dataset that uses the `LocationManager` for this purpose. A request to register a `LocationListener` results in an Exception on emulators leading to benign behavior.

**Current Android Virtual Device Development.** The different techniques we discussed were used by our malware samples to display a benign behavior in case an emulated environment was detected. These techniques rely on either differences in the execution environment or differences in the system behavior during execution. To increase the resilience of emulators against detection these differences can be addressed by closing the gap between bare metal devices and emulators. This would work to our advantage by reducing the false positive rate for our approach.

Most static artifacts can be addressed through changes in the installation images for the AVD. For example, the hard coded telephony values can be randomized and the build properties can be modeled to simulate a real device either by changing them directly or hooking the methods used to query them [4]. Progress in bridging the gap between emulators and bare metal devices is also made by the Android Studio developers in order to improve the testing con-

ditions for application authors. Versions of Android Studio newer than 2.3.2 started to feature AVD images including the GSF, GAPPs and Google Play, complementary to the system images without GSF. Dynamic artifacts can also be emulated but would require more effort in specific cases. A subset of device sensors is already emulated by the AVD and allows for event input, e.g., fake location providers would allow location spoofing and recorded series of events can simulate a moving device.

### 6.3 Limitations

Our detection approach relies on identifying differences between the execution of samples in bare metal and an emulator. Therefore, if execution does not reach the code with anti-analysis features, divergences cannot be observed. Insufficient code coverage is a common problem for dynamic analysis systems, as only executed behavior is usually analyzed. To exploit this, malware can delay the execution of anti-analysis code, or can only execute anti-analysis code after a series of complex GUI interactions that automatic interaction tools are unlikely to reach.

Some malware may be able to detect both environments as analysis systems, because despite the bare metal environment being more similar to a real device, there are still differences that can be exploited, such as information about the user’s behavior (e.g., browsing history and SMS history) and user data. Miramirkhani et al. analyzed these so-called wear-and-tear artifacts for Windows-based operating systems and their viability for analysis environment detection is considered very high because emulators and sandboxes are too “spot-less”, meaning virtually no user data and signs of usage can be detected [19].

When tracing back the origin of some behavior executed in bare metal, we may find some entry point whose source we cannot identify. In these cases we compare this untraced entry point with all instances of the same entry point in the emulated environment. In some cases, this may lead to wrong conclusions. Furthermore, differences in the systems may lead to the execution of different actions that are not related to evading analysis or to the execution of equivalent actions in both systems, but that are considered different in our behavior model. This is the general problem that introduces false positives, e.g., *Lumus* flagging Sample 3 and Sample 5 as evasive. Also, sources of non-determinism that we do not currently handle may lead to the execution of the same high-level behavior, but different actions according to our model. This is the problem that resulted in *Lumus* flagging Sample 7 as evasive. This malware randomly selects the domain name to access from a predefined list, so the domain accessed in bare metal and emulator were different, but the same code path was executed in both cases.

Lastly, the introduction of Google’s SafetyNet Attestation API poses a threat to most dynamic analysis systems. The SafetyNet Attestation API allows an application to assess the security and compatibility of the Android environments in which the application is executed. The attestation is handled off-device by Google’s servers after specific environmental data from the device is collected and the result is sent to any server specified by the malware’s author. This allows

malware to implement easy to use server-side runtime environment checks and even application tamper detection.

## 7 Related Work

Researchers have presented several techniques [9,18,22,29] that Android malware may use to evade detection by making static analysis harder or by circumventing dynamic analysis. Matenaar and Schulz [18] present a method for an app to identify if it is executing inside QEMU, the basis of the Android emulator. Vidas and Christin [29] present anti-analysis techniques based on Android APIs, system properties, network information, QEMU characteristics, performance, hardware and software components. Petsas et al. [22] demonstrate anti-analysis techniques based on Android APIs, system properties, sensors and QEMU characteristics. Instead of manually identifying differences between real and emulated devices, Jing et al. [9] developed Morpheus, a framework that automatically generates heuristics that can identify, based on files, system properties and Android APIs.

Systems that automatically identify malware equipped with anti-analysis techniques have been developed for Windows [2,11–13,15]. Balzarotti et al. [2] propose recording the system calls executed by a sample in a reference environment and replaying the monitored system calls in an emulator to identify if the observed behavior is different. Lindorfer et al. [15] analyze malware samples in different environments and identify differences in the observed actions. Barecloud [12] is a system that dynamically analyzes malware in four different environments and detects evasive malware by comparing the reports provided by these systems in a hierarchical approach. Kolbitsch et al. [13] detect and mitigate malicious programs that stall before executing their malicious behavior. Malgene [11] combines sequence alignment of system call traces, obtained from a bare metal and an emulated environment, with taint tracking to identify evasion signatures of evasive malware.

## 8 Conclusions

In this paper, we presented *Lumus*, a novel approach to identify evasive Android malware by comparing its execution on a bare metal analysis system and on an emulated analysis system. For each action executed only in bare metal, *Lumus* identified the basic cause why it was not successfully performed in the emulated environment, differentiating the cases in which there was evasion from the cases in which there was some analysis problem. Our experiments showed that our approach is much more effective for detecting Android malware with anti-analysis features compared to attempting to directly apply existing Windows-based approaches used to detect evasive malware. We analyzed 1,470 malware samples, from which our technique identified 192 as evasive. We presented detected evasion techniques after manually analyzing the samples.

## Appendix - Action Types and Its Operations and Arguments

Behavior is represented by actions, where each action is a tuple  $a = (action\_type, operation, argument)$ , and *action\_type* is one of  $\{Network, File, Intent, Exec, Phone, Dex, Billing, Multimedia\}$ . Action types, as well as its operations and arguments are described below.

**Network.** For network related actions, operation is one of  $\{INET, UNIX, NETLINK, BLUETOOTH\}$ . *INET* operations represent TCP and UDP connections and *argument* is the destination. Since multiple resolutions of the same DNS name may result in different IP addresses, we consider two actions the same if they use the same IP address or the same DNS name as destination. *UNIX* operations represent connections to UNIX sockets and the argument is the filesystem path used. *BLUETOOTH* operations represent the use of the Bluetooth device and the argument is the operation performed with this device. Lastly, *NETLINK* operations represent connections using NETLINK sockets and the argument used is the protocol parameter passed to the socket.

**File.** The monitored operations on files are *WRITE* and *DELETE*, and the argument of both is the file path.

**Intent.** Intent-related operations include *ACTIVITY*, *SERVICE*, *BROADCAST* and *ALARM*. The argument for all these operations is the “action” argument of the Intent or the destination class of the Intent. *ALARM* operations refer to the use of *AlarmManager* to send Intents.

**Exec.** This action type represents the launch of the `execve` system call, which is used by the API methods *ProcessBuilder.start* and *Runtime.exec*. The argument used is the name of the executable file being invoked.

**Phone.** This action represents the use of phone capabilities. We currently consider only one operation of this type (sending SMS messages); the argument is the destination number of the message.

**Dex.** This action type represents the use of dynamic code loading and its argument is the path of the file being loaded.

**Billing.** This action represents the use of the billing functionality; the argument is the type of action performed.

**Multimedia.** The operations included in this action type are *CAMERA*, *AUDIO* and *WAKELOCK*. The argument in these cases is the type of action being performed, which includes taking pictures, recording videos, recording audio, or acquiring wake locks.

## References

1. Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H., Rieck, K.: DREBIN: effective and explainable detection of Android malware in your pocket. In: NDSS (2014)
2. Balzarotti, D., Cova, M., Karlberger, C., Kirda, E., Kruegel, C., Vigna, G.: Efficient detection of split personalities in malware. In: NDSS (2010)
3. Busch, M., Protsenko, M., Müller, T.: A cloud-based compilation and hardening platform for Android apps. In: Proceedings of the 12th International Conference on Availability, Reliability and Security (ARES). SBA Research, Reggio Calabria (2017)
4. Dresel, L., Protsenko, M., Müller, T.: Artist: the Android runtime instrumentation toolkit. In: Proceedings of the 11th International Conference on Availability, Reliability and Security (ARES). SBA Research, Salzburg (2016)
5. Elish, K.O., Yao, D., Ryder, B.G.: User-centric dependence analysis for identifying malicious mobile apps. In: MOST (2012)
6. Grace, M., Zhou, Y., Zhang, Q., Zou, S., Jiang, X.: Riskranker: scalable and accurate zero-day Android malware detection. In: MOBISYS (2012)
7. Guan, L., Jia, S., Chen, B., Zhang, F., Luo, B., Lin, J., Liu, P., Xing, X., Xia, L.: Supporting transparent snapshot for bare-metal malware analysis on mobile devices. In: Proceedings of the 33rd ACSAC, pp. 339–349. ACM (2017)
8. Hauptert, V., Müller, T.: On app-based matrix code authentication in online banking. In: Furnell, S., Mori, P., Camp, O. (eds.) Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), pp. 149–160. SciTePress, Funchal (2018)
9. Jing, Y., Zhao, Z., Ahn, G.J., Hu, H.: Morpheus: automatically generating heuristics to detect Android emulators. In: ACSAC (2014)
10. Kalysch, A., Götzfried, J., Müller, T.: VMAttack: deobfuscating virtualization-based packed binaries. In: Proceedings of the 12th International Conference on Availability, Reliability and Security, p. 2. ACM (2017)
11. Kirat, D., Vigna, G.: MalGene: automatic extraction of malware analysis evasion signature. In: ACM CCS (2015)
12. Kirat, D., Vigna, G., Kruegel, C.: BareCloud: bare-metal analysis-based evasive malware detection. In: USENIX Security (2014)
13. Kolbitsch, C., Kirda, E., Kruegel, C.: The power of procrastination: detection and mitigation of execution-stalling malicious code. In: ACM CCS (2011)
14. Li, Y.: Droidbot (2012). <http://honeynet.github.io/droidbot/>
15. Lindorfer, M., Kolbitsch, C., Milani Comparetti, P.: Detecting environment-sensitive malware. In: Sommer, R., Balzarotti, D., Maier, G. (eds.) RAID 2011. LNCS, vol. 6961, pp. 338–357. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-23644-0\\_18](https://doi.org/10.1007/978-3-642-23644-0_18)
16. Lindorfer, M., Neugschwandtner, M., Weichselbaum, L., Fratantonio, Y., van der Veen, V., Platzer, C.: Andrubis - 1,000,000 apps later: a view on current Android malware behaviors. In: BADGERS (2014)
17. Maier, D., Müller, T., Protsenko, M.: Divide-and-conquer: why Android malware cannot be stopped. In: Proceedings of the 9th International Conference on Availability, Reliability and Security (ARES). SBA Research, Fribourg (2014)
18. Matenaar, F., Schulz, P.: Detecting Android sandboxes. <http://www.dexlabs.org/blog/btdetect>
19. Miramirkhani, N., Appini, M.P., Nikiforakis, N., Polychronakis, M.: Spotless sandboxes: evading malware analysis systems using wear-and-tear artifacts. In: IEEE Symposium on Security and Privacy (SP), pp. 1009–1024. IEEE (2017)

20. Moser, A., Kruegel, C., Kirda, E.: Limits of static analysis for malware detection. In: ACSAC, pp. 421–430. IEEE (2007)
21. Mutti, S., Fratantonio, Y., Bianchi, A., Invernizzi, L., Corbetta, J., Kirat, D., Kruegel, C., Vigna, G.: Baredroid: large-scale analysis of Android apps on real devices. In: ACSAC (2015)
22. Petsas, T., Voyatzis, G., Athanasopoulos, E., Polychronakis, M., Ioannidis, S.: Rage against the virtual machine: hindering dynamic analysis of Android malware. In: EUROSEC (2014)
23. Poeplau, S., Fratantonio, Y., Bianchi, A., Kruegel, C., Vigna, G.: Analyzing unsafe and malicious dynamic code loading in Android applications. In: NDSS (2014)
24. Reina, A., Fattori, A., Cavallaro, L.: A system call-centric analysis and stimulation technique to automatically reconstruct Android malware behaviors. In: EUROSEC (2013)
25. Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Bringas, P.G., Alvarez, G.: PUMA: permission usage to detect malware in Android. In: Herrero, Á., et al. (eds.) CISIS/ICEUTE/SOCO 2012 Special Sessions. AISC, vol. 189, pp. 289–298. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-33018-6\\_30](https://doi.org/10.1007/978-3-642-33018-6_30)
26. Spreitzenbarth, M.: The evil inside a droid - Android malware: past, present and future. In: Baltic Conference on Network Security & Forensics (2012)
27. Spreitzenbarth, M., Freiling, F., Ehtler, F., Schreck, T., Hoffmann, J.: Mobile-sandbox: having a deeper look into Android applications. In: ACM SAC (2013)
28. Su, X., Chuah, M., Tan, G.: Smartphone dual defense protection framework: detecting malicious applications in Android markets. In: International Conference on Mobile Ad-Hoc and Sensor Networks (2012)
29. Vidas, T., Christin, N.: Evading Android runtime analysis via sandbox detection. In: AsiaCCS (2014)
30. Wu, D.J., Mao, C.H., Wei, T.E., Lee, H.M., Wu, K.P.: DroidMat: Android malware detection through manifest and API calls tracing. In: Asia JCIS (2012)
31. Zheng, M., Sun, M., Lui, J.C.: Droid analytics: a signature based analytic system to collect, extract, analyze and associate Android malware. In: TrustCom (2013)
32. Zhou, Y., Jiang, X.: Dissecting Android malware: characterization and evolution. In: IEEE Symposium on Security & Privacy (2012)
33. Zhou, Y., Wang, Z., Zhou, W., Jiang, X.: Hey, you, get off of my market: detecting malicious apps in official and alternative Android markets. In: NDSS (2012)



# ICUFuzzer: Fuzzing ICU Library for Exploitable Bugs in Multiple Software

Kun Yang<sup>1,2(✉)</sup>, Yuan Deng<sup>3</sup>, Chao Zhang<sup>1,2</sup>, Jianwei Zhuge<sup>1,2</sup>,  
and Haixin Duan<sup>1,2</sup>

<sup>1</sup> Tsinghua University, Beijing, China  
k-yang14@mails.tsinghua.edu.cn

<sup>2</sup> Tsinghua National Laboratory for Information Science and Technology,  
Beijing, China

<sup>3</sup> Ant-financial Light-Year Security Lab, Hangzhou, China

**Abstract.** Software is usually built on top of shared libraries. Vulnerabilities that lie in those dependencies may have huge impact on multiple software. ICU (International Components for Unicode) is one of the most widely used common components in modern software, providing Unicode and Globalization support. ICU is used in a wide range of software from over 70 companies and organizations, including very popular software such as Chrome, Android, macOS, iOS, Windows 10, Edge, Firefox.

In this paper, we proposed a fuzzing method to discover vulnerabilities in ICU library that are reachable from upper layer application software. We also built a prototype named ICUFuzzer to uncover triggerable bugs in browsers' JavaScript Engine, with which we have detected three zero-day vulnerabilities affecting popular browsers like Chrome, Safari and Firefox. According to our further analysis, one of the bugs can be exploited to leak sensitive memory informations to bypass mitigations like ASLR and PIE.

**Keywords:** ICU · Fuzzing · Browser

## 1 Introduction

In recent years, security researchers have done some interesting vulnerability research in compromising modern software by exploiting security flaws from third party libraries. lokihardt successfully turned a buffer overflow bug in libANGLE to get remote code execution in Chrome browser in Pwn2Own 2016. Richard Zhu leveraged a bug in libvorbis to achieve code execution in Firefox in Pwn2Own 2018. Some researcher demonstrated how to use a one byte overflow in DNS library to take remote control of Chrome OS. From the cases above, we can learn

---

This work was partially supported by the National Natural Science Foundation of China (Grant No. 61472209 and No. 61772308), Tsinghua University Initiative Scientific Research Program (Grant No. 20151080436), the CCF-NSFOCUS Kungpeng Award, and the Young Talent Development Program by CCF.



that those shared libraries have become huge attack surfaces in modern software. Since third party libraries are shared code among multiple software, the bugs that lie in those dependencies are also shared. Bugs in fundamental libraries may have higher impact than bugs in code that is not shared. Designing new techniques for automatically finding hidden flaws in those fundamental dependencies now become important task, especially the bugs that are reachable in upper level applications.

Our research focus on a mature, widely used libraries named ICU [10]. ICU is short for International Components for Unicode, providing Unicode and Globalization support for software applications. ICU is widely portable and gives applications the same results on all platforms and between C/C++ and Java software. ICU library is used everywhere, including document processing software like OpenOffice and LibreOffice, browsers like Chrome, Safari and Firefox, operating systems like Android, macOS, iOS, Windows and Linux. ICU is integrated as a fundamental component through high level APIs in various software stack. Only security flaws in ICU that is reachable from the input through API calls could result in security impact. So our methodology is designed for finding bugs in ICU that is triggerable from the entry point of application software.

Fuzzing is a widely used technique to detect vulnerabilities in software. We designed a fuzzing based method to effectively find exploitable bugs in ICU libraries. In our method, we will do a in-depth analysis on the connection between target software and ICU library. Target software mostly use part of APIs from ICU, and does some parameter filtering before invoking. So the analysis result will help us infer the input generation rules for fuzzing, so that we can get rid of useless mutations as many as possible while fuzzing. Moreover, we combine coverage-guided fuzzing technique to improve the code coverage.

**Evaluation.** To evaluate the effectiveness of our design. We implemented a prototype ICUFuzzer to detect vulnerabilities in ICU that can be triggered from browsers' JavaScript engine. We have studied how the browsers implement ECMAScript Internationalization API with the underlying support of ICU and figured out how parameters are passed from the ECMAScript API to low level ICU functions, and how the browsers filter them for security concerns. With our findings, we developed a dedicated input generation engine inside the fuzzer which produces inputs that can bypass the filters. With our methodology, we significantly improve the effectiveness of fuzzing and quickly find three zero-day bugs, 2 of them can be triggered in Chrome and 1 can be triggered in Chrome, Safari, and Firefox.

**Contribution.** This paper makes the following contributions:

- We are the first to conduct fuzzing research on ICU library.
- We designed a fuzzer for finding bugs in ICU that is reachable from application software. By analyzing the data flow from the input in target application to the arguments passed down to ICU, we turn fuzzing target applications

into fuzzing ICU directly to achieve better performance and keep zero false positive at the same time. Coverage guided fuzzing technique is also applied here to further improve the code coverage.

- We implemented ICUFuzzer based on our fuzzer design and effectively find three zero-day bugs that are exploitable in modern browsers.

**Paper Organization.** The paper is organized as follows. In Sect. 2, we give brief introduction on ICU, including the attack surfaces of ICU. In Sect. 3, we use V8 JavaScript engine as an example to describe how ICU is integrated into other applications. In Sect. 4, we will introduce our system design, and how we build the prototype ICUfuzzer. In Sect. 5, we show the results of finding bugs by ICUFuzzer. In the last part of Sects. 6 and 7, we conclude our work and give a summary on other related work.

## 2 Preliminaries

### 2.1 ICU Basics

International Components for Unicode (ICU) is an open source project of mature C/C++ and Java libraries for Unicode support, software internationalization, and software globalization. It is widely portable to many operating systems and environments. It gives applications the same results on all platforms and between C, C++, and Java software. The ICU project is sponsored, supported, and used by IBM and many other companies.

Here are a few highlights of the services provided by ICU:

- Code Page Conversion: Convert text data to or from Unicode and nearly any other character set or encoding.
- Collation: Compare strings according to the conventions and standards of a particular language, region or country.
- Formatting: Format numbers, dates, times and currency amounts according the conventions of a chosen locale.
- Time Calculations: Multiple types of calendars are provided beyond the traditional Gregorian.
- Unicode Support: ICU closely tracks the Unicode standard, providing easy access to all of the many Unicode character properties.

### 2.2 Who Uses ICU?

ICU is used in almost every popular operation systems. In Apple macOS, ICU library is going with the default install under the name `libcucore.A.dylib`. In Microsoft Windows 10, ICU library is distributed in two dynamic link libraries, `icuin.dll` and `icuuc.dll`. In Ubuntu, the package of ICU contains multiple shared libraries named `libcuc*`.so.

ICU is heavily used in document processing software like LibreOffice, OpenOffice, PDF Box and products from Adobe. Document processing software

have been large attack surfaces used in real world APT attacks for years. Document exploits are usually malformed documents distributed by email phishing. So exploitable bugs in ICU are possible to be abused in traditional document attack scenario.

ICU is also used in trending application software like mobile operating systems, IoT devices, and smart cars. ICU can be found in Android and iOS, and even in the automotive from vendors like Alfa Romeo, Audi, Mercedes-Benz, BMW.

ICU is used everywhere. The work for finding bugs in such a fundamental library is critical.

### 2.3 Attack Surfaces in ICU

Core function of ICU that many software rely on is Unicode and internationalization support, which are encoding translation from or to Unicode, locale conversion for numbers, dates, times and currency amounts, Unicode supported regular expression, etc. All the operations above are related to encoding and format translation, which are implemented with lots of memory manipulations. The C/C++ code for handling of large amount of specifications, encodings and locales could become large attack interfaces. In our research, we focus on memory corruption bugs in ICU's C/C++ library.

## 3 ICU in JavaScript Engine

### 3.1 ECMAScript Internationalization API

The ECMAScript Internationalization API [11] is a standard that helps handle locales of dates, numbers, and currencies in JavaScript. There is a specification that defines the application programming interface for ECMAScript objects. Most of the modern browsers implement ECMAScript Internationalization API based on ICU including Chrome, Safari and Firefox.

### 3.2 Architecture

To explain how ICU is integrated with JavaScript engine, we take Chrome's V8 engine as an example. Figure 1 shows the implementation architecture in Chrome's JavaScript engine V8. Every JavaScript Internationalization API call will be first handled by some internal JavaScript code and runtime engine written in C++, and then the execution goes down to the ICU library.

So JavaScript Internationalization API calls can be the surfaces of browser attacks. If malformed arguments of Internationalization API calls are passed down to ICU, bugs in ICU will be triggered. However, according to our analysis, not arbitrary arguments can be passed directly to functions in ICU. There are security checks or filters in V8, which prevents some ICU bugs to be triggered. Our fuzzer will only focus on reachable bugs.

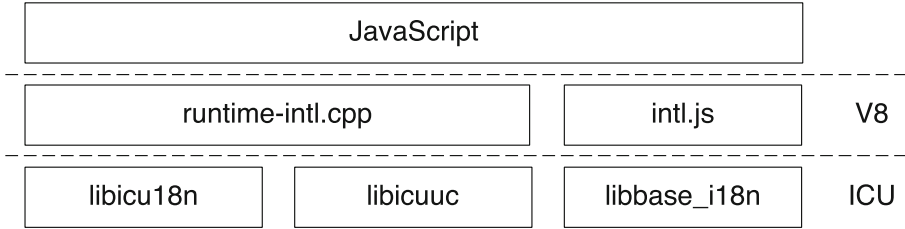


Fig. 1. Internationalization API architecture in V8

### 3.3 JavaScript Internationalization API

The Intl object [12] is the namespace for the ECMAScript Internationalization API, which provides language sensitive string comparison, number formatting, and date and time formatting. The Intl object have 4 properties, which are constructors for Collator, NumberFormat, DateTimeFormat and PluralRules objects:

- The Intl.Collator object is a constructor for collators, objects that enable language sensitive string comparison.
- The Intl.DateTimeFormat object is a constructor for objects that enable language-sensitive date and time formatting.
- The Intl.NumberFormat object is a constructor for objects that enable language sensitive number formatting.
- The Intl.PluralRules object is a constructor for objects that enable plural sensitive formatting and plural language rules.

The objects above are core structures in Internationalization API. The code snippet below demonstrates the character comparison between “ä” in German and “a” in English by creating an Intl.Collator object and calling its method compare.

```
var options = { sensitivity: 'base' };
new Intl.Collator('de', options).compare('ä', 'a');
// 0
```

The code snippet below illustrates the date and time formatting by creating an Intl.DateTimeFormat object and calling its method format.

```
var date = new Date(Date.UTC(2012, 11, 20, 3, 0, 0));
// request a weekday along with a long date
var options = { weekday: 'long', year: 'numeric',
               month: 'long', day: 'numeric' };
new Intl.DateTimeFormat('de-DE', options).format(date);
// "Donnerstag, 20. Dezember 2012"
```

The code snippet below illustrates the currency formatting by creating an Intl.NumberFormat object and calling its method format.

```

var number = 123456.789;
var options = {style:'currency',currency:'EUR' };
// request a currency format
new Intl.NumberFormat('de-DE', options).format(number);
// 123.456,79 €

```

Table 1 summarized all the constructors and methods in JavaScript Internationalization API. We can see all the constructors receive two optional arguments: `locales` and `options`.

**Table 1.** Constructors and methods in JavaScript internationalization API

Objects/Constructors	Methods
<code>Intl.Collator(locales, options)</code>	<code>Intl.Collator.prototype.resolvedOptions()</code>
<code>Intl.DateTimeFormat(locales, options)</code>	<code>Intl.DateTimeFormat.prototype.format(date)</code>
	<code>Intl.DateTimeFormat.prototype.formatToParts(date)</code>
	<code>Intl.DateTimeFormat.prototype.resolvedOptions()</code>
<code>Intl.NumberFormat(locales, options)</code>	<code>Intl.NumberFormat.prototype.format(number)</code>
	<code>Intl.NumberFormat.prototype.formatToParts(number)</code>
	<code>Intl.NumberFormat.prototype.resolvedOptions()</code>
<code>Intl.PluralRules(locales, options)</code>	<code>Intl.PluralRules.prototype.resolvedOptions()</code>
	<code>Intl.PluralRules.prototype.select(number)</code>

**Locales Argument.** The `locales` argument must be either a string holding a BCP 47 language tag [13], or an array of such language tags. A BCP 47 language tag defines a language and minimally contains a primary language code. In its most common form it can contain, in order: a language code, a script code, and a country or region code, all separated by hyphens. The subtags identifying languages, scripts, countries (regions), and (rarely used) variants in BCP 47 language tags can be found in the IANA Language Subtag Registry [10]. While the tag is not case sensitive, it is recommended to use title case for script code, upper case for country and region codes and lower case for everything else. The following are examples of `locales`:

- “hi”: Hindi (primary language).
- “de-AT”: German as used in Austria (primary language with country code).
- “zh-Hans-CN”: Chinese written in simplified characters as used in China (primary language with script and country codes).

BCP 47 also allows for extensions. JavaScript internationalization functions use the “u” (Unicode) extension, which can be used to request additional customization of `Collator`, `NumberFormat`, or `DateTimeFormat` objects. The Unicode extension uses additional keys as subtags.

Let’s take the constructor `Intl.Collator` as an example. `Collator` supports 3 Unicode extension keys, `co`, `kn`, and `kf`. These keys can be used in the `locales`,

and `locales` can be passed to the constructor to customize the `Collator` object:

- `co` is for variant collations for certain locales. Possible values include: "big5han", "dict", "direct", "ducet", "gb2312", "phonebk", "phonetic", "pinyin", etc.
- `kn` specifies whether numeric collation should be used, such that "1" < "2" < "10". Possible values are "true" and "false".
- `kf` specifies whether upper case or lower case should sort first. Possible values are "upper", "lower", or "false" (use the locale's default).

More details about Unicode extension keys for other objects can be found in MDN web docs [12]. Table 2 summarizes extension keys and possible values. Below are some example `locales` that use Unicode extension keys:

- "de-DE-u-co-phonebk": Use the phonebook variant of the German sort order, which expands unlauded vowels to character pairs: ä → ae, ö → oe, ü → ue.
- "en-GB-u-ca-islamic": Use British English with the Islamic (Hijri) calendar, where the Gregorian date 14 October, 2017 is the Hijri date 24 Muharram, 1439.

**Table 2.** Unicode extensions keys and values in `Locales` argument

Constructor	Unicode extension key	Possible values
<code>Collator</code>	<code>co</code>	"big5han", "dict", "direct", "ducet", "gb2312", "phonebk", "phonetic", "pinyin", etc.
	<code>kn</code>	"true", "false"
	<code>kf</code>	"upper", "lower", "false"
<code>DateTimeFormat</code>	<code>nu</code>	"arab", "arabext", "bali", "beng", "deva", "fullwide", "gujr", "guru", "hanidec", etc.
	<code>ca</code>	"buddhist", "chinese", "coptic", "ethioaa", "ethiopic", "gregory", "hebrew", "indian", "islamic", "islaamic", etc.
	<code>hc</code>	"h11", "h12", "h23", "h24"
<code>NumberFormat</code>	<code>nu</code>	"arab", "arabext", "bali", "beng", "deva", "fullwide", "gujr", "guru", "hanidec", etc.

**Options Argument.** The `options` argument must be an object with properties that vary between constructors and functions. Properties in the `options` argument are like optional configurations. Different constructors and functions require different properties. For example, the constructor `Intl.DateTimeFormat` supports some properties as below:

- `localeMatcher` specifies the locale matching algorithm to use. Possible values are “lookup” and “best fit”; the default is “best fit”. This property is supported by all language sensitive constructors and functions.
- `timeZone` specifies the time zone to use, such as "Asia/Shanghai", "Asia/Kolkata", "America/New\_York".
- `hour12` specifies Whether to use 12-h time (as opposed to 24-h time). Possible values are true and false; the default is locale dependent.

Detailed information on properties and functions can be retrieved from MDN web docs [12].

### 3.4 Data Flow from JavaScript to ICU

In previous section, we have gone through the entry level data structures and methods in Javascript International API. In this part, we will dive deep into the implementation of V8 engine to understand how input arguments are handled and passed to ICU.

We will follow the execution of constructor and methods of `Intl.DateTimeFormat` to understand the argument data flow. Figure 2 depicts the basic

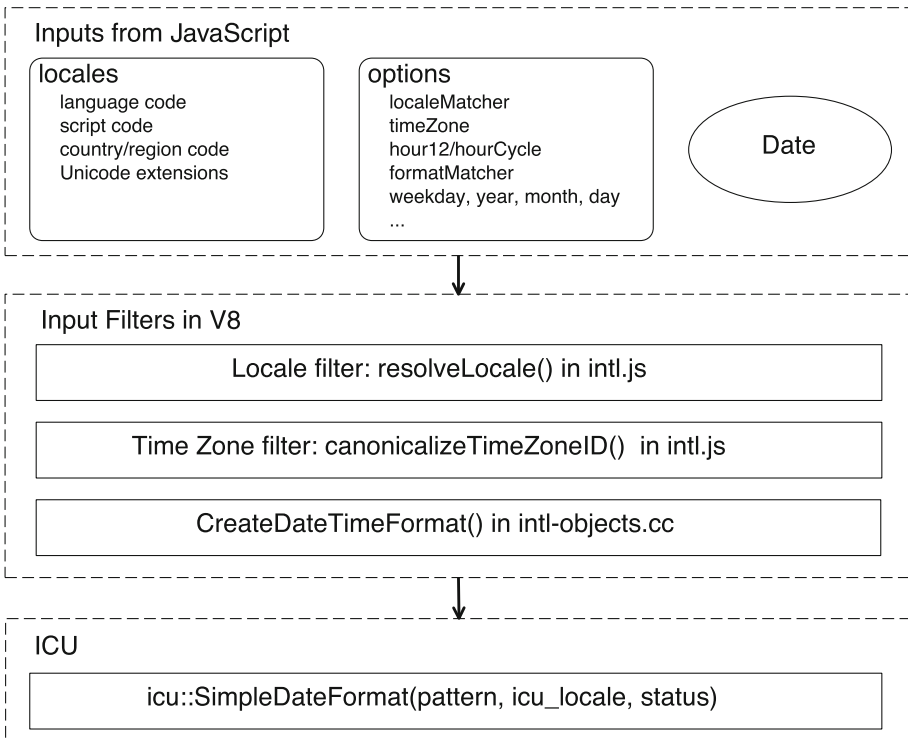


Fig. 2. Date and Time related International API’s Input Filtering Process in V8

procedure. The inputs for `Intl.DateTimeFormat` are `locales`, `options` and `Date`. `locales` will be resolved in function `resolveLocale()` and filtered. Then time zone information will be extracted from `options` and filtered in `canonicalizeTimeZoneID()`. Both filters are implemented in internal JavaScript code `intl.js`.

**Locale Filtering.** For locale filtering in `resolveLocale()`, the following steps will be taken:

1. Canonicalize the language code in function `canonicalizeLanguageTag()`
  - (a) Call function `isStructurallyValidLanguageTag()`, to match the locale by three regular expressions: `LANGUAGE_TAG_RE`, `LANGUAGE_VARIANT_RE` and `LANGUAGE_SINGLETON_RE`
  - (b) check if `locale` conform to strict BCP47 rules by performing a pair of invertible operations: calling `uloc_forLanguageTag()` and `uloc_toLanguageTag()`
2. Split the Unicode extensions from `locale`
3. Check extension keys extracted from `locale` by matching a pre-initialized key list

**Options Filtering.** Just like `locale`, V8 will also filter the argument `options` before passing it to ICU. In the constructor `Intl.DateTimeFormat`, the property `timeZone` in argument `options` will be filtered in function `canonicalizeTimeZoneID()` by following steps:

1. Use regular expression to check if `timeZone` string matches the format of *Area/Location(/Location)\**
2. Convert the location string to be titlecased by function `toTitleCaseTimezoneLocation()`. For example, convert “bueNos.airES” to “Buenos\_Aires”.

## 4 System Design

### 4.1 Design Goals

In Sect. 3, we study how ICU is integrated inside V8 engine. Most of modern application software like browsers are designed and implemented in a hierarchical structure, which depends on a lot of fundamental libraries in the low level. ICU is one of the critical components and has not received much attention on security aspect.

In this paper, we aim at designing a fuzzing methodology to uncover the bugs in ICU library. It is worth emphasizing that the ICU bugs that we are fuzzing for should be reachable from target applications. Our fuzzer is designed for hunting bugs that can affect specific application. The reason is that, if we consider a library separately, the threat model may not be very clear. Libraries are only responsible for exposing dozens of APIs, and library developers may



know little on how these interfaces will be used in the applications. So if we apply bug finding in ICU in the context of application software, the results will bring more practical value.

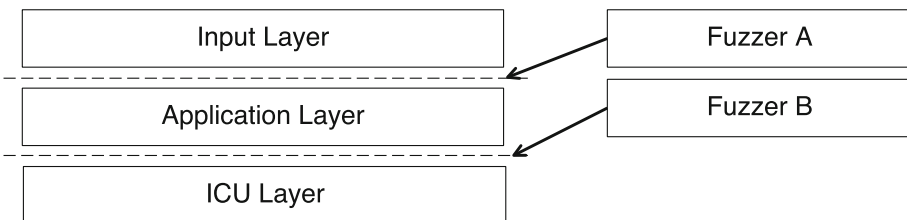
Research on general fuzzing techniques has been lasted for long time and already made very good progress. In our research, we are going to apply the in-depth understanding on ICU into the fuzzer to achieve better performance and results.

## 4.2 Design Challenges

Fuzzing ICU for exploitable bugs in target application could be carried out in two different ways. One is fuzzing the application directly. By mutating and feeding the input that is ICU related to the target software, the crashes you get are real bugs that will surely affect the application. However, according to our analysis on V8, there will be a middle layer between the input layer and ICU that is always checking or filtering the test cases generated by the fuzzer. So the mutation will be struggled in surviving the filters, large amount of computation resources in fuzzing will be wasted in finding paths in the application that can reach ICU. Meanwhile, compared with ICU library, application is more complex software like browsers and pdf readers. Fuzzing ICU in the context of whole application is slow. To conclude, this method has an advantage of 0 false negative, but has low efficiency.

The other method is fuzzing ICU APIs directly. By figuring out which APIs target application relies on, fuzzer can skim the target software and feed mutated arguments directly to ICU APIs. The second method also has a drawback that even if you find a test case that can crash ICU, the bug could not be reproduced in the context of target application. The input will go into the application first and be filtered before being passed to the low level's library. This idea can help improve the fuzzing throughput, but will have very high false positive.

In Fig. 3, it shows the different fuzzing points between two methods discussed above. The challenge we are facing here is that we cannot have low false negative and high fuzzing throughput at the same time.



**Fig. 3.** Different Fuzzing Point Between Two Methods

### 4.3 System Architecture

To overcome the drawbacks in two fuzzing methods discussed above, and keep the good parts for both, we can fuzz the ICU API directly while embedding the filters in target application into the fuzzer to make sure there is no false positive. By analyzing ICU and its depending software, we can obtain the filtering rules like what we have done in Sect. 3 for browsers. Then by embedding the rules inside our fuzzer, we could filter the invalid test cases before feeding them to ICU. In the Fig. 4 below, it illustrates the basic idea of our fuzzer. We call this kind of fuzzer context aware fuzzer, because by applying the filtering rules in the application, the fuzzer is granted with the knowledge from the target software. Every test case will be checked inside fuzzer before going to ICU, and there is no need to run the entire application while fuzzing.

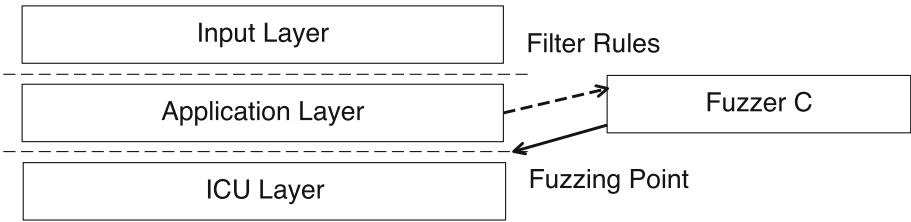


Fig. 4. Context aware ICU fuzzer

In our fuzzer, we can further improve the effectiveness of test cases generation by applying grey-box fuzzing method. We introduced coverage guided fuzzing technique here. In Fig. 5, we can see all of the internal modules for the ICU fuzzer. The *Executor* always get new test case from the *Mutator*, then pass it

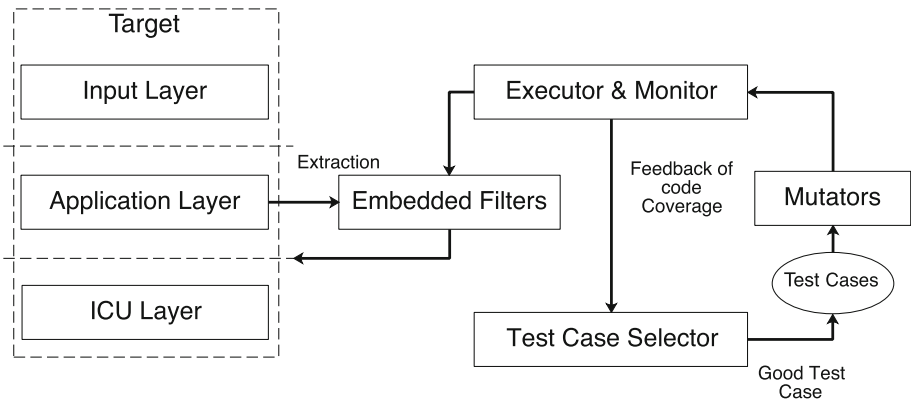
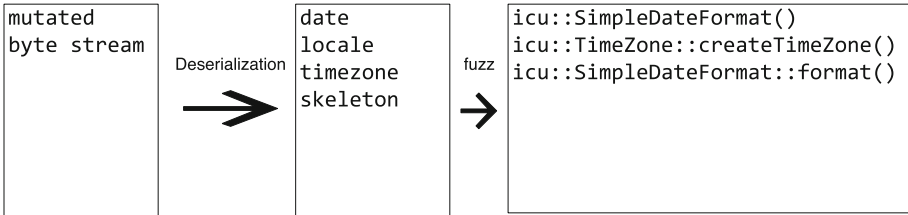


Fig. 5. Context aware ICU fuzzer combined with coverage guided fuzzing

to the *Embedded Filters* extracted from target application. The filter will stop the execution if the test case is not valid, other than the execution will go to the ICU API for testing. *Executor* should be able to retrieve the code coverage information and decide if this test case should be further mutated or abandoned. Good test cases will be remained and the execution will be guided to discover new branches in ICU library.

**Executor and Embedded Filters.** The argument filters in the applications are usually implemented for filtering `locale` strings and various properties in `options`. According to Sect. 3, we can see in the browser case, most of these filters are regular expressions. So we can collect all the rules and program them in the fuzzer as embedded filters. For the test inputs that can bypass the filters, we pass them to native ICU calls.

**Mutation.** Different ICU APIs have different arguments. In order to design a mutation method that is able to adapt to different number and types of arguments. We use a popular mechanism supported in almost every programming language to unify the input data for different ICU APIs—serialization and deserialization. We take byte stream as a unifying input format in fuzzing. Mutation is easy to be implemented on byte stream, e.g. bit flips, byte flips, bit rotations or arithmetic operations. When a test case of byte stream is delivered to *Executor*, the input stream will first be deserialized into corresponding arguments that is compatible with ICU APIs to be tested. Figure 6 demonstrates the deserialization and mutation process.



**Fig. 6.** Context aware ICU fuzzer combined with coverage guided fuzzing

#### 4.4 ICUFuzzer for JavaScript Engine

To evaluate our design, we implemented a prototype named ICUFuzzer to fuzz for exploitable bugs in browsers. We use libFuzzer [19] as underlying support for coverage-guided fuzzing. We focus on fuzzing International APIs—constructors and methods of 4 core objects: `Collator`, `DateTimeFormat`, `NumberFormat` and `PluralRules`, as listed in Table 1. Let’s take `DateTimeFormat` and `NumberFormat` as two examples to show how we map the upper level APIs to low level ICU APIs.

**Fuzzing Intl.DateTimeFormat.** In JavaScript, methods from the object `Date-TimeFormat` are invoked in the following manner:

1. Create an object of `Intl.DateTimeFormat` by the constructor, specifying `locales` and `options`.
2. Use the object initialized in the previous step to call `Format(date)` or `formatToParts(date)` to format `date` into specified `locales` and `options`.

All the arguments specified in JavaScript will be filtered in V8 and passed to APIs in ICU. After passing down to the ICU part, three methods will be invoked:

1. `icu::SimpleDateFormat()` creates an object `SimpleDateFormat` in ICU from `locales` and properties in `options`.
2. `icu::TimeZone::createTimeZone()` creates the `TimeZone` object information in ICU from `timeZone` property in `options`.
3. `icu::SimpleDateFormat::format()` is responsible for arbitrary date formatting according to the configuration.

According to our design, ICUFuzzer will directly fuzz the ICU functions (Figs. 7 and 8).

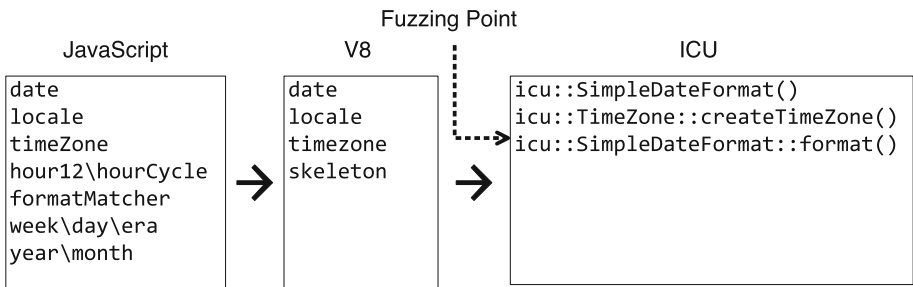


Fig. 7. Data flow in `Intl.DateTimeFormat`

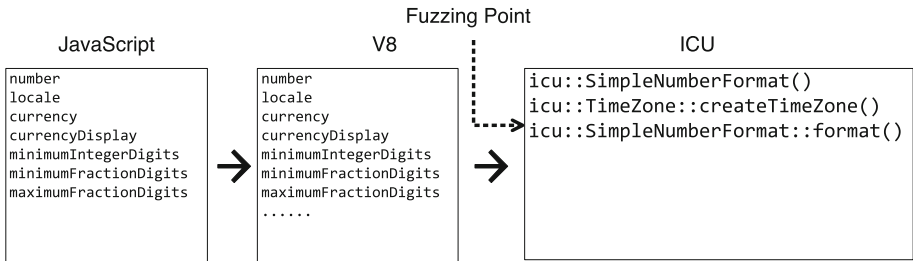


Fig. 8. Data flow in `Intl.NumberFormat`

**Fuzzing Intl.NumberFormat.** In JavaScript, methods from the object `Date-Format` are invoked in the following procedure:

1. Create an object of `Intl.NumberFormat` by the constructor, specifying `locales` and `options`.
2. Use the object initialized in the previous step to call `Format(date)` or `formatToParts(date)` to format `number` into specified `locales` and `options`.

All the arguments specified in JavaScript will be filtered in V8 and passed to APIs in ICU. After passing down to the ICU part, three methods will be invoked:

1. `icu::SimpleNumberFormat()` creates an object `SimpleNumberFormat` in ICU from `locales` and properties in `options`.
2. `icu::TimeZone::createTimeZone()` creates the `TimeZone` object information in ICU from `timeZone` property in `options`.
3. `icu::SimpleNumberFormat::format()` is responsible for arbitrary number formatting according to the configuration.

**Filters in V8.** According to the analysis result in Sect. 2, there are `locale` filter and `options` filter implemented in V8's `Intl.js`. We extracted all the filtering rules for `locale` as multiple regular expressions as below.

```

regex simple_re("^([a-z]{2,3})$", std::regex::ECMAScript);
regex singleton_re("^([0-9]|[A-WY-Za-wy-z])$", std::regex::ECMAScript |
std::regex::icase);
regex variant_re("^(|[a-zA-Z][0-9]){5,8}|([0-9]([a-zA-Z][0-9]){3})$",
std::regex::ECMAScript | std::regex::icase);
regex langtag_re("^(|[a-zA-Z]{2,3}(-([a-zA-Z]{3}(-[a-zA-Z]{3}){0,2}))?[a-zA-Z]{4}|[a-zA-Z]{5,8})(-([a-zA-Z]{4}))?(-([a-zA-Z]{2}|[0-9]{3}))
?(-([a-zA-Z][0-9]){5,8}|([0-9]([a-zA-Z][0-9]){3})))*(-([0-9]|[A-WY
-Za-wy-z])(-([a-zA-Z][0-9]){2,8})+)*(-x(-([a-zA-Z][0-9]){1,8})+))
?|(x(-([a-zA-Z][0-9]){1,8})+)|((en-GB-oed|i-ami|i-bnn|i-default|i-
enochian|i-hak|i-klingon|i-lux|i-mingo|i-navajo|i-pwn|i-tao|i-tay|i-
tsu|sgn-BE-FR|sgn-BE-NL|sgn-CH-DE)|(art-lojban|cel-gaulish|no-bok|no-
nyn|zh-guoyu|zh-hakka|zh-min|zh-min-nan|zh-xiang))$", std::regex::
ECMAScript | std::regex::icase);
regex any_ext_re("-[a-z0-9]{1}-.*", std::regex::ECMAScript);
regex uni_ext_re("-u(-[a-z0-9]{2,8})+", std::regex::ECMAScript);
regex locale_re("^([a-z]{2,3})-([A-Z][a-z]{3})-([A-Z]{2})$", std::regex::
ECMAScript);

```

For `options`, we also listed the checking rules in V8 for different properties in Table 3.

**Table 3.** Part of extracted rules in V8’s options filter.

Property	Type	Filtering rules for value
Weekday	string	narrow: ‘EEEEEE’, short: ‘EEE’, long: ‘EEEE’
Era	string	narrow: ‘GGGGG’, short: ‘GGG’, long: ‘GGGG’
Year	string	2-digit: ‘yy’, numeric: ‘y’
Month	string	2-digit: ‘MM’, numeric: ‘M’, narrow: ‘MMMMM’, short: ‘MMM’, long: ‘MMMM’
Day	string	2-digit: ‘dd’, numeric: ‘d’

The filters above are embedded into ICUFuzzer to get rid of potential false positive test cases, which could only crash ICU but not the browsers.

## 5 Evaluation

### 5.1 Experiment Design

To evaluate bug hunting capability of ICUFuzzer, we run it in a desktop PC that equipped with Intel i7 processor with clock speed of 2.67 GHz, with 8 Gb of RAM, and with Ubuntu 16.04. Within 10 min, we got dozens of crashes. We manually analyzed and classified the crashes, then we realized we found 3 0 days in latest ICU code base, and all of them can crash the latest Chrome browser through loading a snippet of JavaScript where we invoke JavaScript International APIs with malformed arguments. We reported to Chrome and have all the bugs fixed. We will go through the 3 bugs in the next part.

### 5.2 Results

**CVE-2017-15422: Persian Calendar Integer Overflow.** This bug affects Chrome, Safari and Firefox and exists in the code snippet below:

```

//i18n/persncal.cpp
void PersianCalendar::handleComputeFields(int32_t julianDay,
    UErrorCode &/*status*/)
{
    int32_t daysSinceEpoch = julianDay - PERSIAN_EPOCH;//
    year = 1 + ClockMath::floorDivide(33 * daysSinceEpoch + 3,
        12053);
    .....
    dayOfMonth = dayOfYear - kPersianNumDays[month] + 1; // Out
        of bound memory read
}

```

Integer overflow can happen in the expression `33 * daysSinceEpoch`, leading to an unbounded `month` value. The unbounded `month` value will be used as an index in array `kPersianNumDays`, which can be exploited to read out of bound memory. We have successfully turn the bug into a working exploit targeting

Chrome, Safari and Firefox to leak memory addresses, which means modern mitigations ASLR and PIE are bypassed. According our survey, this bug has been existed for more than 5 years.

Here is the proof of concept code that can trigger the bug in the browser.

```
var dateFormatter = new Intl.DateTimeFormat("bs-Cyrl-u-ca-persian");
date = null;
Date.prototype["valueOf"] = function (){}; //date returns NaN
d = dateFormatter.formatToParts(date);
```

### **CVE-2017-15396: NumberingSystem::createInstance Stack Overflow.**

This bug only affects Chrome. Look at the code snippet below:

```
char buffer[ 96 ];
int32_t count = inLocale.getKeywordValue("numbers",buffer,
    sizeof(buffer),status);
if ( count > 0 ) {
    buffer[count] = '\0';    //count = 99
```

The integer variable `count` can exceed the size of `buffer`, leading to a stack out of bound write.

Proof of concept code is as below:

```
var nf = new Intl.NumberFormat('bs-u-nu-bzcu-cab-cabs-avnlubs-avnihu-zcu-cab-cbs-avnllubs-avnihq-zcu-cab-cbs-ubs-avnihu-cabs-flus-xxd-vnluy');
```

**CVE-2017-15406: CanonicalizeLanguageTag Stack Overflow.** This bug only affects Chrome. Look at the code snippet below:

```
char icu_result[ULOC_FULLNAME_CAPACITY];
uloc_forLanguageTag(*locale_id, icu_result,
    ULOC_FULLNAME_CAPACITY, nullptr, &error);
// localeID is not terminated with null byte
...
if (uprv_strlen(localeID) > 0) {    // overflowed
```

`localeID` doesn't need to be null terminated, so the result of `uprv_strlen(localeID)` could be oversized, leading to a stack out of bound write.

Proof of concept code is as below:

```
var dateti1 = new Intl.DateTimeFormat("iw-up-a-caiaup-araup-ai-pdu-sp-bs-up-arscna-zeieiaup-araup-arscia-rews-us-up-arscna-zeieiaup-araup-arsciap-arscna-zeieiaup-araup-arscie-u-sp-bs-uaup-arscia");
```

## 6 Related Work

Fuzzing was originally introduced as one of several tools to test UNIX utilities [1]. Since then, much work has been devoted to improving general fuzzing technique. Guided fuzzing is proposed to direct fuzzers toward specific types of vulnerabilities. One of the typical design for guided fuzzing is selectively choosing optimal test cases [2,3]. Dowser [2] and BuzzFuzz [4] applies taint-tracking to analyze the relations between test cases and code regions in target software. Flayer [5] allows an auditor to skip complex checks in the target application to improve the code coverage of fuzzing. Similarly, Taintscope [6] uses a checksum detection algorithm to remove checksum code from applications. Symbolic execution is also applied to fuzzing to gain maximal code coverage [7–9]. These approaches rely on symbolic execution to generate inputs that will take new code paths.

There are many other research focus on applying fuzzing in some application scenarios and making improvement by using unique characteristics in the scenarios, including our research in the paper. IntentFuzzer [15] applies dynamic fuzzing method in Android IPC mechanism to find permission leak vulnerabilities. jFuzz [16] uses a combination of concrete and symbolic execution to fuzz Java program. MTF [17] proposed a guided fuzzing strategy instead of random testing to fuzz Modbus/TCP protocol. IOTFUZZER [18] applies the knowledge from mobile applications in the fuzzer to find memory corruptions in IoT firmware.

## 7 Conclusion

In this paper, we proposed a fuzzing based method to discover bugs in ICU. Our method is interested in the bugs that are exploitable in the context of application software that use ICU. After we conducted a in-depth study on the implementation details of JavaScript International API, we understand the data flow from entry level input to low level ICU functions and the underlying input filtering mechanism. By applying these knowledge to the fuzzer, we improve the fuzzer to have high throughput and low false positive at the same time. We implemented a prototype named ICUFuzzer to evaluate our design. ICUfuzzer is dedicated for finding ICU bugs in browsers. By running the fuzzer, we have found three zero-day bugs in modern browsers. One of them can be exploited to leak memory information, the others can crash the browser and possibly get remote code execution.

## References

1. Miller, B.P., Fredriksen, L., So, B.: An empirical study of the reliability of UNIX utilities. *Commun. ACM* **33**(12), 32–44 (1990)
2. Haller, I., Slowinska, A., Neugschwandtner, M., et al.: Dowsing for overflows: a guided fuzzer to find buffer boundary violations. In: *USENIX Security Symposium*, pp. 49–64 (2013)



3. Neugschwandtner, M., Milani Comparetti, P., Haller, I., et al.: The BORG: nanoprobing binaries for buffer overreads. In: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, pp. 87–97. ACM (2015)
4. Ganesh, V., Leek, T., Rinard, M.: Taint-based directed whitebox fuzzing. In: Proceedings of the 31st International Conference on Software Engineering, pp. 474–484. IEEE Computer Society (2009)
5. Drewry, W., Ormandy, T.: Flayer: exposing application internals. WOOT **7**, 1–9 (2007)
6. Wang, T., Wei, T., Gu, G., et al.: TaintScope: a checksum-aware directed fuzzing tool for automatic software vulnerability detection. In: IEEE symposium on Security and privacy (SP), pp. 497–512. IEEE (2010)
7. Caselden, D., Bazhanyuk, A., Payer, M., et al.: Transformation-aware Exploit Generation using a HI-CFG. Department of Electrical Engineering and Computer Science, California University, Berkeley (2013)
8. Godefroid, P., Klarlund, N., Sen, K.: DART: directed automated random testing. In: ACM Sigplan Notices, vol. 40, no. 6, pp. 213–223 (2005)
9. Godefroid, P., Levin, M.Y., Molnar, D.: SAGE: whitebox fuzzing for security testing. Commun. ACM **55**(3), 40–44 (2012)
10. ICU - International Components for Unicode. <http://site.icu-project.org/>
11. ECMAScript Internationalization API Specification. <https://www.ecma-international.org/ecma-402/1.0/>
12. Document for Intl object from MDN. [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects)
13. RFC 5646 - Tags for Identifying Languages. <https://tools.ietf.org/html/rfc5646>
14. IANA Language Subtag Registry. <https://www.iana.org/assignments/language-subtag-registry/language-subtag-registry>
15. Yang, K., Zhuge, J., Wang, Y., et al.: IntentFuzzer: detecting capability leaks of android applications. In: Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, pp. 531–536. ACM (2014)
16. Jayaraman, K., Harvison, D., Ganesh, V., et al.: jFuzz: a concolic whitebox fuzzer for Java (2009)
17. Voyiatzis, A.G., Katsigiannis, K., Koubias, S.: A Modbus/TCP fuzzer for testing internetworked industrial systems. In: 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), pp. 1–6. IEEE (2015)
18. Chen, J., Diao, W., Zhao, Q., et al.: IoTfuzzer: Discovering Memory Corruptions in IoT Through App-based Fuzzing
19. libFuzzer - a library for coverage-guided fuzz testing. <https://lvm.org/docs/LibFuzzer.html>



# How Safe Is Safety Number? A User Study on SIGNAL's Fingerprint and Safety Number Methods for Public Key Verification

Kemal Bicakci<sup>1,3</sup>(✉), Enes Altuncu<sup>1</sup>, Muhammet Sakir Sahkulubey<sup>1</sup>,  
Hakan Ezgi Kiziloz<sup>2</sup>, and Yusuf Uzunay<sup>3</sup>

<sup>1</sup> TOBB University of Economics and Technology, Ankara, Turkey  
{bicakci,ealtuncu,msahkulubey}@etu.edu.tr

<sup>2</sup> University of Turkish Aeronautical Association, Ankara, Turkey  
hakan.kiziloz@ceng.thk.edu.tr

<sup>3</sup> Securify Information Tech. and Security Training Consulting Ltd., Ankara, Turkey  
{kemal.bicakci,yusuf.uzunay}@securify.com.tr

**Abstract.** Communication security has become an indispensable demand of smartphone users. End-to-end encryption is the key factor for providing communication security, which mainly relies on public key cryptography. The main and unresolved issue for public key cryptography is to correctly match a public key with its owner. Failing to do so could lead to man-in-the-middle attacks. Different public key verification methods have been proposed in the literature. The methods which are based on verification by the users themselves are preferable with respect to cost and deployability than the methods such as digital certificates that involve the use of trusted third parties. One of these methods, fingerprinting was recently replaced by a method called safety number in the open source messaging application, SIGNAL. The developers of SIGNAL claimed this change would bring usability and security advantages however no formal user study was conducted supporting this claim. In this study, we compare the usability and security aspects of these two methods with a user study on 42 participants. The results indicate with significance that the safety number method leads to more successful results in less time for public key verification as compared to the fingerprint method.

**Keywords:** Public key verification · Safety number · Fingerprint Usability · SIGNAL

## 1 Introduction

Secure communication over the Internet has become a crucial need for all of us, today. End-to-end encryption is the name of the tool and the technology that ensures encryption is performed with the keys stored only at the users' devices,

hence the communication providers and other third parties are not able to read the message content. However, first generation of end-to-end encryption tools such as PGP have not become widespread mainly because they are not easy to use [3, 4, 6, 9].

Since the first days of end-to-end encryption, two important aspects in secure messaging landscape have changed: (i) with smartphones taking place of personal computers (PCs), communication has mostly been through mobile devices, and (ii) the awareness about privacy and security has increased.

The new generation of messaging applications falls into two categories from security point of view; applications that provide encryption between the user and a server and applications that provide end-to-end encryption. Applications in the first category allow the service provider and others to read the messages being sent and received hence offer very little for those caring their security and privacy [7].

The requirements for a mobile messaging application targeting mainly consumer market to provide end-to-end encryption is minimum. Unlike some enterprise settings, digital certificates and trusted third parties are not involved. As soon as the application is downloaded to the smartphone, and registration is made via the mobile number, required keys for end-to-end encryption are generated. A central database stores identities of the registered users, and shares corresponding identities to the users that want to communicate with others. This system is not free of risks. For instance, if the central database is intentionally manipulated by the system administrator or manipulated as the outcome of an attack and user identities and/or their public keys are changed, man-in-the-middle attacks are possible. As a result, the ability to verify communicating parties via their public keys becomes essential for a secure communication.

Public key verification is presented through various user interfaces. Recently, messaging application SIGNAL has made a change and began using a method called safety numbers instead of fingerprints. In our work, our aim is to analyze security and usability aspects of this new public key verification method and compare it against fingerprint method. For this purpose, we design and conduct a user study using previous and current versions of SIGNAL on Android smartphones. SIGNAL Android application is chosen in our work for several reasons. First, its open source nature allows us to set up our own environment for the user study (this is especially important to test the older version). Second, it is one of the few messaging applications which seem to realize the importance of public key verification problem (the rationale behind the transition to safety number method - happened in November 2016 [5] - is well-documented in their official blog, though it lacks any formal supporting arguments). Finally, WhatsApp, another instant messaging application having more than a billion users, uses the SIGNAL protocol for end-to-end encryption [2].

Its leadership as a privacy advocate and general public acceptance of security and trust offered by SIGNAL hints us that the safety number method becomes more and more widely used in applications. Therefore, we strongly believe a timely user study to investigate the security and usability of this public key

verification method fills an important hole. To our best knowledge, our study is the first in evaluating safety number method and comparing it with fingerprint method with a formal user study. For this purpose, we devise a user study in a lab environment with 42 users. The obtained results indicate that public key verification via safety number method is indeed more advantageous than the public key verification via fingerprint method with respect to success rate and verification time.

The rest of the paper is organized as follows. We briefly give background information and present related work in the following section, Sect. 2. In Sect. 3, we describe the user study in detail and define our hypotheses for the study. After presenting the obtained results, we discuss them together with statistical analysis in Sect. 4. Finally, we finish up the paper with concluding remarks in Sect. 5.

## 2 Background and Related Work

In this section, we briefly describe the session establishment and maintaining protocol used in SIGNAL and the public key verification methods used in this study; safety number and fingerprint method. We also present the concerned threat model and summarize the earlier work.

### 2.1 SIGNAL Protocol

The SIGNAL protocol consists of three main parts: registration to the central server, establishment of the messaging session, and the actual message exchange (see Fig. 1).

When a user registers with their mobile number on the SIGNAL server, the key sets are generated in the background. The generated private key is stored at the mobile device, and never shared with any other party (including the central server). When the user sends or receives a message request, the corresponding public keys required to establish a mutual session is downloaded from the central server during the initial setup. Session keys remain same as long as the application is not removed from the device or sessions are not re-established. Hence, they generally last for a long time, e.g. weeks, months or years. Once the messaging session is established, identity keys of the communicating contacts are stored

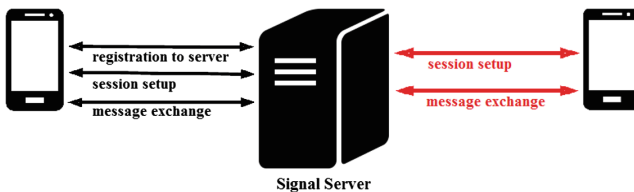


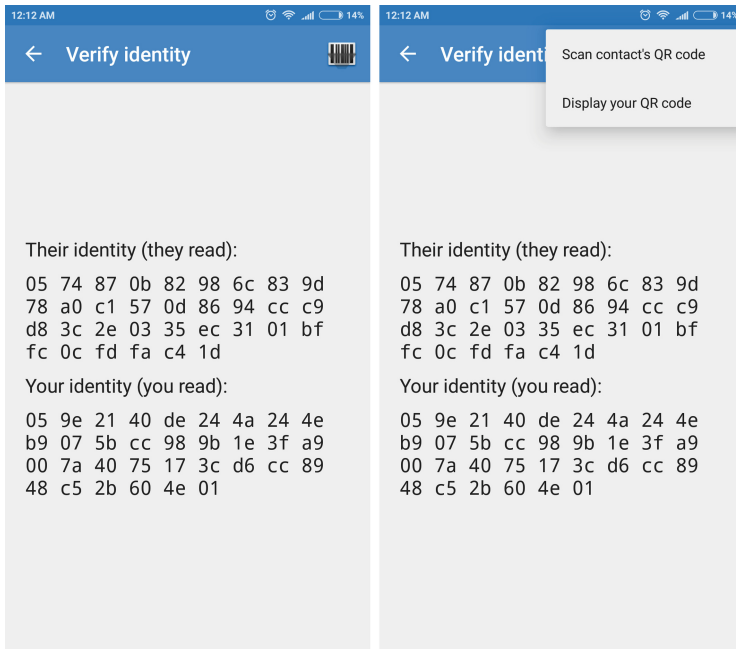
Fig. 1. A basic demonstration of how SIGNAL protocol works.

in the application. These keys can be accessed within the application. The public key verification process is simply determining whether these keys belong to the correct person or not. Normally, users are expected to perform verification upon session establishment or a genuine phone change of the communicating party.

The SIGNAL server is responsible for two tasks: relaying the messages to respective users and distributing public keys. Neither encryption nor decryption is performed on the server. When a user wants to establish a message session with another, (s)he requests their key set from the central server along with the identity key of that user. With the obtained key set, the user generates a symmetric key that is used to encrypt and decrypt messages. The same process is performed at the other side and the same symmetric key is generated. Afterwards, secure messaging takes place using this symmetric key. Both parties keep the identity key of their conversation partner.

### 2.2 Public Key Verification with Fingerprints

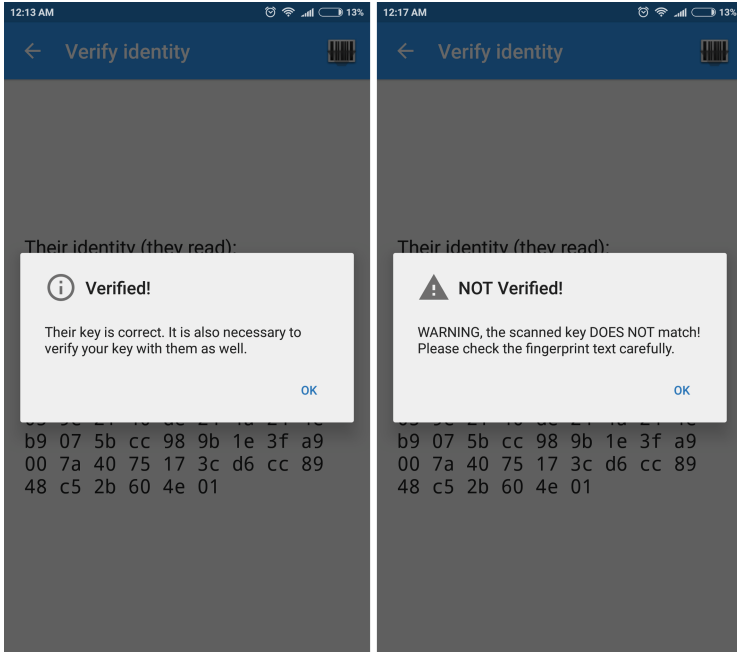
In the fingerprint method, each registered user has a fingerprint that is generated from her public key. This fingerprint is represented as a hexadecimal value consisting of 66 characters. For verification, the user must access the verification page first. This page consists of fingerprints of both users (see Fig. 2). Successful



**Fig. 2.** Fingerprints on the verification page with a barcode symbol on top right. The displayed menu when the barcode symbol is touched is shown on the right side.

verification requires both users to ensure her own fingerprint is identical to the fingerprint on the corresponding part of the conversation partner’s device.

It is also possible to perform the comparison via QR codes. For this purpose, one user must display their QR code via the “Display your QR code” button, while the partner must scan the displayed QR code via the “Scan contact’s QR code” button. Similar to manual verification, this process must be repeated for the other device. Examples for matched (left) and unmatched (right) QR code scan results are given in Fig. 3.



**Fig. 3.** QR scan results in the fingerprint method.

### 2.3 Public Key Verification with Safety Numbers

Unlike the fingerprint method, verification requires comparison of a single safety number that is derived from the public keys of the communicating parties in the safety number method. The safety number is specific to the session; that is, each session has its own different safety number. This safety number consists of 60 characters, represented as groups of 5-digit integers (see Fig. 4).

Upon reaching the verification page, users have two options to verify the safety number: they may either compare the 5-digit integers represented on their mobile devices by themselves, or they can use the QR scanning option similar to the previous method. Since the safety number is same in both devices, one verification is adequate. Examples for matched (left) and unmatched (right) QR code scan results are given in Fig. 5.

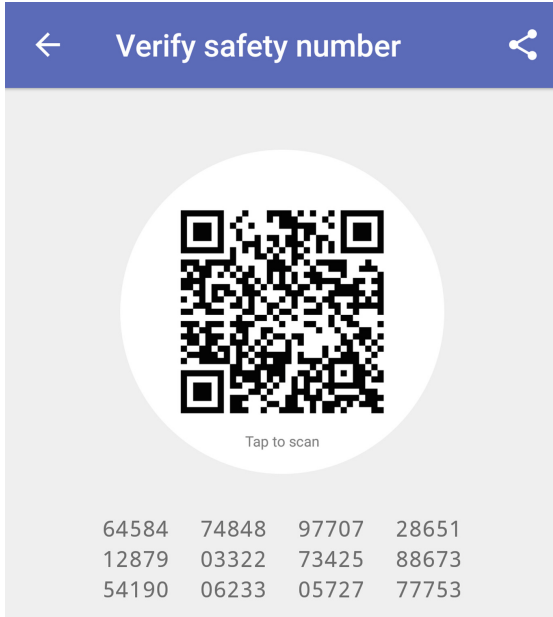


Fig. 4. Safety number and its QR code on the verification page.

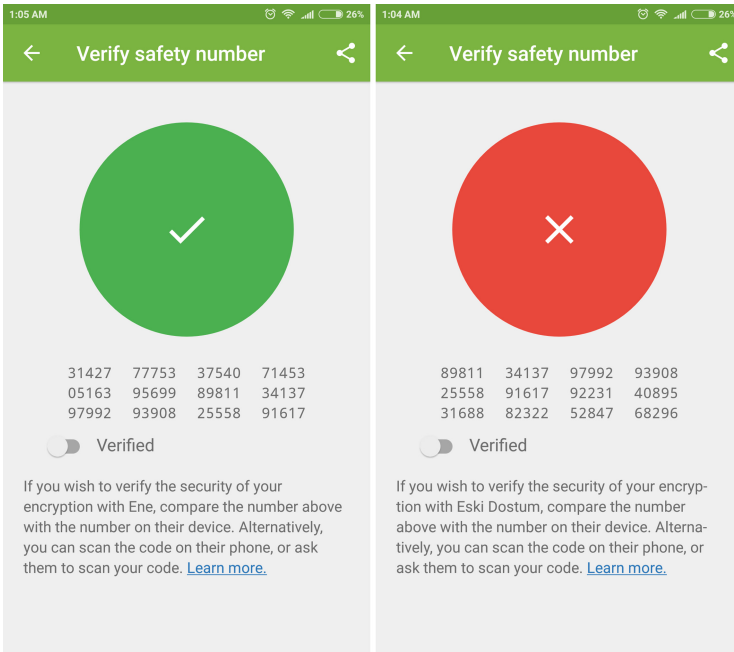


Fig. 5. QR scan results in the safety number method.

## 2.4 Threat Model

As mentioned before, secret session keys are generated and stored at the end devices (smartphones). The cryptographic algorithms and protocols are assumed to be secure hence it is impossible for a third party to generate the session key. However, the SIGNAL central database may be manipulated by a system administrator or by an external attacker, and hence, attacks against the session (either during the initial setup or against the already established session) may become possible.

Our work focuses on attacks against the established session. Specifically, we envision a scenario in which two users have already established a messaging session and then an attacker sends his key set to the users via the central database. In such a case, SIGNAL warns its users and suggests them to perform public key verification. We simulate such an attack scenario in our user study and investigate the effectiveness of implemented measures against this kind of threat.

## 2.5 Related Work

Public key verification is not a new problem, however expecting it to be performed successfully by novice smart phone users who only want to chat with their friends may be considered a more challenging problem. Having said that, the results of Whitten and Tygar’s user study measuring the usability of PGP to send and receive end-to-end encrypted e-mails showed that the problem is a difficult one even for the experienced users [9]. Follow-up studies has shown that email settings are especially problematic with respect to public key verification e.g., [3,4].

For secure messaging settings, the work by Schroder et al. presented a user study for the earlier fingerprint method of the SIGNAL messaging application for public key verification [7]. 21 of the 28 users who participated in this study failed to compare fingerprints.

In a recent work by Tan et al. the authors compared various public key verification interfaces [8]. 661 participants attended the large-scale user study which involve many different fingerprint representations. One of their conclusions is that all the representations and configurations they experimented with exhibited higher rates of successful attack than seems desirable for high-risk situations.

## 3 Methodology

In this section, we describe the methodology of our user study and the hypotheses set prior to the study.

### 3.1 Design

The user study was carried out at the Information Security Lab of TOBB University of Economics and Technology. A total of 42 users were initially split into



two similar groups of 21 users, and each group was tested with a different version of the SIGNAL application. Apart from the application version, which uses a different method for public key verification, all other factors were kept same for each group. Participants were specifically asked to think aloud during the study to understand their train of thought at every step. Participants were asked to answer some questions on their demographic status and their overall opinion about the study during and after the study.

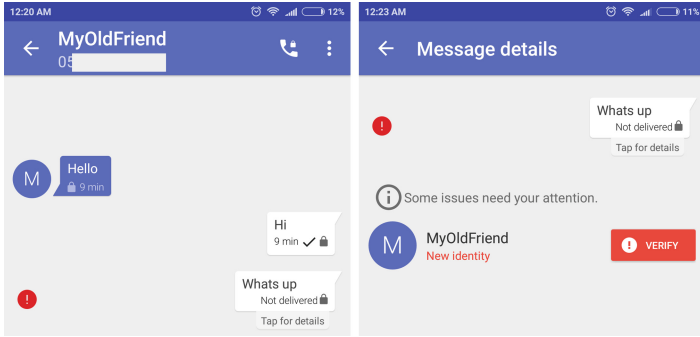
### 3.2 Scenario

At the beginning of the study, we briefly informed the participants about the study, and we gave them three pieces of information. The given information consists of postal address, bank account and credit card information. We asked them to send this information, one by one, to the operator through SIGNAL application that is preinstalled on a smartphone. We told participants to assume that given information were their personal information, and the operator was a close relative.

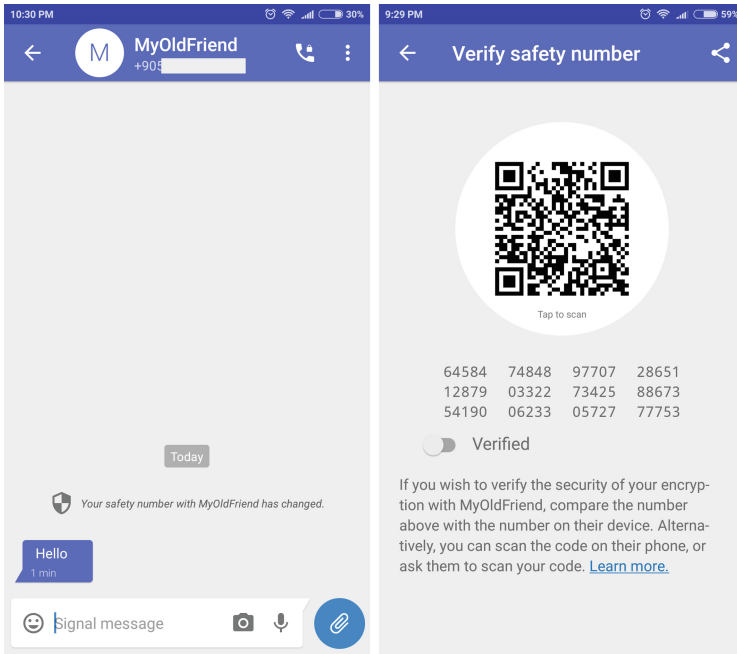
The rest of the scenario can be summarized as follows:

- i. Initially, at the preliminary stage, we asked the user to open the SIGNAL application and send the address part to the operator. At the end of this stage, a session has been established between two users (the participant and the operator).
- ii. Then, before moving onto the next stage, the user is asked to answer demographic questions. While the user was answering these questions, we moved the operator's sim card into another smartphone, and registered to the system with the same mobile phone number. Registering to the application from a different smartphone causes the change of public key for this mobile phone number, and hence, existing key set for the session will no longer match. This action basically simulates a man-in-the-middle attack, since a successful attack would also change the public key of the user. We note that the participants were kept busy with demographic questions and they did not see the actions taken by the operator.
- iii. After answering demographic questions, we asked the user to send her bank account number to the operator. Users in different groups were encountered with different types of error messages at this point. The users of the old version of the application (which uses the fingerprint method) were encountered with an error message with a red exclamation mark (see Fig. 6), while users of the new version of the application (which uses safety number method) were warned with a mere text (Fig. 7). Both versions of the application allow users to access the public key verification page with a single tap at the prompts. At the end of this stage, users who wanted to continue with the next step without entering the public key verification page were asked to verify the identity of their conversation partner within the application.

iv. Finally, we asked the user to send their credit card information to the operator over the SIGNAL application. Since the public key does not match, the correct behavior for a user would be to discontinue the study and not to send the sensitive information.



**Fig. 6.** Left: “Not Delivered” error. Right: Warning that the change of credentials (fingerprint method). (Color figure online)



**Fig. 7.** Left: Warning of safety number change. Right: Safety number verification page (safety number method).

Participants were not forced to send any information to the operator at any stage, and they were informed that they could communicate with the operator face to face whenever they needed. Apart from that, the operator sat at the remote corner of the laboratory and did not play an active role at any stage. Also, he did not inform the user about verification process. After the verification process, we asked the users to answer a short questionnaire about the study and their understandings of man-in-the-middle attack.

### 3.3 Study Environment

We used three smartphones (Android 6.0) and one computer in our study. We used the computer for testing the old version of SIGNAL (version 3.13.0) with two purposes: as the central database server, and also as a wireless access point providing Internet to smartphones. For testing the new version of SIGNAL, we used the current SIGNAL application (version 4.11.5) that was readily available on the app store. The user dealt with only one smartphone, while remaining two smartphones were used by the operator. All the devices were fully restored to their original state for each participant.

There were 42 users aged between 18 and 25 participated in this user study. All participants were TOBB University of Economics and Technology students who were enrolled to the “Introduction to Computer Science” course in Computer Engineering department, and they were rewarded with extra credits in the course for participating. Among participants, 18 (43%) of them were female and remaining 24 participants (57%) were male. Participants were split into two similar, equally sized groups. The study lasted about 15 to 25 min for each participant.

All participants stated that they were actively using WhatsApp (42) as instant messaging application, while SnapChat (28), Skype (22), Facebook Messenger (14), Telegram (5), Discord (2), Viber (1), Bip (1), Signal (1) were also used.

Finally, participants expressed their knowledge level on information security as follows: 31 of them (74%) chose none or very low, 7 of them (17%) chose medium, and 4 of them (9%) chose high. None of the 42 participants expressed themselves as an expert. Among all, 5 participants (12%) have expressed they had prior information about man-in-the-middle attacks.

### 3.4 Hypotheses

On our self-trials, we encountered a system behavior in the previous version of the SIGNAL application which did not allow sending new messages after the conversation partner’s public key changes until verification is completed; whereas, current version warns the user about a possible security problem, yet, allows communication. It has been shown with user studies that, active and obstructive actions, such as blocking communication, does not improve security [1]. Still we wanted to re-evaluate the effects of active (blocking communication)

and passive (allowing communication with a warning) actions on directing users to public key verification process in our study.

Since the main goal of the study is to compare the effectiveness of two methods for public key verification, fingerprint and safety number, we compare user success rates and time for comparison in each method. We expect to have higher success rates and lower comparison times for the safety number method, since comparing 12 blocks of 5-digit integers seems easier than comparing two blocks of 66 hexadecimal characters.

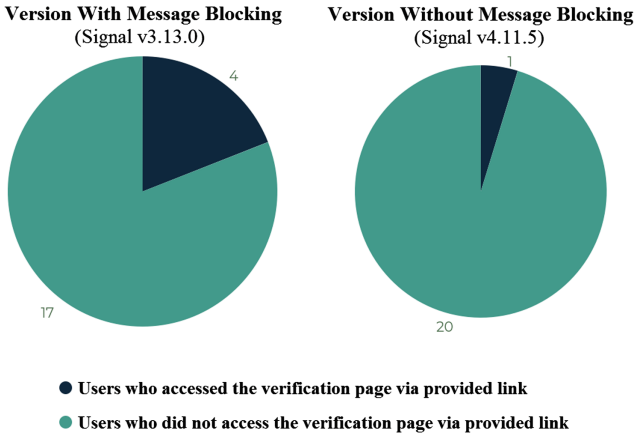
As a result, we specify our null hypotheses as follows:

- H1 In terms of directing users to the public key verification process, there is no difference between allowing or blocking sending new messages after conversation partner's public key changes.
- H2 There is no difference between the safety number and the fingerprint methods in terms of key verification success.
- H3 There is no difference between the safety number and the fingerprint methods in terms of public key comparison time.

## 4 Results and Analysis

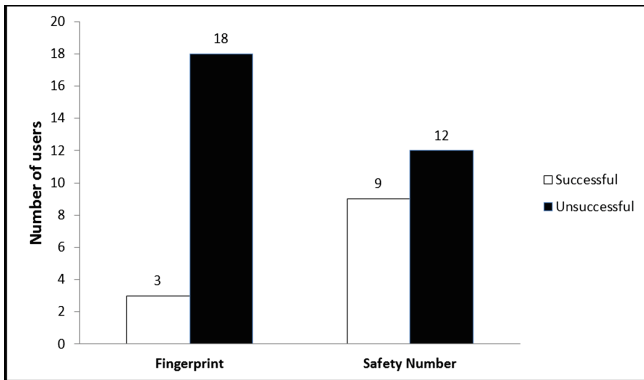
As described in Subsect. 3.2, users were initially asked to send the provided address information to the operator. All participants completed this preliminary task successfully. After sending the address, we asked the participants to answer some demographic questions. As they were answering the questions, we transported the sim card of the operator into another smartphone. This action changed the operator's public keys for simulating the man-in-the-middle attack. Next, users were asked to send their bank account information to the operator. Different versions of the SIGNAL application behaved differently at this point. Older version, which uses the fingerprint method for public key verification, disallows its users to send new messages until public key verification is complete; on the other hand, the current version that uses the safety number method warns the user with a passive message. We observed participants in both groups, and noticed that 4 out of 21 participants (19%) accessed the verification page through the provided link in the older version (v3.13.0), whereas only 1 participant accessed it using the provided link in the current version (v4.11.5) (see Fig. 8). The chi square test for independence suggests no significant difference in effects of employing either a blocking or non-blocking warning message on security ( $\chi^2 = 2.04, p = 0.15$ ), resulting in insufficient evidence to reject the null hypothesis H1.

Participants, who did not access the public key verification page by themselves were directed to it, and the study continued with final phase, where they were asked to send their credit card information. Since the public key does not match, we expected the users to end the communication without sending the credit card number. As a result, we define a successful verification as the combination of realizing the mismatch and not sending the credit card information. We believe that, this definition is reasonable because participants stated that



**Fig. 8.** Performances of the SIGNAL applications used in the study, on directing users to the verification page.

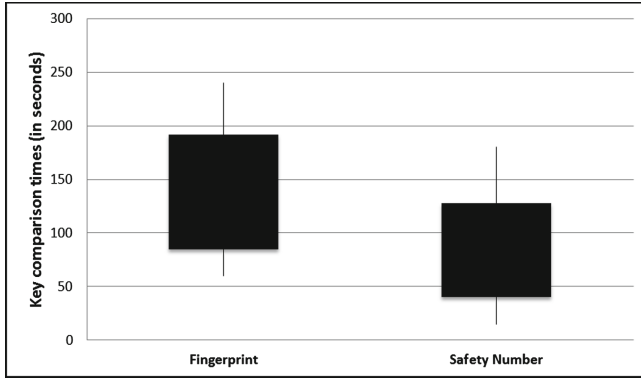
they all use the WhatsApp application in their daily life, and WhatsApp implements the SIGNAL protocol. They might have to deal with such situation by themselves in a real-life scenario. According to this metric, 9 out of 21 participants (43%) were considered successful in the public key verification via safety number method; whereas only 3 of the 21 participants (14%) were successful in the public key verification via fingerprint method (see Fig. 9). The chi square test for independence suggests marginally significant difference in choice of public key verification methods ( $\chi^2 = 4.2, p = 0.04$ ). As a result, we reject the null hypothesis H2 in favor of the safety number method.



**Fig. 9.** Verification success rates of users.

Finally, average public key verification times for the safety number method and the fingerprint method are 82s and 130s, respectively (Fig. 10).

According to the Mann-Whitney-U test, a between-group median comparison test for not normally distributed data, the difference is marginally significant ( $W = 58, p = 0.04$ ), and hence, we reject the null hypothesis H3 favoring the safety number method.



**Fig. 10.** Average public key comparison times of users.

To sum up, there was not enough evidence to reject the null hypothesis H1: a blocking warning message does not increase security as compared to non-blocking warning message. On the other hand, hypotheses H2 and H3 were rejected favoring the safety number method: the safety number method leads to more successful verification in less amount of time, when compared to the fingerprint method. Lastly, we acknowledge that involving more participants that represent the general user profile better could help us draw more significant results.

## 5 Conclusion

End-to-end encryption has become a necessity for secure communication. Smartphone communication applications provide high availability and mobility to its users; hence, their market share increase day by day. On the other hand, they must compete each other to satisfy their users' security demands.

WhatsApp is a popular communication application with more than a billion users, and it uses the SIGNAL protocol for end-to-end encryption. The SIGNAL protocol has recently changed an important feature: its public key verification method. In its older versions, it used the fingerprint method for public key verification, whereas, the safety number method for public key verification is now utilized in its current version.

In this study, we presented a user study which evaluates and compares the security and usability aspects of the fingerprint and the safety number methods

used in older and current versions of the SIGNAL messaging application, respectively. The results of our user study indicate that users achieve more success and spend less time with the safety number method as compared to the fingerprint method for public key verification. Although our results indicate a significant improvement with the new safety number method, we argue that the obtained results does not reflect yet that the problem has been solved i.e., still majority of users could not successfully perform public key verification even with the safety number method before transmitting a sensitive message. Hence, we urge usable security researchers to continue working on the public key verification problem since there is still an obvious room for more improvement.

## References

1. Bicakci, K., Atalay, N.B., Kiziloz, H.E.: Johnny in internet café: user study and exploration of password autocomplete in web browsers. In: Proceedings of the 7th ACM Workshop on Digital Identity Management, pp. 33–42. ACM (2011)
2. Budington, B.: Whatsapp rolls out end-to-end encryption to its over one billion users (2016). <https://www.eff.org/deeplinks/2016/04/whatsapp-rolls-out-end-end-encryption-its-1bn-users>. Accessed 20 Apr 2018
3. Fry, A., Chiasson, S., Somayaji, A.: Not sealed but delivered: the (un) usability of s/mime today. In: Annual Symposium on Information Assurance and Secure Knowledge Management (ASIA 2012), Albany, NY (2012)
4. Garfinkel, S.L., Margrave, D., Schiller, J.I., Nordlander, E., Miller, R.C.: How to make secure email easier to use. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 701–710. ACM (2005)
5. Marlinspike, M.: Safety number updates (2016). <https://signal.org/blog/safety-number-updates/>. Accessed 20 Apr 2018
6. Renaud, K., Volkamer, M., Renkema-Padmos, A.: Why doesn't Jane protect her privacy? In: De Cristofaro, E., Murdoch, S.J. (eds.) PETS 2014. LNCS, vol. 8555, pp. 244–262. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-08506-7\\_13](https://doi.org/10.1007/978-3-319-08506-7_13)
7. Schröder, S., Huber, M., Wind, D., Rottermann, C.: When signal hits the fan: on the usability and security of state-of-the-art secure mobile messaging. In: First European Workshop on Usable Security (EuroUSEC 2016) (2016)
8. Tan, J., Bauer, L., Bonneau, J., Cranor, L.F., Thomas, J., Ur, B.: Can unicorns help users compare crypto key fingerprints? In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 3787–3798. ACM (2017)
9. Whitten, A., Tygar, J.D.: Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In: USENIX Security Symposium, vol. 348 (1999)

# **Symmetric Ciphers and Cryptanalysis**





# Speeding up MILP Aided Differential Characteristic Search with Matsui's Strategy

Yingjie Zhang<sup>1,2,3</sup>, Siwei Sun<sup>1,2,3(✉)</sup>, Jiahao Cai<sup>1,2,3</sup>, and Lei Hu<sup>1,2,3</sup>

<sup>1</sup> State Key Laboratory of Information Security,  
Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

{zhangyingjie,sunsiwei,caijiahao,hulei}@iie.ac.cn

<sup>2</sup> Data Assurance and Communication Security Research Center,  
Chinese Academy of Sciences, Beijing, China

<sup>3</sup> School of Cyber Security,  
University of Chinese Academy of Sciences, Beijing, China

**Abstract.** Being the first generic algorithm for finding the best differential and linear characteristics, Matsui's branch and bound search algorithm (EUROCRYPT 1994) and its variants have played an important role in the security analysis of symmetric-key primitives. However, Matsui's algorithm is difficult to implement, optimize, and be applied to different ciphers with reusable code. Another approach getting popular in recent years is to encode the search problem as a Mixed Integer Linear Programming (MILP) model which can be solved by open-source or commercially available optimizers. In this work, we show how to tweak the objective functions of the MILP models for finding differential characteristics such that a set of constraints derived from the bounding condition of Matsui's algorithm can be incorporated into the models. We apply the new modeling strategy to PRESENT (S-box based SPN design), SIMON (Feistel structure), and SPECK (ARX construction). For PRESENT, the resolution time is significantly reduced. For example, the time to prove that the exact lower bound of the probabilities of the differential characteristics for 7-round PRESENT is reduced from 48638 s to 656 s. For SIMON, obvious acceleration is also observed, and for the ARX cipher SPECK, the new model is unable to speed up the resolution. In the future, it is interesting to investigate how to integrate other search heuristics proposed in the literature of symmetric-key cryptanalysis into the MILP models, and how to accelerate the resolution of MILP models for finding characteristics of ARX ciphers.

**Keywords:** Differential characteristic · Linear characteristic  
Branch and bound · Matsui's algorithm · MILP

## 1 Introduction

Differential and linear attacks are two of the most fundamental methods for analyzing symmetric-key primitives, which many advanced cryptanalytic techniques

are derived from or partly rely on. Performing differential and linear analysis is a tedious routine work for the designers and cryptanalysts of symmetric-key algorithms. Matsui's branch and bound search algorithm [21] is a classical approach for finding the best differential and linear characteristics, and it is extremely efficient for some specific ciphers. But, implementing Matsui's algorithm properly demands for sophisticated programming skills when cipher-specific optimizations are taken into account [5, 6, 12]. Moreover, there seems to be no obvious way to create highly reusable code for Matsui's algorithm targeting different ciphers.

However, the diversity of cryptographic algorithms is an unstoppable trend. In the case of block ciphers, to have a single algorithm work as a security solution for all scenarios is doomed to fail due to the ever-increasing complexity and diversity of today's communication systems. Over recent years, we have witnessed many new block ciphers designed for lightweight devices or dedicated use cases. These include, to just name a few of them, the ISO standard PRESENT [11], SIMON and SPECK [7] designed by the NSA, the SKINNY family presented in CRYPTO 2016 [8], and Rasta with minimizing AND-related metrics as its main design objective [15]. We refer the reader to [9] for a more comprehensive survey. To meet the requirements of the target applications, these newly designed block ciphers typically use lightweight components with relatively weak local cryptographic properties, consume less resources when implemented and executed, and reserve limited security margins aggressively. This approach makes the design and evaluation more difficult, where the security bounds cannot be derived theoretically.

In such situation, the security evaluation against differential and linear attacks have to be performed with the help of search tools. Matsui's algorithm is obviously not a satisfactory choice not only because of its inconvenience but also that it is unable to get useful results in some cases. Another option getting more and more popular in recent years is the Mixed Integer Linear Programming (MILP) based method, where the problem of searching for characteristics is transformed into an MILP model that can be solved with generic MILP solvers.

Similar to SAT/SMT and CP based methods [4, 19, 22, 26], in the MILP based approach [23, 28, 29], the cryptanalysts only need to specify the problem in standard modeling languages without mixing in the actual search algorithms. This *decoupling of formulation and resolution* is the key that makes the MILP based approach more attractive than Matsui's algorithm. Unlike Matsui's algorithm, searching heuristics and optimizations can be issued *externally* without touching the sophisticated code powering the search. In addition, cryptanalysts benefit directly from the advancement of MILP resolution techniques. So far, the MILP based approach covers many cryptanalytic techniques, including differential/linear [18, 28], impossible differential [25], zero-correlation linear [14], and integral cryptanalysis [30].

Despite all these advantages, there are situations where Matsui's algorithm performs far more better than the MILP based approach (*e.g.*, search for the best characteristics of DES and PRESENT in the single-key model). Moreover, both MILP and Matsui's algorithm rarely work for non-lightweight designs under

today’s computational power. Therefore, it is of great importance to improve the efficiency of the MILP based approach, and a natural question to ask is whether it is possible to strengthen the MILP based search with Matsui’s algorithm. In this work, we make a first step towards this direction. Finally, before we present our work, we emphasize that all of our analysis are based on the *Markov assumption* [20], where we assume that each round of an iterative cipher is independent.

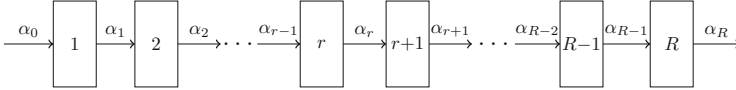
**Motivation and Contribution.** One obvious difference between Matsui’s algorithm and the MILP based approach is worth to highlight. When we search for the best characteristic of an  $R$ -round iterative block cipher, Matsui’s algorithm requires the probabilities of the optimal characteristics of the same cipher reduced to  $r$  rounds for  $1 \leq r < R$ . That is, to get the result of  $R$  rounds, we must first run Matsui’s algorithm for rounds 1, 2,  $\dots$ , and  $R - 1$ . These probabilities are employed to prune the search tree according to certain bounding conditions. In contrast, in the MILP based approach, we always set up an  $R$ -round model directly, and do not exploit the solutions for lower rounds explicitly. This fact motivates us to enhance the  $R$ -round MILP models by taking into account some information of the solutions of lower rounds. We achieve this by adapting the objective function of an  $R$ -round model such that constraints encoding Matsui’s bounding conditions can be incorporated into the model. In practice, this new modeling strategy leaves many choices for the cryptanalysts, since one can choose to include only a subset of the constraints generated from Matsui’s bounding conditions. We perform experiments on PRESENT, SIMON, and SPECK, which shows that the inclusion of the constraints derived from Matsui’s algorithm leads to significantly improved resolution performance for PRESENT. For SIMON, obvious improvement is also observed, and for the ARX cipher SPECK, the new model is unable to accelerate the resolution performance. Our work suggests that trying to combine the power of dedicated search algorithms implemented in general purpose programming language and MILP is a valuable endeavor. In the future, it is interesting to see how to integrate other search heuristics [16,17] to speed up the resolution of the MILP models for finding characteristics of ARX ciphers.

**Organization.** In Sects. 2 and 3, we give a brief introduction of Matsui’s algorithm and the MILP based differential and linear analysis. A method for enhancing the MILP models with constraints generated from Matsui’s bounding condition is presented in Sect. 4. We then show applications of the enhanced MILP models in Sect. 5. Section 6 concludes the paper and suggests future work.

## 2 Matsui’s Algorithm

At Eurocrypt 1994, Matsui presented a branch and bound search algorithm that can be used to identify the maximum probability characteristic of a target block cipher [21]. Matsui’s algorithm, together with its variations, has been an important tool in the practice of security evaluation of symmetric-key primitives. It is improved in subsequent work [3,6,13,24] and adapted to ARX constructions in [10,31].

A general description of Matsui’s algorithm for an iterative block cipher depicted in Fig. 1 is given in Algorithm 1. Our presentation largely follows the work of Banner *et al.* [5]. Also note that Algorithm 1 is an over simplification of Matsui’s algorithm, which does not exhibit the necessary details (*e.g.*, the technique for controlling the number of initial branches, the order in which candidates are enumerated) in actual implementations.



**Fig. 1.** An  $R$ -round iterative cipher, where  $\mathcal{T} = (\alpha_0, \dots, \alpha_R)$  is an  $R$ -round differential characteristic with probability  $\mathbb{P}(\mathcal{T})$ , and the probability of the differential  $\alpha_{r-1} \rightarrow \alpha_r$  is denoted by  $P_{Rd(r)}$ .

With the knowledge of the best probabilities  $P_{Best}(i)$  of  $i$ -round characteristics for  $i \in \{1, \dots, R - 1\}$ , Matsui’s algorithm explores the search space of all possible characteristics in a depth-first approach, and output the optimal  $R$ -round characteristic. The search space conceptually forms a tree structure, and at the  $r$ th level of the tree,  $\mathcal{T}_{[1,r]} = (\alpha_0, \dots, \alpha_r)$  is assigned to actual values by Matsui’s algorithm, and all possible values of  $(\alpha_{r+1}, \dots, \alpha_{R+1})$  form a subtree to be explored. We call  $\mathcal{T}_{[1,r]}$  with  $r < R$  instantiated with actual values a *partial solution* (corresponding to intermediate node of the search tree), and  $\mathcal{T} = \mathcal{T}_{[1,R]}$  instantiated with actual values a *full solution* (corresponding to a leaf node of the search tree). Thus, when Matsui’s algorithm goes one level deeper into the search tree, it extends the current partial solution towards a full solution.

The efficiency of Matsui’s algorithm comes from the fact that it will not try to extend every partial solution. Before trying to extend the current partial solution, the so-called bounding condition specified in line 24 of Algorithm 1 is tested, which essentially states that if this condition is violated, a better characteristic will never be found by extending the current partial solution, and therefore we should give up the current branch, backtrack to the upper level of the search tree, and try another branch.

The variable  $P_{Estim}$  in Matsui’s algorithm keeps track of the best characteristic known so far. Only when a strictly better characteristic is encountered during the search, it will be updated (see line 42 of Algorithm 1).

Moreover, in Matsui’s algorithm, The first and last rounds receive special treatment (see functions `FirstRound()` and `LastRound()` in Algorithm 1), where the input and output difference is determined directly by the output differences of the round 1 and round  $R - 1$ , without the effort of searching through a set of candidates.

---

**Algorithm 1.** Matsui's Algorithm

---

**Input:**  $R \in \mathbb{Z}^*$ ,  $R \geq 2$ ;  $q > 0$ ;  $P_{Best}(1), P_{Best}(2), \dots, P_{Best}(R-1)$   
**Output:** differential characteristic  $\mathcal{T} = (\alpha_0, \alpha_1, \dots, \alpha_{R-1}) \in \mathbb{F}_2^n$  where probability  $\mathbb{P}(\mathcal{T}) = P_{Estim}$

```

1 Algorithm OptimalTrail( $R, q, P_{Best}(1), \dots, P_{Best}(R-1)$ ) // Entry Point
2   for each non-zero  $\alpha_1$  do
3     |  $\mathcal{T} = ()$ ,  $P_{Estim} \leftarrow q$ 
4     | Call FirstRound()
5   end
6   if  $\mathcal{T} \neq ()$  then
7     | return  $\mathcal{T}$ ,  $P_{Estim} = \mathbb{P}(\mathcal{T})$ 
8   end
9 end
10
11 Function FirstRound() // Subroutine
12   |  $P_{Rd(1)} \leftarrow \max_{\alpha} \mathbb{P}(\alpha \rightarrow \alpha_1)$ 
13   |  $\alpha_0 \leftarrow \alpha$ , s.t  $\mathbb{P}(\alpha \rightarrow \alpha_1) = P_{Rd(1)}$ 
14   | if  $R > 2$  then
15     | | Call Round(2)
16   | else
17     | | Call LastRound()
18   | end
19 end
20
21 Function Round( $r$ )( $2 \leq r \leq R-1$ ) // Subroutine
22   | for each candidate  $\alpha$  for  $\alpha_{r-1}$  do
23     |  $P_{Rd(r)} \leftarrow \mathbb{P}(\alpha_{r-1} \rightarrow \alpha)$ 
24     | if  $\prod_{i=1}^r P_{Rd(i)} \cdot P_{Best}(R-r) \geq P_{Estim}$  then
25       | // Matsui's bounding condition
26       | |  $\alpha_r \leftarrow \alpha$ 
27       | | if  $r+1 < R$  then
28       | | | Call Round( $r+1$ )
29       | | else
30       | | | Call LastRound()
31       | | end
32     | end
33   | end
34 end
35 end
36
37 Function LastRound() // Subroutine
38   | for each candidate  $\alpha$  for  $\alpha_{r-1}$  do
39     |  $P_{Rd(R)} \leftarrow \max_{\alpha} \mathbb{P}(\alpha_{R-1} \rightarrow \alpha)$ 
40     |  $\alpha_R \leftarrow \alpha$ , s.t  $\mathbb{P}(\alpha_{R-1} \rightarrow \alpha) = P_{Rd(R)}$ 
41   | end
42   | if  $\prod_{i=1}^R P_{Rd(i)} > P_{Estim}$  then // A strictly better trail is found
43     | |  $\mathcal{T} \leftarrow (\alpha_0, \alpha_1, \dots, \alpha_{R-1})$ 
44     | |  $P_{Estim} \leftarrow \prod_{i=1}^R P_{Rd(i)}$ 
45   | end
46 end

```

---

### 3 MILP Aided Characteristic Search

At first, MILP was used to determine the minimum number of differentially or linearly active S-boxes of word-oriented ciphers [23, 29]. In [28], Sun *et al.* introduced the convex hull computation method which can encode any subset of 0–1 vectors as the solution set of a system of linear inequalities. Thanks to this technique, actual differential and linear characteristics can be found with MILP based method. Subsequently, the MILP aided approach is applied in impossible differential analysis [25], zero-correlation linear analysis [14], and Integral cryptanalysis [30]. It is also extended and adapted to analyze ARX based constructions [18]. In what follows, we give a brief introduction of the MILP modeling technique for finding differential characteristics, which is employed in the following sections.

The key to transfer the problem of searching for differential characteristics into an MILP model is to express the propagation rules of the characteristics as a set of linear inequalities, and encode the overall probability as a linear function.

**Objective Function.** Since the goal is to find the optimal characteristic, we set the objective function to minimize the probability of the underlying differential characteristic. However, we must be able to express the probability as a linear function at the first place to make it valid in MILP. Such representations are available for SIMON, SPECK, and PRESENT [18, 28]. For the sake of simplicity and without loss of generality, we assume the probability (or its equivalence) can be represented by

$$\sum_{i=1}^R \sum_{j=1}^k A_{i,j},$$

and we call  $A_{i,j}$ 's are probability weight variables, where  $A_{i,j}$  for  $j \in \{1, \dots, k\}$  is the probability weight variables of round  $i$  of an iterative cipher. Under this notation, the probability weight contributed by round  $i$  is  $\sum_{j=1}^k A_{i,j}$ .

**Modeling XOR.** Let  $a \oplus b = c$ , where  $a, b, c \in \mathbb{F}_2$  are the bit-level input and output differences of the XOR operation. Then  $(a, b, c)$  is a valid differential characteristic of XOR if and only if  $a + b + c - 2d_{\oplus} = 0$ , where  $a, b$ , and  $c \in \{0, 1\}$ , and  $d_{\oplus}$  is a 0–1 dummy variable.

**Modeling S-box.** The exact differential property of an  $\omega \times \nu$  S-box  $S$  can be modeled by a set of linear inequalities with the convex hull computation method [28]. Let  $\mathcal{D} = \{(\mathbf{a}, \mathbf{b}) \in \{0, 1\}^{\omega+\nu} : P(\mathbf{a} \rightarrow \mathbf{b}) > 0\}$  be the set of all possible input-output differential patterns of  $S$ , where  $\mathbf{a} = (a_0, a_1, \dots, a_{\omega-1})$  and  $\mathbf{b} = (b_0, b_1, \dots, b_{\nu-1})$ . Then, we can compute the H-representation of  $\mathcal{D} \subseteq \mathbb{R}^{\omega+\nu}$ . With the help of the greedy algorithm proposed in [28], we can extract a system of inequalities whose 0–1 solution set is exactly  $\mathcal{D}$ . Sometimes, it is possible to encode the differential probabilities of  $\mathbf{a} \rightarrow \mathbf{b}$  into  $\mathcal{D}$ , and we refer the reader to [18, 27, 28] for concrete examples.

**Modeling Modular Addition [18].** Suppose  $\mathbf{a} = (a_0, a_1, \dots, a_{n-1})$ ,  $\mathbf{b} = (b_0, b_1, \dots, b_{n-1})$  and  $\mathbf{c} = (c_0, c_1, \dots, c_{n-1})$  are the input and output bit-level

XOR-difference of addition module  $2^n$ . The constraints are as follows, where  $d_{\oplus}$  is 0–1 dummy variable,  $s_i (i = 1, \dots, n - 2)$  are 0–1 active markers and  $\sum_{i=1}^{n-2} s_i$  is negative logarithm of the probability  $P[(\mathbf{a}, \mathbf{b}) \rightarrow \mathbf{c}]$ .

$$\left\{ \begin{array}{l} a_{n-1} + b_{n-1} + c_{n-1} \leq 2 \\ a_{n-1} + b_{n-1} + c_{n-1} - 2d_{\oplus} \geq 0 \\ d_{\oplus} - a_{n-1} \geq 0 \\ d_{\oplus} - b_{n-1} \geq 0 \\ d_{\oplus} - c_{n-1} \geq 0 \\ -a_i + b_i + s_i \geq 0 \\ -b_i + c_i + s_i \geq 0 \\ a_i - c_i + s_i \geq 0 \\ a_i + b_i + c_i - s_i \geq 0 \\ -a_i - b_i - c_i - s_i \geq -3 \\ c_i + a_{i-1} + b_{i-1} - c_{i-1} + s_i \geq 0 \\ -a_i - b_i - c_i + 3a_{i-1} + 3b_{i-1} + 3c_{i-1} + 2s_i \geq 0 \\ a_i + b_i + c_i - 3a_{i-1} - 3b_{i-1} - 3c_{i-1} + 2s_i \geq -6 \\ -b_i + a_{i-1} - b_{i-1} - c_{i-1} + s_i \geq -2 \\ c_i + a_{i-1} - b_{i-1} + c_{i-1} + s_i \geq 0 \\ -a_i - b_i - c_i - 3a_{i-1} + 3b_{i-1} - 3c_{i-1} + 2s_i \geq -6 \\ -a_i - a_{i-1} - b_{i-1} + c_{i-1} + s_i \geq -2 \\ a_i + b_i + c_i - 3a_{i-1} + 3b_{i-1} + 3c_{i-1} + 2s_i \geq 0 \\ (i = 1, \dots, n - 2) \end{array} \right. \quad (1)$$

## 4 Enhancing MILP Based Search with Matsui's Bounding Condition

Firstly, let us recall the bounding condition of Matsui's algorithm (see Algorithm 1):

$$\prod_{i=1}^r P_{Rd(i)} \cdot P_{Best}(R - r) \geq P_{Estim}. \quad (2)$$

When we run Matsui's algorithm against an  $R$ -round cipher, the variable  $P_{Estim}$  keeps track of the probability of the best characteristic known by the algorithm so far, and it will be updated dynamically if a strictly better characteristic is encountered during the search. Whenever the algorithm needs to go one level deeper into the search tree, condition (2) is tested. A violation of (2) implies that any extension of the partial solution leads to inferior characteristics with probability less than  $P_{Estim}$  (the probability of a known characteristic). Therefore, the entire subtree is pruned.

To integrate Matsui's bounding condition into the MILP models, we introduce a variable named  $xobj$  acting as the variable  $P_{Estim}$  in Matsui's algorithm, and let

$$\text{Minimize } xobj$$

be the objective function of the new model. Note that this is a very natural choice since the variable  $xobj$  always keeps track of the currently known best

solution during the resolution of the MILP model. To make the  $xobj$  correspond to the probability of the identified characteristic, we put an equation

$$xobj = \sum_{i=1}^R \sum_{j=1}^k A_{i,j}$$

into the constraints section of the model. At this point, the new model is completely equivalent to the original model. What we do is essentially renaming the objective function of the original model.

Assuming we know the probabilities  $P_{Best}(1), P_{Best}(2), \dots, P_{Best}(R-1)$ , we are now ready to express the bounding condition (2) as

$$\sum_{t=1}^i \sum_{j=1}^k A_{t,j} + wt(P_{Best}(R-i)) \leq xobj, \quad i = 1, \dots, R-1 \quad (3)$$

$$\sum_{t=i+1}^R \sum_{j=1}^k A_{t,j} + wt(P_{Best}(i)) \leq xobj, \quad i = 1, \dots, R-1 \quad (4)$$

Therefore, for an  $R$ -round model, we can generate  $2R-2$  more constraints, where  $wt(\cdot)$  make  $P_{best}(i)$  compatible with the probability weight variables. The most different part of the new model is that it takes into account the solutions of the models of lower rounds. In the following, we present three different modeling strategies, which will be compared in the next section.

- $\mathcal{M}^I$ : The original model without any modification.
- $\mathcal{M}^{II}$ : The model with modified objective function, and  $R-1$  additional constraints of (4) generated from Matsui's bounding condition for round 1 to round  $R-1$  respectively.

$$\begin{cases} \min xobj \\ \sum_{i,j} A_{i,j} - xobj = 0 \\ \sum_{t=i+1}^R \sum_{j=1}^k A_{t,j} + wt(P_{Best}(i)) \leq xobj, \quad i = 1, \dots, R-1 \end{cases} \quad (5)$$

- $\mathcal{M}^{III}$ : The model with modified objective function, and all  $2R-2$  additional constraints.

$$\begin{cases} \min xobj \\ \sum_{i,j} A_{i,j} - xobj = 0 \\ \sum_{t=i+1}^R \sum_{j=1}^k A_{t,j} + wt(P_{Best}(i)) \leq xobj, \quad i = 1, \dots, R-1 \\ \sum_{t=1}^i \sum_{j=1}^k A_{t,j} + wt(P_{Best}(R-i)) \leq xobj, \quad i = 1, \dots, R-1 \end{cases} \quad (6)$$



## 5 Applications

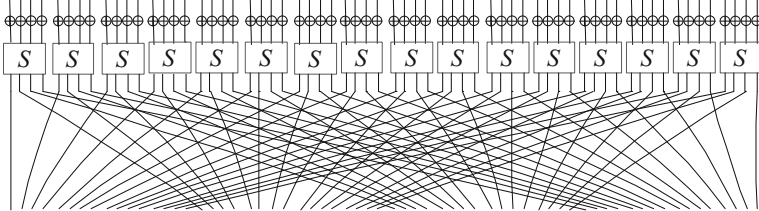
In this section, we apply the modeling strategy presented in Sect. 4 to PRESENT, SIMON, and SPECK. The reasons that these ciphers are selected as the experimental targets are twofold. Firstly, the probabilities (or their equivalences) of the differential characteristics of these ciphers can be expressed as linear functions. Secondly, they represent the most common structures for modern block ciphers, where PRESENT is a SPN network, SIMON is a Feistel cipher with pure bitwise operations, and SPECK is an ARX construction.

However, we admit that in our experiments only lightweight primitives are involved. This is because generally MILP based approach (and actually all currently available automatic search tools) is too inefficient to search for characteristics of non-lightweight ciphers directly, and it is sometimes difficult to modeling the components of non-lightweight ciphers at the first place. For example, only recently, Abdelkhalek *et al.* show how to model the differential property of an  $8 \times 8$  S-box with MILP [2], and even that, the search procedure has to be divided into two steps for a cipher involving  $8 \times 8$  S-boxes, where only truncated differentials are identified in the first step.

In addition, since the focus of this paper is to improve the MILP based method, we will not give a comparison between Matsui’s algorithm and the MILP based approach. Nevertheless, we would like to mention that Matsui’s algorithm is much more better than MILP in the case of PRESENT, while for SIMON and SPECK, it is inferior to MILP. Finally, all of the models presented in this paper are solved by the MILP optimizer Gurobi (version 7.0.2) [1] running at 16 threads on a server with Intel® Xeon® E5-2637V3 CPU 3.50 GHz.

### 5.1 Application to PRESENT

The PRESENT, designed by Bogdanov *et al.*, is an ISO standardized lightweight block cipher [11]. The round function of PRESENT is shown in Fig. 2, and we refer the reader to [11] for more information.



**Fig. 2.** The round function of PRESENT

We construct three models  $\mathcal{M}^I$ ,  $\mathcal{M}^{II}$ , and  $\mathcal{M}^{III}$  according to the strategies presented in Sect. 4. The resolution time for these models are recorded in Table 1.

Note that what we measure is the time cost for the solver to prove that the solution it identified is *optimal*. This timing information is of most importance since in the design process what we care is the *bound*, and the tighter the bound is, the more accurate the security evaluation.

**Table 1.** Experimental results of PRESENT

$R$	$p$	$\mathcal{M}^I$	$\mathcal{M}^{II}$	$\mathcal{M}^{III}$
1	$2^{-2}$	0.01s	0.09s	0.13s
2	$2^{-4}$	0.95s	0.95s	0.06s
3	$2^{-8}$	3.70s	2.82s	2.43s
4	$2^{-12}$	15.78s	10.08s	8.82s
5	$2^{-20}$	629.83s	114.13s	448.61s
6	$2^{-24}$	1740.55s	200.03s	74.56s
7	$2^{-28}$	48638.29s	714.03s	655.36s
8	$2^{-32}$	>10h	2124.51s	1074.45s

From Table 1 we can see that the resolution time can be significantly improved by using the new modeling strategies. For instance, we can prove that the probability of the optimal characteristic of 8-round PRESENT is  $2^{-32}$  in 1074.45 s by using  $\mathcal{M}^{III}$ , while for  $\mathcal{M}^I$  we can not get this result in less 10 h. Moreover, by using the new models, some interesting phenomenons are observed that we cannot explain. For example, the resolution time of  $\mathcal{M}^{III}$  for 6-round PRESENT is faster than that of the 5-round model.

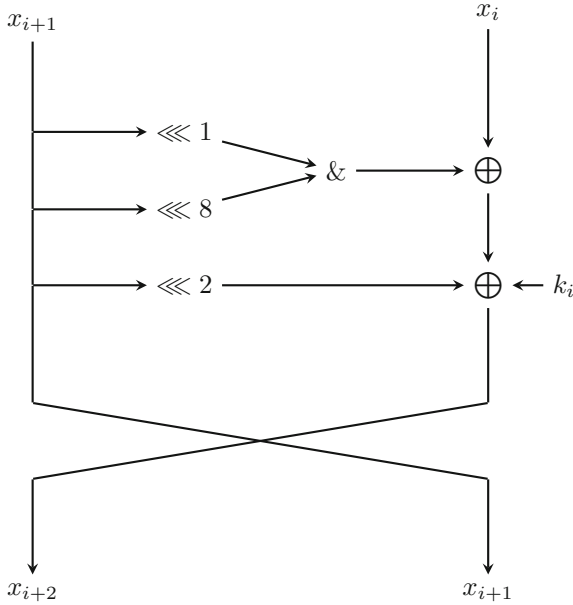
## 5.2 Application to SIMON

SIMON (depicted in Fig. 3) is a family of lightweight block ciphers with Feistel structure involving only bitwise operations: XOR, AND, and Rotation, which is designed by the National Security Agency of USA. The parameters of different SIMON instances involved in our experiments are summarized in Table 2.

**Table 2.** Parameters for SIMON32 and SIMON48

Variant $2n/mn$	Block Size $2n$	Key Size $mn$	Round $r$
32/64	32	64	32
48/72	48	72	36
48/96	48	96	36

We construct three models  $\mathcal{M}^I$ ,  $\mathcal{M}^{II}$ , and  $\mathcal{M}^{III}$  according to the strategies presented in Sect. 4. The resolution time for these models are recorded in Table 3.



**Fig. 3.** The round function of SIMON

**Table 3.** Experimental results of SIMON

Block size $2n$	$R$	$p$	$\mathcal{M}^I$	$\mathcal{M}^{II}$	$\mathcal{M}^{III}$
32	11	$2^{-30}$	75.05s	79.22s	67.92s
	12	$2^{-34}$	657.37s	559.83s	209.09s
48	13	$2^{-38}$	309.58s	376.33s	109.85s
	14	$2^{-44}$	4627.26s	3577.05s	2942.85s
	15	$2^{-46}$	31979.80s	3351.41s	2444.28s
	16	$2^{-50}$	>20h	>15h	26589.96s

From Table 3 we can see that, for larger number of rounds, the improvement is obvious. For example, using  $\mathcal{M}^{III}$  we can prove that the probability of the optimal characteristic of 15-round SIMON48 is  $2^{-46}$  in 2444.28 s, while for  $\mathcal{M}^I$ , the resolution time is 31979.80 s.

### 5.3 Application to SPECK

The SPECK is a family of ARX Feistel block ciphers (depicted in Fig. 4) designed by the National Security Agency of USA. The parameters of different SPECK instances involved in our experiments are summarized in Table 4.

We construct three models  $\mathcal{M}^I$ ,  $\mathcal{M}^{II}$ , and  $\mathcal{M}^{III}$  according to the strategies presented in Sect. 4. The resolution time for these models are recorded in

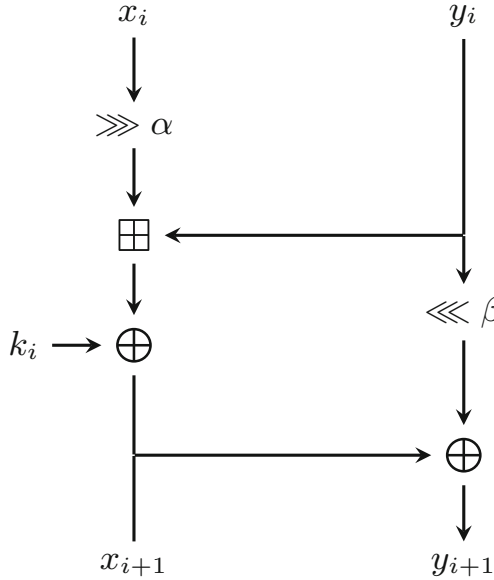


Fig. 4. The round function of SPECK

Table 4. Parameters for SPECK32 and SPECK48

Variant $2n/mn$	Block Size $2n$	Key Size $mn$	Round $r$	$\alpha$	$\beta$
32/64	32	64	22	7	2
48/72	48	72	22	8	3
48/96	48	96	23	8	3

Table 5. Experimental results of SPECK

Block size $2n$	$R$	$p$	$\mathcal{M}^I$	$\mathcal{M}^{II}$	$\mathcal{M}^{III}$
32	5	$2^{-9}$	9.78s	17.15s	26.08s
	6	$2^{-13}$	173.67s	820.82s	390.33s
	7	$2^{-18}$	7175.87s	>10000s	>10000s
48	5	$2^{-10}$	32.90s	358.11s	273.98s
	6	$2^{-14}$	1482.66s	2626.50s	2287.21s
	7	$2^{-19}$	40860.38s	>100000s	>100000s

Table 5. However, the results show that the new modeling strategies are inferior to the original method. This may somehow implies that adding Matsui’s bounding conditions for MILP models of ARX ciphers is not a good choice.

## 6 Conclusion

Borrowing the ideas from Matsui's algorithm, we tweak the MILP models for differential cryptanalysis by altering the objective functions and introducing in special constraints derived from Matsui's bounding condition. We apply this new modeling strategy to PRESENT, SPECK, and SIMON, which demonstrates that the fusion of Matsui's bounding condition and the MILP approach leads to faster resolution in some cases. Therefore, the new modeling approach is expected to reduce the time cost of differential and linear analysis. In particular, during the design process of symmetric-key schemes, a larger design space may be explored within limited time. Our work shows that it is beneficial to include Matsui's bounding condition in the MILP models for differential analysis. More generally, it is interesting to see how to integrate other search heuristics [16, 17] from the literature of symmetric-key cryptanalysis into the MILP models.

**Acknowledgments.** The authors thank the anonymous reviewers for many helpful comments. The work is supported by the National Natural Science Foundation of China (61732021, 61772519), the Youth Innovation Promotion Association of Chinese Academy of Sciences, the Chinese Major Program of National Cryptography Development Foundation, and the Institute of Information Engineering, CAS (Grant No. Y7Z0341103).

## References

1. Gurobi Optimization. Gurobi Optimizer Reference Manual (2013)
2. Abdelkhalek, A., Sasaki, Y., Tolba, M., Youssef, A.M.: MILP modeling for (large) S-boxes to optimize probability of differential characteristics. *IACR Trans. Symmetric Cryptol.* **2017**(4), 99–129 (2017)
3. Aoki, K., Kobayashi, K., Moriai, S.: Best differential characteristic search of FEAL. In: Biham, E. (ed.) *FSE 1997*. LNCS, vol. 1267, pp. 41–53. Springer, Heidelberg (1997). <https://doi.org/10.1007/BFb0052333>
4. Aumasson, J.-P., Jovanovic, P., Neves, S.: Analysis of NORX: investigating differential and rotational properties. In: Aranha, D.F., Menezes, A. (eds.) *LATIN-CRYPT 2014*. LNCS, vol. 8895, pp. 306–324. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-16295-9\\_17](https://doi.org/10.1007/978-3-319-16295-9_17)
5. Banner, A., Nicolas, B., Eric, F.: Automatic search for a maximum probability differential characteristic in a substitution-permutation network. In: *48th Hawaii International Conference on System Sciences, HICSS 2015, Kauai, Hawaii, USA, January 5–8, 2015*, pp. 5165–5174 (2015)
6. Bao, Z., Zhang, W., Lin, D.: Speeding up the search algorithm for the best differential and best linear trails. In: Lin, D., Yung, M., Zhou, J. (eds.) *Inscrypt 2014*. LNCS, vol. 8957, pp. 259–285. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-16745-9\\_15](https://doi.org/10.1007/978-3-319-16745-9_15)
7. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptol. ePrint Arch.* **2013**, 404 (2013)

8. Beierle, C., et al.: 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). [https://doi.org/10.1007/978-3-662-53008-5\\_5](https://doi.org/10.1007/978-3-662-53008-5_5)
9. Biryukov, A., Perrin, L.: State of the art in lightweight symmetric cryptography. IACR Cryptol. ePrint Arch. **2017**, 511 (2017)
10. Biryukov, A., Velichkov, V., Le Corre, Y.: Automatic search for the best trails in ARX: application to block cipher SPECK. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 289–310. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-52993-5\\_15](https://doi.org/10.1007/978-3-662-52993-5_15)
11. Bogdanov, A., et al.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-74735-2\\_31](https://doi.org/10.1007/978-3-540-74735-2_31)
12. Chen, J., Teh, J., Liu, Z., Chunhua, S., Samsudin, A., Xiang, Y.: Towards accurate statistical analysis of security margins: new searching strategies for differential attacks. IEEE Trans. Comput. **66**(10), 1763–1777 (2017)
13. Collard, B., Standaert, F.-X., Quisquater, J.-J.: Improved and multiple linear cryptanalysis of reduced round serpent. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Inscrypt 2007. LNCS, vol. 4990, pp. 51–65. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-79499-8\\_6](https://doi.org/10.1007/978-3-540-79499-8_6)
14. Cui, T., Jia, K., Kai, F., Chen, S., Wang, M.: New automatic search tool for impossible differentials and zero-correlation linear approximations. IACR Cryptol. ePrint Arch. **2016**, 689 (2016)
15. Dobraunig, C., et al.: Rasta: a cipher with low ANDdepth and few ANDs per bit. IACR Cryptol. ePrint Arch. **2018**, 181 (2018)
16. Dobraunig, C., Eichlseder, M., Mendel, F.: Heuristic tool for linear cryptanalysis with applications to CAESAR candidates. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 490–509. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48800-3\\_20](https://doi.org/10.1007/978-3-662-48800-3_20)
17. Eichlseder, M., Mendel, F., Schl affer, M.: Branching heuristics in differential collision search with applications to SHA-512. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 473–488. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46706-0\\_24](https://doi.org/10.1007/978-3-662-46706-0_24)
18. Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: MILP-Based automatic search algorithms for differential and linear trails for speck. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 268–288. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-52993-5\\_14](https://doi.org/10.1007/978-3-662-52993-5_14)
19. Gerault, D., Minier, M., Solnon, C.: Constraint programming models for chosen key differential cryptanalysis. In: Proceedings of Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5–9, 2016, pp. 584–601 (2016)
20. Lai, X., Massey, J.L., Murphy, S.: Markov ciphers and differential cryptanalysis. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 17–38. Springer, Heidelberg (1991). [https://doi.org/10.1007/3-540-46416-6\\_2](https://doi.org/10.1007/3-540-46416-6_2)
21. Matsui, M.: On correlation between the order of S-boxes and the strength of DES. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 366–375. Springer, Heidelberg (1995). <https://doi.org/10.1007/BFb0053451>
22. Mouha, N., Preneel, B.: A proof that the ARX cipher Salsa20 is secure against differential cryptanalysis. IACR Cryptol. ePrint Arch. **2013**, 328 (2013)

23. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C.-K., Yung, M., Lin, D. (eds.) *Inscrypt 2011*. LNCS, vol. 7537, pp. 57–76. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-34704-7\\_5](https://doi.org/10.1007/978-3-642-34704-7_5)
24. Ohta, K., Moriai, S., Aoki, K.: Improving the search algorithm for the best linear expression. In: Coppersmith, D. (ed.) *CRYPTO 1995*. LNCS, vol. 963, pp. 157–170. Springer, Heidelberg (1995). [https://doi.org/10.1007/3-540-44750-4\\_13](https://doi.org/10.1007/3-540-44750-4_13)
25. Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects. In: Coron, J.-S., Nielsen, J.B. (eds.) *EUROCRYPT 2017*. LNCS, vol. 10212, pp. 185–215. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56617-7\\_7](https://doi.org/10.1007/978-3-319-56617-7_7)
26. Sun, S., et al.: Analysis of AES, SKINNY, and others with constraint programming. *IACR Trans. Symmetric Cryptol.* **2017**(1), 281–306 (2017)
27. Sun, S., et al.: Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties. *Cryptology ePrint Archive*, Report 2014/747 (2014). <http://eprint.iacr.org/2014/747>
28. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) *ASIACRYPT 2014*. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-45611-8\\_9](https://doi.org/10.1007/978-3-662-45611-8_9)
29. Shengbao, W., Wang, M.: Security evaluation against differential cryptanalysis for block cipher structures. *IACR Cryptol. ePrint Arch.* **2011**, 551 (2011)
30. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J.H., Takagi, T. (eds.) *ASIACRYPT 2016*. LNCS, vol. 10031, pp. 648–678. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53887-6\\_24](https://doi.org/10.1007/978-3-662-53887-6_24)
31. Yao, Y., Zhang, B., Wu, W.: Automatic search for linear trails of the SPECK family. In: Lopez, J., Mitchell, C.J. (eds.) *ISC 2015*. LNCS, vol. 9290, pp. 158–176. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-23318-5\\_9](https://doi.org/10.1007/978-3-319-23318-5_9)



# Automatic Search for Related-Key Differential Trails in SIMON-like Block Ciphers Based on MILP

Xuzi Wang<sup>1,2</sup>, Baofeng Wu<sup>1,2</sup>, Lin Hou<sup>1,2(✉)</sup>, and Dongdai Lin<sup>1,2</sup>

<sup>1</sup> State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

{wangxuzi,wubaofeng,houlin,ddlin}@iie.ac.cn

<sup>2</sup> School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

**Abstract.** In this paper, we revisit the relationship between the probability of differential trails and the input difference of each round for SIMON-like block ciphers. The key observation is that not only the Hamming weight but also the positions of active bits of the input difference have effect on the probability. Based on this, our contributions are mainly twofold. Firstly, we rebuild the MILP model for SIMON-like block ciphers without quadratic constraints. Accordingly, we give the accurate objective function and reduce its degree to one by adding auxiliary variants to make the model easy to solve. Secondly, we search for optimal differential trails for SIMON and SIMECK based on this model. To the best of our knowledge, this is the first time that related-key differential trails have been obtained. Besides, we not only recover the single-key results in [11], but also obtain impossible differentials through this method.

**Keywords:** SIMON · SIMECK · Related-key differential trails  
Impossible differentials · MILP

## 1 Introduction

Devices of small size, such as smart cards and sensor networks, are increasingly involved in our life. Despite of the convenience, a major concern is that these highly constrained devices cannot afford the computational cost of traditional block ciphers such as DES and AES. To this end, the notion of *lightweight block cipher* was raised, and has seen a flourish of research works in recent years.

Specifically, SIMON and SPECK families of block ciphers [4] proposed by the NSA are amongst the most promising candidates. The distinguishing feature of SIMON (SPECK) is that AND operations (modular additions) serve as non-linear components instead of S-boxes, and this directly yields an implementation advantage on both hardware and software platforms. Later on, Yang *et al.* proposed a variant of SIMON, namely SIMECK [22] which adopts the rotational constants and key schedules of SPECK within the framework of SIMON.



We refer to both SIMON and SIMECK as SIMON-like block ciphers, as introduced in [11].

**Related Work.** For SIMON, no designing rationale or security analysis was explicitly given in the original paper [4]. Lots of subsequent researches have been done for evaluating its security, and a majority of these works are also applicable to the SIMECK case due to their great similarities [22].

There have been many results for SIMON by differential cryptanalysis [2, 3, 6–11, 20] and linear cryptanalysis [1, 12, 14]. To our interest, Kölbl *et al.* [9] gave an exact closed form expression for the differential probability, and obtained single-key differential characteristics through SAT/SMT solvers; in 2017, Liu *et al.* [11] further investigated the relationship between Hamming weight of input difference and differential probability, and proposed an automatic searching algorithm by adapting Matsui’s algorithm [13], and obtained optimal single-key differential trails for SIMON-like block ciphers.

On the other hand, Todo introduced the notion of *division property*, and used it in finding integral distinguishers for SIMON [18]; later on, Todo and Morii proposed a fine-grained variant called *bit-based division property* [19], and thus gave integral distinguisher for SIMON32 with one more round.

Besides, the method of Mixed-Integer Linear Programming (MILP) is widely used in automatic searching recently [15, 17]. Specifically for SIMON, Sun *et al.* modified the original model [15, 17] into an MIP (Mixed-Integer Programming) one by adding quadratic constraints [16], to remove invalid characteristics out of the feasible region. Although they made it theoretically solvable by adding auxiliary variants, it still seems rather sophisticated to make practical use of this model. It is worth noting that based on division property, Xiang *et al.* [21] applied MILP to *automatically* searching integral distinguishers for six lightweight block ciphers including SIMON and SIMECK.

Throughout, no cryptanalysis work has been done for SIMON-like block ciphers in related-key setting, and this issue is also mentioned by the designers of SIMON [5] and SIMECK [22] independently. In fact, the behavior of certain block cipher under related-key differential cryptanalysis is an important criterion for its security, since the secret keys are often updated in security protocols or differences can be incorporated using fault attacks. Meanwhile, avoiding high-probability related-key differential characteristics is one of the goal of the key schedule.

**Our Contributions.** In this paper, we make a fine-grained analysis of the ROTATION-AND operations and construct proper MILP models for SIMON-like block ciphers. As a result, we give related-key differential trails for SIMON-like block ciphers for the first time.

Specifically, we revisit the relationship between the input difference and the probability of differential trails, and reveal that *the active bits’ positions* of the input difference will not only determine which bits of the output difference are likely to be active, but also affect the probability of differential characteristics.

From this we can get all possible output differences of the ROTATION-AND operation and their accurate probabilities *directly from input difference*, rather than using a DDTA (Difference Distribution Table of AND) accompanied with some checking conditions as done in [6, 9, 11]. As a result, we can construct proper MILP models with linear objective function while without quadratic constraints, and search related-key differential trails for SIMON and SIMECK automatically, as well as impossible differentials.

Our main results are listed in the following:

1. We find 10, 9, 9 rounds optimal related-key differential trails for SIMON32/64, SIMON48/96 and SIMON64/128 with probability  $2^{-16}$ ,  $2^{-18}$  and  $2^{-18}$  respectively, costing about 15 days, 6 days and 7 days respectively.<sup>1</sup> Moreover, we find that there is an 8-round period trail with probability  $2^{-n}$  for SIMON2n/4n, and thus all trails can be extended to 19 rounds with probability  $2^{-2n}$ .
2. We find two 11 rounds optimal related-key differential trails for SIMON48/72 and SIMON64/96 with probability  $2^{-22}$  and  $2^{-22}$  respectively, costing about 7 days and 7 days respectively. The extension for SIMON48 reaches 16 rounds with probability  $2^{-50}$ , and the extension for SIMON64 reaches 18 rounds with probability  $2^{-64}$ .
3. We find 15, 16, 16 rounds optimal related-key differential trails for SIMECK32/64, SIMECK48/96, and SIMECK64/128 with probability  $2^{-34}$ ,  $2^{-40}$ , and  $2^{-40}$  respectively, costing about 9.6 h, 3.8 days and 4 days respectively. The extension of SIMECK48/96 reaches 19 rounds with probability  $2^{-48}$ , and the extension for SIMECK64/128 reaches 23 rounds with probability  $2^{-66}$ .

For searching single-key differential trails, without of generality, we assume that there must exist certain round with input difference of Hamming weight one when considering the diffusion of block ciphers. Then by our method, we can recover the results in [11]. In addition, we also get 11, 12 and 13 rounds impossible differentials for SIMON32, SIMON48 and SIMON64 respectively, and get 11, 15 and 17 rounds impossible differentials for SIMECK32, SIMECK48 and SIMECK64 respectively, all in the single-key setting.

**Organization of the Paper.** We introduce notations and recall the constructions of SIMON-like block ciphers in Sect. 2. In Sect. 3, we present the main theorem on relationship between the input difference and the differential probability, and construct proper MILP models for SIMON-like block ciphers. Our results are presented in Sect. 4. Section 5 is a conclusion of this paper.

---

<sup>1</sup> All experiments are performed on a PC with 2.5 GHz Intel Core i7 and 16GB 1600 MHz DDR3.

## 2 Preliminaries

### 2.1 Notations

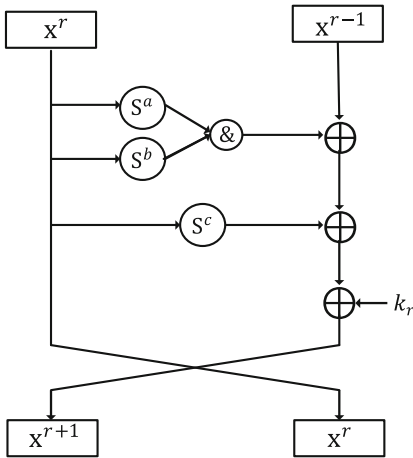
We say a bit is *active* if it is one. For the left half input difference in SIMON2n, each bit has a subscript denoting its position, with that of the most significant bit being 0; all subscripts are in the sense modulo  $n$ . We list main notations in Table 1.

**Table 1.** Notations.

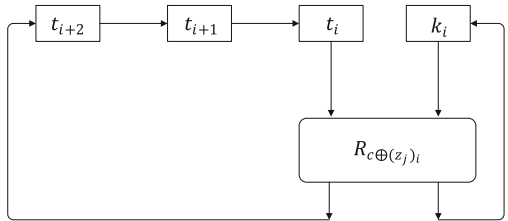
Notation	Description
$\odot, \&$	AND operation
$\oplus$	XOR operation
$S^i$	left circular shift by $i$ bits
$S^{-i}$	right circular shift by $i$ bits
$\Delta x_i^r$	the $i$ -th bit of left half input difference of the $r$ -th round
$\Delta d_i^r$	the $i$ -th bit of output difference of AND operation of the $r$ -th round
$(a, b, c)$	the rotation parameters for SIMON-like block ciphers

### 2.2 A Brief Description of SIMON and SIMECK

The round function of SIMON-like block ciphers is shown in Fig. 1, with the value of  $(a, b, c)$  being  $(8, 1, 2)$  and  $(0, 5, 1)$  for SIMON and SIMECK respectively.



**Fig. 1.** The round function of SIMON-like block ciphers.



**Fig. 2.** The key expansion of SIMECK.

The key schedules of SIMON and SIMECK are totally different. The constant  $C = 2^n - 4 = 0x\text{ff}\dots\text{fc}$ , and the generation of constant sequence  $\{z_j\}$  is referred to [4] (for SIMON) and [22] (for SIMECK). The key of the  $i$ -th round is denoted by  $k_i$ , and the identical permutation is denoted by  $I$ . For SIMON2n/mn, round keys are generated by

$$k_{i+m} = \begin{cases} C \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+1}, & \text{if } m = 2, \\ C \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+2}, & \text{if } m = 3, \\ C \oplus (z_j)_i \oplus k_i \oplus (I \oplus S^{-1})(S^{-3}k_{i+3} \oplus k_{i+1}), & \text{if } m = 4. \end{cases}$$

For SIMECK2n/4n, the key schedules are shown in Fig. 2. The updating function is expressed as

$$\begin{cases} k_{i+1} = t_i, \\ t_{i+3} = k_i \oplus f(t_i) \oplus C \oplus (z_j)_i. \end{cases}$$

where  $f(x) = x \odot S^5(x) \oplus S^1(x)$  is part of the round function.

### 3 Constructing MILP Models for SIMON-like Block Ciphers

In this section, we make a fine-grained analysis of the relationship between input and output difference of the ROTATION-AND operations. We prove that not only the Hamming weight but also the active bits' positions of the input difference can affect the probability of differential characteristics. The former has been proved by Liu *et al.* [11], and we highlight the latter's importance in constructing proper MILP models for SIMON-like block ciphers. Specifically, we give the following theorem:

**Theorem 1.** *Let  $f(x) = S^a(x) \odot S^b(x)$  be a Boolean function from  $\mathbb{F}_2^n$  to itself, and  $\gcd(n, a - b) = 1$ . Let  $\Delta x, \Delta d \in \mathbb{F}_2^n$  be the input and output difference of  $f$  respectively, with  $\text{wt}(\Delta x) = m$ ,  $m < n$ , and  $R = \{\Delta x_{i_0}, \Delta x_{i_1}, \dots, \Delta x_{i_{m-1}}\}$  be the set of all active bits in  $\Delta x$ . If there exist*

1.  $p_1$  pairs of  $\{i_j, i_k\}$  such that  $|i_j - i_k| \equiv |a - b| \pmod n$ ; and
2.  $p_2$  pairs of  $\{i_j, i_k\}$  such that  $|i_j - i_k| \equiv 2|a - b| \pmod n$  and there exists some  $h$  such that  $|h - i_j| \equiv |a - b| \pmod n$ ,  $|h - i_k| \equiv |a - b| \pmod n$ ,  $\Delta x_h \notin R$ ;

*then there will be  $2^{2m-p_1-p_2}$  possible values for  $\Delta d$ , and each has the same probability  $2^{-2m+p_1+p_2}$ .*

To prove this theorem, we use the following lemma, which can be regarded as a generalization of Observation 2 in [8]. All proofs can be found in the Appendix.

**Lemma 1.** Let  $f(x) = S^a(x) \odot S^b(x)$  be a Boolean function from  $\mathbb{F}_2^n$  to itself. Let  $\Delta x, \Delta d \in \mathbb{F}_2^n$  be the input and output difference of  $f$  respectively, and  $x \in \mathbb{F}_2^n$  be an input of  $f$ . Then,

1. In  $\Delta x$ , only two bits, namely  $\Delta x_{i+a}$  and  $\Delta x_{i+b}$  can affect the value of  $\Delta d_i$ , which is an arbitrary bit in  $\Delta d$ ;
2. An arbitrary bit  $\Delta x_i$  in  $\Delta x$ , can affect only two bits  $\Delta d_{i-a}$  and  $\Delta d_{i-b}$  in  $\Delta d$ ;
3. An arbitrary bit  $x_i$  in  $x$  can affect at most two bits  $\Delta d_{i-a}$  and  $\Delta d_{i-b}$  in  $\Delta d$ . Specifically,  $\Delta d_{i-a}$  is affected by  $x_i$ , iff.  $\Delta x_{i-a+b} = 1$ ;  $\Delta d_{i-b}$  is affected by  $x_i$ , iff.  $\Delta x_{i-b+a} = 1$ .

Based on Theorem 1, we can construct proper MILP models for SIMON-like block ciphers in the following.

**Constraints Imposed by XOR Operations.** There are lots of XOR operations in either round functions or key schedules of SIMON-like block ciphers. This turns out to be a bottleneck in constructing efficient models if we follow the XOR constraints given in [15, 17], since there will be too many auxiliary variants. However, we note that all possible points can be figured out easily and linear constraints without auxiliary variants can then be obtained using the SageMath code in [17]. We demonstrate this by the following example.

Let  $x \oplus y \oplus z = w$ , where  $x, y, z, w \in \mathbb{F}_2$ . All possible points for  $(x, y, z, w)$  are  $(0, 0, 0, 0)$ ,  $(0, 0, 1, 1)$ ,  $(0, 1, 0, 1)$ ,  $(0, 1, 1, 0)$ ,  $(1, 0, 0, 1)$ ,  $(1, 0, 1, 0)$ ,  $(1, 1, 0, 0)$  and  $(1, 1, 1, 1)$ . We can easily get the linear constraints as follows:

$$\begin{cases} x + y - z + w \geq 0 \\ x + y + z - w \geq 0 \\ -x + y + z + w \geq 0 \\ x - y + z + w \geq 0 \\ -x - y + z - w \geq -2 \\ x - y - z - w \geq -2 \\ -x + y - z - w \geq -2 \\ -x - y - z + w \geq -2 \end{cases}$$

**Constraints Imposed by ROTATION-AND Operations.** Based on Theorem 1, we divide the  $n$  bits input difference and  $n$  bits output difference of ROTATION-AND operations into  $n$  groups. Specifically, group  $i$  ( $0 \leq i \leq n-1$ ) consists of three input difference bits at positions  $(i, i+t, i+2t)$  and two output difference bits at positions  $(i-b, i+t-b)$ , where  $t = |a-b|$ .

Taking SIMON32 as an example, we list all 16 groups in Table 2, and all possible points with respect to each group in Table 3. Then we can get the following linear constraints by running the SageMath code [17] on input of all possible points, where there is no auxiliary variants and the feasible region of which contains no invalid characteristics.

$$\left\{ \begin{array}{l} \Delta x_{i+t}^r - \Delta x_{i+2t}^r - \Delta d_{i-b}^r + \Delta d_{i+t-b}^r \geq -1 \\ \Delta x_i^r + \Delta x_{i+t}^r - \Delta d_{i-b}^r \geq 0 \\ -\Delta x_i^r + \Delta x_{i+t}^r + \Delta d_{i-b}^r - \Delta d_{i+t-b}^r \geq -1 \\ \Delta x_{i+t}^r + \Delta x_{i+2t}^r - \Delta d_{i+t-b}^r \geq 0 \end{array} \right.$$

**Table 2.** The 16 groups for SIMON32.

Input Bits	0,7,14	7,14,5	14,5,12	5,12,3	12,3,10	3,10,1	10,1,8	1,8,15
Output Bits	15,6	6,13	13,4	4,11	11,2	2,9	9,0	0,7
Input Bits	8,15,6	15,6,13	6,13,4	13,4,11	4,11,2	11,2,9	2,9,0	9,0,7
Output Bits	7,14	14,5	5,12	12,3	3,10	10,1	1,8	8,15

**Objective Functions.** Let the probability of the differential characteristic be  $2^{-w}$ . Then we have the following objective function from Theorem 1:

$$w = \sum_{r=0}^R (2 \sum_{i=0}^{n-1} \Delta x_i^r - \sum_{i=0}^{n-1} \Delta x_i^r \Delta x_{i+t}^r - \sum_{i=0}^{n-1} \Delta x_i^r \Delta x_{i+2t}^r + \sum_{i=0}^{n-1} \Delta x_i^r \Delta x_{i+t}^r \Delta x_{i+2t}^r). \quad (1)$$

However, this objective function of degree three makes it hard to solve the model. To solve this issue, we form  $n$  groups with group  $i$  consisting of three bits input difference  $(\Delta x_i^r, \Delta x_{i+t}^r, \Delta x_{i+2t}^r)$  as well as an auxiliary variants  $p_i^r$ , in order to reduce the degree of the objective function to one.

$$w = 2 \sum_{r=0}^R \sum_{i=0}^{n-1} \Delta x_i^r - \sum_{r=0}^R \sum_{i=0}^{n-1} p_i^r. \quad (2)$$

Then we can obtain the following linear constraints, taking the relationships between  $(\Delta x_i^r, \Delta x_{i+t}^r, \Delta x_{i+2t}^r)$  and  $p_i^r$  as shown in Table 4.

$$\left\{ \begin{array}{l} \Delta x_{i+2t}^r - p_i^r \geq 0 \\ -\Delta x_i^r - \Delta x_{i+2t}^r + p_i^r \geq -1 \\ -\Delta x_{i+t}^r - \Delta x_{i+2t}^r + p_i^r \geq -1 \\ \Delta x_i^r + \Delta x_{i+t}^r - p_i^r \geq 0 \end{array} \right.$$

Since the non-linear key schedules of SIMECK essentially reuse its round function, the objective function of SIMECK turns out to

$$w = 2 \sum_{r=0}^R \sum_{i=0}^{n-1} \Delta x_i^r - \sum_{r=0}^R \sum_{i=0}^{n-1} p_i^r + 2 \sum_{r=1}^{R-3} \sum_{i=0}^{n-1} \Delta k_i^r - \sum_{r=1}^{R-3} \sum_{i=0}^{n-1} p_{ki}^r. \quad (3)$$

## 4 (Related-Key) Differential Trails for SIMON and SIMECK

In this section, we show the (related-key) differential trails for SIMON and SIMECK, which are automatically searched by solving the MILP models in Sect. 3 using Gurobi. Our results are twofold: first and foremost, we give (long) related-key differential trails for SIMON-like block ciphers for the first time; second, using the same method, we give impossible differentials for SIMON-like block ciphers and recover the trails given by Liu *et al.* [11], both in the single-key setting.

**Table 3.** All possible points for each group.

$(\Delta x_i^r, \Delta x_{i+t}^r, \Delta x_{i+2t}^r)$	$(\Delta d_{i-b}^r, \Delta d_{i+t-b}^r)$
(0, 0, 0)	(0, 0)
(0, 0, 1)	(0, 0), (0,1)
(0, 1, 0)	(0, 0), (0,1), (1,0), (1,1)
(0, 1, 1)	(0, 0), (0,1), (1,0), (1,1)
(1, 0, 0)	(0, 0), (1,0)
(1, 0, 1)	(0, 0), (1,1)
(1, 1, 0)	(0, 0), (0,1), (1,0), (1,1)
(1, 1, 1)	(0, 0), (0,1), (1,0), (1,1)

**Table 4.** The value of auxiliary variant  $p_i^r$ .

$(\Delta x_i^r, \Delta x_{i+t}^r, \Delta x_{i+2t}^r)$	$p_i^r$
(0, 0, 0)	0
(0, 0, 1)	0
(0, 1, 0)	0
(0, 1, 1)	1
(1, 0, 0)	0
(1, 0, 1)	1
(1, 1, 0)	0
(1, 1, 1)	1

### 4.1 Related-Key Differential Trails

We present optimal related-key differential trails for SIMON32/64 in Table 6, SIMON48/72 and SIMON48/96 in Table 7, SIMECK32/64 and SIMECK48/96 in Table 8. The optimal trails for SIMON64 and SIMECK64 are identical to those for SIMON48 and SIMECK48 respectively.

Except for SIMECK32/64, constrained by the limited computational resources, it is still difficult to obtain longer optimal related-key differential trails for other parameters, whose probabilities may hopefully reach the security margin. To solve this issue, putting some optimal trail in the middle, we search both forwards and backwards until it reaches the security margin. In addition, we observe that there exists an 8-round period for SIMON32/64 in the related-key setting, which yields a 19-round related-key differential trail with probability  $2^{-32}$ . These results are summarized in Table 5.

## 4.2 Single-Key Differential Trails

For obtaining single-key trails, it indeed costs more time by *directly* solving the MILP models in Sect. 3 than using the method in [11]. However, a key observation is that in optimal single-key differential trails, there is always some round’s input difference with Hamming weight one. This can be explained from the following two perspectives: on the one hand, the upper-bound of probability of each round is negatively related to the Hamming weight of its input difference, as proved in [11]; on the other hand, considering the diffusion property, an active input difference bit of some round can make many forward and backward bits active; thus, it is intuitive to require the Hamming weight of some round’s input difference to be the least (namely one), for obtaining long trails.

Keeping these in mind, we can recover the results in [11] ( $R$ -round optimal single-key differential trails) using much less time, by solving the MILP models with the precondition that there exists some  $r \in \{0, \dots, R - 1\}$  such that the Hamming weight of the  $r$ -th round’s input difference is one.

**Table 5.** The probabilities of optimal and best related-key differential trails for variants of SIMON and SIMECK. To distinguish from optimal trails, best trails are labeled with \*. For simplicity, all probabilities  $p$  are given as  $(-\log_2 p)$  in the table.

Rounds	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
SIMON32/64	0	2	4	8	11	16	16*	24*	24*	24*	24*	32*	32*	32*	32*	-	-	-	-
SIMON48/72	2	4	6	12	14	18	22	30*	33*	40*	42*	50*	-	-	-	-	-	-	-
SIMON48/96	0	2	4	8	12	18	24*	35*	36*	36*	36*	48*	48*	48*	48*	-	-	-	-
SIMON64/96	2	4	6	12	14	18	22	30*	34*	42*	46*	54*	55*	64*	70*	-	-	-	-
SIMON64/128	0	2	4	8	12	18	26*	36*	41*	48*	48*	64*	64*	64*	64*	-	-	-	-
SIMECK32/64	0	2	4	8	10	14	18	22	26	30	34	-	-	-	-	-	-	-	-
SIMECK48/96	0	2	4	8	10	14	18	22	26	30	34	40	42*	46*	48*	-	-	-	-
SIMECK64/128	0	2	4	8	10	14	18	22	26	30	34	40	42*	46*	48*	54*	56*	62*	66*

## 4.3 Impossible Differentials

Considering the *miss-in-the-middle* approach and the diffusion property, we search impossible differentials for SIMON-like block ciphers in single-key setting, under that there is only one active bit in either the input difference or the



**Table 6.** 10 rounds optimal related-key differential trails for SIMON32/64, where the numbers represent the positions of active bits of input difference of each round while ‘-’ represents that there is no active bits.

R	SIMON32/64		
	$\Delta x^l$	$\Delta x^r$	$\Delta k$
0	-	1,3,5,7,9,11,13,15	1,3,5,7,9,11,13,15
1	-	-	-
2	-	-	0,2,4,6,8,10,12,14
3	0,2,4,6,8,10,12,14	-	1,3,5,7,9,11,13,15
4	-	0,2,4,6,8,10,12,14	0,2,4,6,8,10,12,14
5	-	-	-
6	-	-	1,3,5,7,9,11,13,15
7	1,3,5,7,9,11,13,15	-	1,3,5,7,9,11,13,15
8	-	1,3,5,7,9,11,13,15	1,3,5,7,9,11,13,15
9	-	-	-
10	-	-	0,2,4,6,8,10,12,14
11	0,2,4,6,8,10,12,14	-	1,3,5,7,9,11,13,15

**Table 7.** Optimal related-key differential trails for SIMON48.

R	SIMON48/72			SIMON48/96		
	$\Delta x^l$	$\Delta x^r$	$\Delta k$	$\Delta x^l$	$\Delta x^r$	$\Delta k$
0	-	7,8,9,10,22	4,6,7,8,9,10,22	-	9,11,12,13,16,17,19	9,11,12,13,16,17,19
1	4,6	-	2,4,22	-	-	11,12,14,15
2	22	4,6	1,2,4,6,20	11,12,14,15	-	9,12,14
3	1,2	22	0,22,23	11	11,12,14,15	9,10,14,15
4	-	1,2	1,2,22	11,12	11	9
5	22	-	20	11	11,12	9,10,11,12
6	-	22	22	-	11	11
7	-	-	22	-	-	-
8	22	-	1,2,20	-	-	13
9	1,2	22	0,4,6,22,23	13	-	9,10,16,17
10	4,6	1,2	1,4,7,8,9,10,22	9,10,11,16,17	-	
11	7,8,9,10,22	4,6				

output difference. Then if the MILP models are infeasible under this condition, we get impossible differentials.

Taking the rotational invariance property of SIMON-like block ciphers [20], for each variant of SIMON $2n$  and SIMECK $2n$ , an impossible differential additionally yields  $(n - 1)$  impossible differentials by rotation. Our main results are listed in Table 9.

**Table 8.** Optimal related-key differential trails for SIMECK32 and SIMECK48.

R	SIMECK32/64			SIMECK48/96		
	$\Delta x^l$	$\Delta x^r$	$\Delta k$	$\Delta x^l$	$\Delta x^r$	$\Delta k$
0	-	4,8,9,10	4,8,9	-	18	18
1	10	-	9	-	-	-
2	-	10	10	-	-	20
3	-	-	-	20	-	-
4	-	-	-	19	20	18
5	-	-	-	20	19	19
6	-	-	10	19	20	20
7	10	-	-	18,19	19	17
8	9	10	-	-	18,19	-
9	4,8,9,10	9	9	18,19	-	-
10	5,7	4,8,9,10	10	17,19	18,19	16,20
11	6,8,9	5,7	-	20	17,19	17
12	-	6,8,9	8,9	-	20	-
13	6	-	5	20	-	19
14	-	6	10	-	20	20
15	6,10	-	-	-	-	17
16				17	-	

**Table 9.** Impossible differentials for SIMON and SIMECK in single-key.

	ROUNDS	Trails
SIMON32	11	$(0,4000) \nrightarrow (80,0);$ $(0,4000) \nrightarrow (20,0)$
SIMON48	12	$(0,400000) \nrightarrow (800000,0);$ $(0,400000) \nrightarrow (200000,0)$
SIMON64	13	$(0,40000000) \nrightarrow (8000000,0);$ $(0,40000000) \nrightarrow (800000,0);$ $(0,40000000) \nrightarrow (200000,0);$ $(0,40000000) \nrightarrow (80,0);$ $(0,40000000) \nrightarrow (20,0);$ $(0,40000000) \nrightarrow (2,0)$
SIMECK32	11	$(0,4000) \nrightarrow (200,0);$ $(0,4000) \nrightarrow (8,0)$
SIMECK48	15	$(0,400000) \nrightarrow (800000,0);$ $(0,400000) \nrightarrow (200000,0);$ $(0,400000) \nrightarrow (20000,0);$ $(0,400000) \nrightarrow (8,0)$
SIMECK64	17	$(0,40000000) \nrightarrow (8000000,0);$ $(0,40000000) \nrightarrow (2,0)$

## 5 Summary

In this paper, we mainly studied the security of SIMON-like block ciphers in the related-key setting, by a fine-grained analysis of the ROTATION-AND operations. We hope our work helpful in designing key schedules for SIMON-like block ciphers. For future works, it is desirable to obtain longer optimal differ-

ential trails in related-key setting, maybe by combining our work with other automatic searching algorithm, e.g., SAT/SMT solver.

**Acknowledgement.** The authors thank the anonymous reviewers of ISC2018 for useful comments. This work was supported by the NSFC under grant #61379139.

## A Proof of Lemma 1

*Proof.* Let  $x$  and  $(x \oplus \Delta x)$  be two inputs of the function  $f$ . We have

$$\begin{aligned} \Delta d &= f(x) \oplus f(x \oplus \Delta x) \\ &= (S^a(x) \odot S^b(x)) \oplus (S^a(x \oplus \Delta x) \odot S^b(x \oplus \Delta x)) \\ &= S^a(x) \odot S^b(\Delta x) \oplus S^a(\Delta x) \odot S^b(x) \oplus S^a(\Delta x) \odot S^b(\Delta x) \end{aligned} \quad (4)$$

Then for any bit  $\Delta d_i$  in  $\Delta d$  ( $i = 0, \dots, n-1$ ), we have

$$\Delta d_i = x_{i+a} \odot \Delta x_{i+b} \oplus \Delta x_{i+a} \odot x_{i+b} \oplus \Delta x_{i+a} \odot \Delta x_{i+b} \quad (5)$$

Obviously, only two bits in  $\Delta x$ , namely  $\Delta x_{i+a}$  and  $\Delta x_{i+b}$  can affect the value of  $\Delta d_i$ .

Fix an arbitrary  $i$ , assume that  $\Delta d_k$  is affected by  $\Delta x_i$ . First, we have

$$\Delta d_k = x_{k+a} \odot \Delta x_{k+b} \oplus \Delta x_{k+a} \odot x_{k+b} \oplus \Delta x_{k+a} \odot \Delta x_{k+b}, \quad (6)$$

from Eq. (5). If  $\Delta d_k$  is affected by  $\Delta x_i$ , then we have

$$i \equiv k + a \pmod{n} \quad (7)$$

or

$$i \equiv k + b \pmod{n} \quad (8)$$

Put it in another form, we have

$$k \equiv i - a \pmod{n} \quad (9)$$

or

$$k \equiv i - b \pmod{n} \quad (10)$$

So proved that an arbitrary bit  $\Delta x_i$  can affect only two bits  $\Delta d_{i-a}$  and  $\Delta d_{i-b}$ .

From Eq. (5), we have that

- (1) if  $\Delta x_{i+a} = 0, \Delta x_{i+b} = 0$ , then  $\Delta d_i = 0$ ;
- (2) if  $\Delta x_{i+a} = 1, \Delta x_{i+b} = 1$ , then  $\Delta d_i = (x_{i+a} \oplus x_{i+b}) \odot 1 \oplus 1$ ;
- (3) if  $\Delta x_{i+a} = 1, \Delta x_{i+b} = 0$ , then  $\Delta d_i = \Delta x_{i+a} \odot x_{i+b} = x_{i+b}$ ;
- (4) if  $\Delta x_{i+a} = 0, \Delta x_{i+b} = 1$ , then  $\Delta d_i = x_{i+a} \odot \Delta x_{i+b} = x_{i+a}$ .

Let  $x_i$  denote an arbitrary bit in  $x$ .  $\Delta d_k$  is affected by  $x_i$ , iff.  $k \equiv i - a \pmod{n}$  and  $\Delta x_{k+b} = \Delta x_{i-a+b} = 1$ , or  $k \equiv i - b \pmod{n}$  and  $\Delta x_{k+a} = \Delta x_{i-b+a} = 1$ .  $\square$

## B Proof of Theorem 1

*Proof.* Let  $R_d$  be the collection of bits in  $\Delta d$  which are affected by bits in  $R$ . From Lemma 1,

$$R_d = \{\Delta d_{i_0-a}, \Delta d_{i_0-b}, \Delta d_{i_1-a}, \Delta d_{i_1-b}, \dots, \Delta d_{i_{m-1}-a}, \Delta d_{i_{m-1}-b}\}$$

There may be duplicate elements in the collection  $R_d$ .

1. Since  $a \neq b$ , then  $i_\ell - a \not\equiv i_\ell - b \pmod n$ , for  $\ell = 0, \dots, m-1$ ;
2. For  $0 \leq j \neq k \leq m-1$ ,  $i_j - a \not\equiv i_k - a \pmod n$ , since  $i_j \neq i_k$ ;
3. For  $0 \leq j \neq k \leq m-1$ , if  $i_j - a \equiv i_k - b \pmod n$ , then  $i_j - i_k \equiv a - b \pmod n$ ;
4. For  $0 \leq j \neq k \leq m-1$ , if  $i_j - b \equiv i_k - a \pmod n$ , then  $i_j - i_k \equiv b - a \pmod n$ ;

If there exist  $p_1$  pairs of  $\{i_j, i_k\}$  such that  $|i_j - i_k| \equiv |a - b| \pmod n$ , we have

$$i_j - i_k \equiv a - b \pmod n \quad (11)$$

or

$$i_j - i_k \equiv b - a \pmod n \quad (12)$$

we claim that Eqs. (11) and (12) cannot hold true simultaneously, otherwise it contradicts with  $\gcd(n, a - b) = 1$ . Let  $R'_d$  denote the set obtained by removing duplicate elements from the collection  $R_d$ . Then if there exist  $p_1$  pairs of  $\{i_j, i_k\}$  such that  $|i_j - i_k| \equiv |a - b| \pmod n$ ,  $|R_d| - |R'_d| = p_1$ .

Now we turn to discuss the relationships amongst bits in  $\Delta d$ . First, for  $\Delta d_k \notin R'_d$ , we have  $\Delta x_{k+a} = 0$  and  $\Delta x_{k+b} = 0$  from Lemma 1; specifically,  $\Delta d_k = 0$  holds with probability 1, regardless of the values of  $x_{k+a}$  and  $x_{k+b}$ . Thus, we need only to discuss the relationships amongst bits in  $R'_d$ . For  $\Delta d_k \in R'_d$ , it has been proved by Lemma 1 that at least one of  $\Delta x_{k+a}$  and  $\Delta x_{k+b}$  is active. Specifically,

1.  $\Delta x_{k+a} = 1$ ,  $\Delta x_{k+b} = 0$ . In this case,  $\Delta d_k = x_{k+b}$ . If there exists some other bit  $\Delta d'_k \in R'_d$  such that  $\Delta d'_k$  is dependent of  $\Delta d_k$ , then  $k' \equiv k + b - a \pmod n$ , since  $\Delta d_{k+b-a}$  is the only bit which *may be* affected by  $x_{k+b}$  except for  $\Delta d_k$  from Lemma 1.

$$\begin{aligned} \Delta d_{k+b-a} &= \Delta x_{k+b} \odot x_{k+2b-a} \oplus x_{k+b} \odot \Delta x_{k+2b-a} \\ &\quad \oplus \Delta x_{k+b} \odot \Delta x_{k+2b-a} \end{aligned} \quad (13)$$

If  $\Delta x_{k+2b-a} \in R$ , then  $\Delta d_{k+b-a} = x_{k+b} = \Delta d_k$ ; otherwise,  $\Delta d_{k+b-a} = 0$  holds with probability 1 (independent of  $\Delta d_k$ ).

2.  $\Delta x_{k+a} = 0$ ,  $\Delta x_{k+b} = 1$ . In this case,  $\Delta d_k = x_{k+a}$ . If there exists some other bit  $\Delta d'_k \in R'_d$  such that  $\Delta d'_k$  is dependent of  $\Delta d_k$ , then  $k' \equiv k + a - b \pmod n$ , since  $\Delta d_{k+a-b}$  is the only bit which *may be* affected by  $x_{k+a}$  except for  $\Delta d_k$  from Lemma 1.

$$\begin{aligned} \Delta d_{k+a-b} &= \Delta x_{k+a} \odot x_{k+2a-b} \oplus x_{k+a} \odot \Delta x_{k+2a-b} \\ &\quad \oplus \Delta x_{k+a} \odot \Delta x_{k+2a-b} \end{aligned} \quad (14)$$

If  $\Delta x_{k+2a-b} \in R$ , then  $\Delta d_{k+a-b} = x_{k+a} = \Delta d_k$ ; otherwise,  $\Delta d_{k+a-b} = 0$  holds with probability 1 (independent of  $\Delta d_k$ ).

3.  $\Delta x_{k+a} = 1, \Delta x_{k+b} = 1$ . In this case,  $\Delta d_k = (x_{k+a} \oplus x_{k+b}) \odot 1 \oplus 1$ . From Lemma 1, the only other bit which may be affected by  $x_{k+a}$  is  $\Delta d_{k+a-b}$ . Specifically, the following equation holds if  $\Delta x_{k+2a-b} \in R$ .

$$\Delta d_{k+a-b} = (x_{k+2a-b} \oplus x_{k+a}) \odot 1 \oplus 1 \quad (15)$$

From Lemma 1, the only other bit which may be affected by  $x_{k+b}$  is  $\Delta d_{k+b-a}$ . Specifically, the following equation holds if  $\Delta x_{k+2b-a} \in R$ .

$$\Delta d_{k+b-a} = (x_{k+b} \oplus x_{k+2b-a}) \odot 1 \oplus 1 \quad (16)$$

It is obvious that  $\Delta d_k$  can be dependent of other bit(s) in  $R'_d$ , only in the case that  $\Delta x_{k+2a-b}, \Delta x_{k+2b-a} \in R$ . However, since  $\Delta d_{k+b-a}$  and  $\Delta d_{k+a-b}$  introduce the new bits (variants) of  $x_{k+2a-b}$  and  $x_{k+2b-a}$  respectively, we should involve more elements in  $R'_d$  to reduce the effects of  $x_{k+2a-b}$  and  $x_{k+2b-a}$ . Again from Lemma 1, the only other bit affected by  $x_{k+2a-b}$  ( $x_{k+2b-a}$ ) is  $\Delta d_{k+2a-2b} = (x_{k+3a-2b} \oplus x_{k+2a-b}) \odot 1 \oplus 1$  ( $\Delta d_{k+2b-2a} = (x_{k+2b-a} \oplus x_{k+3b-2a}) \odot 1 \oplus 1$ ) on condition that  $\Delta x_{k+3a-2b} \in R$  ( $\Delta x_{k+3b-2a} \in R$ ).

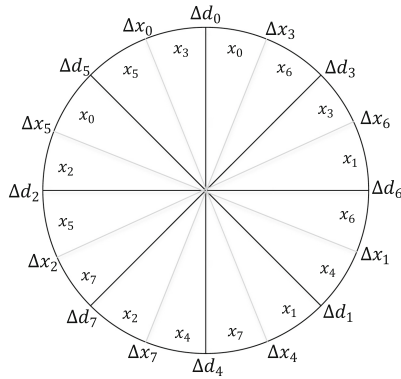
Thus, in order to eliminate the effects of  $x_{k+2a-b}$  and  $x_{k+2b-a}$ , the *only* choice (from Lemma 1) is involving the new bits of  $\Delta d_{k+2a-2b}$  and  $\Delta d_{k+2b-2a}$ , which can indeed eliminate  $x_{k+2a-b}$  and  $x_{k+2b-a}$  however introduce two new variants of  $x_{k+3a-2b}$  and  $x_{k+3b-2a}$ . Under the condition  $\gcd(n, a-b) = 1$ , this *eliminating-while-introducing* process will succeed iff.  $|R| = n$ , and the probability of each possible value of  $\Delta d$  is  $2^{-(n-1)}$  which coincides with the result in [11]. On the other hand,  $\Delta d_k = (x_{k+a} \oplus x_{k+b}) \odot 1 \oplus 1$  is independent of other bits in  $R'_d$  when  $|R| < n$ .

□

For a better understanding, we give an example with  $(n, a, b) = (8, 0, 3)$  as shown in Fig. 3. Assume that  $\Delta x_0 = 1, \Delta x_3 = 1$ . Only in the case where all input difference bits are active, can  $\Delta d_0$  be dependent of other bits in  $\Delta d$ , namely  $\Delta d_0 = \Delta d_1 \oplus \dots \oplus \Delta d_7$ .

1.  $\Delta d_0 = (x_0 \oplus x_3) \odot 1 \oplus 1$
2.  $\Delta d_5 = (x_0 \oplus x_5) \odot 1 \oplus 1$ , when  $\Delta x_5 = 1$ ;  $\Delta d_3 = (x_6 \oplus x_3) \odot 1 \oplus 1$ , when  $\Delta x_6 = 1$
3.  $\Delta d_2 = (x_2 \oplus x_5) \odot 1 \oplus 1$ , when  $\Delta x_2 = 1$ ;  $\Delta d_6 = (x_6 \oplus x_1) \odot 1 \oplus 1$ , when  $\Delta x_1 = 1$
4.  $\Delta d_7 = (x_2 \oplus x_7) \odot 1 \oplus 1$ , when  $\Delta x_7 = 1$ ;  $\Delta d_1 = (x_1 \oplus x_4) \odot 1 \oplus 1$ , when  $\Delta x_4 = 1$
5.  $\Delta d_4 = (x_4 \oplus x_7) \odot 1 \oplus 1$

Essentially, given  $\gcd(n, a-b) = 1$ , there is only one cycle  $\left( \begin{pmatrix} 3 & 6 & 1 & 4 & 7 & 2 & 5 & 0 \\ 6 & 1 & 4 & 7 & 2 & 5 & 0 & 3 \end{pmatrix} \right)$  in this example). More generally, when  $\gcd(n, a-b) = t$ , there will be  $t$  cycles, and this in some way explains the rationalities of such requirement  $\gcd(n, a-b) = 1$ .



**Fig. 3.** The affected relationship between input and output difference bits of ROTATION-AND.

## References

1. Abdelraheem, M.A., Alizadeh, J., Alkhzaimi, H.A., Aref, M.R., Bagheri, N., Gauravaram, P.: Improved linear cryptanalysis of reduced-round SIMON-32 and SIMON-48. In: Biryukov, A., Goyal, V. (eds.) INDOCRYPT 2015. LNCS, vol. 9462, pp. 153–179. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-26617-6\\_9](https://doi.org/10.1007/978-3-319-26617-6_9)
2. Abed, F., List, E., Lucks, S., Wenzel, J.: Differential cryptanalysis of round-reduced SIMON and SPECK. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 525–545. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46706-0\\_27](https://doi.org/10.1007/978-3-662-46706-0_27)
3. Alkhzaimi, H.A., Lauridsen, M.M.: Cryptanalysis of the SIMON Family of Block Ciphers. Cryptology ePrint Archive, Report 2013/543 (2013). <https://eprint.iacr.org/2013/543>
4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The Simon and Speck Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404 (2013). <https://eprint.iacr.org/2013/404>
5. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: Notes on the design and analysis of SIMON and SPECK. Cryptology ePrint Archive, Report 2017/560 (2017). <https://eprint.iacr.org/2017/560>
6. Biryukov, A., Roy, A., Velichkov, V.: Differential analysis of block ciphers SIMON and SPECK. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 546–570. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46706-0\\_28](https://doi.org/10.1007/978-3-662-46706-0_28)
7. Biryukov, A., Velichkov, V.: Automatic search for differential trails in ARX ciphers. In: Benaloh, J. (ed.) CT-RSA 2014. LNCS, vol. 8366, pp. 227–250. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-04852-9\\_12](https://doi.org/10.1007/978-3-319-04852-9_12)
8. Chen, Z., Wang, N., Wang, X.: Impossible Differential Cryptanalysis of Reduced Round SIMON. Cryptology ePrint Archive, Report 2015/286 (2015). <https://eprint.iacr.org/2015/286>
9. Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 161–185. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-47989-6\\_8](https://doi.org/10.1007/978-3-662-47989-6_8)

10. Kondo, K., Sasaki, Y., Iwata, T.: On the design rationale of SIMON block cipher: integral attacks and impossible differential attacks against SIMON variants. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 518–536. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-39555-5\\_28](https://doi.org/10.1007/978-3-319-39555-5_28)
11. Liu, Z., Li, Y., Wang, M.: Optimal differential trails in SIMON-like ciphers. *IACR Trans. Symmetric Cryptol.* **2017**, 358–379 (2017). <https://doi.org/10.13154/tosc.v2017.i1.358-379>
12. Liu, Z., Li, Y., Wang, M.: The Security of SIMON-like Ciphers Against Linear Cryptanalysis. Cryptology ePrint Archive, Report 2017/576 (2017). <https://eprint.iacr.org/2017/576>
13. Matsui, M.: On correlation between the order of S-boxes and the strength of DES. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 366–375. Springer, Heidelberg (1995). <https://doi.org/10.1007/BFb0053451>
14. Shi, D., Hu, L., Sun, S., Song, L., Qiao, K., Ma, X.: Improved linear (hull) cryptanalysis of round-reduced versions of SIMON. **60**, 39101, May 2016. <https://doi.org/10.1007/s11432-015-0007-1>
15. Sun, S., et al.: Towards Finding the Best Characteristics of Some Bit-oriented Block Ciphers and Automatic Enumeration of (Related-key) Differential and Linear Characteristics with Predefined Properties. Cryptology ePrint Archive, Report 2014/747 (2014). <https://eprint.iacr.org/2014/747>
16. Sun, S., et al.: Constructing Mixed-integer Programming Models whose Feasible Region is Exactly the Set of All Valid Differential Characteristics of SIMON (2015). <https://eprint.iacr.org/2015/122>
17. Sun, S., Hu, L., Wang, P., Qiao, K., Ma, X., Song, L.: Automatic security evaluation and (related-key) differential characteristic search: application to SIMON, PRESENT, LBlock, DES(L) and other bit-oriented block ciphers. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 158–178. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-45611-8\\_9](https://doi.org/10.1007/978-3-662-45611-8_9)
18. Todo, Y.: Division property: efficient method to estimate upper bound of algebraic degree. In: Phan, R.C.-W., Yung, M. (eds.) Mycrypt 2016. LNCS, vol. 10311, pp. 553–571. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-61273-7\\_30](https://doi.org/10.1007/978-3-319-61273-7_30)
19. Todo, Y., Morii, M.: Bit-based division property and application to SIMON family. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 357–377. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-52993-5\\_18](https://doi.org/10.1007/978-3-662-52993-5_18)
20. Wang, Q., Liu, Z., Varıcı, K., Sasaki, Y., Rijmen, V., Todo, Y.: Cryptanalysis of reduced-round SIMON32 and SIMON48. In: Meier, W., Mukhopadhyay, D. (eds.) INDOCRYPT 2014. LNCS, vol. 8885, pp. 143–160. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-13039-2\\_9](https://doi.org/10.1007/978-3-319-13039-2_9)
21. Xiang, Z., Zhang, W., Bao, Z., Lin, D.: Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 648–678. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53887-6\\_24](https://doi.org/10.1007/978-3-662-53887-6_24)
22. Yang, G., Zhu, B., Suder, V., Aagaard, M.D., Gong, G.: The Simeck family of lightweight block ciphers. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 307–329. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48324-4\\_16](https://doi.org/10.1007/978-3-662-48324-4_16)



# Linear Cryptanalysis of Reduced-Round Speck with a Heuristic Approach: Automatic Search for Linear Trails

Daniël Bodden<sup>(✉)</sup>

imec-COSIC, KU Leuven, Leuven, Belgium  
dbodden@esat.kuleuven.be

**Abstract.** Previous research on linear cryptanalysis with Speck has proved that good linear trails and a meaningful distinguisher for variants of Speck can be found. In this paper we use two different linear approximations of modular addition to search for even better linear trails. Also, we have added a heuristic to search for large bias approximations for the state conversion approach. We will explain how the automatic search works and discuss its performance. Finally we illustrate that linear approximations with large bias exist in variants of Speck.

**Keywords:** Speck · Lightweight ciphers · Heuristic  
Automatic search · Linear cryptanalysis

## 1 Introduction

In the last couple of years a surge of new block cipher designs has urged the need for cryptanalysts to scrutinize their security. Lightweight ciphers are designed for resource-constrained devices. Designing ciphers for these devices means that one has to make design trade-off's keeping in mind the limitations that such an environment presents, such as limited memory, restricted computational and energy resources, etc.

A popular construction for block ciphers is Addition Rotation and XOR, abbreviated as ARX. In this construction a block is split into 2 or more words, which are then added, XORed and rotated by the round function. The popularity of the ARX construction stems from its good performance in software. Confusion is achieved by using modular addition in the round function, whereas diffusion is achieved by using bitwise rotation and xor.

In this paper we try to analyse the security of Speck, a block cipher that has been published by the NSA in 2013 [6]. Speck is a lightweight block cipher designed to achieve good performance in software. The block consists of two words, each 16/24/32/48 or 64 bit, which are processed a number of times by the round function. At the end of the last round a ciphertext is obtained. Speck comes in different variants for which different security and performance levels are provided, see Table 1.



Since the publication of the cipher in 2013, several papers have analysed its security [1, 4, 10]. The best published attacks on Speck are differential cryptanalytic attacks [10]. However, recently linear cryptanalysis applied against Speck has proved to deliver good linear approximations that can be used for testing the security of the cipher [14, 21]. Linear cryptanalysis is a known plaintext attack developed in 1993 by Matsui [15]. When using this method the adversary searches for possible correlations between bits of the input and bits of the output. Once such a correlation is found, the known plaintexts and ciphertexts can be used to recover the secret key.

This paper applies two known approaches to linear approximate modular addition, using them separate and combined. In addition we introduce a heuristic for searching linear trails using state conversion.

The rest of this paper is structured as follows. In Sect. 2 we describe Speck. In Sect. 3 we discuss shortly the related work. In Sect. 4 we explain the 2 approaches of linear approximating modular addition. In Sect. 5 we explain the automatic search method that has been used to obtain linear trails for Speck. In Sect. 6 we present the linear trails we found for Speck. Finally, in Sect. 7 we conclude this paper.

## 2 A Brief Description of Speck

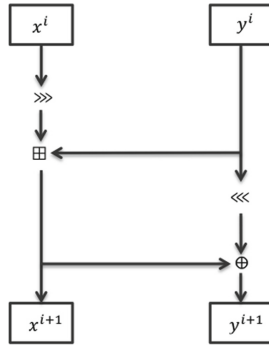
The cipher comes in several versions sharing the same Feistel structure and round function. The different parameters of all Speck versions are presented in Table 1.

The round function of Speck uses three operations:

- Modular addition,  $\boxplus$ .
- Left and right circular bit shifting,  $\alpha$  and  $\beta$ .
- Bitwise XOR,  $\oplus$ .

**Table 1.** Variants of the Speck family [6]

Block size $n$	Key size	Word size	$\alpha$	$\beta$	Rounds
32	64	16	7	2	22
48	72	24	8	3	22
	96				23
64	96	32	8	3	26
	128				27
96	96	48	8	3	28
	144				29
128	128	64	8	3	32
	192				33
	256				34



**Fig. 1.** Speck round function

The output of the round function is:

$$\begin{aligned} x^{i+1} &= (x^i \ggg \alpha) \boxplus (y^i) \\ y^{i+1} &= ((x^i \ggg \alpha) \boxplus (y^i)) \oplus (y^i \lll \beta) \end{aligned}$$

The output words  $x^{i+1}$  and  $y^{i+1}$  are the input words for the next round. In Fig. 1 the round function of Speck is shown.

### 3 Related Work

Since the publication of Speck there has been a fair amount of analysing done on the security and performance of the cipher. The best attacks to date are of the family of differential cryptanalysis [1, 4, 10]. Differential cryptanalysis has been developed by Biham and Shamir in 1990 [5]. When using differential cryptanalysis the adversary investigates how differences in the input affect the output, trying to discover non-random behaviour. Results of previous differential attacks concerning Speck are presented in Table 2.

From the results of differential cryptanalysis on Speck we gather that these methods are successful in attacking a large number of rounds in the chosen plaintext model. Looking at the design of Speck we might expect that linear cryptanalysis might be effective as well. This has been demonstrated by recent research on linear cryptanalysis on variants of Speck [14, 21]. Results of previous linear attacks concerning Speck are presented in Table 3.

The results of linear cryptanalysis on Speck are at the moment not better than the differential attacks. Yet, they are not far off the mark. Considering the recent gains made with linear cryptanalysis on Speck, this paper complements previous work by evaluating the resistance of Speck against linear cryptanalysis in the known plaintext model. The contribution of this paper is to introduce a heuristic search method for linear trails that uses several approaches to approximate modular addition [7, 19] to find long linear trails in a short search time in a relevant search space.

**Table 2.** Trail lengths of differential cryptanalysis attacks concerning Speck

Variant $n$	Distinguished rounds/ Total rounds	Probability	Reference
32	9/22	$2^{-30}$	[11]
	10/22	$2^{-30}$	[10]
48	11/22	$2^{-45}$	[11]
	11/22	$2^{-40}$	[10]
64	15/22	$2^{-62}$	[11]
	15/26	$2^{-60}$	[10]
96	14/28	$2^{-84}$	[10]
	16/28	$2^{-87}$	[11]
128	15/32	$2^{-112}$	[10]
	19/32	$2^{-119}$	[11]

**Table 3.** Trail length of linear cryptanalysis attacks concerning Speck

Variant $n$	Distinguished rounds/ Total rounds	Bias	Reference
32	9/22	$2^{-14}$	[21]
	9/22	$2^{-14}$	[11]
	9/22	$2^{-14}$	[18]
48	9/22	$2^{-20}$	[21]
	10/22	$2^{-22}$	[11]
	10/22	$2^{-22}$	[18]
64	12/26	$2^{-31}$	[21]
	12/26 <sup>a</sup>	$2^{-30}$	[18]
	13/26	$2^{-30}$	[11]
96	6/28	$2^{-11}$	[21]
	15/28	$2^{-45}$	[18]
	15/28	$2^{-45}$	[11]
128	6/32	$2^{-11}$	[21]
	16/32	$2^{-61}$	[18]
	16/32	$2^{-58}$	[11]

<sup>a</sup> In discussion with the authors [14] it was established that the length of the linear trail is correct, but that the input of the last two rounds should be for  $\lambda_{x,i} = 0x41000040$ ,  $\lambda_{y,i} = 0x4c000040$  and  $\lambda_{x,i} = 0x2698200$ ,  $\lambda_{u,i} = 0x62080200$  with the same cost as in the paper.

## 4 Linear Approximation of Modular Addition

Linear cryptanalysis is a powerful cryptanalytic method with regard to cryptanalysis of block ciphers. When using linear cryptanalysis, an adversary tries to find a linear expression that approximates a non-linear function with a probability different than  $\frac{1}{2}$  [15]. The approximation has the form:

$$P_i \oplus \dots \oplus P_j \oplus C_k \dots \oplus C_l = K_m \dots \oplus K_n \quad (1)$$

with  $P_i \dots P_j$  being bits from the plaintext,  $C_k \dots C_l$  bits from the ciphertext and  $K_m \dots K_n$  bits from the key. Once a good approximation is found the adversary can retrieve one key bit. To combine linear approximations we use the Piling Up Lemma [15]. This will tell us the bias of the combined approximation. For two approximations, with  $\epsilon_1$  and  $\epsilon_2$  as their respective biases, combining them gives an overall bias:

$$\epsilon = 2 \cdot \epsilon_1 \epsilon_2 \quad (2)$$

This can be generalized for  $\sigma$  approximations:

$$\epsilon = 2^{\sigma-1} \cdot \prod_{i=1}^{\sigma} \epsilon_i \quad (3)$$

The combined approximations can be used to form a linear distinguisher  $\zeta$  for the cipher. This  $\zeta$  can be used to detect non-random behaviour in supposed to be random signals. Meaning that  $\zeta$  can be used to detect if a certain cipher is being used. When a good  $\zeta$  is found, the input bits are masked by  $\{\lambda_{x^1}; \lambda_{y^1}\}$ , and the output bits are masked using  $\{\lambda_{x^r}; \lambda_{y^r}\}$ , resulting in:

$$T = \# \{ \lambda_{x^1} \cdot x \oplus \lambda_{y^1} \cdot y \oplus \lambda_{x^r} \cdot x \oplus \lambda_{y^r} \cdot y = 0 \} \quad (4)$$

with  $r$  being the length of  $\zeta$ ,  $T$  the counter and  $\cdot$  being the dot product. This should be repeated for at least  $\epsilon^{-2}$  messages. If  $T$  is around:

$$\epsilon^{-2} \cdot \left( \frac{1}{2} \pm \epsilon \right) \quad (5)$$

we have discovered the cipher in a signal.

### 4.1 Approach 1

The modular addition operation, which is used as the non-linear operation of Speck, consists of an xor and a carry. A chain of carries means using the previous carry in the current operation. This makes the behaviour of modular addition non-linear. To approximate the behaviour of Speck, we have to deal with modular addition in such a way that takes out this non-linear behaviour. The property of modular addition that we use is exploring the correlation between two neighbouring bits. Suppose we have 3 words  $x$ ,  $y$  and  $z$  with  $z = x \boxplus y$  and  $x$ ,  $y$

$\in \{0, 1\}^n$ . According to Cho and Pieprzyk [7], each single  $z_{(i)}$  bit written as function of  $x_{(i)}, \dots, x_{(0)}$  and  $y_{(i)}, \dots, y_{(0)}$  bits can be expressed as:

$$z_{(i)} = x_{(i)} \oplus y_{(i)} \oplus x_{(i-1)} \cdot y_{(i-1)} \oplus \sum_{j=0}^{i-2} x_{(j)} \cdot y_{(j)} \prod_{k=j+1}^{i-1} [x_{(k)} \oplus y_{(k)}], i = 1, 2, \dots, n-1. \tag{6}$$

with  $x_i$  and  $y_i$  representing 1 bit each and  $z_{(0)} = x_{(0)} \oplus y_{(0)}$ . The carry  $R(x, y)$  of the modular addition is represented by:

$$R(x, y)_{(i)} = x_{(i)} \cdot y_{(i)} \oplus \sum_{j=0}^{i-1} x_{(j)} \cdot y_{(j)} \prod_{k=j+1}^i [x_{(k)} \oplus y_{(k)}], i = 0, 1, \dots, n-2. \tag{7}$$

Then we can also write  $z_{(i)} = x_{(i)} \oplus y_{(i)} \oplus R(x, y)_{(i-1)}$  for  $i = 1, \dots, (n-1)$ . We use a property of modular addition mentioned in another paper of Cho and Pieprzyk [8] that removes the carry chain from Eq. 6. This property uses consecutive bits as underlying requirement to keep the probability for the approximation constant. The parity of two consecutive bits can be approximated as:

$$z_{(i)} \oplus z_{(i-1)} = x_{(i)} \oplus x_{(i-1)} \oplus y_{(i)} \oplus y_{(i-1)}, \quad \Pr[R_i(x, y) \oplus R_{i-1}(x, y) = 0] = \frac{3}{4}. \tag{8}$$

hence, if a mask  $\lambda$  contains only two consecutive bits, we can say:

$$\Pr[\lambda \cdot (x \boxplus y) = \lambda \cdot (x \oplus y)] = \frac{3}{4}. \tag{9}$$

This expression says that using a mask  $\lambda$  to mask out the bits we want to throw away and keep the bits we are interested in, the consecutive bits, and we linearise the left side of the expression by replacing it with the right side. This approximation holds with probability  $\frac{3}{4}$  only for 2 consecutive bits. For an ARX construction, Eq. 9 remains valid as long as the following two conditions are avoided:

1. Bitwise rotation moves a pair of approximated bits to the MSB (most significant bit position) and LSB (least significant bit position) hence not adhering to the Cho and Pieprzyk framework, take for instance the following example  $0000000011000000 \ggg 7 = 1000000000000001$ .
2. Or, xor breaks the consecutive bits input into single bits output, hence not adhering to the Cho and Pieprzyk framework, take for instance the following example  $0000000011000000 \oplus 0000000110000000 = 0000000101000000$ .

### 4.2 Approach 2

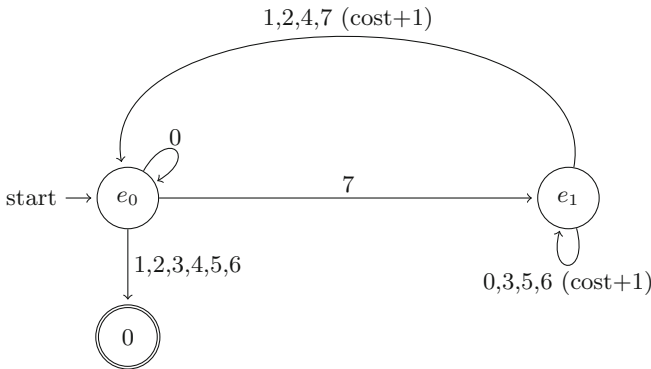
Another approach to linearly approximate modular addition is using the research of Wallén [19, 20]. Later Nyberg and Wallén used this approach to search for linear approximations of the stream cypher Snow2.0 [17]. Recently, Lui and Yao applied this approach to variants of Speck [14, 21]. Our approach 2 for variants of Speck is based on these works.

Let  $u$  be the output mask of the modular addition and  $v, w$  be the input masks. The correlation is:

$$c(u, v, w) \triangleq 2 \Pr(u \cdot (Z1 \boxplus Z2) \oplus v \cdot Z1 \oplus w \cdot Z2 = 0) - 1 . \quad (10)$$

with  $Z1$  and  $Z2$  to be independent uniform distributed random variables. This expression says that a modular addition can be approximated by replacing the modular addition with 3 variables  $u, v$  and  $w$ . Several follow up approaches exist how to search for valid  $u, v$  and  $w$ . One possible approach would be to enumerate all possible  $u, v, w$  masks and use branch—and bound techniques to find a good linear approximation [19–21]. An alternative approach is to produce a partial linear mask table by using state conversion [18]. In this paper we use an adjusted version of the second approach to search for valid  $u, v$  and  $w$  masks that deliver good linear approximations with large bias.

We use state conversion [17, 19] to compute the correlation from Eq. 10. The state conversion can be computed using the standard automaton, illustrated in Fig. 2.



**Fig. 2.** The standard automaton, graphically illustrated

This standard automaton, a state machine, checks the validity of each  $u_i, v_i, w_i$ . Starting at the LSB or MSB, the standard automaton combines the bits  $u_i, v_i, w_i$  and checks the current state, iterating over the whole mask. The cost of the approximation is in this approach also expressed in bias. The bias will increase by one every time the state passes  $e_1 \rightarrow e_1$  or  $e_1 \rightarrow e_0$ . Free passes are  $e_0 \rightarrow e_0$  or  $e_0 \rightarrow e_1$ . The approximation of the modular addition fails if state transition goes  $e_1 \rightarrow 0$ .

For a certain iteration we have  $u, w$  and  $v$ . The standard automaton starts in state  $e_0$  with checking the state conversion of the most significant bit or least significant bit of  $u_i, w_i$  and  $v_i$ , taking each a bit of these masks and combines them into  $\zeta_i$ . According standard automaton, the following state conversion can happen:

- if  $e_0$  and  $\zeta_i \in \{1, 2, 3, 4, 5, 6\}$ , the state is then  $O$
- if  $e_0$  and  $\zeta_i \in \{0\}$  counter no addition, the states stays  $e_0$
- if  $e_0$  and  $\zeta_i \in \{7\}$  counter no addition, the states changes to  $e_1$
- if  $e_1$  and  $\zeta_i \in \{0, 3, 5, 6\}$  counter +1, the states stays  $e_1$
- if  $e_1$  and  $\zeta_i \in \{1, 2, 4, 7\}$  counter +1, the state changes to  $e_0$ .

The above mentioned state conversion goes on until  $n$ . At the end of the Automaton the counter represents the bias of the state conversion for  $n$  over  $\zeta_i$  concerning  $u$ ,  $w$  and  $v$ .

## 5 Automated Search for Linear Trails

This paper is not the first to introduce a automated search for primitive characteristics [9,12,13]. Yet, there are important differences. First, Leurent’s research focuses on automated differential cryptanalysis on ARX ciphers. Second, Dobraunig et al. focuses on linear cryptanalysis approaches for substitution-permutation networks. The automated search introduced in this paper focuses on linear cryptanalysis of ARX ciphers with 2 main parts: combining several linear cryptanalysis approaches and introducing an heuristic approach.

In order to automate the search for good linear trails, we have designed a parallel algorithm. The algorithm is implemented using a parallel programming language called OpenCL [16,18]. Unique to OpenCL is that it takes an abstraction of the hardware layer and is vendor independent. A particular asset using this parallel programming language is that it dynamically use the available resources to maximise computing in parallel mode.

The configuration on which the computation has run was a 40 core Intel Xeon machine with a clock speed of 3.10 GHz, exclusively using CPU’s. The time it took to compute all pairs of consecutive bits for the 32 bit version was less than 0.01 s to a few minutes for the largest variant of Speck.

### 5.1 Combined Approach

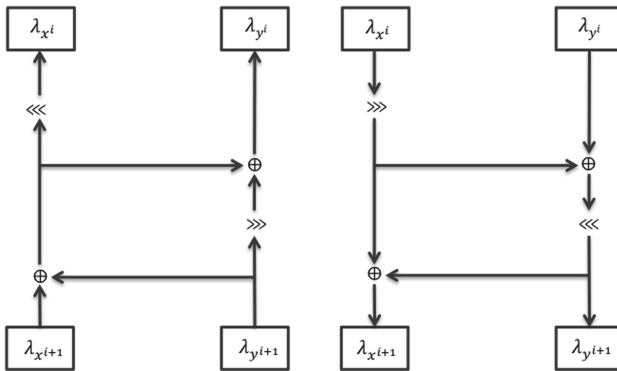
Approach 1 has been introduced and later implemented in a automated search by Ashur et al. [2,3]. In this paper we work further on that work combined with approach 2 introduced by Yao et al. and improved by Liu et al. [14,21] that we fit into a heuristic.

Combining both approaches for searching good linear trails makes sense. The first approach iterates over a small space by checking all possible pairs of consecutive bits over the length of the mask. Yet, this method has a lot of restrictions under which the search for linear trail breaks. Whereas, the second approach gives more freedom, i.e. an extra variable for every modular addition, meaning  $2^{\frac{n}{2}}$  per modular addition more space to search for a long trail. On the contrary, the downside is that this method cannot run on the complete available search space, as the number of mask combinations to search from is to intractable for present-day computers.

The combined approach starts first using the first approach until the computation breaks and then switches to the second approach. We started by searching in the space of all possible pairs of consecutive bits over the length of a mask (e.g. starting with one pair of two consecutive bits up to  $\frac{n}{2}$  pairs of two consecutive bits). Practically, this means applying the following steps on the round function of a cipher:

1. Replace every modular addition with XOR.
2. Following two actions have to be done in parallel
  - (a) Replace branch with XOR
  - (b) Replace prior existing XORs with branch
3. Reverse the direction of the horizontal arrows

We have done this analysis in the forward direction, backward direction and combining forward and backward together. The forward direction is exactly the same as the original cipher, whereas in the backward direction the vertical arrows and the circular rotation are reversed. In Fig. 3 we show a simplified cipher using the framework of Cho and Pieperzyk for searching linear trails, respectively in the backward and in the forward direction [2].



**Fig. 3.** Approach 1: a transformation of the Speck one round function (on the left (right) to search for linear 1 round approximation in the backward (forward) direction)

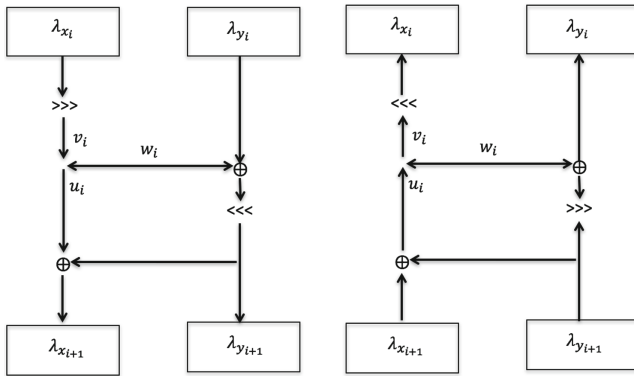
We have iterated over all pairs of consecutive bits until one of two stop conditions have been met. The first stop condition is when a mask containing non-consecutive bits is encountered in the round function entering into the modular addition. The other condition upon which the computation will be stopped is when the counter exceeds minimum bias.

When the first stop condition is triggered using the first approach, the combined approach switches to the second approach. The search space for the second approach is not limited to only consecutive bits. The following steps are rule of thumb to convert the round function of the cipher for the second approach:



1. Replace every modular addition with a branch consisting of  $v$ ,  $w$  and  $u$ , with  $w$  having a double sided arrows. This branch will not be changed later to xor.
2. Following two actions have to be done in parallel
  - (a) Replace branch with XOR
  - (b) Replace prior existing XORs with branch
3. Reverse the direction of the horizontal arrows

This approach follows the same set up as the first approach, a forward direction, a backward direction and a combination of the both. When working exclusively with approach 2, finding linear trails in both direction makes no difference. However with the first approach it does. As the second approach starts at the point the first approach stops, the second approach will go in the direction that has been started during the computation for the first approach. In Fig. 4 we show a simplified cipher using the theory of Wallén for searching linear trails, respectively in the backward and in the forward direction [20].



**Fig. 4.** Approach 2: a transformation of the Speck one round function (on the left (right) to search for linear 1 round approximation in the backward (forward) direction)

In order to find approximations with large bias we restrict the allowable bias to  $T \leq \frac{n}{2} + 1$ . The bias for one pair of bits is  $\Pr[R_i(x, y) \oplus R_{i-1}(x, y) = 0] - \frac{1}{2} = \frac{1}{4}$ . Generalized to  $\ell$  pairs of bits the bias can be calculated using the Piling Up Lemma:

$$2^{\ell-1} \cdot (1 - \Pr[R_i(x, y) \oplus R_{i-1}(x, y) = 0])^\ell. \tag{11}$$

with values for  $\ell = 1, \dots, \frac{n}{2}$ . The bias is calculated with the value obtained by the hamming weight of the mask passing the modular addition in the round function divided by 2, multiplied by the bias of the previous rounds, calculated on the approximated modular addition  $\lambda \cdot (x \oplus y)$ .

### 5.2 Heuristic

Not only both the approaches are combined, but we also introduce a fast heuristic search for masks in approach 2 to speed up the switch from approach 1 to 2.

Namely, by finding an  $u$ ,  $v$  and  $w$  that is valid for approach 2, given the last correct output of the linear trail that broke with approach 1. An approach would be to just iterate over all possibilities of  $w_i$  and  $u_i$  or  $v_i$ . However, this approach is slow and for the larger impossible to check all possible masks. To solve this difficulty we use the Automaton to search only for fixes for the 2 dependent mask, for the forward direction  $w$  and  $u$ , that have a large bias. The heuristic works as follows for the forward direction:

1. When consecutive bits check fails, use the heuristic
2. Given  $v$ , choose a  $w$  and  $u$  as follows
  - (a) if (state 0 AND bit 0 for  $v_i$ )  $\Rightarrow w_i = 0, u_i = 0$
  - (b) else if (state 0 AND bit 1 for  $v_i$ )  $\Rightarrow w_i = 1, u_i = 1$
  - (c) else if (state 1 AND bit 0 for  $v_i$ )  $\Rightarrow w_i = 0, u_i = 1, \text{cost} + 1$
  - (d) else if (state 1 AND bit 1 for  $v_i$ )  $\Rightarrow w_i = 1, u_i = 1, \text{cost} + 1$
3. Result is a valid  $w, u, v$  for the Automaton with minimal cost. Perform Automaton and calculate new bias

The heuristic also works in the backward direction. Instead of  $v$ , now  $u$  is given. The only change compared to the explanation of the heuristic in the forward direction, is to interchange  $v$  and  $u$  from position. In Fig. 5 we illustrate how the heuristic searches for a fix with a low cost.

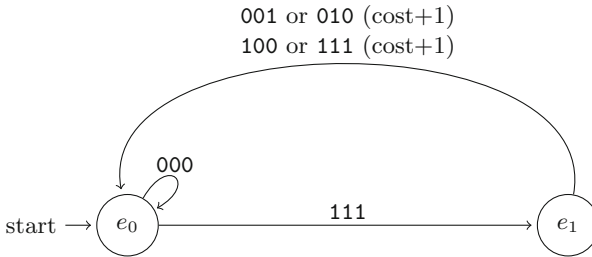


Fig. 5. Heuristic automaton: finding  $u, v$  and  $w$  with a low cost

### 5.3 Searching in a Smaller Relevant Space

A final improvement was to use the hamming weight for filtering the input space of approach 1 on masks that had a certain hamming weight, depending on the  $n$  blocksize of the Speck variant. After analysing several trails, by branching every trail, we found that long linear trails had a low hamming weight. In the long linear trails we never saw a hamming weight larger than 12. The improvement helped to reduced the search time for the larger variants of Speck.

## 6 Linear Trails of the Round Function of Speck

Linear cryptanalysis relies on collecting a large amount of input and output pairs in order to verify whether the approximation has been satisfied or not. In this work we experimented with several variants of the approaches to linear approximate modular addition. We concluded the following experiments:

- approach 1 searching with pairs of consecutive bits limited to a certain hamming weight
- approach 2 with a heuristic searching all possible bits limited to a certain hamming weight
- combined approach with a heuristic searching all possible bits limited to a certain hamming weight

The overall best result for the several approaches are shown in Table 4. From the result in Table 4 we observe that both approach 2 and the combined approach find longer linear trails compared to approach 1. This observation is clear for all variants of Speck, except for the 64-bit variant. In the following sections we explain in more details these results and the differences between the results of the approaches.

**Table 4.** Best trail length of the different approaches

Variant $n$	Approach 1 Distinguished rounds/ Total rounds	Approach 2 Distinguished rounds/ Total rounds	Combined approach Distinguished rounds/ Total rounds
32	7/22	8/22	8/22
48	8/22	9/22	9/22
64	11/26	11/26	11/26
96	11/28	12/28	12/28
128	13/32	14/32	13/32

## 6.1 Results Approach 1

The search space for the first approach has been limited to start fixing one mask  $\lambda_{x^1}$  with one pair of consecutive bits (0x3, 0x6, ...) and keeping the other mask  $\lambda_{y^1}$  zero, checking how the masks evolve both in the forward and backward direction.

After receiving promising results for one pair we extended the computation to all possible pairs of two consecutive bits given a block size. We found out that the consecutive bits search space still wasted time on checking masks that would in this approach not result in long trails. This is because approach 1 uses hamming weight to calculate the cost for each pass through the linear approximated modular addition. After careful analysis of long trails we found out that there is a certain upperbound on the hamming weight of masks. Above this limit the trails stopped early because reaching the upperbound for the allowed bias to find good linear trails. The hamming weight upperbound depends largely on the variant of Speck that is employed for searching linear trails. By limiting the maximum allowable hamming weight for masks we could search in the more relevant search space, and by archiving this we found long linear trails in a short time. In Table 5 we show the best linear trails found for variants of Speck by approach 1. From the

**Table 5.** Approach 1: length linear trails for different variants of Speck

Variant $n$	Distinguished rounds/ Total rounds	Bias	Upperbound hamming weight	Mask searched	Search time in sec.
32	7/22	$2^{-14}$	2	$9^2$	0.0120
48	8/22	$2^{-22}$	2	$13^2$	0.0115
64	11/26	$2^{-32}$	2	$17^2$	0.1148
96	11/28	$2^{-42}$	2	$25^2$	0.0190
128	13/32	$2^{-58}$	2	$64^2$	0.0090

results of approach 1 we can derive that this method doesn't find the longest trails already discovered by other methods in the research community. However, this approach finds in a relatively short time trails that are not far off the mark of the longest linear trails.

We have looked further in to the causes why approach 1 doesn't find the longest trails of all existing methods to find linear trails. One of the possible causes are that approach 1 restricts too much the search space to only consecutive bits. Secondly, the method to linear approximate modular addition that is used in this approach puts heavy restrictions on the conditions that valid masks are found.

## 6.2 Results Approach 2

The second approach in this paper is based on the state machine. The search space we used was limited to all  $n$  bits with a certain upperbound hamming weight, that can be set freely. In order to overcome the time complexity problem that other papers hit upon [21], we introduce a heuristic to find good linear trails in a short search time. In Table 6 we show the best linear trails found for variants of Speck by approach 2.

**Table 6.** Approach 2: length linear trails for different variants of Speck

Variant $n$	Distinguished rounds/ Total rounds	Bias	Upper bound hamming weight	Mask searched	Search time in sec.
32	8/22	$2^{-16}$	2	$137^2$	0.5087
48	9/22	$2^{-23}$	2	$301^2$	0.5175
64	11/26	$2^{-32}$	2	$529^2$	21.0613
96	12/28	$2^{-47}$	2	$1177^2$	22.1236
128	14/32	$2^{-65}$	2	$2081^2$	7.8315

From the results of approach 2 we can conclude that we get mixed results compared to approach 1. For the 32, 48, 96 and 128 variant approach 2 finds

one more round, while for 64 the found trail length stays the same. A downside of approach 2 is the apparent longer search time. The search time for approach 2 heavily depends upon the upperbound that is set for the hamming weight on the input masks. A higher upperbound results in a larger search time and space. A larger search space has more possible masks to search for a long trail, yet has a longer search time. A way to counter the longer search time is to partition the search space in small chunks. The partial result of a chunk can give feedback on the current best result.

An interesting result from the experiments on this approach is that by changing the parameters for the heuristic the length of the trails differ. One can tune the heuristic parameters by changing the following values:

- if  $e_1$  and  $v_i == 0$ 
  - set current bit of  $u_i$  to 0 and set  $w_i$  to 1, or
  - set current bit of  $u_i$  to 1 and set  $w_i$  to 0
- if  $e_1$  and  $v_i == 1$ 
  - set current bit of  $u_i$  to 1 and set  $w_i$  to 1, or
  - set current bit of  $u_i$  to 0 and set  $w_i$  to 0

From running several experiments with different parameters for the heuristic on variants of Speck we found an indication that the following parameter setting finds the longest linear trails.

- if  $e_1$  and  $v_i == 0$ 
  - set current bit of  $u_i$  to 0 and set  $w_i$  to 1
- if  $e_1$  and  $v_i == 1$ 
  - set current bit of  $u_i$  to 1 and set  $w_i$  to 1

There is no clear indication why this particular parameter setting should be optimal.

### 6.3 Results Combined Approach

A last approach that we investigated in this paper is a combination of both previous mentioned approaches. From the first approach we expected to deliver reasonable long trails in a short time, with approach 2 picking up on the moment approach 1 fails and try to further extend linear trails. An overview of the obtained results from the experiments are illustrated in Table 7. The combined approach finds the same trails length for almost all variants of Speck. Yet, on the 32 variant of Speck one round less is found. An interesting observation is that for 32 much more masks has to be searched leading to a longer search time compared to approach 2 with the same trail length as the combined approach. In addition, the search performance of 32 to the other variants of Speck for the combined approach is significantly worse. The combined approach seems to work faster in terms of search performance compared to approach 2.

Looking into more detail to a complete linear trail we can investigate interesting aspects of its behaviour. In Table 8 a full linear trail is illustrated for

**Table 7.** Combined approach: linear trail length for different versions of Speck

Variant $n$	Distinguished rounds/ Total rounds	Bias	Upperbound hamming weight	Mask searched	Search time in sec.
32	8/22	$2^{-15}$	8	$39203^2$	113.1070
48	9/22	$2^{-23}$	2	$301^2$	2.8183
64	11/26	$2^{-31}$	2	$529^2$	10.0541
96	12/28	$2^{-48}$	2	$1177^2$	0.9943
128	13/32	$2^{-58}$	2	$2081^2$	2.5044

**Table 8.** Linear trail for Speck variant 48, using the combined approach

Trail length	Cost	Action log
1	3	$\lambda_{x^i}$ is 0x036300 and $\lambda_{y^i}$ is 0x00630f
2	2	$\lambda_{x^i}$ is 0x030003 and $\lambda_{y^i}$ is 0x030360
3	1	$\lambda_{x^i}$ is 0x030000 and $\lambda_{y^i}$ is 0x000300
4	1	$\lambda_{x^i}$ is 0x000300 and $\lambda_{y^i}$ is 0x000000
5	2	$\lambda_{x^i}$ is 0x00001b and $\lambda_{y^i}$ is 0x000018
6	3	$\lambda_{x^i}$ is 0xc300c0 and $\lambda_{y^i}$ is 0xd800c0
7	4	$\lambda_{x^i}$ is 0x06dd00 and $\lambda_{y^i}$ is 0xc61e00 breaks consecutive, fixed with heuristic $v_i$ is 0x0006dd, $u_i$ is 0x0006dd and $w_i$ is 0x0006df
8	4	$\lambda_{x^i}$ is 0x30c023 and $\lambda_{y^i}$ is 0x30c6fe breaks consecutive, fixed with heuristic $v_i$ is 0x2330c0, $u_i$ is 0x2330c0 and $w_i$ is 0x3330c0
9	4	$\lambda_{x^i}$ is 0x3c8130 and $\lambda_{y^i}$ is 0x1fb1f0 breaks consecutive, fixed with heuristic $v_i$ is 0x303c81, $u_i$ is 0x303c81 and $w_i$ is 0x303cc1
fails	6	$\lambda_{x^i}$ is 0x4c5508 and $\lambda_{y^i}$ is 0x7c6989 breaks consecutive, fixed with heuristic $v_i$ is 0x084c55, $u_i$ is 0x084c55 and $w_i$ is 0x0c6c7f, but iteration breaks because upper bound is reached $24 + 6 \Rightarrow 48/2$

Speck variant 48. In this linear trail is clear that as of trail length 7 the heuristic switches between the two approaches. In addition, this linear trail demonstrates that the using multiply approaches to linear approximate of modular addition can further extend linear trails.

Next, we illustrate how the heuristic switches to the second approach and searches for a fix that works with the Automaton. We take the example of forward iteration of round 7 for Speck variant 48.

The input of round 6  $\lambda_{x^i}$  is 0x30c023 and  $\lambda_{y^i}$  is 0x30c6fe. One of the conditions for the first approach is to check if  $0x30c023 \ggg \alpha$  consist exclusively of consecutive bits. We can check that this is not the case:

hexadecimal  $0x30c023 \ggg \alpha$  is 0x2330c0

binary  $001100001100000000100011 \ggg \alpha$  is  $001000110011000011000000$

**Table 9.** How the heuristic finds a valid  $w$  and  $v$  in the forward direction

State	Cost	Action log
0	0	1st first bit (msb)of $v_i$ is 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
0	0	2nd bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
1	0	3rd bit of $v_i$ is 1 and we are in state 0, heuristic picks 1 for both $w_i$ and $u_i$ , cost of zero and change state to 1
0	1	4th bit of $v_i$ is 0 and we are in state 1, heuristic picks 0 $u_i$ and 1 for $w_i$ , cost of one and change state to 0
0	0	5th bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
0	0	6th bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
1	0	7th bit of $v_i$ is 1 and we are in state 0, heuristic picks 1 for both $w_i$ and $u_i$ , cost is zero and change state to 1
0	1	8th bit of $v_i$ is 1 and we are in state 1, heuristic picks 1 for both $w_i$ and $u_i$ , cost is one and change state to 0
0	0	9th bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
0	0	10th bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
1	0	11th bit of $v_i$ is 1 and we are in state 0, heuristic picks 1 for both $w_i$ and $u_i$ , cost is zero and change state to 1
0	1	12th bit of $v_i$ is 1 and we are in state 1, heuristic picks 1 for both $w_i$ and $u_i$ , cost is one and change state to 0
0	0	13th bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
0	0	14th bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
0	0	15th bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
0	0	16th bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
1	0	17th bit of $v_i$ is 1 and we are in state 1, heuristic picks 1 for both $w_i$ and $u_i$ , cost is zero and change state to 1
0	1	18th bit of $v_i$ is 1 and we are in state 0, heuristic picks 1 for both $w_i$ and $u_i$ , cost is one and change state to 0
0	0	19th bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
0	0	20th bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
0	0	21st bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
0	0	22nd bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
0	0	23rd bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0
0	0	24th bit of $v_i$ is 0 and we are in state 0, heuristic picks 0 for both $w_i$ and $u_i$ , cost is zero and stay in state 0

From these results we can conclude the condition of consecutive bits doesn't hold any more. The heuristic then switches to the second approach and tries to find  $w$ ,  $u$  and  $v$  with a large bias. The heuristic tries to find a valid  $u_i$ ,  $v_i$  and  $w_i$ . The mask  $v_i$  is given as

`0x30c023`  $\ggg$   $\alpha$  is `0x2330c0` and in binary `001000110011000011000000`

iterating over all bits of  $v_i$  from the most significant bit position (e.g. left to right).

The detailed search of the heuristic to find a valid  $w_i$  and  $v_i$  with a large bias is illustrated in Table 9.

In the end the heuristic has found a valid  $w$  and  $u$ , respectively `0x2330c0` and `0x3330c0`, with a bias of 4.

## 7 Conclusions

In this paper we investigated the linear behaviour of Speck in the known plaintext model. We explained the theory behind our method, presented a new heuristic search method for finding linear trails and gave the linear trails for each version of Speck found by several approaches to approximate modular addition. We demonstrated that using a heuristic search method can significantly shorten the time find long linear trails in a relevant search space.

The analysis in this paper tested the strength of the Speck round function and demonstrated that the cipher offers sufficient resistance against linear cryptanalysis. The heuristic search method introduced in this paper uses a very greedy search approach. Further research could investigate how the heuristic can be changed to explore different possible optimal paths. For example, exploring every round which parameter setting for the heuristic is optimal for the masks under investigation.

In this paper we employed a state of the art parallel computing technique, called OpenCL, which supported the fast search in a relevant search space for linear trails. Several novel parallel search techniques have used, like partitioning the relevant search space in optimal chunks for the hardware used. Another, heuristic search technique used in this paper is the use of lookup tables for creating the relevant search space up to the hamming weight upperbound. In future research the employed parallel computing technique could further be improved. One improvement could be to limit the computing space to the newly added search space. For example, if in the previous experiment the hamming weight upperbound was set to  $n - 1$  and the current upperbound is set to  $n$ , one could compute all masks combinations again. Which will mean that a large part of the search space is searched again without changing any parameters and also will result in a longer search time. By only computing the new search space, instead of all previously computed space of lower hamming weight masks there will be a significant reduction in the time needed to find linear trails.

In future research better heuristics can be investigated to find good linear trails. We believe more research into heuristic search in linear cryptanalysis can further improve the length of linear trails.



**Acknowledgments.** This research was partially supported by the Research Fund of the KU Leuven, grant C16/18/004.

## References

1. Abed, F., List, E., Lucks, S., Wenzel, J.: Differential cryptanalysis of round-reduced SIMON and SPECK. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 525–545. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46706-0\\_27](https://doi.org/10.1007/978-3-662-46706-0_27)
2. Ashur, T., Dunkelman, O.: Linear analysis of reduced-round CubeHash. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 462–478. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-21554-4\\_27](https://doi.org/10.1007/978-3-642-21554-4_27)
3. Ashur, T., Bodden, D.: Linear cryptanalysis of reduced-round speck. In: Proceedings of the 37th Symposium on Information Theory in the Benelux (2016)
4. Biryukov, A., Roy, A., Velichkov, V.: Differential analysis of block ciphers SIMON and SPECK. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 546–570. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46706-0\\_28](https://doi.org/10.1007/978-3-662-46706-0_28)
5. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer, New York (1993). <https://doi.org/10.1007/978-1-4613-9314-6>
6. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference, pp. 1–6 (2015)
7. Cho, J.Y., Pieprzyk, J.: Algebraic attacks on SOBER-t32 and SOBER-t16 without stuttering. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 49–64. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-25937-4\\_4](https://doi.org/10.1007/978-3-540-25937-4_4)
8. Cho, J.Y., Pieprzyk, J.: Multiple modular additions and crossword puzzle attack on NLSv2. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 230–248. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-75496-1\\_16](https://doi.org/10.1007/978-3-540-75496-1_16)
9. Dobraunig, C., Eichlseder, M., Mendel, F.: Heuristic tool for linear cryptanalysis with applications to CAESAR candidates. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part II. LNCS, vol. 9453, pp. 490–509. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48800-3\\_20](https://doi.org/10.1007/978-3-662-48800-3_20)
10. Dinur, I.: Improved differential cryptanalysis of round-reduced speck. In: Joux, A., Youssef, A. (eds.) SAC 2014. LNCS, vol. 8781, pp. 147–164. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-13051-4\\_9](https://doi.org/10.1007/978-3-319-13051-4_9)
11. Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: MILP-based automatic search algorithms for differential and linear trails for speck. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 268–288. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-52993-5\\_14](https://doi.org/10.1007/978-3-662-52993-5_14)
12. Leurent, G.: Analysis of differential attacks in ARX constructions. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 226–243. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-34961-4\\_15](https://doi.org/10.1007/978-3-642-34961-4_15)
13. Leurent, G.: Construction of differential characteristics in ARX designs application to skein. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 241–258. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-40041-4\\_14](https://doi.org/10.1007/978-3-642-40041-4_14)
14. Liu, Y., Fu, K., Wang, W., Sun, L., Wang, M.: Linear cryptanalysis of reduced-round SPECK. *Inf. Process. Lett.* **116**(3), 259–266 (2016)

15. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseht, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994). [https://doi.org/10.1007/3-540-48285-7\\_33](https://doi.org/10.1007/3-540-48285-7_33)
16. Munshi, A.: The OpenCL specification. In: 2009 IEEE Hot Chips 21 Symposium (HCS), pp. 1–314 (2009)
17. Nyberg, K., Wallén, J.: Improved linear distinguishers for SNOW 2.0. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 144–162. Springer, Heidelberg (2006). [https://doi.org/10.1007/11799313\\_10](https://doi.org/10.1007/11799313_10)
18. Stone, J., Gohara, S.: OpenCL a parallel programming standard for heterogeneous computing systems. *Comput. Sci. Eng.* **12**(3), 66–73 (2010)
19. Wallén, J.: Linear approximations of addition modulo  $2^n$ . In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 261–273. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-39887-5\\_20](https://doi.org/10.1007/978-3-540-39887-5_20)
20. Wallén, J.: On the differential and linear properties of addition (2003). <http://www.tcs.hut.fi/Publications/bibdb/HUT-TCS-A84.pdf>
21. Yao, Y., Zhang, B., Wu, W.: Automatic search for linear trails of the SPECK family. In: Lopez, J., Mitchell, C.J. (eds.) ISC 2015. LNCS, vol. 9290, pp. 158–176. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-23318-5\\_9](https://doi.org/10.1007/978-3-319-23318-5_9)



# Conditional Cube Searching and Applications on Trivium-Variant Ciphers

Xiaojuan Zhang<sup>1,2(✉)</sup>, Meicheng Liu<sup>1,2</sup>, and Dongdai Lin<sup>1,2</sup>

<sup>1</sup> State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

[zhangxiaojuan@iie.ac.cn](mailto:zhangxiaojuan@iie.ac.cn)

<sup>2</sup> School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

**Abstract.** In this paper, we describe a new cube searching method called conditional searching. The main idea of this new searching method is to reduce the searching space and contains two main steps: finding complementary variables and searching conditional cubes. At the first step, we introduce a concept of complementary variables corresponding to cube variables to ensure that cube variables are not multiplied with each other in the first few propagations. According to the taps in the feedback functions, two main strategies are given to find complementary variables. At the second step, we first give a simple algorithm to estimate the maximal size of conditional cubes that don't contain any complementary variable. Then another algorithm is given to search conditional cubes. We can confirm the maximum numbers of initialization rounds of some NFSR-based cryptosystems such that the generated keystream bit does not achieve the maximum algebraic degree with our cube searching method and the algebraic degree estimated method numeric mapping. We apply our method to Trivium to verify the validity and our searching space is about  $2^{12.5}$  much smaller than that of existing results. We also introduce two Trivium-variants named Par-Trivium and Loc-Trivium, and apply the method to them. We can get an upper bound of the maximum initialization rounds when we change the parameters or the key and IV loading locations in Trivium. The applications provide some insights into the taps used in the feedback functions of such stream ciphers. We believe that our method is useful in both cryptanalysis and design of NFSR-based cryptosystems.

**Keywords:** Cryptanalysis · Numeric mapping · Stream cipher  
Trivium · Trivium variants

## 1 Introduction

A nonlinear feedback shift register (NFSR) is widely used in modern cryptographic primitives, especially in radio-frequency identification devices (RFID)

© Springer Nature Switzerland AG 2018

L. Chen et al. (Eds.): ISC 2018, LNCS 11060, pp. 151–168, 2018.

[https://doi.org/10.1007/978-3-319-99136-8\\_9](https://doi.org/10.1007/978-3-319-99136-8_9)

and wireless sensor networks applications. Most NFSR-based cryptosystems can be described as tweakable Boolean functions with respect to both secret variables (e.g., key bits) and public variables (e.g., plaintext bits or initial value (IV) bits). The algebraic degrees of these Boolean functions are of great importance in the security of the corresponding primitives. For a cryptographic primitive with low algebraic degree, it is vulnerable to many known attacks, such as cube attacks [1–4] and higher order differential attacks [5,6] which are the most powerful cryptanalytic tools against NFSR-based cryptosystems. Cube attack is introduced by Dinur and Shamir [1], which is a chosen plaintext key-recovery attack. Since then, cube attack has attracted much attention in recent public cryptographic literatures. At Eurocrypt 2015, Dinur et al. [2] publish a key-recovery attack on Keccak keyed modes, where the cube variables are selected not to multiply with each other after the first round, then the output degree of the polynomials is reduced. Later Huang et al. [7] propose a new conditional cube attack on Keccak keyed modes and present an 8-round attack on Keyak. Recently, conditional cube attack is applied to round-reduced ASCON [8] and River Keyak [9]. Also, algebraic attacks [10,11] and integral attacks [12] are easy to perform on a cryptographic primitive with low algebraic degree.

For modern cryptographic primitives, it is difficult to compute the exact values of the algebraic degrees. But, there are several theoretical tools can be used to estimate the upper bounds on the algebraic degrees of iterated permutations, and concurrently exploited to attack iterated ciphers [1,13–15]. Yet for NFSR, there are few tools for estimating its algebraic degree, besides symbolic computation and statistical analysis. Some techniques highly depend on computational capabilities which restrict the cryptanalytic results. A variant of cube attacks called dynamic cube attacks can reach much higher attack complexity, but they are still limited by the size of the cubes [1,4]. Based on this point, in either cube attacks or cube testers, the cubes with size larger than 54 have never been utilized in cryptanalysis of NFSR-based cryptosystems. Recently, at CRYPTO 2017, two works on cube attacks use the cubes with size larger than 50. The one by Todo et al. [16] presents possible key recovery attacks against Trivium [17], Grain-128a [18] and ACORN [19] using the cubes of sizes 72, 92 and 64. They mainly make use of the propagation of the bit-based division property of stream ciphers. The other by Liu [20] gives a tool called numeric mapping to iteratively obtain the upper bounds on the algebraic degrees of NFSR-based cryptosystems.

**Our Contributions.** In this paper, we propose a new cube searching method named conditional searching. The main idea of this new searching method is to reduce the searching space through controlling the propagation of the IV bits and contains two main steps: finding complementary variables and searching conditional cubes. At the finding complementary variables step, we introduce a concept of complementary variables corresponding to cube variables. To ensure that cube variables are not multiplied with each other in the first few propagations, we give two main strategies to find complementary variables according to the taps used in the feedback functions and the size of cubes needed. At the

second step, we first give a simple algorithm to estimate the maximal size of conditional cubes which means that there is not any corresponding complementary variable. Then another algorithm is given to search conditional cubes. With our cube searching method and the algebraic degree estimated method numeric mapping introduced by Liu [20], we can confirm the maximum numbers of initialization rounds of some NFSR-based cryptosystems such that the generated keystream bit does not achieve the maximum algebraic degree.

We apply our method to Trivium to verify the validity with the searching space of size about  $2^{12.5}$ . While in [20], the searching space is  $2^{25}$ . Our searching space is smaller than that of the existing results. We also apply the method to Trivium-variants containing Par-Trivium and Loc-Trivium. For Par-Trivium, where the parameters in Trivium are changed, we estimate the algebraic degrees and can get an upper bound of the maximum initialization rounds such that the generated keystream bit does not achieve the maximum algebraic degree. The experiments show that the maximum round of Par-Trivium is 863 which is the worst case. So, parameters in Par-Trivium can be as big as possible on the premise of the security against other attacks. And for Loc-Trivium, where the key and IV loading locations are changed in Trivium, we can get similar results. The experiments show that the maximum initialization round of all considered Trivium-variants is 910, which is the worst case and should be avoided to be resistant to cube attacks or cube tests when new ciphers are designed. The applications provide some insights into the taps used in the feedback functions and the key and IV loading locations of such stream ciphers, which are useful in both cryptanalysis and design of NFSR-based cryptosystems.

**Organization of the Paper.** The rest of this paper is structured as follows. In Sect. 2, basic definitions and notations are provided. Section 3 shows the general framework of our conditional searching method, while its applications on Trivium and Trivium-variants are given in Sects. 4 and 5. Section 6 concludes the paper.

## 2 Preliminaries

**Boolean Functions and Algebraic Degree.** Let  $\mathbb{F}_2$  be the binary field and  $\mathbb{F}_2^n$  the  $n$ -dimensional vector space over  $\mathbb{F}_2$ . An  $n$ -variable Boolean function is a mapping from  $\mathbb{F}_2^n$  into  $\mathbb{F}_2$ . An  $n$ -variable Boolean function  $f$  can be uniquely represented as a multivariate polynomial over  $\mathbb{F}_2$ ,

$$f(x_1, x_2, \dots, x_n) = \bigoplus_{c=(c_1, c_2, \dots, c_n) \in \mathbb{F}_2^n} a_c \prod_{i=1}^n x_i^{c_i}, \quad a_c \in \mathbb{F}_2,$$

called the algebraic normal form (ANF). The algebraic degree of  $f$ , denoted by  $\text{deg}(f)$ , is defined as  $\max\{wt(c) | a_c \neq 0\}$ , where  $wt(c)$  is the Hamming weight of  $c$ .

**Numeric Mapping.** Let  $f(x_1, x_2, \dots, x_n) = \bigoplus_{c=(c_1, c_2, \dots, c_n) \in \mathbb{F}_2} a_c \prod_{i=1}^n x_i^{c_i}$  ( $a_c \in \mathbb{F}_2$ ) be a Boolean function. Denote by  $\mathbb{B}_n$  the set of all  $n$ -variable Boolean functions. The numeric mapping [20], denoted by DEG is defined as

$$\begin{aligned} \text{DEG} : \quad \mathbb{B}_n \times \mathbb{Z}_n &\rightarrow \mathbb{Z}, \\ (f, D) &\mapsto \max_{a_c \neq 0} \left\{ \sum_{i=1}^n c_i d_i \right\}, \end{aligned} \tag{1}$$

where  $D = (d_1, d_2, \dots, d_n)$  and  $a_c$ 's are coefficients of the ANF of  $f$ . Let  $g_i$  ( $1 \leq i \leq n$ ) be Boolean functions on  $m$  variables, and denote  $\text{deg}(G) = (\text{deg}(g_1), \text{deg}(g_2), \dots, \text{deg}(g_n))$  for  $G = (g_1, g_2, \dots, g_n)$ . The numeric degree of the composite function  $h = f \circ G$  is defined as  $\text{DEG}(f, \text{deg}(G))$ , denoted by  $\text{DEG}(h)$  for short. The algebraic degree of  $h$  is always less than or equal to the numeric degree of  $h$ . The algebraic degrees of the output bits with respect to the internal states can be estimated iteratively for NFSR-based cryptosystems by using numeric mapping.

**Cube Testers.** Given a Boolean function  $f$  and a term  $t_I$  containing variables from an index subset  $I$  that are multiplied together, the function can be written as

$$f(x_1, x_2, \dots, x_n) = f_S(I) \cdot t_I \oplus q(x_1, x_2, \dots, x_n),$$

where the terms in  $q(x_1, x_2, \dots, x_n)$  miss at least one variable from  $I$  and  $f_S(I)$  is called the superpoly of  $I$  in  $f$ . The basic idea of cube testers is that

$$\sum_{x' \in C_I} f = f_S(I),$$

where  $C_I$  are all possible values of the subset of variables in the term  $t_I$ . The target of cube testers work by evaluating superpolys of carefully selected terms  $t_I$ 's which are products of public variables (e.g., IV bits), and trying to distinguish them from a random function. A cube tester can detect the nonrandomness in cryptographic primitives by extracting the testable properties of the superpoly, such as unbalance, constantness and low degree, with the help of property testers. Especially, the superpoly  $f_S(I)$  is equal to a zero constant, if the algebraic degree of  $f$  in the variables from  $I$  is smaller than the size of  $I$ .

### 3 A New Method for Searching Cube

In this section, we propose a new model for searching cube, called conditional searching. The new searching method consists of two phases, finding complementary variables and searching conditional cubes. First, we will give a generalized model of the initialization phases of NFSR-based cryptosystems.

### 3.1 Generalized Model

For NFSR-based cryptosystems, especially NFSR-based stream ciphers, the initialization phase is used to initialize the internal state using secret variables (e.g., key bits) and public variables (e.g., plaintext bits or IV bits). Then the encryption phase just consists of an exclusive or (XOR) with the continuously updated keystream. The generalized model of initialization phases of some NFSR-based cryptosystems can be depicted as Fig. 1, which is helpful in the sense that we could study some special properties/choices more clearly in a unified framework.

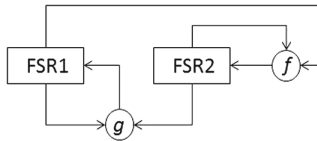


Fig. 1. Generalized initialization phase of NFSR-based cryptosystem

FSR1 and FSR2 are two registers. Here we stress that FSR1 in the model can be further decomposed into a series of cascaded smaller NFSRs or LFSRs, which could also be treated by our cryptanalysis. There are two Boolean functions involved in the model: a (either linear or non-linear) Boolean function  $g$  and a non-linear Boolean function  $f$ . FSR2 is initialized by the padded IV. It is obvious that our generalized model could cover initialization processes of Grain v1 [21], Trivium and so on. Denote by  $S_t$  and  $B_t$  the initial states of FSR1 and FSR2 at time  $t$  with size of  $m_1$  and  $m_2$ . At each step, FSR2 is updated by  $f$ , sometimes without any taps from FSR1, and FSR1 is updated by  $g$  as follows:

- FSR1 is updated recursively by  $g$  as  $S_{t+1} = (s_{t+1}, s_{t+2}, \dots, s_{t+m_1})$  with  $s_{t+m_1} = g(S_t, B_t)$ .
- FSR2 is updated recursively by  $f$  as  $B_{t+1} = (b_{t+1}, b_{t+2}, \dots, b_{t+m_2})$  with  $b_{t+m_2} = f(B_t, S_t)$  or  $b_{t+m_2} = f(B_t)$ .
- We assume these processes are invertible, and the inverse processes are  $S_{t-1} = (s_{t-1}, s_t, \dots, s_{t+m_1-2})$  with  $s_{t-1} = g^{-1}(S_t, B_t)$  and  $B_{t-1} = (b_{t-1}, b_t, \dots, b_{t+m_2-2})$  with  $b_{t-1} = f^{-1}(B_t, S_t)$  or  $b_{t-1} = f^{-1}(B_t)$ .

### 3.2 Conditional Searching Method

Huang et al. [7] propose a new conditional cube attack on Keccak keyed modes and present an 8-round attack on Keyak. By restraining some bit conditions of the key, they obtain a new set of cube variables which not only do not multiply with each other after the first round, but also contains one cube variable that does not multiply with other cube variables after the second round, and then the output degree over cube variables is further reduced. Based on this method, we give our conditional searching method which is a searching tool with some conditions

to reduce the searching space. The conditional cubes used in our paper is different, which means that the cubes do not contain any complementary variables. Before the method is given, we will give a definition of complementary variables. In the cube attack or cube test against stream cipher, cube variables are chosen from the IV bits with length  $l$ , where  $IV = (iv_1, iv_2, \dots, iv_l)$ . When  $iv_i$  is chosen as a cube variable, the variables that must not be chosen are called complementary variables corresponding to the cube variable  $iv_i$ . Our conditional searching method contains two steps: finding complementary variables and searching conditional cubes. We call the cubes that do not contain any corresponding complementary variables, conditional cubes for simplicity. The conditional cubes and the maximum numbers of initialization rounds (maximum rounds for simplicity) are corresponding to the situation that the generated keystream bit does not achieve the maximum algebraic degree.

**Finding Complementary Variables.** The main way to find complementary variables is to control the propagation of cube variables. As shown in the generalized model, the IV bits are loaded in FSR2 and cube variables are chosen from them. This way, the propagation paths of the IV bits are of importance to choose cube variables and determine the corresponding complementary variables. In the beginning, the IV bits only appear in FSR2 and take part in the update of FSR2 (or FSR1) through the feedback function  $f$  (or  $g$ ). Several steps later, they will appear in both FSR2 and FSR1.

It is intuitive that if the cube variables are not multiplied with each other in the first few propagations, the maximum initialization round, such that the generated keystream bit does not achieve the maximum algebraic degree, will be larger. The complementary variables are determined by these variables that would be multiplied with each other. For the multiplied variables, if one of them is chosen to be cube variable, the others are defined as complementary variables. This way, the taps in the feedback functions play a leading role in finding complementary variables. It is easy to see that the more propagations are controlled, the less conditional cubes are satisfied. Two main strategies to choose cube variables are used here.

- In the beginning, where the IV bits take part in the first iteration, cube variables should not be multiplied with each other.
- When the IV bits as a part of the feedback value take part in the update function, cube variables must not be multiplied with each other.

By an iteration we mean two or more rounds which depends on the maximal tap in the feedback function, more details, one can see the following example.

*Example 1.* Let  $m_1 = m_2 = 8$  and

$$\begin{aligned} s_{t+8} &= s_{t+6}s_{t+1} + s_t + b_{t+2}, \\ b_{t+8} &= b_{t+4}b_{t+3} + b_t + s_{t+3}, \\ S_1 &= (s_1, s_2, \dots, s_8) = (iv_1, iv_2, \dots, iv_8), \end{aligned}$$



where  $t \geq 1$ . For FSR2, the maximal tap in the feedback function is  $s_{t+6}$  and an iteration means two rounds, where

$$\begin{aligned}
 &\text{the first iteration} \quad \begin{cases} \mathbf{s}_9 = s_7s_2 + s_1 + b_3 \\ s_{10} = s_8s_3 + s_2 + b_4 \end{cases} \\
 &\text{the second iteration} \quad \begin{cases} \mathbf{s}_{11} = \mathbf{s}_9s_4 + s_3 + b_5 \\ s_{12} = s_{10}s_5 + s_4 + b_6 \end{cases} \\
 &\text{the third iteration} \quad \begin{cases} s_{13} = \mathbf{s}_{11}s_6 + s_5 + b_7 \\ \dots \end{cases}
 \end{aligned}$$

What we need to control is the first two iterations, that is  $t \leq 4$ . In the first iteration,  $s_7, s_2$  and  $s_8, s_3$  are multiplied with each other. In the second iteration,  $s_9$  is multiplied with  $s_4$  and  $s_{10}$  is multiplied with  $s_5$ , where  $s_9$  and  $s_{10}$  are feedback values. Take the IV bits into account, the multiplied pairs are  $(iv_2, iv_7), (iv_3, iv_8), (iv_1, iv_4)$  and  $(iv_2, iv_5)$ , during the first two iterations. We can predict that if  $iv_i$  is a cube variable,  $iv_{i+3}$  and  $iv_{i+5}$  will be multiplied with  $iv_i$  at some point. So, the complementary variables are  $iv_{i+3}$  and  $iv_{i+5}$  corresponding to the cube variable  $iv_i$ . Similarly, for FSR1, when  $t = 9$ , the number of the iterations with respect to  $S_1$  is larger than two. The multiplied pairs are  $(iv_4, iv_5), (iv_5, iv_6)$  and  $(iv_6, iv_7)$ , during the first two iterations and the complementary cube variable is  $iv_{i+1}$  corresponding to the cube variable  $iv_i$ . In summary, if  $iv_i$  is chosen to be a cube variable, the set of complementary variables is  $\{iv_{i+1}, iv_{i+3}, iv_{i+5}\}$ .

**Searching Conditional Cube.** Once the complementary variables are obtained, we need to search the conditional cubes that do not contain any complementary variables. But before that, we have to determine the maximal size of conditional cubes and then to search this kind of cubes. If the number of variables that can be used as cube variables is small, the problem is very easy. As shown in Example 1, the maximal size of conditional cubes is four, which are  $\{iv_1, iv_3, iv_5, iv_7\}$  and  $\{iv_2, iv_4, iv_6, iv_8\}$ . We can verify that among conditional cubes, cube variables will not be multiplied with each other in the first few rounds and there is not any complementary variables  $\{iv_{i+1}, iv_{i+3}, iv_{i+5}\}$  corresponding to  $iv_i$ .

In the NFSR-based stream ciphers, the length of the IV bits is so large that we can't easily estimate the maximal size and search the cubes. Two algorithms are given to solve these two problems. Let the set of the complementary variables corresponding to  $iv_i$  be  $C' = \{iv_{i+j_1}, iv_{i+j_2}, \dots, iv_{i+j_m}\}$ . Algorithm 1 is used to estimate the maximal size of conditional cubes. The main idea is to choose one special cube according to the indexes of the IV bits from small to large. It is obvious that the size  $dim$  of this special cube is the maximal size and all the conditional cubes are of size less than  $dim$ .

Algorithm 2 is given to search conditional cubes with the maximal size  $dim$ . First, we evenly divide the IV bits into several parts. For the first part, search all sub-conditional cubes with possible sizes. For other parts, we can get the corresponding sub-conditional cubes by changing the subscripts, as show in line 7 of

---

**Algorithm 1.** Estimation of maximal size

---

**Require:** the complementary variables set  $C'$ , the set  $C = \phi$ , and the length  $l$  of the IV bits

**Ensure:** the maximal size of conditional cubes

- 1: **for**  $i$  from 1 to  $l$  **do**
  - 2:   **if**  $iv_i \notin C'$  **then**
  - 3:     Add  $iv_i$  to  $C$ .
  - 4:     Add  $iv_{i+j_1}, iv_{i+j_2}, \dots, iv_{i+j_m}$  to  $C'$ .
  - 5:   **end if**
  - 6: **end for**
  - 7: Let  $dim = |C|$ , where  $|C|$  is the size of set  $C$ .
  - 8: **return**  $dim$  is the maximal size of conditional cubes.
- 

---

**Algorithm 2.** Searching conditional cubes

---

**Require:** the maximal size  $dim$ , the complementary variables set  $C'$ ,  $l$ ,  $m$

**Ensure:** the maximal round of conditional cubes

- 1: **for**  $i$  from 1 to  $m$  **do**
  - 2:   In the set  $\{iv_1, iv_2, \dots, iv_m\}$ , search all possible sub-conditional cubes with size  $i$  and denoted by  $C_i$ . Denote the size of  $C_i$  by  $SC_i$ .
  - 3: **end for**
  - 4: **for** all combinations of sub-conditional cubes with size  $subdim_j$ , such that  $subdim_1 + subdim_2 + \dots + subdim_{l/m} = dim$  **do**
  - 5:   **for**  $j$  from 1 to  $l/m$  **do**
  - 6:     Choose one sub-conditional cube in  $C_{subdim_j}$
  - 7:     Add  $(j-1) \cdot m$  to the subscripts of the sub-conditional cube.
  - 8:   **end for**
  - 9:   Put the sub-conditional cubes with size  $subdim_j$  together to obtain a cube with size  $dim$
  - 10:   Test whether the cube with size  $dim$  is a conditional cube.
  - 11:   **if** the cube is an conditional cube **then**
  - 12:     Estimate the maximum round with numeric mapping method.
  - 13:   **end if**
  - 14: **end for**
  - 15: **return** the maximal round of conditional cubes
- 

Algorithm 2. Second, examine the combinations of all possible sub-conditional cubes to obtain the conditional cubes. Denote the length of the IV bits and the evenly divided factor by  $l$  and  $m$ . Evenly divide the IV bits into  $l/m$  parts, denoted by part  $j$ , where  $1 \leq j \leq l/m$ . Denoted by  $subdim_j$  the sub-size of sub-conditional cubes chosen from part  $j$ . The criteria of testing whether the cube with size  $dim$  is a conditional cube is to verify that if  $iv_i$  is a cube variable, whether the complementary variables in  $C'$  are cube variables. For example, if the length of the IV bits is 80, we can evenly divide them into 4 parts,  $iv_1$  to  $iv_{20}$ ,  $iv_{21}$  to  $iv_{40}$ ,  $iv_{41}$  to  $iv_{60}$ , and  $iv_{61}$  to  $iv_{80}$ . The first part is  $iv_1$  to  $iv_{20}$  and all sub-conditional cubes with possible sizes can be obtained by the simple exhaustive search method. For other parts, add 20, 40, 60 to the subscripts

of the sub-conditional cubes. If the size of conditional cubes is 38, we need to consider all possible combinations of sub-conditional cubes, where the sum of the sub-sizes is 38. Then for each cube variable  $iv_i$ , verify whether the corresponding complementary variables are cube variables. If not, this cube is a conditional cube.

The main time complexity of Algorithm 2 is to test the combinations of sub-conditional cube with size of  $subdim_j$ , where  $1 \leq j \leq l/m$ . According to the length of the IV bits and the maximum size of conditional cubes,  $m$  need to be chosen carefully, which plays an important role in the time complexity. Whether the IV bits need to be divided depends on the complementary variables. For Trivium, two complementary variables are  $iv_{i+1}$  and  $iv_{i-1}$  and we can exhaust search all the cubes containing no adjacent indexes. Then the criteria is used to test whether the cubes are conditional cubes.

## 4 Applications to Trivium

Trivium is a stream cipher designed in 2005 and has been selected as one of the portfolio for hardware ciphers (Profile 2) by the eSTREAM project. Though Trivium is designed to provide a flexible trade-off between speed and gate count in hardware, it also provides extremely efficient software implementation. Trivium has attracted much attention in recent public cryptographic literatures for its simplicity and very good performance, such as [26, 27].

In this section, we apply our conditional searching method to Trivium to verify the validity of our method.

### 4.1 A brief description of Trivium

Trivium generates up to  $2^{64}$  bits of keystream from an 80-bit secret key and an 80-bit IV. For the sake of simplicity, we give an alternative description of the algorithm different from the already existed ones. Let  $A$ ,  $B$  and  $C$  be three registers of sizes 93, 84 and 111. Denoted by  $A^t$ ,  $B^t$  and  $C^t$  the corresponding states at time  $t$  ( $t \geq 0$ ),

$$\begin{aligned} A^t &= (x_1^t, x_2^t, \dots, x_{93}^t), \\ B^t &= (y_1^t, y_2^t, \dots, y_{84}^t), \\ C^t &= (z_1^t, z_2^t, \dots, z_{111}^t), \end{aligned}$$

and the three quadratic update functions are

$$\begin{aligned} x_{93}^t &= z_1^{t-1} + z_2^{t-1} \cdot z_3^{t-1} + z_{46}^{t-1} + x_{25}^{t-1}, \\ y_{84}^t &= x_1^{t-1} + x_2^{t-1} \cdot x_3^{t-1} + x_{28}^{t-1} + y_7^{t-1}, \\ z_{111}^t &= y_1^{t-1} + y_2^{t-1} \cdot y_3^{t-1} + y_{16}^{t-1} + z_{25}^{t-1}. \end{aligned}$$

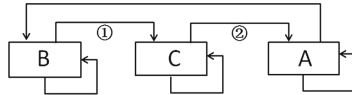
The algorithm is initialized by loading an 80-bit secret key and an 80-bit IV into the 288-bit initial state, and setting all remaining bits to 0, except for  $z_1$ ,  $z_2$  and  $z_3$ ,

$$\begin{aligned} (x_1^0, x_2^0, \dots, x_{93}^0) &\leftarrow (0, \dots, 0, k_0, \dots, k_{79}), \\ (y_1^0, y_2^0, \dots, y_{84}^0) &\leftarrow (0, \dots, 0, iv_0, \dots, iv_{79}), \\ (z_1^0, z_2^0, \dots, z_{111}^0) &\leftarrow (1, 1, 1, 0, \dots, 0). \end{aligned}$$

Let  $h$  be the output function. After an initialization of  $N$  rounds, in which the internal state is updated for  $N$  times, the cipher generates a keystream bit by  $h(A^t, B^t, C^t)$  for each  $t \geq N$ .

### 4.2 Conditional Searching for Trivium

The 80-bit IV is loaded into the shift register  $B$  and the propagation paths are shown in Fig. 2. To find the complementary variables, we just need to control the paths ① and ② in the first few rounds. For path ①, we guarantee that the cubes are not multiplied with each other in the first iteration. According to the taps in the feedback functions, the complementary variables are  $iv_{i-1}$  and  $iv_{i+1}$  corresponding to cube variable  $iv_i$ . When the IV bits in the register  $C$  begin to be transmitted to the register  $A$ , we need to control the path ②. The complementary variables are  $iv_{i+14}$  and  $iv_{i+16}$  corresponding to cube variable  $iv_i$ . In summary, if  $iv_i$  is chosen to be a cube variable, the set of complementary variables is  $\{iv_{i-1}, iv_{i+1}, iv_{i+14}, iv_{i+16}\}$ .



**Fig. 2.** Propagation paths of IV

With Algorithm 1, we know that the sizes of conditional cubes are less than 38. Then, with Algorithm 2, we can obtain all conditional cubes of size 37 and 38 in a dozen seconds on a common PC. The result, see Table 1, shows that the output of 837-round Trivium has degree strictly less than 37 over a subset of the IV bits with size 37, which agrees with [20]. The corresponding cubes

**Table 1.** Results of Trivium

	Maximum round/cube size	Searching space
[20]	837/37	$2^{25}$
Our method	837/37	$2^{12.5}$

are the same and the amount of the conditional cubes is  $5945 \approx 2^{12.5}$ . Before this paper, the amount of cubes need to be searched is about  $2^{25}$  [20]. We can conclude that our conditional searching method is valid for Trivium and it has better performance.

## 5 Applications to Trivium-Variant Ciphers

The analysis of Trivium-like ciphers is to find a variant that will remove one of the biggest deficiencies in the Trivium design: the huge number of initial 1152 rounds, which makes the original Trivium stream cipher to have very high initial latencies when used in IoT devices. The parameters used in Trivium are subtle that a little change will affect the security a lot. Although there are some Trivium-based cryptosystems, such as Kreyvium [22], TriviA-SC [23] Bivium [24], Trivium-N [25] and so on, how do the parameters work is still a problem to be solved. In this section, we introduce two Trivium-variants named Par-Trivium and Loc-Trivium, and apply the conditional searching method to them to give some guidelines for choosing parameters according to the algebraic degree.

### 5.1 Applications to Par-Trivium

Par-Trivium is a variant of Trivium, where the parameters in the feedback functions of Trivium are changed. Determined by propagation paths of the IV bits, the algebraic degrees of conditional cubes are mainly associated with the taps from the register where the IV bits are loaded, and distances between the indexes of multiplied variables. In order to keep the elegant structure, the feedback functions of Par-Trivium at time  $t$  can be write as

$$\begin{aligned} x_{93}^t &= z_1^{t-1} + z_\alpha^{t-1} \cdot z_{\alpha+\delta}^{t-1} + z_{46}^{t-1} + x_{25}^{t-1}, \\ y_{84}^t &= x_1^{t-1} + x_\alpha^{t-1} \cdot x_{\alpha+\delta}^{t-1} + x_{28}^{t-1} + y_\gamma^{t-1}, \\ z_{111}^t &= y_1^{t-1} + y_\alpha^{t-1} \cdot y_{\alpha+\delta}^{t-1} + y_\beta^{t-1} + z_{25}^{t-1}. \end{aligned}$$

For Trivium,  $\alpha = 2$ ,  $\beta = 16$ ,  $\gamma = 7$ ,  $\delta = 1$ . For simplicity, we denote by  $R_{\alpha,\beta,\gamma,\delta}$  the maximum round that the conditional cubes of size  $37 \leq n \leq 40$  have reached maximum degrees. Sometimes a part of the subscripts would be omitted and the symbol becomes  $R_{\beta,\gamma}$ ,  $R_\alpha$  and so on.

The impacts of parameters  $\alpha, \beta, \gamma$  and  $\delta$  on the algebraic degrees are summarized in the following three properties.

**Property 1.** *When  $\beta$ ,  $\gamma$  and  $\delta$  are fixed, the maximum round that the conditional cubes have reached maximum degrees decreases with the growth of  $\alpha$ .*

It is obvious that the larger  $\alpha$  is, the earlier the feedback values take part in the iterations and the smaller the maximum round is. For  $\beta = 16$ ,  $\gamma = 7$ ,  $\delta = 1$ , the result is listed in Table 2. We can see that the maximum rounds  $R_\alpha$  is decreased with the growth of  $\alpha$ .

**Table 2.** The maximum rounds  $R_\alpha$ , when  $\beta = 16$ ,  $\gamma = 7$ ,  $\delta = 1$

$\alpha$	2	7	12	17	22	27	32	37	42	47	52	57	62	67	72	77	82
$R_\alpha$	837	741	708	704	640	601	575	524	487	449	407	364	315	272	226	189	156

For Par-Trivium with parameters  $(\alpha, \beta, \gamma, \delta)$ , the complementary variables are  $iv_{i-\delta}$ ,  $iv_{i+\delta}$ ,  $iv_{i+\beta-\delta-1}$  and  $iv_{i+\beta+\delta-1}$  corresponding to cube variable  $iv_i$ . When  $i + \beta + \delta - 1 > 79$ ,  $iv_{i+\beta+\delta-86+\gamma}$  will be multiplied with  $iv_{i+\delta-86+\gamma}$  according to the propagation paths of the IV bits. Also, when  $i + \beta - \delta - 1 > 79$ ,  $iv_{i+\beta-\delta-86+\gamma}$  will be multiplied with  $iv_{i-\delta-86+\gamma}$ . For example, when  $\alpha = 2$ ,  $\beta = 16$ ,  $\gamma = 7$  and  $\delta = 2$ , we can know that

$$\begin{aligned}
 z_{111}^1 &= y_{16}^0 + \dots = iv_{11} + \dots, \\
 \dots & \\
 z_{111}^4 &= y_{16}^3 + y_2^3 \cdot y_4^3 + \dots = iv_{14} + iv_0 \cdot iv_2 + \dots, \\
 z_{111}^5 &= y_{16}^4 + y_1^4 + y_2^4 \cdot y_4^4 + \dots = iv_{15} + iv_0 + iv_1 \cdot iv_3 + \dots, \\
 \dots & \\
 z_{111}^{88} &= y_{16}^{87} + y_1^{87} + y_2^{87} \cdot y_4^{87} + \dots = iv_{78} + iv_{63} + iv_{64} \cdot iv_{66} + \dots, \\
 z_{111}^{89} &= y_{16}^{88} + y_1^{88} + y_2^{88} \cdot y_4^{88} + \dots = iv_{79} + iv_{64} + iv_{65} \cdot iv_{67} + \dots, \\
 z_{111}^{90} &= y_{16}^{89} + y_1^{89} + y_2^{89} \cdot y_4^{89} + \dots = iv_2 + iv_{65} + iv_{66} \cdot iv_{68} + \dots, \\
 \dots & \\
 z_{111}^{103} &= y_{16}^{102} + y_1^{102} + \dots = iv_{15} + iv_{78} + \dots, \\
 z_{111}^{104} &= y_{16}^{103} + y_1^{103} + \dots = iv_{16} + iv_{79} + \dots, \\
 z_{111}^{105} &= y_{16}^{104} + y_1^{104} + \dots = iv_{17} + iv_2 + \dots
 \end{aligned}$$

according to the feedback functions. When  $t = 198$ ,  $z_{111}^{88}$  and  $z_{111}^{90}$  will take part in the feedback of  $x_{93}^{198}$  and  $iv_2$  will be multiplied with  $iv_{63}$ . Also, at time  $t = 213$ ,  $z_{111}^{103}$  and  $z_{111}^{105}$  will take part in the feedback of  $x_{93}^{213}$  and  $iv_2$  will be multiplied with  $iv_{15}$ . Take these into consideration, we can use our conditional searching method to estimate the maximum round  $R_{\alpha, \beta, \gamma, \delta}$  with respect to the conditional cubes of size  $37 \leq n \leq 40$  and we get the following result.

**Property 2.** For parameters  $\alpha$ ,  $\beta$ ,  $\gamma$  and  $\delta$ , let  $q = 79 - \delta \pmod{2\delta}$  and

$$\begin{aligned}
 \beta^* &= \lfloor \beta / (2\delta) \rfloor \cdot 2\delta + 1, \\
 \gamma^* &= \begin{cases} \delta + 1 + 2k\delta, & 0 \leq q \leq \delta - 2 \\ q - \delta + 2k\delta, & \delta - 1 \leq q \leq 2\delta - 1, \end{cases}
 \end{aligned}$$

where  $k$  is the maximum integer satisfying  $\gamma \geq \gamma^*$ . When  $\alpha$  and  $\delta$  are fixed, the maximum round  $R_{\beta, \gamma}$  that the conditional cubes of size  $37 \leq n \leq 40$  have reached maximum degrees, satisfies that

$$R_{\beta, \gamma} \leq R_{\beta^*, \gamma^*}$$

and  $R_{\beta^*, \gamma^*}$  decreases with the growth of  $\beta^*$  or  $\gamma^*$ .

When  $\alpha$  and  $\delta$  are fixed, for any  $\beta$  and  $\gamma$ , we can obtain an upper bound of the number of initialization rounds such that the generated keystream bit does not achieve the maximum algebraic degree. For example, when  $\alpha = 2$  and  $\delta = 2$ , we can get  $q = 1$  and list a part of the maximum rounds for different  $\beta$  and  $\gamma$  in Table 3. We can see that in each square, the upper bound of the maximum rounds appears at the lower left. For  $21 \leq \beta \leq 24$  and  $9 \leq \gamma \leq 12$ , it shows that  $R_{\beta,\gamma} \leq R_{\beta^*,\gamma^*} = R_{21,9}$ , where  $\beta^* = 21$  and  $\gamma^* = 9$ , which is indicated by the gray part. Also,  $R_{\beta^*,\gamma^*}$  is inversely proportional to  $\beta^*$  or  $\gamma^*$ , see the numbers in bold in Table 3. Parameters  $\beta$  and  $\gamma$  for the maximum rounds in bold are the corresponding  $\beta^*$  and  $\gamma^*$ . So, in design of Trivium-Like stream cipher, it is needed to choose  $\beta$  and  $\gamma$  as big as possible, and  $\beta^*$  and  $\gamma^*$  should be avoided. While the location of the lower bound is indeterminate in spite of the fact in Table 3. It's a coincidence that the maximum round in the first column and second row is the smallest number.

**Table 3.** The maximum rounds  $R_{\beta,\gamma}$ , where  $\alpha = 2$  and  $\delta = 2$

$\beta \backslash \gamma$	5	6	7	8	9	10	11	12	13	17	21	25
33	<b>823</b>	810	801	812	<b>823</b>	810	801	812	<b>823</b>	<b>823</b>	<b>823</b>	<b>823</b>
32	813	811	821	821	813	811	825	821	808	797	795	795
31	809	821	814	808	811	823	814	808	813	814	816	823
30	815	827	831	815	809	826	829	817	807	807	803	803
29	<b>830</b>	821	808	820	<b>830</b>	821	809	818	<b>830</b>	<b>830</b>	<b>830</b>	<b>830</b>
28	823	809	825	829	823	811	829	829	822	818	812	804
27	791	814	824	813	809	818	824	813	810	814	816	820
26	824	833	831	815	819	833	827	817	817	817	813	809
25	<b>836</b>	824	817	826	<b>836</b>	824	817	822	<b>836</b>	<b>836</b>	<b>836</b>	<b>836</b>
24	829	823	822	833	829	817	822	833	827	825	819	815
23	812	824	822	821	805	822	822	818	796	796	816	813
22	827	829	825	825	827	829	825	825	827	827	821	817
21	<b>842</b>	830	825	832	<b>842</b>	830	825	830	<b>842</b>	<b>842</b>	<b>840</b>	<b>837</b>
20	839	831	823	829	839	827	823	828	838	834	829	821
19	-	-	-	-	-	-	-	-	-	-	-	-
18	831	825	825	827	831	827	823	827	831	831	827	823
17	<b>846</b>	842	833	838	<b>846</b>	840	833	837	<b>846</b>	<b>844</b>	<b>841</b>	<b>837</b>
16	836	840	840	830	837	841	834	828	837	837	837	833
15	-	-	-	-	-	-	-	-	-	-	-	-
14	789	829	835	833	822	826	835	833	826	825	825	825
13	<b>850</b>	848	839	842	<b>850</b>	844	839	843	<b>848</b>	<b>845</b>	<b>841</b>	<b>837</b>

- means that the conditional cubes of size  $37 \leq n \leq 40$  doesn't exist.

**Property 3.** Let  $\alpha = 2$ . When  $\beta$  and  $\gamma$  run through all possible combinations, the maximum round that the conditional cubes have reached maximum degree decreases with the growth of  $\delta$ .

It may be more persuasive to consider the relationship between the maximum round and  $\delta$  if  $\beta$  and  $\gamma$  are also fixed. But when  $\beta$  and  $\gamma$  are also fixed, different

$\delta$  would lead to the sizes of the corresponding conditional cubes smaller than 37, which makes the comparison meaningless. For  $\alpha = 1$ ,  $\beta = 16$ ,  $\gamma = 7$ , the size of the conditional cubes is 38 corresponding to  $\delta = 1$ , but if  $\delta = 6$ , the size of the conditional cubes is smaller than 33. So we run through all possible  $\beta$  and  $\gamma$  to compare the maximum rounds. When  $1 \leq \delta \leq 5$ , the results are listed in Table 4. Here  $\alpha = 2$  and three taps from register  $B$ , which are  $y_1$ ,  $y_2$  and  $y_{2+\delta}$ , are fixed.  $y_\gamma$  takes part in the feedback of register  $B$  separately and  $y_\beta$  takes part in the feedback of register  $C$  with  $y_1$ ,  $y_2$  and  $y_{2+\delta}$  together, as

$$y_{84}^t = x_1^{t-1} + x_2^{t-1} \cdot x_{2+\delta}^{t-1} + x_{28}^{t-1} + y_\gamma^{t-1},$$

$$z_{111}^t = y_1^{t-1} + y_2^{t-1} \cdot y_{2+\delta}^{t-1} + y_\beta^{t-1} + z_{25}^{t-1}.$$

So the range of values for  $\gamma$  is 1 to 84 and for  $\beta$  is 2 to 84. When  $\delta$  is given, the minimum values of  $\beta^*$  and  $\gamma^*$ , and the maximum value of rounds are determined by Property 2. But some pairs of  $(\beta, \gamma)$  aren't in consideration. For example, when  $\delta = 3$ ,  $\beta_{min}^* = 7$ ,  $\gamma_{min}^* = 1$  and  $R_{7,1} = 851$ . But for  $2 \leq \beta \leq 6$ , we need to estimate the corresponding maximum rounds separately.

**Table 4.** The maximum rounds  $R_{\beta,\gamma,\delta}$ , where  $\alpha = 2$  and  $k$  is a positive integer

$\delta$	Taps in $B$	$\beta^*$	$\gamma^*$	$R_{\beta,\gamma}$
1	$\{y_1, y_2, y_3, y_\beta, y_\gamma\}$	$1 + 2k$	$1 + 2k$	$\max\{R_{\beta,\gamma}   2 \leq \beta \leq 3, \gamma = 1\} = R_{3,1} = 863$
2	$\{y_1, y_2, y_4, y_\beta, y_\gamma\}$	$1 + 4k$	$1 + 4k$	$\max\{R_{\beta,\gamma}   2 \leq \beta \leq 5, \gamma = 1\} = R_{5,1} = 861$
3	$\{y_1, y_2, y_5, y_\beta, y_\gamma\}$	$1 + 6k$	$1 + 6k$	$\max\{R_{\beta,\gamma}   2 \leq \beta \leq 7, \gamma = 1\} = R_{3,1} = 854$
4	$\{y_1, y_2, y_6, y_\beta, y_\gamma\}$	$1 + 8k$	$5 + 8k$	$\max\{R_{\beta,\gamma}   2 \leq \beta \leq 9, 1 \leq \gamma \leq 5\} = R_{9,1} = 847$
5	$\{y_1, y_2, y_7, y_\beta, y_\gamma\}$	$1 + 10k$	$5 + 10k$	$\max\{R_{\beta,\gamma}   2 \leq \beta \leq 11, 1 \leq \gamma \leq 5\} = R_{3,1} = 842$

The results give us some design principles for designing Trivium-Like ciphers. Ideally, the smaller the maximum rounds of NFSR-based cryptosystems such that the generated keystream bit does not achieve the maximum algebraic degree the better. So the parameters in Par-Trivium should be as big as possible on the premise of the security against other attacks. From the experiment results, we can know that the maximum round of Par-Trivium is 863 which is the worst case.

### 5.2 Applications to Loc-Trivium

Loc-Trivium is a variant of Trivium, where the key and the IV loading locations are changed in Trivium. In Trivium, the key bits are loaded into the register  $A$  and the IV bits are loaded into the register  $B$ . The maximum round that conditional cubes reach maximum degree is influenced mainly by propagation paths of the IV bits. So, if we change the key and IV loading locations, the maximum round will be changed. The experiments show that the maximum round is influenced mainly by the IV loading location and the corresponding taps. Here, an example is given to illustrate our findings.



*Example 2.* For Loc-Trivium, if we load the IV bits into the register  $A$  and the key bits into the register  $C$ , and other parameters remain unchanged, the loading procedure becomes

$$\begin{aligned} (x_1^0, x_2^0, \dots, x_{93}^0) &\leftarrow (0, \dots, 0, iv_0, \dots, iv_{79}), \\ (y_1^0, y_2^0, \dots, y_{84}^0) &\leftarrow (1, 1, 1, 0, \dots, 0), \\ (z_1^0, z_2^0, \dots, z_{111}^0) &\leftarrow (0, \dots, 0, k_0, \dots, k_{79}). \end{aligned}$$

With the tool numeric mapping, we can estimate the maximum round  $R$  of Loc-Trivium. The results are listed in the second column of Table 5. We can see that the key bits loaded into the register  $A$  and the IV bits loaded into the register  $B$  are not the best choice in view of the algebraic degree. The best choice is to load the key bits into register  $A$  and the IV bits into the register  $C$  and the maximum round is 814 smaller than 837.

**Table 5.**  $R$  with different key and IV loading locations

<i>Key, IV location</i>	$R$ (taps unchanged)	$R$ (taps changed)
$(A, B) \leftarrow (Key, IV)$	837	863
$(C, B) \leftarrow (Key, IV)$	828	864
$(A, C) \leftarrow (Key, IV)$	814	904
$(B, C) \leftarrow (Key, IV)$	825	910
$(B, A) \leftarrow (Key, IV)$	825	876
$(C, A) \leftarrow (Key, IV)$	825	853

If we also change the taps from the register loaded with the IV bits as shown in Sect. 5.1, we can get similar results, except the complementary variables. The feedback functions at time  $t$  turn into

$$\begin{aligned} x_{93}^t &= z_1^{t-1} + z_\alpha^{t-1} \cdot z_{\alpha+\delta}^{t-1} + z_{46}^{t-1} + x_\gamma^{t-1}, \\ y_{84}^t &= x_1^{t-1} + x_\alpha^{t-1} \cdot x_{\alpha+\delta}^{t-1} + x_\beta^{t-1} + y_7^{t-1}, \\ z_{111}^t &= y_1^{t-1} + y_\alpha^{t-1} \cdot y_{\alpha+\delta}^{t-1} + y_{16}^{t-1} + z_{25}^{t-1}. \end{aligned}$$

The complementary variables are  $iv_{i-\delta}$ ,  $iv_{i+\delta}$ ,  $iv_{i+\beta-\delta-1}$  and  $iv_{i+\beta+\delta-1}$  corresponding to the cube variable  $iv_i$ . When  $i + \beta + \delta - 1 > 79$ ,  $iv_{i+\beta+\delta-113+\gamma}$  will be multiplied with  $iv_{i+\delta-113+\gamma}$  according to the propagation paths of the IV bits. Also, when  $i + \beta - \delta - 1 > 79$ ,  $iv_{i+\beta-\delta-113+\gamma}$  will be multiplied with  $iv_{i-\delta-113+\gamma}$ .

In the third column of Table 5, we list the maximum rounds  $R$  when the corresponding parameters are also changed. The result shows that Loc-Trivium with the key loaded in the register  $B$  and IV loaded in the register  $C$  has the maximum rounds 910, which is the worst case. The corresponding feedback functions are

$$\begin{aligned}
x_{93}^t &= z_1^{t-1} + z_2^{t-1} \cdot z_3^{t-1} + \mathbf{z}_3^{\mathbf{t}-1} + x_{25}^{t-1}, \\
y_{84}^t &= x_1^{t-1} + x_2^{t-1} \cdot x_3^{t-1} + x_{28}^{t-1} + y_7^{t-1}, \\
z_{111}^t &= y_1^{t-1} + y_2^{t-1} \cdot y_3^{t-1} + y_{16}^{t-1} + \mathbf{z}_1^{\mathbf{t}-1}.
\end{aligned}$$

Here some tap positions are the same. Ensure that all tap positions are distinct, we obtain that the maximum rounds is 906 and the corresponding feedback functions are

$$\begin{aligned}
x_{93}^t &= z_1^{t-1} + z_2^{t-1} \cdot z_3^{t-1} + \mathbf{z}_5^{\mathbf{t}-1} + x_{25}^{t-1}, \\
y_{84}^t &= x_1^{t-1} + x_2^{t-1} \cdot x_3^{t-1} + x_{28}^{t-1} + y_7^{t-1}, \\
z_{111}^t &= y_1^{t-1} + y_2^{t-1} \cdot y_3^{t-1} + y_{16}^{t-1} + \mathbf{z}_4^{\mathbf{t}-1}.
\end{aligned}$$

The results show that the key and IV loading locations play important roles in the algebraic degree of NFSR-based cryptosystems. When a new cipher with the similar structure is designed, both the parameters and the key and IV loading locations should be taken into account. The worst case that the maximum rounds is 910 or 906, which should be avoided to be resistant to cube attacks or cube tests when new ciphers are designed.

## 6 Conclusions

In this paper, we have shown a new cube searching method. The main idea is to reduce the searching space through controlling the propagation of the IV bits. With our cube searching method and the algebraic degree estimated method numeric mapping, we can confirm the maximum numbers of initialization rounds of some NFSR-based cryptosystems such that the generated keystream bit does not achieve the maximum algebraic degree. As illustrations, we applied our method to Trivium to verify the validity, our searching space is much smaller than that of the existing results. We also applied our method to two Trivium-variants, named Par-Trivium and Loc-Trivium, and we can get an upper bound of the maximum initialization rounds.

We believe that our method is useful in both cryptanalysis and design of NFSR-based cryptosystems. In design of Trivium-Like stream cipher, it is needed to choose  $\alpha$ ,  $\beta$ ,  $\gamma$  and  $\delta$  as big as possible on the premise of the security against other attacks, and  $\beta^*$  and  $\gamma^*$  should be avoided. The experiments show that the maximum round of Par-Trivium is 863 which is the worst case. For the key and IV loading locations, if other parameters are not changed, the loading locations used in Trivium are the worst case, but it doesn't threaten the security. If other parameters are also changed, the worst and best choices are also given. For Loc-Trivium, the maximum initialization round of all considered Trivium-variants is 910 which is the worst case and should be avoided to be resistant to cube attacks or cube tests.

**Acknowledgment.** The authors would like to thank the anonymous reviewers for their comments and suggestions which significantly improve the quality and presentation of this paper. This work was supported by the National Natural Science Foundation of China (Grant No. 61672516), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDA06010701), and the Fundamental Theory and Cutting Edge Technology Research Program of Institute of Information Engineering, CAS (Grant No. Y7Z0331102).

## References

1. Dinur, I., Güneysu, T., Paar, C., Shamir, A., Zimmermann, R.: An experimentally verified attack on full Grain-128 using dedicated reconfigurable hardware. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 327–343. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-25385-0\\_18](https://doi.org/10.1007/978-3-642-25385-0_18)
2. Dinur, I., Morawiecki, P., Pieprzyk, J., Srebrny, M., Straus, M.: Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 733–761. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46800-5\\_28](https://doi.org/10.1007/978-3-662-46800-5_28)
3. 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). [https://doi.org/10.1007/978-3-642-01001-9\\_16](https://doi.org/10.1007/978-3-642-01001-9_16)
4. 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). [https://doi.org/10.1007/978-3-642-21702-9\\_10](https://doi.org/10.1007/978-3-642-21702-9_10)
5. Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995). [https://doi.org/10.1007/3-540-60590-8\\_16](https://doi.org/10.1007/3-540-60590-8_16)
6. Moriai, S., Shimoyama, T., Kaneko, T.: Higher order differential attack of a CAST cipher. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 17–31. Springer, Heidelberg (1998). [https://doi.org/10.1007/3-540-69710-1\\_2](https://doi.org/10.1007/3-540-69710-1_2)
7. Huang, S., Wang, X., Xu, G., Wang, M., Zhao, J.: Conditional cube attack on reduced-round Keccak sponge function. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 259–288. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56614-6\\_9](https://doi.org/10.1007/978-3-319-56614-6_9)
8. Li, Z., Dong, X., Wang, X.: Conditional cube attack on round-reduced ascon. IACR Trans. Symmetric Cryptol. **1**, 175–202 (2017)
9. Bi, W., Li, Z., Dong, X., Li, L., Wang, X.: Conditional cube attack on round-reduced River Keyak. Des. Codes Crypt. **86**(6), 1295–1310 (2018)
10. Courtois, N.T.: Fast algebraic attacks on stream ciphers with linear feedback. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 176–194. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-45146-4\\_11](https://doi.org/10.1007/978-3-540-45146-4_11)
11. 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). [https://doi.org/10.1007/3-540-36178-2\\_17](https://doi.org/10.1007/3-540-36178-2_17)
12. Knudsen, L., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-45661-9\\_9](https://doi.org/10.1007/3-540-45661-9_9)
13. Boura, C., Canteaut, A., De Cannière, C.: Higher-order differential properties of KECCAK and *Luffa*. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 252–269. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-21702-9\\_15](https://doi.org/10.1007/978-3-642-21702-9_15)

14. Canteaut, A., Videau, M.: Degree of composition of highly nonlinear functions and applications to higher order differential cryptanalysis. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 518–533. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-46035-7\\_34](https://doi.org/10.1007/3-540-46035-7_34)
15. Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 287–314. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46800-5\\_12](https://doi.org/10.1007/978-3-662-46800-5_12)
16. Todo, Y., Isobe, T., Hao, Y., Meier, W.: Cube attacks on non-blackbox polynomials based on division property. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 250–279. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63697-9\\_9](https://doi.org/10.1007/978-3-319-63697-9_9)
17. De Cannière, C., Preneel, B.: TRIVIUM. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 244–266. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-68351-3\\_18](https://doi.org/10.1007/978-3-540-68351-3_18)
18. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of Grain-128 with optional authentication. *Int. J. Wirel. Mob. Comput.* **5**(1), 48–59 (2011)
19. Wu, H.: ACORN: A Lightweight Authenticated Cipher (v3) (2016). <http://competitions.cr.yup.to/round3/acornv3.pdf>
20. Liu, M.: Degree evaluation of NFSR-based cryptosystems. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 227–249. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63697-9\\_8](https://doi.org/10.1007/978-3-319-63697-9_8)
21. Hell, M., Johansson, T., Maximov, A., Meier, W.: The Grain family of stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 179–190. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-68351-3\\_14](https://doi.org/10.1007/978-3-540-68351-3_14)
22. Canteaut, A., et al.: Stream ciphers: a practical solution for efficient homomorphic-ciphertext compression. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 313–333. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-52993-5\\_16](https://doi.org/10.1007/978-3-662-52993-5_16)
23. Chakraborti, A., Chattopadhyay, A., Hassan, M., Nandi, M.: TriviA: a fast and secure authenticated encryption scheme. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 330–353. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48324-4\\_17](https://doi.org/10.1007/978-3-662-48324-4_17)
24. Raddum, H.: Cryptanalytic results on Trivium. eSTREAM, ECRYPT Stream Cipher Project, Report, 39 (2006)
25. Schilling, T.E., Raddum, H.: Analysis of Trivium using compressed right hand side equations. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 18–32. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-31912-9\\_2](https://doi.org/10.1007/978-3-642-31912-9_2)
26. Fu, X., Wang, X., Dong, X., Meier, W.: A Key-recovery Attack on 855-round Trivium. IACR Cryptology ePrint Archive 2018, 198 (2018)
27. 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). [https://doi.org/10.1007/978-3-662-43933-3\\_26](https://doi.org/10.1007/978-3-662-43933-3_26)

# **Data Privacy and Anonymization**



# Practical Attacks on Relational Databases Protected via Searchable Encryption

Mohamed Ahmed Abdelraheem<sup>1</sup>(✉), Tobias Andersson<sup>2</sup>,  
Christian Gehrman<sup>3</sup>, and Cornelius Glackin<sup>1</sup>

<sup>1</sup> Intelligent Voice Ltd., London, UK  
[moh.ahm.abdelraheem@gmail.com](mailto:moh.ahm.abdelraheem@gmail.com)

<sup>2</sup> RISE SICS, Lund, Sweden

<sup>3</sup> Lund University, Lund, Sweden

**Abstract.** Searchable symmetric encryption (SSE) schemes are commonly proposed to enable search in a protected unstructured documents such as email archives or any set of sensitive text files. However, some SSE schemes have been recently proposed in order to protect relational databases. Most of the previous attacks on SSE schemes have only targeted its common use case, protecting unstructured data. In this work, we propose a new inference attack on relational databases protected via SSE schemes. Our inference attack enables a passive adversary with only basic knowledge about the meta-data information of the target relational database to recover the attribute names of some observed queries. This violates query privacy since the attribute name of a query is secret.

## 1 Introduction

Searchable symmetric encryption (SSE) schemes provide one of the practical solutions for searching on encrypted data. It was firstly proposed by Song et al. in [26] and later improved by Curtmola et al.'s [10]. Based on Curtmola et al.'s security model many SSE schemes were proposed such as [7–9]. The efficiency of SSE schemes comes at the cost of some leakage that might make them vulnerable to inference attacks [6, 15].

Traditionally, SSE schemes are designed to protect a set of unstructured documents (e.g. email archives or a backup or any set of sensitive text files). However, recently Cash et al. proposed an elegant scheme [8] that targets relational databases by increasing the functionality through supporting a large subset of Boolean queries. Their scheme also has good performance as it achieves a query speed that is comparable to the query speed under the unprotected MySQL (release 5.5) but with a storage cost of up to seven times the unencrypted data [8, 14].

One can argue that SSE schemes offer better security than other encrypted search schemes such as deterministic or order preserving encryption schemes

---

M. A. Abdelraheem—Most of this work was done while the author was a postdoc at RISE SICS in Sweden.

since they do not leak the frequency of an attribute-value pair before querying it. The security of SSE schemes has been well studied against inference attacks [6, 15] in their traditional use case scenario where document datasets such as email archives are protected. More recently, the work in [1] presented the first security analysis of searchable encrypted relational databases (i.e. relational databases protected via SSE) by proposing an inference attack, called *Relational-Count*, exploiting the structural properties of relational databases and using only knowledge about the frequency distribution of the attribute-value pairs which is much less than the prior knowledge (i.e. joint frequency of attribute-value pairs which is equivalent to knowledge of the whole plaintext database) required by the *Count* attack [6].

However, in this work we further improve the analysis of searchable encrypted relational databases and show that there is an extra leakage that could reveal the attribute name (also known as column name) of all the queries (i.e. encrypted attribute-value pairs) belonging to a specific database column.

Henceforth, we will use the words “query” and “query token” interchangeably to mean an encrypted attribute-value pair that has been queried (or an encrypted keyword in the context of unstructured databases). Thus, query recovery means finding the actual plaintext of the query which is an attribute-value pair (or a keyword). Also, relational database tables protected via SSE schemes will either be called SSE-protected databases or searchable encrypted relational databases. Finally, the words “attribute” and “column” will also be used interchangeably throughout the paper.

**Our Contribution.** The contribution of this paper is twofold. *First*, we propose a new attack on searchable encrypted relational databases. Our attack recovers the attribute names or column names of queries and thus we call it the *Attribute-Name recovery* attack. Our attack breaks the query privacy by recovering the secret attribute names of some of the observed queries. We propose two simple algorithms (one is deterministic and the other is heuristic), that take as input the set of issued queries and divide it into different subsets where each subset represents all the values belonging to a specific column. Table 2 shows the number of columns recovered by our attack on three different real world databases protected by an SSE scheme. It is apparent from Table 2 that for the Adult [19] and the Bank [25] database tables which have 14/32561 and 17/4521 columns/records respectively, our attack recovers almost all the columns and thus completely breaks the privacy of queries. However, regarding larger database tables such as the Census which has 40/299285 columns/records, we are able to recover slightly more than half of the columns. Most notably, our attack works using only the meta-data information (i.e. column names and their cardinalities where the cardinality of a column is defined as the number of unique elements in the column) and the number of records about the protected relational database table under attack. This lesser amount of required knowledge makes our Attribute-Name recovery attack more applicable in practice than all previous attacks on SSE schemes [1, 6, 15] as they require at least prior knowledge about the frequency distribution of the keywords or attribute-value pairs in the target database.

*Second*, we combine our Attribute-Name recovery attack with the *Relational-Count* attack [1]. This gives us a second attack that recovers the values of queries but requires knowledge about the frequency distribution of the attribute-value pairs in the target database. The results about the number of recovered queries in different SSE-protected relational databases are shown in Table 3.

**Related Work.** Query recovery attacks exploiting the access pattern leakage of SSE schemes were proposed by Islam et al. [15] and later improved by Cash et al. [6]. The recent inference attack on searchable encrypted relational databases by Abdelraheem et al. [1] exploits the properties of relational databases. This reduces the attacker’s knowledge from full database knowledge required by previous inference attacks [6, 15] to only partial knowledge represented by the frequency distribution of the attribute-value pairs in the target database. Unlike Abdelraheem et al.’s *Relational-Count* attack [1] which still requires the frequency distribution of the attribute-value pairs in the target database, our first proposed attack recovers the attribute names of queries by requiring only the attacker’s knowledge about the meta-data information of the target protected database. Moreover, as shown in Table 3, our second proposed attack recovers more queries than those recovered by Abdelraheem et al.’s *Relational-Count* attack [1].

Naveed et al. [22] proposed a number of attacks targeting relational database columns encrypted using deterministic encryption [4] or order preserving encryption algorithms [2, 5] in CryptDB [24] where encrypted values belonging to the same column whose name is encrypted are collected together as one set or more precisely one column vector. While the approach seems similar, their column finder procedure is significantly different than our attribute recovery attack. It takes as input the set of encrypted values (i.e. column vector of encrypted values) belonging to the same unknown column whose name is encrypted. Their approach recovers the encrypted name by matching the number of distinct encrypted values with each column’s cardinality defined in the set of plaintext columns (i.e. a column name and its possible values are known) belonging to the attacker’s auxiliary or background data. Their procedure relies mainly on the attributes’ (i.e. columns) cardinalities.

In contrast, our attack described in Algorithm 1 takes as input all the observed encrypted queries in an SSE scheme. Then using the access pattern leakage inherent in SSE schemes in addition to basic background data about the number of records and attributes’ cardinalities, it divides the observed encrypted queries into different classes where each class contains a set of encrypted queries belonging to the same attribute. Thus, each class or set of encrypted queries found by our attack is actually the input used by Naveed et al.’s column finder procedure.

The recent generic attacks [17] proposed at CCS 2016 target any secure database systems supporting range queries but leaking the access pattern without any prior knowledge about the database under attack. Most notably, one of their generic attacks target even secure encrypted search methods supporting range queries and only leaking the communication volume (i.e. query result



size) such as fully homomorphic encryption or ORAM schemes. Their attacks only target range queries and they require the attacker to gather at least  $N^4$  queries ( $N$  is the domain size) to mount the attack successfully. However, our work targets relational databases protected by SSE schemes dealing with equality queries. There are only two things in common between our attack and theirs, and that is the requirement of knowing the number of records and the query result size (or communication volume as defined in [17]).

**Organization of the Paper.** Section 2 gives a brief overview of SSE schemes and inference attacks. In Sect. 3, we point out the security risks of using SSE schemes in relational databases by proposing a new frequency attack exploiting the properties of relational databases. In Sect. 4, we present detailed experimental results demonstrating our attacks and then discuss some countermeasures. Section 5 concludes the paper.

## 2 Background About SSE Schemes

**Definition.** An SSE scheme takes as inputs a plaintext database index together with the client's secret keys and outputs an encrypted index where each keyword  $w$  in the plaintext index is transformed into a token  $t$  using a deterministic encryption algorithm and its corresponding documents' identifiers are encrypted using a randomized encryption algorithm. An SSE scheme encrypts the original documents by employing a randomized encryption algorithm using the client's secret keys and stores the encrypted documents in an encrypted database indexed by the document identifiers. Both the encrypted index and the encrypted documents database are sent to the cloud server. To search for a keyword  $w$ , the client issues a query by generating its token  $t$  using its secret keys and sends it to the server. The server responds by sending the corresponding encrypted documents from the encrypted documents database to the client.

Our focus will be on analyzing the security of relational databases protected by SSE schemes that are adaptively secure as defined by Curtmola et al. [10]. The use of SSE schemes to protect SSE schemes was firstly proposed by Cash et al. [7,8] where the database records are considered as documents and the attribute-value pairs are considered as keywords (e.g. each keyword is represented as  $w_i = (\text{attribute}_i, v_i)$  where  $\text{attribute}_i$  is the attribute name and  $v_i$  is its value).

**Leakage Profile.** An SSE scheme leaks the *access pattern*: the result of the query or the record (or document) IDs corresponding to the queried keyword  $w_i$ ,  $\text{DB}(w_i)$ , and also leaks the *search pattern*: the fact that whether two searches are the same or not.

**Attack Model.** All recent SSE schemes follow the adaptive security definition proposed by Curtmola et al. [10] where security is achieved against an honest-but-curious server. That means a passive adversary following the protocol but curious to use the leakage profile to learn about the queries and the encrypted records.

**Inference Attacks on SSE Schemes.** Classical ciphers were broken by frequency analysis which is a standard example of an inference attack where an attacker can recover a plaintext character by inferring some information about its corresponding ciphertext character using language statistics. Similarly, using publicly-available auxiliary data, an inference attack can be mounted on adaptive SSE schemes to recover the plaintext of queries issued by the client and observed by the attacker (e.g. honest-but-curious server or passive external attacker). This kind of attack performs *query recovery* and was proposed by Islam, Kuzu, and Kantarcioglu (IKK) in [15]. Their attack, known in the literature as the *IKK* attack, targets the strongest kinds of SSE schemes which are those proved to be secure under the adaptive security definition. The *IKK* attack assumes knowledge about the *joint frequency (or co-occurrence count)* of any two plaintext attribute-value pairs (or two keywords)  $w_i, w_j \in W$ . It also assumes knowledge about the plaintext of a subset of queries issued by the client.

A similar inference attack has been proposed recently by Cash et al. is called the *Count* attack [6]. The *Count* attack assumes the attacker’s knowledge about the number of occurrences of each keyword/attribute-value over all the documents/records (i.e.  $|\text{DB}(w_i)|$  where  $w_i \in W$  and  $\text{DB}$  is the original plaintext dataset). This is also called keyword frequency knowledge and we denote this knowledge by  $\mathcal{K}_{\mathcal{F}}$ . Similar to the *IKK* attack, it also assumes knowledge about the joint frequency (or co-occurrence count) of any two plaintext attribute-value pairs (or two keywords)  $w_i, w_j \in W$ . Both, the *IKK* and the *Count* attacks, represent the joint frequency knowledge in a matrix called the *co-occurrence knowledge-matrix*,  $C_w$ . Therefore, both attacks require a complete knowledge about the plaintext dataset under attack in order to form the co-occurrence knowledge-matrix. Both attacks exploit the access pattern leakage inherent in SSE schemes, in this case enabling the computation of the result size of any observed query and also the observed joint frequency of any two observed queries. In the relational database setting, this is equal to the size of the set resulting from the intersection between the result sets of the two queries. A joint *co-occurrence query-matrix*,  $C_t$ , is then formed and compared to the *co-occurrence knowledge-matrix*,  $C_w$  in both attacks.

The recent *Relational-Count* [1] inference attack also performs query recovery but on searchable encrypted relational databases. Its query recovery rate is less than the *Count* attack but it is more practical than the *Count* attack since it requires only the knowledge of the frequency distribution of the plaintext attribute-value pairs in the target database rather than the joint frequency knowledge required by the *Count* attack. The difference lies in the fact that the *Relational-Count* attack uses Observation 1 (See Sect. 3) to discard the wrong candidates in the list of possible candidates for a query with non-unique result size in contrast to the *Count* attack which uses prior joint frequency knowledge to discard wrong candidates from the same list.

### 3 Attacking Searchable Encrypted Relational Databases

As demonstrated in the recent *Relational-Count* inference attack [1], the structure of relational databases does enable an attack on searchable encrypted relational databases without resorting to the co-occurrence knowledge-matrix  $C_w$  about the relational database under attack. In the following, we describe our Attribute-Name recovery attack. Our attack works under the assumption that enough queries have been observed and that the attacker knows the meta-data information about the target database (e.g. column names  $\mathcal{A}$  and the cardinality of each column which is equivalent to the number of unique values in the column).

Note that the meta-data information represents the least possible prior knowledge that could be acquired by an attacker. We define this as the attacker’s basic background knowledge  $\mathcal{K}_B$ . This is definitely much less than the prior knowledge required by previously discussed inference attacks and could be guessed or acquired from public data. For example, if an honest-but-curious server holds an encrypted medical data, then an attacker can easily acquire the columns’ or attributes’ names of the protected data by referring to publicly-available information such as the meta-data about database tables used in standard medical software applications such as OpenEMR [23]<sup>1</sup>.

In addition to this basic knowledge, our attack assumes also the attacker’s knowledge of the number of records in the target database which can be dynamic depending on the protected database under attack. The number of records could be gained either through a guess-and-determine process especially for small or medium sized databases or through a leakage of the SSE scheme under attack which is the case in some notable SSE schemes such as [9, 16, 20, 27]. Moreover, our attack could also recover the attribute value of a given attribute name ‘ $a$ ’ under the assumption that the attacker knows  $\mathcal{K}_F$ .

#### 3.1 Attribute-Name Recovery Attacks (First Attack)

We make use of the following simple observation, proposed in [1], about the joint frequency (or the co-occurrence count) of observed queries sharing the same attribute name on searchable encrypted relational databases.

**Observation 1.** *The observed joint frequency (or observed co-occurrence count) between any two different queries is non-zero only when their corresponding attribute names are different.*

The observation should be clear from the fact that each relational database record has only one value for each column or attribute name. For example, let  $t_1$  be the query token corresponding to the plaintext attribute-value pair “Sex : Male”,  $t_2$  be the query token corresponding to the plaintext attribute-value pair “Sex : Female”, and  $t_3$  be the query token corresponding to the plaintext

<sup>1</sup> OpenEMR is a well known open source medical software supporting Electronic Medical Records (EMR).

attribute-value pair “Age : 18”. Now, the observed joint frequency of  $t_1$  and  $t_2$  must be zero as there cannot be a relational database record whose “Sex” value is both “Male” and “Female”. Also there is no guarantee that the observed joint frequency of  $t_1$  and  $t_3$  (or  $t_2$  and  $t_3$ ) is zero since their corresponding attribute names are different.

More generally, Observation 1 might allow an attacker to answer the following question “Do the queries  $t_i$  and  $t_j$  have the same attribute name?”. Here the attacker does not need any knowledge other than the observed access pattern leakage  $C_t$ . If the value  $C_t[t_i, t_j]$  does not equal zero, then  $t_i$  and  $t_j$  definitely have different attribute names. Otherwise, the attacker cannot answer.

Our first proposed attack, the attribute-name recovery attack, is based on Observation 2 which follows immediately from Observation 1 and the fact that the total frequency of all the domain values of an attribute  $a$  in a searchable encrypted relational database (EDB) equals the number of records in EDB. Going back to our previous example and assuming that the number of records in EDB is  $n$ , one can see that since  $t_1$  and  $t_2$  have the same attribute name and  $|\text{Sex}| = 2$  (i.e. the cardinality of “Sex”) then the following equation will be satisfied:  $|t_1| + |t_2| = n$ . This gives an example that explains the following observation.

**Observation 2.** *Let  $\mathbf{t} = \{t_1, \dots, t_l\}$  be the set of observed queries where  $l \leq |W|$ . Let  $|t_i|$  denote the result size of query token  $t_i$ . Let  $C_t$  be the observed co-occurrence query-matrix. Let  $n$  be number of records in a searchable encrypted relational database EDB. Let ‘ $a$ ’ be an attribute name in the EDB with cardinality  $|a|$ . Then there exists a subset<sup>2</sup>  $\mathbf{s} \subseteq \mathbf{t}$  where  $\sum_{t_i \in \mathbf{s}} |t_i| = n$ ,  $|\mathbf{s}| = |a|$ , and  $\forall t_i, t_j \in \mathbf{s}, C_t[t_i, t_j] = 0$  when  $l = |W|$ .*

Observation 2 can be used to develop an algorithm that distinguishes between observed queries. Such a distinguisher algorithm can be mounted by a weak attacker (hence a strong attack) who observes only the access pattern leakage of queries and has no prior knowledge other than  $\mathcal{K}_B$  (i.e. number of records, column names and their cardinalities).

**Description of Algorithm 1.** Algorithm 1 takes as input a set of observed query tokens  $\mathbf{t} = \{t_1, \dots, t_l\}$  and the attacker’s basic background knowledge  $\mathcal{K}_B$ . It divides and classifies these tokens according to their attribute names into different sets where each set  $G_a$  belongs to one attribute name  $a = (a_i)$  or multiple attributes names  $a = (a_i, a_j)$  sharing the same cardinality  $|a_i|$ . Note that the pseudo-code of Algorithm 1 provides only an overview about the steps and does not represent a description of our implementation. In the following, we discuss how to efficiently implement Algorithm 1.

Step 7 and Step 8 can give us an idea about the time complexity of Algorithm 1 since they form the known  $k$ -SUM problem (i.e. Given  $A = \{a_1, \dots, a_s\}$  and a target sum  $t$ . Is there any subset of indices  $\{i_1, \dots, i_k\}$  such that  $\sum_{j=1}^k |a_{i_j}| = t$ ?) with an additional condition that the observed joint frequency (or co-occurrence count) value between any two elements in the subset is

<sup>2</sup> If the cardinality  $|a|$  is not unique, then  $k$  subsets will exist where  $k$  is the number of attributes whose cardinalities are equal to  $|a|$ .

**Algorithm 1.** Attribute Recovery Attack

---

**Require:**  $\mathcal{K}_{\mathcal{B}}$  and observed query tokens  $\mathbf{t} = \{t_1, \dots, t_l\}$  and their results.  $|a| \equiv$  cardinality of  $a \in \mathcal{A}$  where  $a$  can represent a single attribute name or multiple attribute names sharing the same cardinality  $|a|$ .  $|t_i| \equiv$  result size of query token  $t_i$ .

**Ensure:** Recover the attribute name of observed queries.

- 1: Set  $R = \{\}$ . Compute the co-occurrence query-matrix  $C_t$  from query tokens  $\mathbf{t}$ .
- 2: For each  $t_i$ , create a list  $Q_i$  holding  $t_i$  in its head (i.e.  $Q_i[1] = t_i$ ) and all other query tokens  $t_j$ 's where  $C_t[t_i, t_j] = 0$ .
- 3: Sort all the lists  $Q_i$  according to their size in ascending order. Add all the sorted lists  $Q_i$ 's to a lists' container  $L$  where  $L[i, j]$  is the  $j$ th query in the  $i$ th list  $L[i]$ .
- 4: Set  $G_a \leftarrow \{\}$  for each attribute  $a$  and  $\text{ctr} = 1$ .
- 5: **while**  $\text{ctr} \leq l$  **AND**  $\mathcal{A} \neq \emptyset$  **do**
- 6:      $L[\text{ctr}] \leftarrow L[\text{ctr}] \setminus R$
- 7:     **for all**  $S \subseteq L[\text{ctr}]$  **where**  $L[\text{ctr}, 1] \in S$  **do**
- 8:         **if**  $\sum_{t_u \in S} |t_u| = n$  **and**  $C_t[t_u, t_v] = 0, \forall t_u, t_v \in S$  **then**
- 9:             **for all**  $a \in \mathcal{A}$  **do**
- 10:                 **if**  $|S| = |a|$  **then**
- 11:                      $G_a \leftarrow G_a \cup S$
- 12:                      $R \leftarrow R \cup G_a$
- 13:                      $\mathcal{A} \leftarrow \mathcal{A} \setminus a$
- 14:                     **print**("Possible solution for cardinality ",  $|a|$ , " is ",  $G_a$ )
- 15:                     **goto** Step 16
- 16:      $\text{ctr} = \text{ctr} + 1$
- 17: **return**  $R$ .

---

zero. The  $k$ -Sum problem is a parameterized version of the subset sum problem which is a known NP-complete problem. The brute force algorithm for the  $k$ -SUM problem takes  $O(s^k)$  where  $s$  is the size of the given set. There are simple algorithms solving this problem in  $O(s^{\frac{k}{2}} \log s)$  when  $k$  is even and  $O(s^{\frac{k+1}{2}})$  when  $k$  is odd [3, 11, 13].

However, employing the observed joint frequency (or co-occurrence count) condition might reduce the above complexity times but this needs further investigation. Obviously, Algorithm 1 performs well when  $k$  is small (i.e. the attribute cardinality is small). When the target sum  $t$  is not very large (i.e. number of records  $n$  is not very large), one can use the known dynamic programming technique to solve the subset sum problem in pseudo-polynomial time  $O(st)$  [18].

Another way to look at Algorithm 1 is to consider the co-occurrence query-matrix as the adjacency matrix of a weighted graph  $G_{\mathcal{C}_0}$  whose nodes are the queries and any two nodes are connected by an edge whose weight is the observed joint frequency (or co-occurrence count) value between the two connected nodes (i.e. a zero value for the joint frequency (or co-occurrence count) means no edge or edge with weight zero). This allows us to consider the elements of the list  $L[\text{ctr}]$  created in Algorithm 1 as nodes in another smaller weighed graph  $G_{L[\text{ctr}]}$  whose adjacent matrix is a submatrix of the co-occurrence query-matrix (i.e. adjacent matrix of the bigger graph  $G_{\mathcal{C}_0}$  containing all queries).

Now rather than looking at the subsets of  $L[\text{ctr}]$  (Note that the first element  $L[\text{ctr}, 1]$  should be included in all subsets) whose size is equivalent to a given cardinality  $|a|$ , one can look at the independent sets of the graph  $G_{L[\text{ctr}]}$  corresponding to the list  $L[\text{ctr}]$  whose size is equivalent to  $|a|$  with the additional condition that the total sum of the frequency of each node (i.e. query) equals the total number of records  $n$ . This is the known independent set NP-Complete problem with an additional filtering condition. We have implemented Step 8 in Algorithm 1 using a greedy solution to the independent set problem where a node with the maximum degree is included in the solution and all its neighbors are discarded (See Sect. 4). Appendix A gives an example where the queries are represented in a graph and the problem of finding the attribute name of each query is equivalent to finding solutions to the independent set problem of a given graph.

Now, one can ask, which approach (i.e. independent set algorithms or  $k$ -SUM algorithms or dynamic programming of subset sum) is better to implement Step 7 and Step 8 in Algorithm 1. The answer depends on the target database table and the available queries. We have implemented the dynamic programming algorithm for the subset problem where all the solutions are traced back and the joint frequency (or co-occurrence count) condition is evaluated after finding each solution. This is practical when  $O(st)$  is pseudo-polynomial which means that the number of records  $t$  is not large and for each cardinality there exists a constructed list  $L[\text{ctr}]$  whose size  $s$  is not large.

However, Algorithm 1 will not be practical to apply when the constructed lists are large. Therefore, we propose Algorithm 2, which is an efficient heuristic algorithm that can find valid solutions faster. Algorithm 2 takes as input the constructed lists  $L[\text{ctr}]$ , the number of records, and the observed query-matrix. The algorithm is based on the observation that “*the intersection between lists whose heads belong to a similar attribute name gives a list containing queries whose result sizes add up to the number of records*”. The problem here lies in identifying the lists whose heads belong to the same attribute name. Our heuristic approach allows the intersection between lists whose heads are queries that are disjoint. Its time complexity is  $O(|W|^2)$  where  $|W|$  is the number of attribute-value pairs which is equivalent also the number of lists  $L[i]$ .

Unlike Algorithm 1 which needs to find all the solutions of a subset sum problem (see Step 8 in Algorithm 1), the heuristic approach only needs to find the intersection (see Step 8 in Algorithm 2) between two sorted sets at each iteration which takes  $O(m_1 \log(m_2))$  where  $m_1$  is the size of the small list and  $m_2$  is the size of the larger. For each list  $L[i]$ , it outputs a possible solution  $R[i]$ . Note that a correct solution  $R[i]$  is more likely to appear and to be confirmed many times but not more than  $|R[i]|$  times. No doubt false positives will appear especially for columns with small cardinalities, even if the number of confirmations for a solution  $R[i]$ ,  $C[R[i]]$ , is exactly  $|R[i]|$ . The order of lists  $L[i]$  affects the results of the intersections, so the algorithm might not succeed in finding the right solution. So further work is required to study the success rate of our heuristic approach.

**Algorithm 2.** Heuristic Approach

---

```

1: procedure HEURISTIC( $L, C_t, n, \mathbf{t} = \{t_1, \dots, t_l\}, \mathcal{A}$ )
2:    $R = \{\}$ 
3:    $C = \{\}$  ▷ Hash table to map similar entries in R to one entry
4:   while  $L$  is decreasing do
5:     for all  $L[i]$  do
6:        $R[i] \leftarrow L[i]$ 
7:       for all  $L[j]$  where  $L[j, 1] \in L[i]$  do
8:          $S \leftarrow R[i] \cap L[j]$ 
9:         if  $\sum_{t_u \in S} |t_u| \geq n$  then
10:           $R[i] \leftarrow S$ 
11:          if  $\sum_{t_u \in R[i]} |t_u| = n$  &  $C_t[t_u, t_v] = 0, \forall t_u, t_v \in R[i]$  then
12:             $R[i] \leftarrow \text{Sort}(R[i])$  ▷ Sort according to query no.
13:            if  $R[i] \in C$  then
14:               $C[R[i]] \leftarrow C[R[i]] + 1$ 
15:            else
16:               $C[R[i]] \leftarrow 0$ 
17:          for all  $R[i]$  where  $|R[i]| \in |\mathcal{A}|$  &  $(R[i] \cap R \setminus R[i] = \phi)$  do
18:            if  $C[R[i]] \approx |R[i]|$  then
19:              print( $R[i]$ , “ holds queries of an attribute with card. ”,  $|R[i]|$ )
20:              for all  $L[j]$  do
21:                if  $L[j, 1] \in R[i]$  then
22:                   $L \leftarrow L \setminus L[j]$ 
23:                else
24:                   $L[j] \leftarrow L[j] \setminus R[i]$ 

```

---

However, experiments show that such false positive solutions are discarded by Step 17. As long as there are no intersections between solutions, any number of confirmations more than one will give us confidence that our solution is correct. In our experiments, for small cardinalities  $\leq 15$ , we require that the number of confirmations to be exactly equivalent to the number of confirmations. But for other cardinalities, we require that the number of confirmations  $\geq 10$ .

Experiments show that Algorithm 2 managed to recover many attributes in large databases such as the Census database [21] while Algorithm 1 succeeded in recovering almost all the columns (12/14) in the Adult database [19] which is a small database with only 498 attribute-value pairs.

Algorithms 1 and 2 break query privacy after observing enough queries using a less amount of required knowledge. To realize their effect, assume that an attacker has observed enough queries on a searchable encrypted relational database, then Algorithm 1 can give us an answer to: “Have all possible values about attribute ‘ $x$ ’ been queried?”. To answer such a question, an attacker needs to know the number of records  $n$  which could be possible as noted above. The attacker also needs to know the cardinality of  $x$ ,  $|x|$ , which could also be possible as many relational database tables are standard such as the sensitive OpenEMR [23] relational databases. Employing Algorithm 1, after observing enough queries, might return all the queries with the same attribute that have

result sizes whose sum is equivalent to  $n$ . If there is only one attribute whose cardinality equals  $|x|$ , then Algorithm 1 will yield one solution if all values of  $x$  have been queried. The ability to answer the above question does break the query privacy meant to be provided by using SSE and confirms the recent results [1] that the leakage resulting from protecting relational databases with SSE schemes is more than the leakage resulting from protecting unstructured data with SSE schemes.

### 3.2 Recovering Attribute Values (Second Attack)

Algorithms 1 and 2 might enable an attacker to recover the attribute names of some queries without any knowledge except the basic knowledge,  $\mathcal{K}_B$ . Now with  $\mathcal{K}_B$ , the attacker knows the domain or space values of a given attribute name. Moreover, with the access pattern leakage, the attacker knows the result set size (i.e. number of records holding the corresponding attribute-value pair for the query token) of each observed query.

However, in order to recover the attribute values of observed queries whose attribute names are recovered with Algorithm 1, without any prior joint frequency knowledge such as those used in the *Count* attack  $\mathcal{K}_F$  and  $C_w$ , an attacker needs to know at least the rank-size or rank-frequency distribution of the attribute values. Note that the knowledge of the rank-size or rank-frequency distribution of a given attribute value does not necessarily imply the knowledge of the frequency of each attribute value  $\mathcal{K}_F$ . For example, an attacker might know that in a certain country, from Census data, that the number of females exceeds the number of males without knowing the exact numbers.

Using rank-size distribution knowledge only instead of  $\mathcal{K}_F$  knowledge, an attacker can create a list  $L_a$  containing the attribute values of an attribute named  $a$  that is sorted according to their rank-size in descending order. Let  $L_q$  be a list containing the result-size of each query such that  $L_q[i]$  contains the result size of query token  $t_i$ . Let  $\text{Sort}(L_q)$  be the list obtained after sorting  $L_q$  in descending order. Let  $\text{Find}(\text{Sort}(L_q), L_q[i])$  be a function that gives the location corresponding to  $t_i$  in the sorted list. Then the value of the query token  $t_i$  will be  $L_a[\text{Find}(\text{Sort}(L_q), L_q[i])]$ . If all the result sizes of queries in  $L_q$  are unique, then the above attack succeeds with probability one. Otherwise, there might be an error whenever we have a tie in the result sizes between two or more queries in  $L_q$ .

Using  $\mathcal{K}_F$  knowledge which is a very strong assumption compared to the rank-size distribution knowledge, an attacker can populate a list of lists data structure, say  $L_a$ , where  $L_a[j]$  gives the value (or a list of values) of the attribute ‘ $a$ ’ whose frequency value (values) equals  $j$ . Assuming, for example, that Algorithm 1 has recovered the attribute name of a query token  $t_i$  to be ‘ $a$ ’, then one can see that by adding the result-size of observed queries to a dictionary  $D_q$  (i.e.  $D_q[t_i]$  gives the result-size of query token  $t_i$ ), then  $L_a[D_q[t_i]]$  will be the attribute value (or the possible attribute values) of an observed query token  $t_i$  whose attribute name is  $a$ . If each list in  $L_a$  (i.e.  $L_a[j]$ ) has size 1, then this process yields one value for the query token  $t_i$  whose frequency matches the result-size of  $t_i$ ,



$D_q[t_i]$ . Otherwise, it will return the list of all the possible values of the query token  $t_i$  which appear  $D_q[t_i]$  times over all the database records. The two above procedures are standard frequency analysis attacks similar to the one described by Naveed et al. [22] for attacking columns of a relational database encrypted using deterministic encryption.

However, our attacks target searchable encrypted relational databases by firstly recovering the attribute name of a given query using only the meta-data information about the databases and then secondly recovering its value when background knowledge about the databases (i.e. the distribution of the attribute-value pairs) under attack is known. Moreover, rather than firstly recovering all the attribute names of the issued queries and then secondly recovering all the values using the attacker’s frequency knowledge, we can use the frequency knowledge and use our attribute-name recovery attack and at the same time apply the *Relational-Count* attack proposed by Abdelraheem et al. [1]. Combining the *Attribute-Name Recovery* attack and the *Relational-Count attack* allows us to recover more queries. This combination represents our second proposed attack. See the graph on the right in Fig. 1 at Appendix A to see the effect of our second attack when combined with our first attack. In Sect. 4, we give some experimental results on the Bank database [25] where this combination significantly increases the number of recovered queries and closes the security gap between deterministic encryption and searchable encryption schemes.

## 4 Experimental Results and Countermeasures

In this section, we show the practical viability and threat posed by our first proposed attack (attribute-name recovery) as well as our second proposed attack which recovers the actual plaintext values of the queries.

Our first attack is represented by Algorithm 1 (subset sum approach or Independent set approach) and Algorithm 2. We also combine both Algorithms 1 and 2 (subset sum approach) by using Algorithm 2 first and then using the reduced and unresolved lists produced by Algorithm 2 as an input for Algorithm 1 (subset sum approach) in order to tackle the unresolved lists with small sizes. Our attack takes as input the access pattern of the observed queries (i.e. access pattern = record IDs and result size), which enables the attacker to compute the observed co-occurrence matrix  $C_t$ . It also takes as input the basic background information represented in *only* the meta-data information about the relational database under attack (i.e.  $\mathcal{K}_{\mathcal{B}}$ ). Our goal is to distinguish between queries belonging to different attribute names and gather all queries belonging to the same attribute in a single unit column. This will reduce that the security offered by SSE and make it close to the security offered by deterministic encryption when enough queries are issued which will enable an attacker with background knowledge about the actual database or its distribution to recover all queries and thus all attribute values of the target database as demonstrated by Naveed et al. [22] and discussed above in Sect. 3.2.

We tested our attacks against the three relational database tables used in [1], namely, the Adult [19], the Bank [25], and the Census [21]. The implementation

of our attacks can be accessed from the link <https://goo.gl/WxmSU4>. Before describing our experimental results, we give a short description about these three databases.

**Relational Databases.** The three relational database tables are publicly available online at the UCI Machine Learning Repository [12]: (1) Adult [19]: consists of 32561 rows (records), 14 columns (attributes), and has 498 distinct attribute-value pairs. (2) Bank [25]: consists of 4521 rows, 17 columns (attributes), and has 3720 distinct attribute-value pairs. (3) Census [21]: consists of 299285 rows, 40 columns (attributes), and has 4001 distinct attribute-value pairs.

**Query Generation.** Using a single-keyword Bitmap-based SSE scheme similar to the ones described in [15,27], each target relational database is transformed into a separate searchable encryption relational database where all possible queries within it are issued. The queries were chosen randomly from the set of all available queries (498 queries for the Adult database, 3720 queries for the bank database and 3993 queries for the Census database) by a client and each query result set and its access pattern leakage were recorded by an honest-but-curious server. Using this observed-queries knowledge, our attacker (i.e. the honest-but-curious server) can compute the joint frequency (or co-occurrence count) value between any two queries and thus the whole co-occurrence query-matrix,  $C_t$ .

**Sizes of the Sorted Lists.** Both Algorithms 1 and 2 depend on the lists  $L[ctr]$  whose sizes determine the time complexity of both algorithms. The sizes of the lists  $L[ctr]$  vary depending on the database under attack and the number of issued queries. For example, in the Adult database when all the 498 queries are issued, the smallest list has size 13 and the largest list has size 485 whereas in the Bank database when all the 3720 queries are issued, the smallest list has size 37 and the largest list has size 3704 and in the Census database when all the 4001 queries are issued, the smallest list has size 11 and the largest list has size 3962 (Table 1).

**Table 1.** Sizes of the Sorted Lists (increasing order) in each of the three databases (i.e. Adult, Bank and Census) when all the possible queries are issued in each protected database. The 1st list (L[1]) for the Adult database has size 13, the 2nd list (L[2]) has size 28 and so on. last refers to the last list which corresponds to the 498th list in the Adult database, the 3720th list in the Bank database and the 4001th list in the Census database.

Data Set	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...	last
Adult	13	28	41	45	52	67	89	89	93	118	118	119	129	135	136	...	485
Bank	37	288	374	662	662	662	770	852	954	1184	1465	1518	1812	1878	1963	...	3704
Census	11	25	25	30	34	55	107	110	111	127	127	152	157	197	198	...	3962

#### 4.1 Experimental Results of Our First Attack

**Experiments with Algorithm 1.** Algorithm 1 is implemented using two approaches. The first approach is the known dynamic programming with a backtracking procedure to solve the subset sum problem in each list  $L[\text{ctr}]$  and the second approach is a greedy algorithm that solves the independent set problem. Table 2 shows the results of using the two approaches of implementing Algorithm 1 on the above three databases. The results show that the subset sum implementation is more successful than the independent set implementation when the database table has a small number of attribute-value pairs such as the Adult database which has only 498 attribute-value pairs and thus at most 498 queries. So regarding the Adult database, all the columns or attribute names whose cardinalities are unique have been recovered successfully. However, columns with the same cardinality such as the “sex” and salary “class”, where each has two values, have been distinguished from the other attributes but one can not tell which of the two values point to the “sex” attribute and which point to the “class” attribute. Similarly, each of the “education” and “education-num” attributes has 16 values<sup>3</sup>. However, all their values were in one list ranked at position  $\text{ctr} = 13$  in the sorted lists’ container and at the same time each value in each attribute has zero joint frequency (or co-occurrence count) value with all the other values except one value. This makes it impossible to separate the values of each attribute as we did for the “sex” and “class” attributes. In fact, our dynamic programming implementation of Algorithm 1 generated exactly  $32678 = 2^{15}$  solutions. The reason is that the first element of each list is included in each solution but each of all the other 15 values has two possibilities which gives us in total  $2^{15}$  solutions. However, the set of all queries belonging to “education” and “education-num” will be identified and separated from the other queries but not from each other. In such scenarios Algorithm 1 fails completely to recover the attribute name of a class of queries.

Regarding the Census and the Bank databases, the greedy approach of the independent set implementation of Algorithm 1 is more effective as it recovers more columns. Note that the subset sum approach is deterministic but it takes too long time to resolve a single list  $L[\text{ctr}]$  with a large size. For example, we are only able to resolve the first three lists in the Bank database using the subset sum implementation but we cannot resolve the 4th ranked list in the Bank database whose size is 662 and all the other lists corresponding to the Bank database in a reasonable time using the subset sum implementation. However, when we use the greedy approach for the independent set (note that this approach is heuristic so we check the intersections between solutions similar to Step 17 in Algorithm 2 to discard false positives), we are able to recover 10/17 columns compared to only 3/17 recovered columns when using the subset sum implementation.

**Experiments with Algorithm 2.** The heuristic approach presented in Algorithm 2 was more effective than the deterministic approach presented in Algorithm 1. We tested Algorithm 2 against the three above databases.

<sup>3</sup> There is one-to-one correspondence between education and education-num.

**Table 2.** Attribute-Name Recovery Results on Different Relational Databases when all the possible queries are issued. The entry 22/40 indicates that we have recovered 22 columns out of 40 columns in the Census database.

Algorithm	Census	Bank	Adult
1 (Subset sum)	6/40	3/17	12/14
1 (Independent set)	6/40	10/17	2/14
2 (Intersection)	20/40	12/17	10/14
2 (Int.)+1 (Sub.)	22/40	12/17	12/14

The results are depicted in Table 2 and they show clearly that the heuristic approach presented in Algorithm 2 recovers more columns. For example, it recovered 20/40 columns in the Census database compared to only 6/40 when Algorithm 1 (independent set approach) is employed. Moreover, when we combine the intersection heuristic approach with the subset sum approach, we recover even more columns in the Census database as shown in the last row of Table 2 where 22/40 queries are recovered.

## 4.2 Experiments to Recover the Attribute Values

Under the assumption that the attacker has background knowledge represented only in the frequency distribution, we apply both the *Attribute-Name Recovery* attack and the *Relational-Count* attack on the Bank database. Our goal is to recover more queries compared to the case where only the *Relational-Count* attack is applied.

Combining Algorithm 2 and the *Relational-Count* attack allows us to recover more queries than those recovered when only the *Relational-Count* is applied [1]. Table 3 summarizes the results of experiments to recover the attribute values using our second attack (i.e. combination of the Attribute-Name and the *Relational-Count* attack). The table shows that when all queries are issued, our second attack recovers more queries than those recovered using the *Relational-Count* attack. So one can conclude that for some relational database tables (e.g. small number of columns, unique cardinality, non-zero co-occurrence between attribute-value pairs belonging to different columns), SSE schemes security level is very close to the security level provided by deterministic encryption schemes.

**Table 3.** The table shows the number of queries recovered from three SSE-protected relational databases (Adult, Bank and Census) using both the *Relational-Count attack* [1] and our second attack.

Attack	Adult	Bank	Census
Our Second Attack	240/272	147/150	760/829
<i>Relational-Count</i> Attack [1]	236/272	122/150	757/829

Note that the maximum number of queries that can be recovered in each SSE-protected database is upper bounded by the number of queries that can be recovered when the columns' of the relational database is protected using a deterministic encryption algorithm. This upper bound is equivalent to 272, 150 and 829 in the Adult, Bank and Census databases respectively due to the existence of some values within the same column that have the same frequency in these databases.

### 4.3 Countermeasures

The padding countermeasure, which is proposed in [6, 15], hides the actual result size of each query and therefore might prevent our *Attribute-Name recovery* attack. However, depending on the padding level, a variant of the *Attribute-Name recovery* attack that does not look for the exact number of records but for a range of possible values for the number of records might still allow the attacker to find queries belonging to the same column.

## 5 Conclusion

In this paper, we proposed two attacks on relational databases protected via SSE schemes. Our first attack breaks query privacy as it classifies the set of issued queries into different subsets where each subset holds the queries belonging to a specific column. Remarkably, our first attack is more practical than all other proposed attacks on SSE schemes [1, 6, 15] which require prior knowledge about the frequency distribution of the attribute-value pairs in the target database. This is because our attribute-name recovery attack does not require any prior knowledge about the target database other than the meta-data information and the number of records. An important message of this paper is that SSE schemes leaking the number of records  $n$  should not be used.

Moreover, we improved the *Relational-Count* attack [1] by combining it with our first attack. This combination allows us to recover more queries on some databases. Our work is important because it shows that protecting relational databases via SSE schemes breaks query privacy as it allows an attacker to recover the attribute names of some of the observed queries. In particular, the queries belonging to columns with low cardinality will be easily distinguished from other queries without waiting for all queries to be issued.

Our experiments assume that all the possible queries on a searchable encrypted database are issued which is one limitation of our work. Observation 2 will not hold for some attributes when the number of issued queries is less than the number of all the attribute-value pairs in the encrypted database since the existence of a subset of observed queries belonging to a certain attribute is not certain. Estimating the existence probability of a certain subset of observed queries depends on the distribution of the issued queries which varies from one user to another. So further work is required to estimate this probability.

**Acknowledgments.** This work was supported by European Union’s Horizon 2020 research and innovation programme under grant agreement No 644814, the PaaSWord project within the ICT Programme ICT-07-2014: Advanced Cloud Infrastructures and Services.

## A Example Explaining Our Attribute-Name Recovery Attack

In the following, we give a toy example to demonstrate our attack. Assume that we have a relational database table as shown in Table 4. Using a deterministic encryption algorithm to encrypt the “Sex” and “Education” columns and using an order preserving encryption will transform our relational database table to an encrypted relational database table as shown in Table 5. However, most secure SSE schemes will transform an inverted index such as the one displayed in Table 6 into a length-hiding encrypted index where the server does not know the frequency or result length of each keyword token before being queried.

After observing all the queries issued on the encrypted index shown in Table 6. Our attribute-name recovery attack tries to resolve the attribute name of each observed query by exploiting the access pattern leakage. Figure 1 shows three graphs whose nodes represent the observed queries. The graph on the left

**Table 4.** The table shows a plaintext relational database table.

ID	Sex	Education	Age
1	M	Bsc	40
2	F	Msc	39
3	F	PhD	30
4	M	PhD	45
5	M	Bsc	25
6	F	Bsc	23
7	M	Msc	30

**Table 5.** The table shows an encrypted relational database table. *DET* refers to Deterministic Encryption and *OPE* refers to Order Preserving Encryption. Column names can either be replaced by random labels or deterministically encrypted using the table unique ID.

ID	$DET(Education  tableID, K_0)$	$DET(Age  tableID, K_0)$	$DET(Sex  tableID, K_0)$
1	$DET(Bsc, K_2)$	$OPE(40, K_3)$	$DET(M, K_1)$
2	$DET(Msc, K_2)$	$OPE(39, K_3)$	$DET(F, K_1)$
3	$DET(PhD, K_2)$	$OPE(30, K_3)$	$DET(F, K_1)$
4	$DET(PhD, K_2)$	$OPE(45, K_3)$	$DET(M, K_1)$
5	$DET(Bsc, K_2)$	$OPE(25, K_3)$	$DET(M, K_1)$
6	$DET(Bsc, K_2)$	$OPE(23, K_3)$	$DET(F, K_1)$
7	$DET(Msc, K_2)$	$OPE(30, K_3)$	$DET(M, K_1)$

**Table 6.** The table shows an inverted index for the relational database table shown in Table 4. Each keyword  $w$  is represented as  $w = (a : v)$  where  $a$  refers to its attribute name and  $v$  refers to its value.

Keyword	Record IDs
Sex:F	2, 3, 6
Sex:M	1, 4, 5, 7
Education:Bsc	1, 5, 6
Education:Msc	2, 7
Education:PhD	3, 4
Age:23	6
Age:25	5
Age:30	3, 7
Age:39	2
Age:40	1
Age:45	4

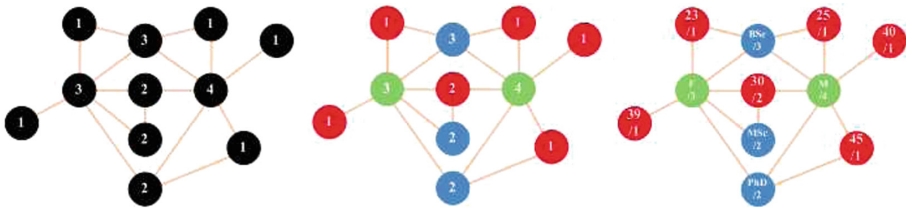
shows the server’s knowledge (represented by the frequencies or result lengths of observed queries gained from the access pattern leakage) before launching our attacks. When we apply the Attribute-Name recovery attack using only as background knowledge the meta-data information about the table and the number of records, the Server will know only the attribute names, “Education”, “Sex”

**Table 7.** The table shows a length-hiding encrypted index before being randomly permuted for the inverted index shown in Table 6. *DET* refers to deterministic encryption and *Enc* refers for a randomized encryption algorithm. Note that this encrypted index is not secure as the rows needs to be securely and randomly shuffled and the number of rows needs to be padded to the maximum number of keywords possible.

Keyword	Record IDs
$DET(Sex : F, K_0)$	$Enc(2  3  6  *, K_1)$
$DET(Sex : M, K_0)$	$Enc(1  4  5  7, K_1)$
$DET(Education : Bsc, K_0)$	$Enc(1  5  6  *, K_1)$
$DET(Education : Msc, K_0)$	$Enc(2  7  *  *, K_1)$
$DET(Education : PhD, K_0)$	$Enc(3  4  *  *, K_1)$
$DET(Age : 23, K_0)$	$Enc(6  *  *  *, K_1)$
$DET(Age : 25, K_0)$	$Enc(5  *  *  *, K_1)$
$DET(Age : 30, K_0)$	$Enc(3  7  *  *, K_1)$
$DET(Age : 39, K_0)$	$Enc(2  *  *  *, K_1)$
$DET(Age : 40, K_0)$	$Enc(1  *  *  *, K_1)$
$DET(Age : 45, K_0)$	$Enc(4  *  *  *, K_1)$

and “Age” represented by the graph on the middle in Fig. 1. Note that Naveed et al. [22] attack recovers column names and values of the encrypted database table shown in Table 5 using public background data. However, our attribute-name recovery attack recovers the query issued on the encrypted index shown in Table 7 using only meta-data information about the database table in addition to the number of records which can be leaked by some SSE schemes or guessed by the attacker.

Moreover, when we apply both the Attribute-Name recovery attack and the *Relational-Count* attack using the frequency distribution knowledge, the Server will know both the attribute names and their corresponding actual values. This additional knowledge is represented in Fig. 1 by the graph on the right.



**Fig. 1.** Nodes on the graphs represent all the possible queries that can be issued in the relational database index table shown in Table 6 after being encrypted by an SSE scheme. An edge between nodes (queries) exists if the intersection between their corresponding result sets is non-zero. The graph on the left shows queries as nodes of a graph labeled by their result lengths before applying our attacks. The graph on the middle shows what the server will learn after applying our attribute-name recovery attack. Note that the green color refers to the “Sex” attribute name and the blue color refers to the “Education” attribute name and the red color refers to the “Age” attribute name. The graph on the right shows what the server could learn after applying our second attack (Attribute-Name recovery attack combined with the *Relational-Count* attack).

## References

1. Abdelraheem, M.A., Andersson, T., Gehrman, C.: Searchable encrypted relational databases: risks and countermeasures. In: The 12th Data Privacy and Management Workshop (2017)
2. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (2004)
3. Ailon, N., Chazelle, B.: Lower bounds for linear degeneracy testing. *J. ACM (JACM)* **52**(2), 157–171 (2005)
4. Bellare, M., Boldyreva, A., O’Neill, A.: Deterministic and efficiently searchable encryption. In: Annual International Cryptology Conference (2007)



5. Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: improved security analysis and alternative solutions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 578–595. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-22792-9\\_33](https://doi.org/10.1007/978-3-642-22792-9_33)
6. Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. In: CCS 2015 (2015)
7. Cash, D., Jaeger, J., Jarecki, S., Jutla, C., Krawczyk, H., Rosu, M., Steiner, M.: Dynamic searchable encryption in very-large databases: Data structures and implementation. IACR Cryptology ePrint Archive (2014)
8. 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). [https://doi.org/10.1007/978-3-642-40041-4\\_20](https://doi.org/10.1007/978-3-642-40041-4_20)
9. Chase, M., Kamara, S.: Structured encryption and controlled disclosure. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 577–594. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-17373-8\\_33](https://doi.org/10.1007/978-3-642-17373-8_33)
10. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: CCS (2006)
11. Erickson, J.: Lower bounds for linear satisfiability problems. In: SODA 1995 (1995)
12. Center for Machine Learning and Intelligent Systems. University of California, Irvine. <https://archive.ics.uci.edu/ml/datasets.html>. Accessed June 2017
13. Gold, O., Sharir, M.: Improved bounds for 3sum, k-sum, and linear degeneracy. CoRR, abs/1512.05279 (2015)
14. IARPA. Poster about protecting privacy and civil liberties. [https://www.iarpa.gov/images/files/programs/spar/09-SPAR\\_final\\_v21.pdf](https://www.iarpa.gov/images/files/programs/spar/09-SPAR_final_v21.pdf)
15. Islam, M.S., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In: NDSS 2012 (2012)
16. 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). [https://doi.org/10.1007/978-3-642-39884-1\\_22](https://doi.org/10.1007/978-3-642-39884-1_22)
17. Kellaris, G., Kollios, G., Nissim, K., O'Neill, A.: Generic attacks on secure out-sourced databases. In: CCS (2016)
18. Kleinberg, J., Tardos, E.: Algorithm design. Pearson Education India (2006)
19. Kohavi, R., Becker, B.: Adult data set (1996). <https://archive.ics.uci.edu/ml/machine-learning-databases/adult/>. Accessed June 2017
20. Kurosawa, K., Ohtaki, Y.: UC-secure searchable symmetric encryption. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 285–298. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-32946-3\\_21](https://doi.org/10.1007/978-3-642-32946-3_21)
21. Lane, T., Kohavi, R.: Census-income (kdd) data set (2000). <https://archive.ics.uci.edu/ml/machine-learning-databases/census-income-mld/>. Accessed June 2017
22. Naveed, M., Kamara, S., Wright, C.: Inference attacks on property-preserving encrypted databases. In: CCS 2015 (2015)
23. OpenEMR. <http://www.open-emr.org/>. Accessed Mar 2017
24. Popa, R.A., Redfield, C., Zeldovich, N., Balakrishnan, H.: Cryptdb: protecting confidentiality with encrypted query processing. In: ACM Symposium on Operating Systems Principles (2011)

25. Laureano, R., Moro, S., Cortez, P.: Using data mining for bank direct marketing: an application of the crisp-dm methodology. In: Novais, P., et al. (eds.) Proceedings of the European Simulation and Modelling Conference - ESM 2011, pp. 117–121, Guimarães, Portugal, EUROSIS, October 2011. <https://archive.ics.uci.edu/ml/datasets/Bank+Marketing>. Accessed June 2017
26. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE Security and Privacy, S&P 2000
27. Van Liesdonk, P., Sedghi, S., Doumen, J., Hartel, P., Jonker, W.: Computationally efficient searchable symmetric encryption. In: Workshop on Secure Data Management (2010)



# A Simple Algorithm for Estimating Distribution Parameters from $n$ -Dimensional Randomized Binary Responses

Staal A. Vinterbo<sup>(✉)</sup> 

Department of Information Security and Communication Technology,  
Norwegian University of Science and Technology, Trondheim, Norway  
`Staal.Vinterbo@ntnu.no`

**Abstract.** Randomized response is attractive for privacy preserving data collection because the provided privacy can be quantified by means such as differential privacy. However, recovering and analyzing statistics involving multiple dependent randomized binary attributes can be difficult, posing a significant barrier to use. In this work, we address this problem by identifying and analyzing a family of response randomizers that change each binary attribute independently with the same probability. Modes of Google’s Rappor randomizer as well as applications of two well-known classical randomized response methods, Warner’s original method and Simmons’ unrelated question method, belong to this family. We show that randomizers in this family transform multinomial distribution parameters by an iterated Kronecker product of an invertible and bisymmetric  $2 \times 2$  matrix. This allows us to present a simple and efficient algorithm for obtaining unbiased maximum likelihood parameter estimates for  $k$ -way marginals from randomized responses and provide theoretical bounds on the statistical efficiency achieved. We also describe the efficiency – differential privacy tradeoff. Importantly, both randomization of responses and the estimation algorithm are simple to implement, an aspect critical to technologies for privacy protection and security.

**Keywords:** Privacy · Data collection · Randomized response  
Disclosure control

## 1 Introduction

Randomized response, introduced by Warner in 1965 [23], works by allowing survey respondents to sample their response according to a particular probability distribution. This provides privacy while still allowing the surveyor to gain insights about the queried population. Due to its suitability for large scale privacy preserving data collection, randomized response has lately enjoyed a resurgence in interest from Apple and Google, among others.

As an example of randomized response, consider a population of parties each holding an independent sensitive bit  $b$  with  $P(b = 1) = p$  where  $p$  is unknown. We want to estimate  $p$  and therefore randomly select  $m$  parties  $i$  to ask for their bit values  $b_i$  for this purpose. Tools from information security allow us to collect the bit values and compute the estimate  $\hat{p} = m^{-1} \sum_i b_i$  of  $p$  without access to any proper subset sum of bit values. However, if  $\hat{p} = 1$  we can infer that  $b_i = 1$  for all observed bits. Even if we are trusted with this knowledge, disseminating  $\hat{p}$  allows outsiders to infer  $b_i = 1$  for any party  $i$  known to be a contributor. Knowing this, parties might not be willing to share their bit-value directly. However, if each contributor is allowed to lie with a probability  $q < \frac{1}{2}$ , then we can argue that  $\frac{1-q}{q}$  is the upper limit to which any adversary can update their belief regarding the true value of any contributed bit. If parties then agree to contribute, we can still estimate  $p$ , albeit at a loss of statistical efficiency.

In 1977, Tore Dalenius defined *disclosure* about an object  $x$  by a statistic  $v$  with respect to a property  $p$  to have happened if the value  $p(x)$  can be determined more accurately with knowledge of  $v$  than without [7]. A goal of information security is preserving the integrity circles of trust. Mechanisms to achieve this include access control, communication security, and secure multi-party computation. These mechanisms have in common that the protected information is well circumscribed, and the states of allowed access are discrete. Disclosure control, on the other hand, provides a tool for considering questions regarding the *consequence* of access, and how to deal with information not necessarily well circumscribed. An emerging standard for defining privacy based on disclosure control is differential privacy [9]. The likelihood ratio  $\frac{1-q}{q}$  from the example above is an example of a quantification of disclosure risk, and the log transformation of this ratio is parameterized in differential privacy. We can also view the above randomization as enabling continuously graded access to each contributor's bit, quantifiable by entropy, for example.

In the past, when surveys were conducted manually with responses recorded on paper, surveys were generally limited to a single randomized dichotomous question. It was simply too expensive to survey enough individuals to support efficient recovery of parameter estimates with multiple randomized questions. With the advent of computerized surveys, enrollment is much easier, particularly if the data is collected automatically. With increased enrollment comes the ability to consider multiple randomized values per respondent and still obtain efficient estimates of population distribution parameters. On the other hand, multiple randomized values per response significantly increases the difficulty of analysis. For example, the first publication regarding Google's 2014 Rappor technology for automatically collecting end user data with randomized response [10] only considered each bit in an  $n$ -bit response independently as if it were the only bit randomized. The consideration of several bits jointly, had to wait for a subsequent publication [11]. Neither of these publications provided theoretical bounds of efficiency loss due to randomization. The point is that analyzing multi-question randomized response can be difficult, potentially causing surveyors to adopt less effective privacy protections.

We address this problem by defining a family of very easily implementable randomizers of length  $n$ -bit strings or surveys with  $n$  sensitive dichotomous questions. For the randomizers in this family, we provide simply computable distribution parameter estimators as well as statistical efficiency bounds for these. As these randomizers act on each response bit independently, marginals can be queried and recovered independently. This is helpful when bit  $k$  to query is chosen based on the length  $k - 1$  based marginal already queried, or when the bits of responses are distributed among multiple sources.

### 1.1 Contributions in Detail

A randomized response method can be seen as a randomized algorithm  $M$  that takes a response  $x$  as input and produces a randomized response  $r = M(x)$ . Encoding both  $x$  and  $r$  as length  $n$  bit strings, the algorithm  $M$  can be characterized by a  $2^n \times 2^n$  matrix  $C$  where the entry indexed by  $(r, x)$  contains  $P(r = M(x))$ . If  $M$  is applied independently to each of  $m$  strings sampled according to a multinomial distribution with parameters  $m \in \mathbb{N}$  and  $\pi \in [0, 1]^{2^n}$ , the resulting strings are expected to be multinomially distributed with parameters  $m$  and  $C\pi$ . Consequently, if  $C$  is invertible we can obtain a maximum likelihood estimate for  $\pi$  from the histogram  $y$  of observed randomized responses as  $m^{-1}C^{-1}y$ . If  $C$  is not invertible, using the expectation maximization algorithm can be a suitable, albeit more complicated alternative for obtaining estimates. In general, the expression of  $C$  can be such that estimators for the population parameters are not available in closed form [3].

We first recognize that a randomizer  $M$  that randomizes each bit in a length  $n$  string  $x$  independently in an identical manner can be represented by the iterated Kronecker product  $C$  of a bisymmetric  $2 \times 2$  matrix (Theorem 1 and Proposition 2). For the family of such randomizers, our contributions are developments of

- a definition of  $C^{-1}$  in terms of an iterated Kronecker products of a bisymmetric  $2 \times 2$  matrix,
- closed form formulas for the individual entries of both  $C$  and  $C^{-1}$ , of which at most  $n + 1$  are distinct in each matrix (Theorem 2 and Corollary 1),
- a closed form formula for the trace of the covariance matrix for the unbiased maximum likelihood estimator  $m^{-1}C^{-1}Y$  (Lemma 1),
- a closed form formula for the effective loss in sample size for estimating  $\pi$  due to randomization (Theorem 3) together with concentration bounds for uniformly distributed  $\pi$  (Proposition 4),
- an analysis of the loss of effective sample size in terms of the afforded level of differential privacy (Theorem 4).

As the Kronecker product can be implemented in linear time in the number of entries in the result,  $C$  and  $C^{-1}$  for iterated bisymmetric randomizers can be computed by an algorithm that is linear in the number of entries of these matrices (Proposition 5). We show that this algorithm is simple to state and simple to implement (Sect. 4.2), which facilitates adoption as well as verification of

implementation correctness. These are both critical aspects of algorithms applied for privacy protection and security.

Finally, we show that our results apply to modes of Rappor, application of Warner’s original randomizer [23] and Simmon’s unrelated question randomizer [13].

## 2 Related Work

Randomized response was first introduced primarily as a technique to reduce bias introduced by absent or untruthful responses to a single potentially sensitive dichotomous question [23]. Much research into randomized response is in the context of an interview tool for social sciences research. Here, randomization devices generally consist of a physical source of randomness like a spinner or a coin, together with a protocol for how the respondent should use it. These devices are then evaluated in terms of both human factors, e.g., protocol compliance and response rates, as well as the statistical utility of their randomized output [17, 22]. Randomized response surveys carry a double burden of requiring additional time and effort on behalf of the respondents, as well as requiring an enrollment that increases rapidly in the number of questions that require randomization. This might explain why randomized response designs for single dichotomous sensitive questions [4, 12, 13, 23] are much more common in the literature than for multiple sensitive questions [3, 5] or polychotomous questions [1]. Furthermore, multiple authors point out that while there exists a substantial body of methods research, “there have been very few substantive applications [of randomized response techniques]” [4, 17, 22].

However, as automated data collection on very large populations has become available, interest in randomized response involving multiple independent questions has emerged. Two examples are Google’s Rappor [10] technology for collecting end-user data, and Apple’s technology for collecting analytics data in MacOS and iOS [2, 21].

The view of randomizers as transformations of multinomial distribution parameters has been investigated in the context of local differential privacy [8]. Kairouz et al. [15] analyze what they call staircase mechanisms, which in the context of this paper can be thought of as family of randomized response mechanisms where  $C = BD$ , where  $B$  is a matrix that contains at most two values, located on the diagonal and elsewhere, respectively, and  $D$  is a diagonal matrix. In particular, they investigate a randomized response mechanism  $k$ -RR where  $D$  is the identity matrix. For  $k$ -RR they show that this mechanism is optimal with respect to the tradeoff between differential privacy and utility defined in terms of KL-divergence. In subsequent work [14] they further analyze Warner’s original proposal, their  $k$ -RR staircase mechanism and Rappor under general loss functions. They show that for  $n = 1$  Warner’s proposal is optimal for any loss and any differential privacy level. This is the only case where the  $k$ -RR family of staircase mechanisms intersects the family of randomizers presented here in this paper. Furthermore, their analysis is based on entire responses being known up

front and it is not clear how to apply their work in the case where a response is an interactively queried sequence of  $n > 1$  randomized bits.

### 3 Randomizing Mechanisms

#### 3.1 Length $n$ Bit Strings and the Multinomial Distribution

Let  $\mathbb{B} = \{0, 1\}$ , and let  $\wedge, \vee, \oplus$  denote logical and, or, and exclusive or. For  $x, y, u \in \mathbb{B}^n$  let  $x \geq y \iff x \wedge y = y$ ,  $x =_u y \iff x \wedge u = y \wedge u$ , and  $[x]_u = \{y \mid x =_u y\}$ . Now, let  $e_i \in \mathbb{B}^n$  be the string with a single 1 at position  $i$ , and let for a set of indices  $K$ ,  $e_K \geq e_i \iff i \in K$ . For  $x = (x_0, x_1, \dots, x_{n-1}) \in \mathbb{B}^n$ ,  $|x| = \sum_{i=0}^{n-1} x_i$ . Also let  $\eta : \mathbb{B}^n \rightarrow \mathbb{N}$  be defined as  $\eta(x_0, x_1, \dots, x_{n-1}) = \sum_{i=0}^{n-1} x_i 2^i$ , and let  $\zeta = n^{-1}$ . The function  $\eta$  lets us treat element  $x \in \mathbb{B}^n$  as a 0-based index  $\eta(x)$  which we will do often. Also, let  $\odot$  denote the coordinate-wise (Hadamard) product.

Consider an experiment that produces an outcome in  $\{0, 2, \dots, k - 1\}$  for some positive integer  $k$ , where outcome  $i$  is produced with probability  $p_i$ . Let  $m$  indicate a fixed number of independent experiments and let  $X_i$  denote the number of times outcome  $i$  is observed among the  $m$  experiments. Note that  $\sum_{i=0}^{k-1} X_i = m$ . Then  $X = (X_0, X_1, \dots, X_{k-1})$  follows a multinomial distribution  $\text{Mult}(m, \pi)$  where  $\pi = (p_0, p_1, \dots, p_{k-1})$ . The variables in  $X$  have expectations  $m\pi$  and variances  $m\pi \odot (1 - \pi)$ , and we can think of a realization of  $X$  as a histogram over the outcomes of the  $m$  experiments. Also, when  $m = 1$ ,  $X$  follows a categorical distribution, and when  $m = 1$  and  $k = 2$ ,  $X$  follows a Bernoulli distribution. If experiments produce outcomes in  $\mathbb{B}^n$  for  $n \in \mathbb{N}$ , we let  $X_i$  be the number of times  $\zeta(i) \in \mathbb{B}^n$  is observed among  $m$  experiments.

The estimator  $\hat{\pi}^*(m) = m^{-1}X$  is a maximum likelihood estimator for  $\pi$  (e.g., [16, example 6.11]) and the covariance matrix of  $\hat{\pi}^*(m)$  is

$$\text{cov}(\hat{\pi}^*(m)) = m^{-1}(\text{diag}(\pi \odot (1 - \pi) + \pi \odot \pi) - \pi\pi^T) = m^{-1}(\text{diag}(\pi) - \pi\pi^T)$$

where  $\text{diag}(v)$  is the square matrix with the elements of  $v$  along the diagonal.

#### 3.2 Randomizers as Linear Transformations

For a string of bits of length  $n$ , we will think of a randomizing mechanism as a function  $M : \mathbb{B}^n \times \mathbb{B}^\infty \rightarrow \mathbb{B}^n \times \mathbb{B}^\infty$  that takes a bit string and an infinite sequence of independent uniformly distributed random bits that serves as the source of randomness, and returns the randomized string and what remains of the source of randomness. Since the source of randomness consists of uniformly and independently distributed bits, we will let the randomness be implicit and state that randomizer  $M$  is a randomized algorithm  $M : \mathbb{B}^n \rightarrow \mathbb{B}^n$ . We can define randomizer  $M$  in terms of a  $2^n \times 2^n$  conditional probability matrix

$$C_{Mr,x} = P(r = M(x)).$$

Then, if  $C = C_M$  and  $X$  is  $\text{Mult}(m, \pi)$ , then  $Y = CX$  is  $\text{Mult}(m, C\pi)$ . If  $C$  invertible, then we can let  $\hat{\pi}(m) = m^{-1}C^{-1}Y$  be an estimator for  $\pi$  with

$$E(\hat{\pi}(m)) = m^{-1}C^{-1} E(Y) = m^{-1}C^{-1}m C\pi = \pi,$$

from which we see that it is unbiased. The invariance property of maximum likelihood estimators [6, theorem 7.2.10] states that if  $\hat{\theta}$  is a maximum likelihood estimator for parameter  $\theta$ , then for any function  $g$  we have that  $g(\hat{\theta})$  is a maximum likelihood estimator for  $g(\theta)$ . Consequently, for invertible matrix  $C$  we have that  $\hat{\pi}(m) = C^{-1}m^{-1}Y = m^{-1}C^{-1}Y$  is an unbiased maximum likelihood estimator of  $\pi$ . The covariance matrix of  $\hat{\pi}(m)$  is

$$\text{cov}(\hat{\pi}(m)) = \text{cov}(m^{-1}C^{-1}Y) = m^{-1} \left( C^{-1}\text{diag}(C\pi)C^{-1T} - \pi\pi^T \right).$$

**Proposition 1.** *If*

$$L(m) = \frac{\text{Tr}(\text{cov}(\hat{\pi}(m)))}{\text{Tr}(\text{cov}(\hat{\pi}^*(m)))}$$

where  $\text{Tr}(A)$  denotes the sum of the elements along the diagonal of square matrix  $A$ . Then  $L = L(m) = L(m')$  for any  $m, m' > 0$ , and for  $\alpha \geq 1$ ,

$$E(\|\hat{\pi}(\alpha Lm) - \pi\|_2^2) \leq E(\|\hat{\pi}^*(m) - \pi\|_2^2).$$

By the above proposition, the loss of quality of estimation when using randomized responses over non-randomized responses can be described by  $L$ .

## 4 Bitwise Independent Randomizers

Let  $(x : xs) \in \mathbb{B}^n$  be a length  $n$  sequence of bits with first bit  $x$  and length  $n - 1$  tail  $xs$ , and let  $M = (M' : Ms)$  be a sequence of bit randomizers. Now define

$$\begin{aligned} R(\epsilon, x) &= R(M, \epsilon) = \epsilon \\ R(M' : Ms, x' : xs) &= M'(x') : R(Ms, xs) \end{aligned}$$

where  $\epsilon$  is the empty sequence. We will think of  $R$  as a function that maps a length  $n$  sequence of bit randomizers  $M$  to a randomizer  $R(M)$  of length  $n$  bit strings. We note that  $R(M)$  randomizes each bit independently, and that any randomizer of length  $n$  bit strings that randomizes each bit independently can be written as  $R(M')$  for some sequence of bit randomizers  $M'$ . We will call these bitwise independent randomizers.

We first repeat a result regarding bitwise independent randomizers, first established by Bourke [5], namely that  $R(M)$  is defined by the Kronecker product  $\otimes$  of its constituent bit randomizers.

**Theorem 1 (Bourke 1982).** *For a sequence  $M = (M_0, M_1, \dots, M_{n-1})$  of independent bit randomizers,  $C_{R(M)} = C_{M_0} \otimes C_{M_1} \otimes \dots \otimes C_{M_{n-1}}$ .*

We now examine a special case of bitwise independent randomizers in more detail.



### 4.1 Iterated Bisymmetric Randomizers

Let a bisymmetric bit randomizer  $M$  be a bit randomizer that has a matrix

$$C_M = C_{a,b} = \begin{pmatrix} a & b \\ b & a \end{pmatrix}$$

that is symmetric about both its main diagonals, i.e., is a bisymmetric matrix. We now consider bitwise independent randomizers generated by a sequence of identical bisymmetric bit randomizers.

**Definition 1.** *The iterated Kronecker product of bisymmetric  $C_{a,b}$  is*

$$C_{a,b}(n) = \begin{cases} C_{a,b} \otimes C_{a,b}(n-1) & \text{if } n > 0, \\ (1) & \text{otherwise.} \end{cases}$$

**Proposition 2.** *Let  $M = (M_0, M_1, \dots, M_{n-1})$  be a sequence of identical bisymmetric bit randomizers with  $C_{M_i} = C_{a,b}$ . Then  $C_{R(M)} = C_{a,b}(n)$ .*

The iterated Kronecker product preserves properties of  $C_{a,b}$  in the sense of the following.

**Proposition 3.** *The matrix  $C_{a,b}(n)$*

- a. is bisymmetric,*
- b. has a constant diagonal, and*
- c. has a constant anti-diagonal, and*
- d. contains at most  $n + 1$  distinct entries.*

As a consequence we will call  $M$  with  $C_M = C_{a,b}(n)$  *iterated bisymmetric randomizers*. We also note that since a column in  $C_M$  contains a distribution, the sum of entries must be 1. Consequently,  $b = 1 - a$ , and we can define  $C_a(n) = C_{a,1-a}(n)$ , and let  $C_M = C_a(n)$  for iterated bisymmetric randomizer  $M$ . Our main results regarding iterated bisymmetric randomizers are the following. First we turn to the results regarding  $C_M$  and  $C_M^{-1}$ .

**Theorem 2.** *Let  $M$  be a randomizer with  $C_M = C_a(n)$ . Then*

$$C_{M_{r,x}} = a^{n-d}(1-a)^d$$

where  $d = |r \oplus x|$ . If  $a \neq \frac{1}{2}$ , then

$$C_M^{-1} = C_{\frac{a}{2a-1}}(n).$$

**Corollary 1.** *If  $a \neq \frac{1}{2}$ ,*

$$C_{M_{x,r}}^{-1} = \frac{a^{n-d}(a-1)^d}{(2a-1)^n}$$

for  $d = |x \oplus r|$ .

We now apply the above results to determine bounds for the statistical efficiency of iterated bisymmetric randomizers.

**Lemma 1.** *Let  $\hat{\pi}$  be the unbiased estimator defined in Sect. 3.2 associated with the randomizer  $M$  with invertible  $C = C_M = C_a(n)$ . If  $a \neq \frac{1}{2}$  the trace of the variance-covariance matrix of  $\hat{\pi}$  is given by*

$$\text{Tr}(\text{cov}(\hat{\pi})) = m^{-1}(c - s)$$

where

$$c = \left( \frac{a^2 + (1 - a)^2}{(2a - 1)^2} \right)^n,$$

and  $s = \pi^T \pi = \sum_x p_x^2$ .

**Theorem 3.** *For  $\pi^T \pi < 1$ ,  $a \neq 1/2$ , and  $\hat{\pi}$  associated with  $M$  as in Lemma 1 the loss as defined in Proposition 1 is given by*

$$L = f_L(s) = \frac{\left( \frac{a^2 + (1 - a)^2}{(2a - 1)^2} \right)^n - s}{1 - s}$$

for  $s = \pi^T \pi = \sum_x p_x^2$ . Also,  $f_L(2^{-n}) \leq L$ .

When  $\pi$  is a random uniformly sampled probability distribution on  $2^n$  categories<sup>1</sup>, the quantity  $\pi^T \pi$ , also known as the Greenwood statistic, has expected value  $E(\pi^T \pi) = \frac{2}{2^n + 1}$  and variance  $\text{Var}(\pi^T \pi) = \frac{4(2^n - 1)}{(2^n + 1)^2 (2^n + 2) (2^n + 3)}$  [18]. As  $\pi$  is usually unknown, we can approximate the loss  $L$  as  $L(n) = f_L(\frac{2}{2^n + 1})$ . We state the following about the quality of this approximation.

**Proposition 4.** *For  $n > 2$  and  $\pi$  a random uniformly sampled probability distribution on  $n$  categories,*

$$P(1 \leq \frac{E(f_L(\pi^T \pi))}{f_L(\frac{2}{2^n + 1})} \leq 1 + \delta(n)) \geq 0.99.$$

where  $\delta(n) \in O(2^{-3n})$  and  $\delta(3) < 0.2386$ .

Also,  $\delta(4) < 0.0029$ . Finally, we can compare iterated bisymmetric randomizers in terms of the efficiency of their estimators  $\hat{\pi}$ , which in turn means comparing their associated values for  $c$  in Lemma 1. Here smaller is better.

## 4.2 A Simple Algorithm

A randomizer with  $C_M = C_a(n)$  can be implemented as  $M(x) = x \oplus u$ , where  $u$  is a length  $n$  sequence of independent Bernoulli trials with success probability  $1 - a$ .

<sup>1</sup> Distributed as the flat Dirichlet distribution of order  $2^n$ .

Let  $(r_0, r_1, \dots, r_{m-1})$  be  $m$   $n$ -bit randomized responses where each bit has been randomized by an independent bisymmetric bit randomizer with  $C_M = C_a$ , and let  $r_i[K]$  denote the sub-sequence of  $r_i$  indexed by  $K \subseteq \{0, 1, \dots, n-1\}$  in order. By Theorem 2 we can then estimate the marginal multinomial distribution parameter  $\hat{\pi}_K$  for the bits indexed by the  $k$  bits in  $K$  as follows:

1. let  $y = (y_0, y_1, \dots, y_{2^k-1})$  where  $y_i = |\{j \mid \eta(r_j[K]) = i\}|$ , i.e.,  $y$  is the histogram over observed sub-sequences, and
2.  $\hat{\pi}_K = m^{-1} C_{\frac{a}{2^a-1}}(k) y$ .

**Proposition 5.** *The simply recursive algorithm  $C_a(n)$  can be implemented to run in time that is linear in the number of entries of the output matrix.*

A Python code example for implementing the algorithm above is as follows.

**Implementation 1.**

```
from numpy import array, kron, log2, bincount as bc, arange
```

```
def C(n, a):
    c, z = array([[a, 1-a], [1-a, a]]), array([1])
    return z if n < 1 else kron(c, C(n-1, a))
```

```
def pihat(a, Y):
    m, n = float(sum(Y)), int(log2(len(Y)))
    return C(n, a/(2*a - 1)).dot(Y)/m
```

```
def hist(R):
    _,n = R.shape
    x = 2**(n-1)-arange(n)
    return bc(R.dot(x), minlength = 2**n)
```

For input  $R$  being a  $m \times n$  numpy array of  $m$  randomized length  $n$  binary responses and  $K$  a list of column indices, the call `pihat(a, hist(R[:,K]))` computes the value for  $\hat{\pi}_K$ .

**4.3 Privacy Considerations**

The results established so far are about efficiency aspects of estimating multinomial parameters as functions of population and randomization parameters. We now briefly turn to a measure of privacy risk for bit-wise independent randomizers.

Now, let  $M$  be a bit randomizer such that entries in  $C_M$  all are positive, i.e., randomization happens for both possible inputs. We will in this section only consider such bit randomizers. Let

$$l_M(r) = \frac{\max_x P(r = M(x))}{\min_x P(r = M(x))}.$$

The likelihood ratio  $l_M(r)$  can be thought of representing the best evidence for preferring one hypothetical input over another when given the randomized output  $r$ . This is reflected in the definition of Differential Privacy [9], where a randomized algorithm  $M$  can be considered  $\alpha$ -differentially private if, for any measurable subset  $S$  of possible outputs, and inputs  $D$  and  $D'$  obtained from any two sets of individuals that overlap in all but one individual, we have that

$$\frac{P(M(D) \in S)}{P(M(D') \in S)} \leq \exp(\alpha),$$

and the probabilities are over the randomness available to the algorithm.

Now let  $l_M = \max_r l_M(r)$ , then it follows that  $M$  is a  $\log(l_M)$ -differentially private algorithm. Now consider  $R(M)$  for  $n$  bit randomizers  $M = (M_i)_{i=0}^{n-1}$  such that  $l_{M_i} \geq l_{M_{i+1}}$ , and let  $k \geq |x \oplus x'|$  for any  $x, x'$  given as input to  $R(M)$ . Then  $l = \prod_{i=0}^{k-1} l_{M_i}$  can be an upper bound of privacy loss for any respondent. Specifically,  $R(M)$  is then a  $\log(l)$ -differentially private algorithm.

Now assume that  $M$  is bisymmetric. Exploiting the structure of  $M$ , we get that  $l_M = l_a = r(a)$  where  $r(a) = \max(a^{-1}(1 - a), a(1 - a)^{-1})$ . If  $R(M)$  is an iterated bisymmetric randomizer, i.e.,  $C_{R(M)} = C_a(m)$ , then  $l = (l_a)^k$ , and we have that  $R(M)$  is  $\alpha$ -differentially private for  $\alpha = \log(l_a^k) = k \log(r(a))$ . This is particularly useful if  $a$  is an invertible function  $a_\phi(\phi)$ . Then, we can write

$$\begin{aligned} \alpha_\phi(\phi) &= k \log(r(a_\phi(\phi))), \text{ and} \\ \phi_\alpha(\alpha) &= a_\phi^{-1}(r^{-1}(\exp(\alpha/k))). \end{aligned}$$

We have from Lemma 1 that  $c$  is a function  $c_a(a)$ . We can now view  $c$  as a function of  $\phi$  by  $c_\phi = c_a \circ a_\phi$ , where  $\circ$  denotes function composition. Expanding further, we can let  $c$  be a function of  $\alpha$  as  $c_\alpha = c_\phi \circ \phi_\alpha$ . This means that

$$\begin{aligned} c_\alpha &= c_a \circ a_\phi \circ \phi_\alpha \\ &= c_a \circ a_\phi \circ a_\phi^{-1} \circ r^{-1} \circ \exp \circ (x \mapsto x/k) \\ &= c_a \circ r^{-1} \circ \exp \circ (x \mapsto x/k). \end{aligned}$$

The last equation shows that  $c_\alpha$  is independent of the functional shape of  $a(\phi)$ , and therefore this holds for  $L$  as well. In other words, from a perspective of differential privacy, the performance of iterated bisymmetric randomizers in terms of  $L$  is independent of the functional shape of invertible  $a(\phi)$ . The above reasoning proves the following Theorem.

**Theorem 4.** *For any  $\alpha$ -differentially private iterated bisymmetric randomizer for inputs not differing in more than  $k$  bits, for  $c$  from Lemma 1,*

$$c \geq c_\alpha(\alpha) = \left( \frac{\exp(2\alpha/k) + 1}{(\exp(\alpha/k) - 1)^2} \right)^n,$$

and for  $L$  from Theorem 3,

$$L \geq L(\alpha) = \frac{c_\alpha(\alpha) - s}{1 - s}$$

where  $s = \pi^T \pi$ .

Combining the above with Theorem 2 in [14], stating the optimality of Warner’s randomizer with regards to the privacy-utility tradeoff for any loss function and privacy level, we conclude the following.

**Corollary 2.** *For  $n = 1$ , bisymmetric iterated randomizers are optimal with respect to loss  $L$  for any privacy level  $\alpha$ .*

## 5 Case Studies

### 5.1 The Unrelated Uniform Question Device

In Simmons’ unrelated question method, the interviewer asks the respondent to answer a question randomly selected between the sensitive question of interest and an unrelated question that the respondent presumably has no problem answering truthfully. The unrelated question is chosen with probability  $p$ . Here we analyze the variant of this method where the unrelated question is “Flip a coin. Is it heads?”. In other words, the case where interviewer knows that the answer to the unrelated question is uniformly distributed in the study sample.

Let  $A : \{0, 1\}^n \times \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  be given by

$$A(x, u, z) = x \odot (\mathbf{1} - u) + z \odot u$$

where  $\mathbf{1}$  is the string with all elements 1. Letting  $B_p^n$  denote a sequence of  $n$  independent Bernoulli variables each taking value 1 with probability  $p$ , the randomizer  $M(x)$  can then be defined as  $M(x) = A(x, u, z)$  where  $u$  and  $z$  are realizations of  $B_p^n$  and  $B_{0.5}^n$  variables, respectively.

We start by noting that if  $A(x, u, z) = r$ , then  $u \geq d = x \oplus r$ . Now, let  $U$ , and  $Z$  be independent  $B_p^n$  and  $B_{0.5}^n$  variables, respectively. Then

$$P(r = A(x, U, Z)) = \sum_{u \geq d} P(U = u)P(Z =_u r),$$

where  $d = r \oplus x$ . Now,  $P(U = u) = p^{|u|}(1 - p)^{n - |u|}$  and  $P(Z =_u r) = (1/2)^{|u|}$ . Furthermore, there are  $2^{n - |d|}$  strings  $u$  such that  $u \geq d$ , and of those  $\binom{n - |d|}{i}$  have  $|u| = i + |d|$ . Consequently,

$$c_{r,x} = P(r = A(x, U, Z)) = \sum_{i=0}^{n - |d|} \binom{n - |d|}{i} \left(\frac{p}{2}\right)^{i + |d|} (1 - p)^{n - (i + |d|)}.$$

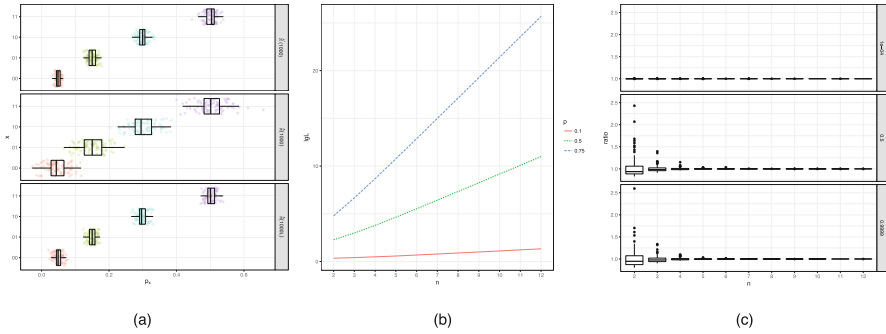
If we instead note that each bit is randomized independently by a bit randomizer  $M_S$  with matrix  $C_{\frac{2-p}{2}}$ , then by applying Theorem 2 we get that for  $n \geq 1$

$$C_{M_{r,x}} = \frac{p^{|x \oplus r|} (2 - p)^{n - |x \oplus r|}}{2^n}.$$

Algebraic manipulations yield that  $c_{r,x} = C_{M_{r,x}}$ , and the value for  $c$  in Lemma 1 is

$$c_M = \left( \frac{p^2 - 2p + 2}{2(p - 1)^2} \right)^n .$$

Figure 1(a) shows the effect of increasing the number of randomized response data points by a factor  $L$  for computing  $\hat{\pi}$ . As expected, the plot for  $\hat{\pi}(L1000)$  is close to the target  $\hat{\pi}^*(1000)$ . Figure 1(b) shows the growth of  $\log(f_L(2(2^n + 1)^{-1}))$  in  $n$  for three values of  $p$ . Figure 1(c) plots the ratio of  $L$  and the approximated loss  $f_L(2(2^n + 1)^{-1})$  for 100 uniformly random distributions  $\pi$  and three probabilities  $p$  of 0.0001, 0.5, and 0.9999.



**Fig. 1.** (a): Plot of  $\hat{\pi}^*(1000)$ ,  $\hat{\pi}(1000)$ , and  $\hat{\pi}(L1000)$  for 100 randomly generated datasets with the same fixed  $\pi = (0.05, 0.15, 0.3, 0.5)^T$ ,  $p = 0.5$ , and  $L = 9.75$ . (b):  $\log(L) = \log(f_L(2(2^n + 1)^{-1}))$  for  $n = 1, 2, \dots, 12$  and three values of  $p$ . (c): The ratio  $\frac{f_L(\pi^T \pi)}{f_L(2(2^n + 1)^{-1})}$  for 100 random uniformly sampled  $\pi$  for each  $2 \leq n \leq 12$  and three values of  $p$

### 5.2 Warner’s Original Device: A Randomizer Comparison

Warner’s original randomizer involved a spinner with two areas “Yes” and “No”, with a probability  $p$  for indicating “Yes”. The respondent was then asked to spin the spinner unseen by the interviewer and respond with “yes” if the spinner indicated the respondent’s true answer to the sensitive question and “no” otherwise. The corresponding bit randomizer  $M_W$  has matrix  $C_p$ , which is invertible if  $p \neq 0.5$ . Furthermore, we have that the value for  $c$  as defined in Lemma 1 is

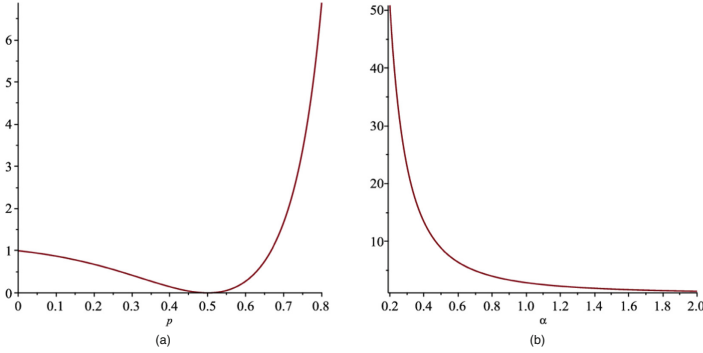
$$c_W = \left( \frac{2p^2 - 2p + 1}{(2p - 1)^2} \right)^n .$$

We can use the ratio  $c_M/c_W$  to compare the estimators for  $\hat{\pi}$  corresponding to the independent question and Warner’s original method, respectively. We have that  $c_M/c_W < 1$  for  $p \in (0, \frac{2}{3})$ . Figure 2(a) shows a plot of this ratio for  $n = 1$ . We see that when  $p < \frac{2}{3}$ , the unrelated question randomizer is preferable with

respect to estimating  $\pi$ , and particularly so around  $p = 0.5$  ( $c_W$  is undefined at  $p = 0.5$ ). The preference regions are emphasized as  $n$  increases. However, if we express  $c_M$  and  $c_W$  as functions of privacy level  $\alpha$  using the results from Sect. 4.3, we get that the two randomizers perform identically as

$$c_M(\alpha) = c_W(\alpha) = \left( \frac{\exp(2\alpha) + 1}{(\exp(\alpha) - 1)^2} \right)^n.$$

Figure 2(b) shows  $c_M(\alpha) = c_W(\alpha)$  for  $\alpha \in [0.2, 2]$  and  $n = 1$ .



**Fig. 2.** (a) The ratio  $c_M/c_W$  for  $p \in (0, 0.8)$  and  $n = 1$ . (b)  $c_M(\alpha) = c_W(\alpha)$  for  $n = 1$  and  $\alpha \in [0.2, 2]$

### 5.3 The Rappor Randomizer

In Rappor, randomization is applied after an hashing of ordinal values onto a bit string. Here we only examine the randomizer.

The Rappor randomizer is a bit-wise independent randomizer  $R(M_1, \dots, M_n)$  where  $M_i = M$  for all  $i$ . The bit randomizer  $M$  is a combination of two bit randomizers, “Permanent Randomized Response” (PR) and “Instantaneous Randomized Response” (IR), respectively. The PR randomizer is the above  $M_S$  randomizer with  $p = f$ , while

$$C_{M_{IR}} = \begin{pmatrix} 1-p & 1-q \\ p & q \end{pmatrix}.$$

A bit  $b$  decides the combination, where  $b = 1$  is called “one-time” mode, and  $M(b) = (1 - b)(M_{IR} \circ M_S) + b M_S$ . Consequently,  $C_{M(1)} = C_{\frac{2-f}{2}}$ . When  $p = 1 - q$ , we recognize  $M_{IR}$  as Warner’s  $M_W$  with parameter  $q$ , and  $C_{M(0)} = C_q C_{\frac{2-f}{2}} = C_{q-(q-\frac{1}{2})f}$ . This means that the Rappor randomizer is an iterated bisymmetric randomizer when  $p = 1 - q$  or  $b = 1$ .

## 6 Conclusion

A family of randomized response methods is described and analyzed. Instances of both well known classical and recently developed methods belong to this family. The analysis resulted in an efficient algorithm for estimating multinomial population parameters from randomized responses, and the statistical efficiency of the produced estimates was described.

The investigated statistical loss grows exponentially in the number of dimensions  $n$ , as does the size of the matrix  $C$  that describes the effect of randomization on the multinomial parameters. Consequently, the estimation of these multinomial parameters is only practical for small  $n$ , even with a large number of observations. However, the knowledge of how statistical loss grows with dimensionality allows the determination of a value  $k$  for which it is feasible to estimate parameters for size  $k$  marginals. Since individual variables are randomized independently, only the variables in the relevant feasible marginals need to be obtained. Furthermore, due to the independent randomization of variables these can be queried interactively across distributed data sources.

Brevity is said to be a hallmark of simplicity [20]. Simple algorithms are more likely to be implemented and trusted by practitioners, their implementations are easier to maintain and adapt to changing contexts, and they are easier to implement in constrained environments such as in hardware [19]. Simple algorithms are also easier to debug and implement correctly, which is critical in systems that need to implement privacy and security requirements. The algorithm presented here is simple. It centers on a short recursive definition of the matrix  $C^{-1}$ , which is shown implemented in four lines of Python, a multi-purpose programming language with a significant current market-share. Furthermore, an implementation of the full process of computing parameter estimates from binary randomized input was implemented in an additional nine lines of Python code, making iterated bisymmetric randomizers a potentially attractive alternative for randomized response applications.

**Acknowledgments.** Thanks go to the anonymous reviewers for their comments. This work was in part funded by Oppland fylkeskommune.

## A Proofs

We start by making a key observation.

**Observation 1:** Consider the  $2^n \times 2^n$  matrix  $C$ . If we let entry  $C_{ix', jy'} = \eta((ix') \oplus (jy')) = 2^{\eta(i \oplus j)} \eta(x' \oplus y')$ , we get that  $C_{x,y} = 1^{n-|x \oplus y|} 2^{|x \oplus y|}$ . Since we can write  $C = J_1 \otimes J_2 \otimes \dots \otimes J_n$  where  $J_k$  is the  $2 \times 2$  matrix  $J$  such that  $J_{i,j} = 2^{i \oplus j}$ , i.e.,  $J = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$ , we get that  $D_{x,y} = a^{n-|x \oplus y|} b^{|x \oplus y|}$  for

$$D = C_{a,b}(n) = B_1 \otimes B_2 \otimes \dots \otimes B_n \text{ where } B_k = \begin{pmatrix} a & b \\ b & a \end{pmatrix}.$$



**Proof of Proposition 1:** Note that we can write

$$\begin{aligned} \text{Tr}(\text{cov}(\hat{\pi}(m))) &= \sum_x \text{Var}(\hat{\pi}_x(m)), \quad \text{Tr}(\text{cov}(\hat{\pi}^*(m))) = \sum_x \text{Var}(\hat{\pi}_x^*(m)) \\ \text{Var}(\hat{\pi}_x(m)) &= m^{-1}F(x, \pi), \quad \text{Var}(\hat{\pi}_x^*(m)) = m^{-1}G(x, \pi) \end{aligned}$$

where  $F$  and  $G$  are functions independent of  $m$ . Then for any positive integer  $m$ ,

$$L(m) = \frac{m^{-1} \sum_x F(x, \pi)}{m^{-1} \sum_x G(x, \pi)} = \frac{\sum_x F(x, \pi)}{\sum_x G(x, \pi)} = L.$$

and

$$\begin{aligned} \sum_x \text{Var}(\hat{\pi}_x(\alpha Lm)) &= \sum_x \alpha^{-1} m^{-1} L^{-1} F(x, \pi) \leq m^{-1} L^{-1} \sum_x F(x, \pi) \\ &= \sum_x m^{-1} G(x, \pi) = \sum_x \text{Var}(\hat{\pi}_x^*(m)). \end{aligned}$$

Furthermore,

$$\begin{aligned} \text{E}(\|\hat{\pi}(m) - \pi\|_2^2) &= \text{E}\left(\sum_x (\hat{\pi}_x(m) - p_x)^2\right) = \sum_x \text{E}((\hat{\pi}_x(m) - p_x)^2) \\ &= \sum_x \text{Var}(\hat{\pi}_x(m)), \end{aligned}$$

and similarly  $\text{E}(\|\hat{\pi}^*(m) - \pi\|_2^2) = \sum_x \text{Var}(\hat{\pi}_x^*(m))$ . □

**Proof of Proposition 2:** The proposition follows directly from Theorem 1. □

**Proof of Proposition 3:** We first note that for  $i \in \{0, 1, \dots, 2^n - 1\}$  we have that  $\zeta((2^n - 1) - i) = \mathbf{1} \oplus \zeta(i)$ . From this and that  $\oplus$  commutes, we get

1.  $|\zeta(i) \oplus \zeta(n - i)| = n$ , and
2.  $|\zeta(i) \oplus \zeta(j)| = |\zeta(2^n - 1 - j) \oplus \zeta(2^n - 1 - i)|$ .

The above and that the entry  $C_{a,b}(n)_{i,j} = g(|\zeta(i) \oplus \zeta(j)|, a, b)$  for some  $g$ , the proposition follows. □

**Proof of Theorem 2:** The first equation follows directly from Observation 1. We have that  $C_{a,b}(1)$  is invertible if  $a^2 \neq b^2$ . From this and that  $(A \otimes B) = (A^{-1} \otimes B^{-1})$  we complete the proof. □

**Proof of Lemma 1:** From Sect. 3.2 we have that

$$\text{cov}(\hat{\pi}(m)) = m^{-1} \left( C^{-1} \text{diag}(C\pi) C^{-1T} - \pi\pi^T \right).$$

By properties of the trace of matrix products and symmetry of  $C^{-1}$ ,

$$\begin{aligned} &\text{Tr} \left( m^{-1} \left( C^{-1} \text{diag}(C\pi) C^{-1T} - \pi\pi^T \right) \right) \\ &= m^{-1} \left( \text{Tr} \left( C^{-1} \text{diag}(C\pi) C^{-1T} \right) - \text{Tr}(\pi\pi^T) \right) \\ &= m^{-1} \left( \text{Tr}(C^{-1} C^{-1} \text{diag}(C\pi)) - s \right) \end{aligned}$$

From  $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$  it follows that  $C_a(n)C_a(n) = C_{a^2+(1-a)^2}(n)$ . From this and Theorem 2 and Corollary 1 we get that the entry  $(C^{-1}C^{-1})_{0,0} = f(n, a)$  where

$$f(n, a) = \left( \frac{a^2 + (1 - a)^2}{(2a - 1)^2} \right)^n.$$

Furthermore, from Proposition 3 the diagonal entries of  $C^{-1}C^{-1}$  are all  $f(n, a)$ . Combining this, that  $\text{Tr}(AB) = \sum_{i,j} (A \odot B^T)_{i,j}$ , and  $\sum_x C_x \pi = 1$ ,

$$\text{Tr}(C^{-1}C^{-1} \text{diag}(C\pi)) = \sum_x f(n, a) C_x \pi = f(n, a) \sum_x C_x \pi = f(n, a),$$

and consequently,  $\text{Tr}(\text{cov}(\hat{\pi}_x(m))) = m^{-1} (f(n, a) - s)$ . □

**Proof of Theorem 3:** We have that

$$\text{Tr}(\text{cov}(\hat{\pi}^*)) = m^{-1} \text{Tr}(\text{diag}(\pi) - \pi\pi^T) = m^{-1}(1 - s).$$

From Lemma 1 and Proposition 1 we get that  $L = f_L(s) = \frac{c-s}{(1-s)}$  for  $c = \left( \frac{a^2+(1-a)^2}{(2a-1)^2} \right)^n$ . From  $0 \leq p_x \leq 1$  and  $\sum_x p_x = 1$ ,  $s$  has a minimum when  $p_x = 1/2^n$  for all  $x$ , and maximum when  $p_x = 1$  for a fixed  $x$ , and  $p_y = 0$  for  $y \neq x$ . These values are then  $\frac{2^n}{(2^n)^2} = 2^{-n}$  and 1, respectively. The  $m$ 'th derivative of  $f_L(s) = \frac{c-s}{1-s}$  wrt.  $0 \leq s < 1$  is  $f_L^{(m)}(s) = \frac{m!}{(1-s)^m} (f_L(s) - 1)$ . The loss  $f_L$  therefore achieves its minimum at  $f_L(2^{-n})$ . □

**Proof of Proposition 4 (sketch):** The  $m$ 'th derivative of  $f_L(s) = \frac{c-s}{1-s}$  wrt.  $0 \leq s < 1$  is  $f_L^{(m)}(s) = \frac{m!}{(1-s)^m} (f_L(s) - 1)$ . Since  $c \geq 1$ ,  $f_L^{(m)} \geq 0$  for all  $m > 0$ . In particular, we have that  $f_L$  is convex, as is  $f_L^{(m)}$  for all  $m$ . Using the expectation for a first order Taylor approximation for convex  $f_L$  we have that for random variable  $S$

$$f_L(\mathbb{E}(S)) \leq \mathbb{E}(f_L(S)) \leq f_L(\mathbb{E}(S)) + \frac{\lambda}{2} \text{Var}(S) \tag{*}$$

where  $\lambda = \max_{x \in \mathcal{I}} f_L^{(2)}(x) \geq 0$  for suitable interval  $\mathcal{I}$ . Dividing (\*) by  $f_L(\mathbb{E}(S)) = L(n)$ , we get

$$1 \leq \frac{\mathbb{E}(f_L(S))}{f_L(\mathbb{E}(S))} \leq 1 + \delta,$$

where

$$\delta = \frac{\lambda \text{Var}(S)}{2f_L(\mathbb{E}(S))}.$$

Let  $S = \pi^T \pi$ . Recalling that  $c = c(a)^n$  and expanding both numerator and denominator at  $n = 3$  (where the minimum occurs since  $f_L$  is increasing and  $\text{Var}(S)$  and  $\mathbb{E}(S)$  are both decreasing in  $n$ ), we see that  $\delta(n) \in O(2^{-3n})$ . Applying Chebyshev's inequality, we have that  $P(S \geq \mathbb{E}(S) + 10 \text{Var}(S)^{\frac{1}{2}}) \leq 0.01$ . Evaluating  $\delta$  at  $\mathbb{E}(S) + 10 \text{Var}(S)^{\frac{1}{2}}$  and  $n = 3$ , we arrive at the numerical bound. □

**Proof of Proposition 5:** Let the computation of  $Z \otimes R$  require  $t_f(n^2)$  time for  $2 \times 2$  matrix  $Z$  and  $R$  of size  $n \times n$ . Then we can compute  $R_{a,b}(n)$  at a time cost of  $t(n) = t_f(2^{2(n-1)}) + t(n-1) = \sum_{i=0}^{n-1} t_f(2^{2i}) = \sum_{i=0}^{n-1} t_f(4^i)$ . Letting  $t_f(n) = k4n$  for some  $k$ , then  $t(n) = 4k \sum_{i=0}^{n-1} 2^{2i} = 4k \sum_{i=0}^{n-1} 4^i = 4k(1 + \frac{1-4^n}{1-4}) = 4k(1 + \frac{4^n-1}{3})$ . Now we have that  $t(n) = O(4^n) = O(2^{2n}) = O(|R_{a,b}(n)|)$ . In other words, the singly recursive algorithm  $R_{a,b}(n)$  is linear in the time in the number of elements of the output matrix as we can perform  $t_f$  in linear time in the size of input  $R$ , in fact we can expect that the Kronecker product can be implemented with  $k \leq 3$ , due to reading, multiplication, and writing.  $\square$

## References

1. Abul-Ela, A.L.A., Greenberg, B.G., Horvitz, D.G.: A multi-proportions randomized response model. *J. Am. Stat. Assoc.* **62**(319), 990–1008 (1967). <https://doi.org/10.2307/2283687>
2. Apple: Learning with Privacy at Scale - Apple. December 2017. <https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html>
3. Barabesi, L., Franceschi, S., Marcheselli, M.: A randomized response procedure for multiple-sensitive questions. *Stat. Papers* **53**(3), 703–718 (2012). <https://doi.org/10.1007/s00362-011-0374-5>
4. Blair, G., Imai, K., Zhou, Y.Y.: Design and analysis of the randomized response technique. *J. Am. Stat. Assoc.* **110**(511), 1304–1319 (2015). <https://doi.org/10.1080/01621459.2015.1050028>
5. Bourke, P.D.: Randomized response multivariate designs for categorical data. *Commun. Stat. Theory Methods* **11**(25), 2889–2901 (1982). <https://doi.org/10.1080/03610928208828430>
6. Casella, G., Berger, R.L.: *Statistical Inference*. Duxbury/Thomson Learning, Australia; Pacific Grove, CA (2002)
7. Dalenius, T.: Towards a methodology for statistical disclosure control. *Statistisk Tidskrift* **15**(429–444), 2–1 (1977)
8. Duchi, J.C., Jordan, M.I., Wainwright, M.J.: Local privacy and statistical minimax rates. In: 2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 1592–1592, October 2013. <https://doi.org/10.1109/Allerton.2013.6736718>
9. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: *Proceedings of the Conference on Theory of Cryptography* (2006). <https://doi.org/10.1007/11681878%5F14>
10. Erlingsson, Ú., Pihur, V., Korolova, A.: RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1054–1067. CCS 2014, ACM, New York (2014). <https://doi.org/10.1145/2660267.2660348>
11. Fanti, G., Pihur, V., Erlingsson, Ú.: Building a RAPPOR with the unknown: privacy-preserving learning of associations and data dictionaries. [arXiv:1503.01214](https://arxiv.org/abs/1503.01214) [cs], March 2015. <http://arxiv.org/abs/1503.01214>
12. Folsom, R.E., Greenberg, B.G., Horvitz, D.G., Abernathy, J.R.: The two alternate questions randomized response model for human surveys. *J. Am. Stat. Assoc.* **68**(343), 525–530 (1973). <https://doi.org/10.2307/2284771>

13. Greenberg, B.G., Abul-Ela, A.L.A., Simmons, W.R., Horvitz, D.G.: The unrelated question randomized response model: theoretical framework. *J. Am. Stat. Assoc.* **64**(326), 520–539 (1969). <https://doi.org/10.2307/2283636>
14. Kairouz, P., Bonawitz, K., Ramage, D.: Discrete distribution estimation under local privacy. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning, vol. 48, pp. 2436–2444. ICML 2016, JMLR.org, New York (2016). <http://dl.acm.org/citation.cfm?id=3045390.3045647>
15. Kairouz, P., Oh, S., Viswanath, P.: Extremal mechanisms for local differential privacy. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems*, vol. 27, pp. 2879–2887. Curran Associates, Inc. (2014). <http://papers.nips.cc/paper/5392-extremal-mechanisms-for-local-differential-privacy.pdf>
16. Lehmann, E.L., Casella, G.: *Theory of Point Estimation*. STS. Springer, New York (1998). <https://doi.org/10.1007/b98854>
17. Lensvelt-Mulders, G.J.L.M., Hox, J.J., van der Heijden, P.G.M., Maas, C.J.M.: Meta-analysis of randomized response research: thirty-five years of validation. *Sociol. Methods Res.* **33**(3), 319–348 (2005). <https://doi.org/10.1177/0049124104268664>
18. Moran, P.A.P.: The random division of an interval. *Suppl. J. Roy. Stat. Soc.* **9**(1), 92–98 (1947). <https://doi.org/10.2307/2983572>
19. Müller-Hannemann, M., Schirra, S. (eds.): *Algorithm Engineering*. LNCS, vol. 5971. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-3-642-14866-8>
20. SOSA: Symposium on Simplicity in Algorithms, January 2018. <https://simplicityalgorithms.wixsite.com/sosa/cfp>
21. Tang, J., Korolova, A., Bai, X., Wang, X., Wang, X.: Privacy loss in apple’s implementation of differential privacy on MacOS 10.12. [arXiv:1709.02753](https://arxiv.org/abs/1709.02753) [cs], September 2017. <http://arxiv.org/abs/1709.02753>
22. Umesh, U.N., Peterson, R.A.: A critical evaluation of the randomized response method: applications, validation, and research agenda. *Sociol. Methods Res.* **20**(1), 104–138 (1991). <https://doi.org/10.1177/0049124191020001004>
23. Warner, S.L.: Randomized Response: a survey technique for eliminating evasive answer bias. *J. Am. Stat. Assoc.* **60**(309), 63–69 (1965). <https://doi.org/10.1080/01621459.1965.10480775>

# **Outsourcing and Assisted Computing**



# Enforcing Access Controls for the Cryptographic Cloud Service Invocation Based on Virtual Machine Introspection

Fangjie Jiang<sup>1,2,3</sup>, Quanwei Cai<sup>1,2(✉)</sup>, Le Guan<sup>4</sup>, and Jingqiang Lin<sup>1,2,3</sup>

<sup>1</sup> Data Assurance and Communication Security Research Center, CAS,  
Beijing 100093, China

{jiangfangjie, caiquanwei, linjingqiang}@iie.ac.cn

<sup>2</sup> State Key Laboratory of Information Security, Institute of Information  
Engineering, CAS, Beijing 100093, China

<sup>3</sup> School of Cyber Security, University of Chinese Academy of Sciences,  
Beijing 100049, China

<sup>4</sup> Pennsylvania State University, State College, USA  
lug14@ist.psu.edu

**Abstract.** Most cloud providers afford their tenants with cryptographic services that greatly escalate the protection of users' private keys. Isolated from the guest operating systems (OSes), the keys are kept confidential even if the OS kernel is compromised. However, existing cryptographic services are ineffective in the access control of these critical services. In particular, they enforce controls for the key accesses mainly based on non-cryptographic authentication/authorization information (i.e., the identity and the password). Some platforms leverage other information such as the resource identification of the Virtual machine (VM) (e.g., IP address). Therefore, once the password is leaked, the attacker could invoke the cryptographic service in the victim VM. Moreover, sophisticated attackers can exploit vulnerabilities in the guest OS kernel and stealthily invoke cryptographic services. In this paper, we propose a new scheme named En-ACCI to improve the security of cryptographic service invocation in the cloud and achieve better access controls as well as auditing by leveraging the rich VM context provided by virtual machine introspection (VMI). To the best of our knowledge, we are the first in the literature to discuss these security issues involved in the invocation of cryptographic services in the cloud. We address the challenges by using an access control mechanism atop a set of optimization to VMI. We have implemented a prototype of En-ACCI, and our evaluation demonstrates that En-ACCI effectively addresses the authorization and audit issues in the cloud-based cryptographic service and the introduced performance overhead is modest.

**Keywords:** Cryptographic service · Cloud security  
Invocation security · VMI · Access control  
Virtualization-based security

## 1 Introduction

Cloud computing is becoming more and more popular due to its agility, elasticity, reliability, and scalability. Using cloud computing, an enterprise will reduce the investment in IT infrastructures and focus on their core business. They also greatly benefit from add-on services offered by cloud providers. For example, virtual cryptographic computing and key management services bring enhanced protections for the tenants' cryptographic keys.

In these cryptographic computation services, tenants do not store the private keys. Instead, the cloud providers keep the keys securely and provide interfaces for tenants to invoke the computations of signing/encryption/decryption. The cloud providers, as the delegate of the tenants, protect the private keys by storing them in physically-separated devices (e.g., hardware security module) and never leak them into unsecure environments. They may further block unauthorized invocations to the cryptographic services with strict access control policies. For instance, only an invoker possessing the correct identity and the corresponding password is authorized to execute the cryptographic computation. Due to the increasing security requirements from tenants, such add-on services have become one of the major competitions in the cloud computing market. Typical cryptographic computing services include AWS CloudHSM [3] and Alibaba Aliyun encryption service [1]. The key management service (KMS) includes AWS KMS [4], Alibaba Aliyun KMS [2], Microsoft Azure Key Vault [8], Tencent Tencyun KMS [10], etc.

Compared with other add-on services, the cryptographic service is much more important because it is used to ensure the security properties (e.g., data confidentiality, authentication, etc.) of other services. The cloud providers can improve the security of cryptographic services in two ways. First, they provide secure storage for the cryptographic keys. This can be achieved by existing techniques such as data encryption or dedicated devices. Second, the cloud enforces strict access control policies to the cryptographic service invocation and implements audit mechanisms of these invocations (e.g., AWS CloudTrail [5]). The latter records the invoker's identity, the source IP address, the time, the requested operation and the parameters of the operation in the specified storage for further compliance checks, security analysis, and troubleshooting.

However, existing approaches seem to be insufficient. For example, existing access control implementations of cryptographic services [2, 4, 8, 10] are based on the identity of invokers and the corresponding passwords, which may be leaked due to dishonest developers or operators. The monitor tools only record the account information, identity and IP address of VM, the requested operation information, and do not log the inner context of the VM. An adversary can easily circumvent such monitoring. In particular, the adversary having the correct identity and the password may trigger a malicious process in the victim VM that invokes the cryptographic service directly, or even remotely exploit existing process (e.g., through network connections or file handlers) for cryptographic service invocation.

We observe that existing defense leverages coarse-grained information, which cannot provide rich information about the context of service invocation. We envision a new defense that incorporates diverse in-context information to log the invocation of critical services. Except for the basic information such as identity and password, the cloud may also check the VM context. Combining these rich information can greatly raise the bar for the attacker to circumvent the audit system. For example, the tenant may specify a sound access control policy, allowing the cryptographic computation to be invoked only by legitimate processes in the whitelist under authorized uid at a specified time frame. Request for cryptographic computation is fulfilled only if the invoking process is not compromised and the information of the invoker comply with the predefined access control policy. On receiving the request, the cloud provider checks the identity, the password, and the inner context of the VM, before performing the cryptographic computation. In addition, the cloud also records the inner context of VM during computation, e.g., the processes, the network connections, opened files' information, account, requested operation information, VM's identifier and IP address, for a better audit.

There are two approaches to collect context information of service invocation. One is that the VM reports such information when interacting with the hypervisor. However, a malicious process could potentially manipulate the results submitted to the hypervisor, or even hijack the system calls to return false results. The other approach is based on VMI [17], which actively analyzes the VM state using memory forensic techniques. Although this method may also get the incorrect context information of service invocation, it is transparent to the VM. Also, it reads the memory of VM directly, so it is not subject to the attack which tampers with the results. Adopting VMI technology to collect the context information is promising. However, it still remains challenging:

- **Introspection timing.** Ideally, when the cryptographic service is invoked, the VMI components should be triggered immediately to check the VM state. However, VMI components are not tightly coupled with the programs in the VM that invoke cryptographic service. In order to be informed about the cryptographic events in time, we have two options. First, the VMI component is triggered only when the cryptographic service is invoked. In our scheme, the VMM invokes the VMI component when the cryptographic service request is received. Second, the VMI components monitor the whole VM continuously. Obviously, this would introduce considerable overhead.
- **Authenticity of the context of cryptographic services.** The VM kernel may be infected by rootkits. The rootkits may tamper with the invoker's process context, misleading the VMI components.

In this paper, we propose a new scheme named En-ACCI to improve the security of cryptographic service invocation in the cloud and achieve better audit by leveraging rich VM context provided by the VMI. The cloud returns the results of a cryptographic request only after checking that the VM is in a trustworthy state. Moreover, the cloud records the context information during crypto computing. En-ACCI innovates by adapting VMI for cryptographic service with



improved performance. Many schemes provide cryptographic service, such as KMS [2, 4, 8, 10], virtio-ct [20], vTPM [18] and etc. KMS provides cryptographic service via https networks connections. In virtualization platform, VMM emulates network card. That is to say, all the network I/O operations of the VM are trapped into the VM monitor (VMM). Similar to KMS, virtio-ct implemented a software HSM. Each cryptographic service request also causes the VM to be trapped into the VMM. To address the aforementioned challenges, we need to modify the I/O handle module in the VMM. Once the I/O module finds that the VM-Exit event is related to the cryptographic service, En-ACCI will be triggered. To get the correct context information of the invoker, we use the VMI tool to bridge the semantic gaps. Moreover, before collecting the context information of VM, En-ACCI scans rootkit in the VM. We have implemented an En-ACCI prototype based on QEMU/KVM [7, 9], and integrated it with a cryptographic service named virtio-ct [20] which stores the cryptographic keys in a dedicated storage and completes the cryptographic computation in the trusted VMM. The main contributions of En-ACCI are as follows:

- To the best of our knowledge, we are the first in the literature to discuss the security issues involved in the invocation of cryptographic service in the cloud environment. We propose to address the associated challenges by using a new access control mechanism atop virtual machine introspection techniques.
- We have developed a set of optimization to existing VMI techniques to improve the performance of En-ACCI.
- We have implemented a prototype of En-ACCI, and our evaluation demonstrates that En-ACCI effectively addresses the authorization and audit issues in the cloud-based cryptographic service and the introduced performance overhead is modest.

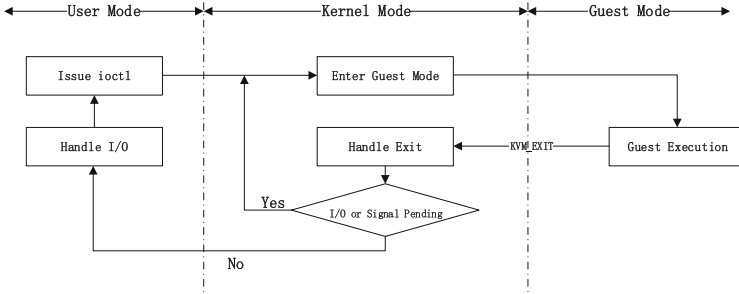
The rest of the paper is organized as follows. We introduce the background in Sect. 2 and describe the design of En-ACCI in Sect. 3, followed by the implementation in Sect. 4. In Sect. 5, we analyze the performance and security of En-ACCI. Related work is introduced in Sect. 6. Finally, we draw the conclusion in Sect. 7.

## 2 Background

In this section, we firstly give a description of Kernel-based Virtual Machine (KVM), a popular virtualization solution based on QEMU. Then, we introduce essential knowledge about the virtual machine introspection (VMI) [17] and Executable and Linking Format [15] to better understand our solution.

### 2.1 Kernel-Based Virtual Machine

The KVM [23] is a popular virtualization solution based on QEMU. Taking advantage of hardware-assisted virtualization extensions such as Intel VT and AMD-V, it supports executing guest intrusions naively in the host system, thus improving performance significantly. At a lower level, it is implemented as a



**Fig. 1.** Guest execution loop

loadable kernel module, which provides a set of `ioctl()` system calls to QEMU. QEMU, as a user-land process, is responsible for machine emulation, scheduling, resource allocation, and isolation etc. Therefore, QEMU can be viewed as a Virtual Machine Monitor (VMM).

In QEMU-KVM, except for the traditional kernel mode and user mode, another execution mode called guest mode is added, as shown in Fig. 1. The guest mode is essentially the mode in which the guest OS runs. When an exception or whatever critical system event configured by the KVM is caught, the VM exits, which is intercepted by the KVM module running in the kernel mode. Then KVM handles the exception either in the kernel mode directly and returns, or forwards it to the user mode. In the latter case, after handling the exception, QEMU calls another `ioctl` system call to resume the guest execution.

## 2.2 Virtual Machine Introspection

The Virtual Machine Introspection (VMI) is a technology in which the VMM dynamically inspects the execution context (internal state) of each VM. The purposes are mainly for security checking [17, 19], software patching [13, 14], and digital forensics [11]. To implement VMI functions, the VMM has to recover the semantic information of the inspected VM from the view of physical memory.

Depending on the VM OS, the VMI tools get the internal state of VM directly or reconstruct the high-level semantics. Installing a secret data collection module in the virtual machine [19, 26] makes VMI tools effectively get the VM state. But there is a risk that the data collection module may be compromised. Utilizing the OS knowledge (e.g., system symbol map and kernel data struct) of the VM, VMI tools [11, 17] reconstruct high-level semantics from the underlying binary data. And these VMI tools inspect the VM memory directly without installing an assistant module in the VM. VMM also obtains the state data of specific hardware devices of VM externally and then deduces high-level semantic information with the help of hardware architecture knowledge. Antfarm [22] proposed a hardware-based scheme to enable the VMM to track the processes and infer critical process events such as creation, context switch and exit.

### 2.3 Executable and Linking Format

This section describes an object file format called the Executable and Linking Format (ELF) [15], which is widely used by many OSes including Linux, Solaris, IRIX and OpenBSD. The proposed En-ACCI needs to access information stored in the ELF file to perform integrity check of the code segments. There are three types of ELF files: relocatable file, executable file and shared object file. All of them share a very similar format but are used for different purposes. As implied by the name, the executable file is directly loaded into the memory by OS for execution. The relocatable file contains code and data for individual program modules, which cannot be executed by its own, but needs to be statically linked with other object files to assemble an executable file. Similar to relocatable files, the shared object file is a module of the whole program. However, it is not statically linked during compilation. Rather, it is dynamically loaded into the memory by a dynamic loader implemented by relevant run-time.

Code and data in an object file are organized into sections, which are pointed to by a descriptive **section header table**. The sections are the basic modules which are linked together by the linker during linking process. In particular, sections with the same attributes are combined together to form a new section. Another important meta-data in an ELF file is the **program header table**, which provides program information for the loader. Therefore, it is mandatory for an executable file. A segment is essentially a chunk of aligned data/code with the same attributes. An entry in the **program header table** designates each segment's starting/ending addresses, the corresponding offset to the file image and attributes. The loader is responsible for loading the contents from file to the virtual memory.

## 3 System Design

In this section, we first present the threat model, followed by an overview of En-ACCI. Then, we detail our design from two aspects – access control policy and memory analysis methodology.

### 3.1 Threat Model

En-ACCI is designed to prevent unauthorized invocation to the cryptographic service. We consider an adversary model in which the VM OS is partially compromised by the attacker. Concretely, the attacker could invoke the cryptographic service by executing any user-space code in the VM. He could also conceal his unauthorized access to cryptographic service by leveraging rootkits that manipulate system log, process, network and file handle.

As with other VMI systems, we assume the integrity of the basic memory layer of an OS kernel. That is, the logical addresses of the kernel symbols and static kernel data structure are not modified [25].

In a cryptographic cloud service, the client invokes the cryptographic service through the provided program interface. We assume that the cryptographic key is

well protected by the provider, and the cryptographic computation is performed in a protected environment. The adversary cannot infer the cryptographic key from the provider through side channel attacks. Moreover, we assume the operators of the cryptographic service are honest and never leak the key, invoke the service nor modify the log information illegally. We assume that the design and implementation of cryptographic algorithms are secure. Finally, VMM is free of bugs. This is a very practical assumption considering the much smaller code base of VMM compared with full-fledged OSes.

### 3.2 Overview

A conceptual architecture of the proposed solution is illustrated in Fig. 2. As shown in the figure, En-ACCI is a software component in the VMM, which is non-bypassable when an app intends to invoke the cryptographic service. If the request is granted, En-ACCI forwards the request to the cryptographic service, which is securely implemented.

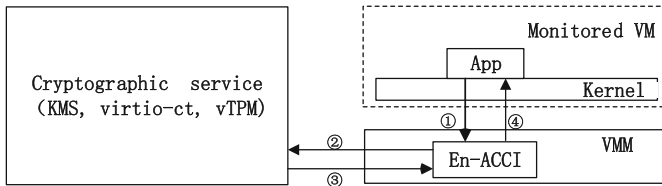


Fig. 2. En-ACCI architecture

To enforce security checking and auditing, we design our system to include the following processes.

- **Generating profile and hash meta-data.** For each version of OS, the operator needs to generate the corresponding profile, which includes the logic address of the essential kernel symbols and the offsets of elements in the kernel structure. This process is invoked only once for each version of the OS. The profile may be reused for different VMs with the same OS version and doesn't need to be regenerated when any kernel module is installed or removed. The tenant also needs to calculate the hashes of code segments in authorized binaries and relevant dynamic linking libraries. The hash meta-data include executable name, virtual addresses and length about the code segments. The hash meta-data are used to verify the integrity of the authorized processes.
- **Specifying access control policy.** The tenant may specify the access control policy for the service invocation on each cryptographic key. The access control policy defines the conditions that must be satisfied when the cryptographic service is invoked. The tenant may set the effective time for each policy and modify the policy when needed. In our implementation, the policy may specify the user ID, user group ID, name, start time, opened files

and established network connections of the process who may invoke the cryptographic service. Note that the list is flexible and can be extended in the future. The details of the access control policy are provided in Sect. 3.3.

- **Rootkit detection.** Once receiving the cryptographic service invocation, En-ACCI runs rootkit detector to check the state of the VM. Rootkit detector includes many VMI tools to check the critical kernel data and kernel text. If a rootkit is detected, the incident is reported.
- **Sampling and analyzing the VM memory.** Once receiving the cryptographic service request, VMM samples the necessary memory region of the VM and analyzes it to obtain the elements specified in the access control policy. We traverse the list of the processes in the VM and determine the process who invokes the cryptographic service according to the file handle or network connection corresponding to the cryptographic key.
- **Access control enforcement.** For each cryptographic service invocation, after sampling and analyzing the VM memory, we obtain the elements contained in the access control policy and compare the values of the elements obtained from the analysis with these defined in the policy. We fulfill the cryptographic service requests only when the policy is satisfied.

### 3.3 Access Control Policy

The tenant may define the access control policy based on different properties. These properties are extracted from the inner VM context. In the current version, En-ACCI supports the following access control properties.

- **PID, UID and GID.** After executing the application in the monitored VM, the tenant obtains the process identifier (PID), the identifier of the user that executes the application (UID), and the identifier of the corresponding user group (GID).
- **Process Name.** The name of the process is usually the name of the execution file, and is limited to 16 characters by default in Linux.
- **Location in the process tree.** The tenant may specify the blacklist and whitelist of the processes (according to PID, process name) that run concurrently in the VM or in the path from the init process to the application that invokes the service.
- **Process start time.** The tenant may obtain the accurate start time of the application externally, and specify it in the policy about the time frames during which cryptographic service can be invoked.
- **Opened file list.** The application may need to open a set of files in the different periods of normal execution. The cryptographic service is invoked in a critical region that should be well protected. Therefore, the tenant may specify the blacklist and whitelist of the files (through the file name and path) for this application during the cryptographic service invocation.
- **Allowed network connections.** Similar to the opened file list of the process, the tenant may specify the 5-tuple in the form of (src ip, src port, dst ip, dst port, protocol) of established connections during the service invocation.

The combination of the elements in the policy raises the bar for the adversary to invoke the cryptographic service without being noticed. Moreover, the tenant may find whether the VM contains suspicious processes or kernel modules that stealthily invoke the cryptographic service illegally, by comparing the results returned by the command with the ones logged in En-ACCI, and update the access control policy (e.g., forbidding the service invocation until the VM state is recovered) in time.

### 3.4 Memory Analysis

En-ACCI adopts memory analysis to sample the memory of the monitored VM, figure out the memory region related with the application that invokes the service, and obtain the information required by the access control policy.

Memory analysis is performed in VMM, which has access to the memory image of the monitored VM. According to the profile of the corresponding OS, we obtain the logical address of the first process (e.g., `init` in Linux). After transferring it to the address in the VMM address, we analyze it to get the semantic information, based on the offset (defined in the profile) of each element. With the semantic information of the first process, En-ACCI gets the addresses of all the processes in the VM, and obtains their semantic information. En-ACCI figures out the memory region of the cryptographic service invoker by comparing the information of file list or networking connections with the ones corresponding to the cryptographic key handler. After figuring out the region corresponding to the service invoker, En-ACCI parses the figured memory region to obtain the semantic information, compares the parsed values with the ones specified in the policy, and finally returns the decision to the cryptographic service.

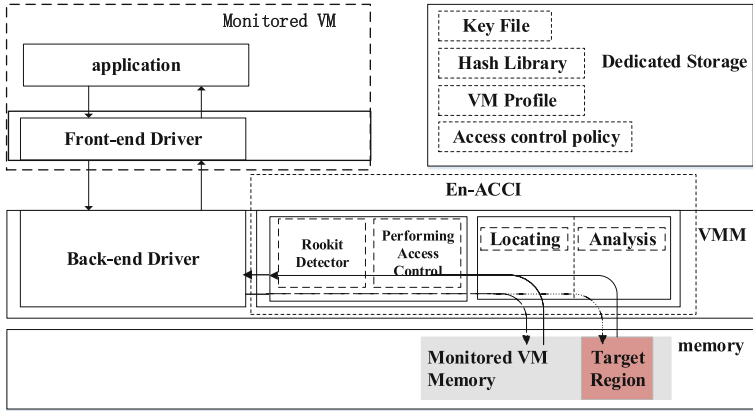
However, after obtaining the address of the process, the corresponding process may exit, which means that semantic information parse fails. In this case, we need to re-parse the previous process in the process chain to obtain the address of the cryptographic service invoker process again. The previous process may also exit before parsing. In the worst case, we need to trace back to the first process (i.e., `init` in Linux).

## 4 Implementation

We have implemented a prototype of En-ACCI based on QEMU-KVM v1.7.1. En-ACCI supports Linux distributions with kernel version 2.6 or above, and Windows OS as VMs. In the following, we describe our implementation on the CentOS v6.6 (Linux 3.13.7).

### 4.1 Framework Implementation

Figure 3 demonstrates the architecture of En-ACCI. We showcase how En-ACCI is used to protect `virtio-ct` [20], a cryptographic token implemented on KVM.



**Fig. 3.** The architecture of En-ACCI when integrating with virtio-ct [20].

In virtio-ct, the application in the monitored VM invokes the cryptographic service through a file handle, which is associated with the front-end driver. The parameters for the cryptographic computation include the identity of the cryptographic key, the corresponding PIN and the plaintext/ciphertext. The front-end driver routes these parameters to the virtio bus. The back-end driver in QEMU fetched the arguments on the bus and performs the corresponding cryptographic computation. The back-end driver invokes En-ACCI before performing the actual computation. En-ACCI conducts the following two steps. It first determines whether the VM is infected by rootkits, and then checks whether the inner context of the monitored VM satisfies the specified access control policy. More specifically:

- **Step 1:** Before performing the cryptographic computation, En-ACCI checks whether the request is invoked in a benign environment. This prevents the back-end driver from being abused by attackers, e.g., accessing keys without authorization or DoS attack by invoking intensive cryptographic requests. Here, the rootkit detector sequentially invokes all the included tools to check (known) rootkits. To check whether the inner VM state satisfies the specified policy, En-ACCI first identifies the memory region of the process that invokes the cryptographic service. Regarding virtio-ct, we can locate the invoker based the opened files of the process, as the invoker needs to open the special file that represents the cryptographic key. Then, En-ACCI parses the identified memory region, and performs the access control according to the obtained semantic information. If any rootkit is detected or the access control policy is not satisfied, the back-end driver refuses to fulfill the cryptographic computation request.
- **Step 2:** Before returning the result to the front-end driver, En-ACCI performs checking to avoid the result to be obtained by the illegal user who comprises the VM or invoker process during the cryptographic computation.

In this step, En-ACCI reuses the result obtained in the previous step. For example, there is no need to identify the invoker’s memory region. It directly invokes the rootkit detectors and performs access control by parsing the previously identified memory region. If the parse fails, or any rootkit exists, or the access control policy is not satisfied, the back-end driver refuses to return the computation results.

Except for virtio-ct [20], En-ACCI supports other types of the cryptographic cloud services. The major difference is the way to identify the corresponding cryptographic processes. In virtio-ct, we rely on the process’s opened file information, which must include the corresponding the cryptographic key. For AWS CloudHSM and Alibaba Aliyun KMS, we identify a cryptographic process using the network connection information, because the process must maintain a network connection to the cryptographic service provider.

### 4.2 En-ACCI Implementation

Linux kernel adopts the data structure `task_struct` to store the metadata of a process. As shown in Fig. 4, the fields `comm`, `cred`, `pid` and `start_time` contain information about the process name, UID/GID, PID, process start time, while fields `files` and `fs` store information about open file information (including network connections) and filesystem information. Note that `tasks` is a list data structure used by Linux to keep all the processes in a linked list.

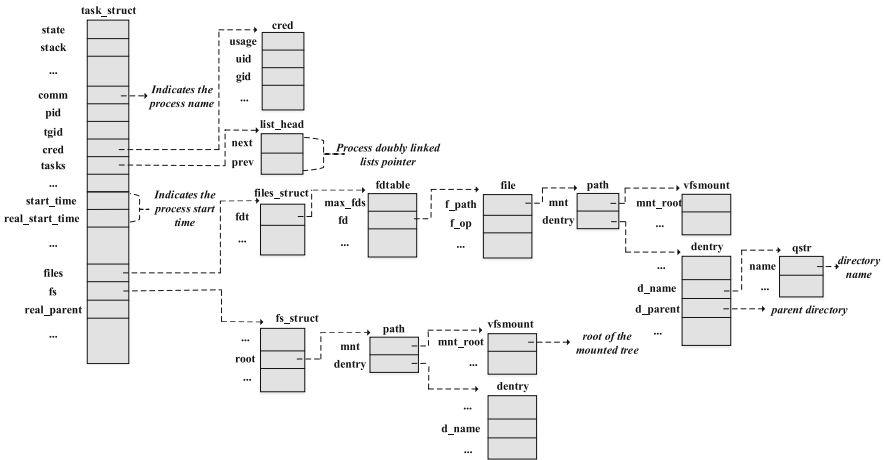


Fig. 4. The Linux process descriptor

We have generated an aforementioned profile For Linux 3.13.7. The profile records the logical address `init_task` and the offsets of the variables in `task_struct` shown in Fig. 4. En-ACCI adopts these information to parse the



semantic information of the `init` process and then follows the linked list indicated by the `tasks` field (offset is 1640) to obtain the logical addresses of the other processes. For each process, the PID, UID, GID, process name, and start time are obtained directly from the `task_struct`, while, En-ACCI needs to parse the fields `files` (of type `files_struct`) and `fs` (of type `fs_struct`) to infer the information of associated files. The variable `fs` provides the root dentry through the variable `root` (of type `path`). The variable `fdt` (of type `fdtable`) in the structure `files_struct` contains the information of all files opened by this process. That is, the member `fd` of structure `fdtable` lists the dentries for each opened file. The network connection is represented by a file in Linux. We distinguish it from normal files through the variable `f_op` in the structure `file`. The `f_op` points to the operation that can be invoked on this file, but points to the function `socket_file_ops` when the file is a socket.

The profile only provides the guest logical address of each symbol, while En-ACCI, being a component of VMM, can access the contents through guest physical address in the underlying host OS. Therefore, we need to perform the address translation. En-ACCI invokes `cpu_physical_memory_rw` provided by QEMU-Monitor to transfer the guest logical address to the guest physical address, and then access it for further parse. En-ACCI also records the address returned by `cpu_physical_memory_rw`, and uses it with the offset to access the memory in the host directly when no new guest logical address needs to be processed. We implemented the aforementioned procedure in a lightweight way. The whole implementation comprises about 700 lines of code.

The hash library provides the authorized process names, virtual addresses of the code segments when the authorized processes is loaded into memory and the digest of each code page. We analyze the authorized binary files offline in advance. We calculate the digest of each page in the code segments from start. For pages contributing to both code segments and data segments, we replace the data regions with all zero. Then we calculate the digest of each page and store them in the hash library. En-ACCI determines if the virtual page has been loaded into physical memory by `bit0` in the `page table entry` and verifies the integrity of the virtual page.

In our prototype, En-ACCI is integrated with `virtio-ct` [20], in which the application invokes the cryptographic service through a virtual device (i.e., a file). En-ACCI uses the information of this file to figure the memory regions corresponding to the application. The semantic analysis of this memory region and the access control process are performed twice, one before performing the cryptographic computation and the other before returning the result to the application. Enforcing the access control is implemented in less than 100 lines of code.

En-ACCI also logs the information for each cryptographic service invocation. In addition to the user identity, IP address and the geographical region of VM, the parameters and invocation time of the cryptographic service, it also includes the PID, UID, GID, process name, process start time, the parent process, opened files, established network connections, and the result of integrity checking.

For rootkit detector, we have implemented three VMI tools to detect a set of kernel rootkits that alter the control flow by modifying the critical kernel data

(e.g., IDT and system call table) or directly manipulate the kernel text. The three VMI tools check the integrity of IDT, system call table and kernel text respectively. Since our implementation is based on KVM, other popular VMI tools can be easily integrated in our framework.

## 5 Analysis

In this section, we evaluate the performance overhead introduced by En-ACCI and analyze the security improvement brought to cryptographic service invocations. Finally, we discuss the limitations of En-ACCI in our current prototype.

### 5.1 Performance Evaluation

The prototype of En-ACCI is implemented as a module for QEMU/KVM v1.7.1 using the C programming language. The profile of the target operating system is generated using memory forensics tool volatility [11] and dwarf-tools. We set up our evaluation environment with a Dell Optiplex 9020 powered an Intel i7-4770 CPU (3.4 GHz) and 16 GB RAM. We assigned 4 vCPUs and 4 GB RAM for the VMs. The host operating system and the VM run CentOS v7.0 and CentOS v6.6 respectively.

We compared the results obtained from En-ACCI with that obtained from the original virtio-ct and LibVMI-virtio-ct, which performs access control for virtio-ct based on LibVMI. LibVMI-virtio-ct is implemented as follows: on receiving the service invocation, it suspends the VM, invoking the LibVMI to analyze the memory for the entries specified in the access control policy, and returns the computation results to the resumed VM.

The performance was evaluated at different concurrency levels. In the experiment, we used an application with at most 8 threads to invoke 2048-bit RSA decryptions. The throughput is demonstrated in Fig. 5. Compared with the original virtio-ct, the throughput of En-ACCI is reduced by 17.77% (the number

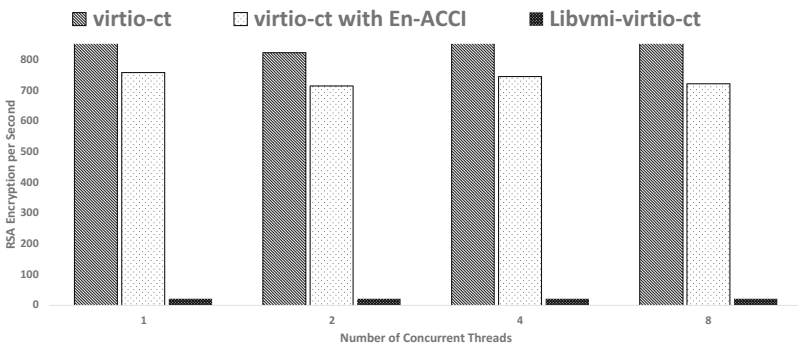


Fig. 5. Throughput of cryptographic service invocation.

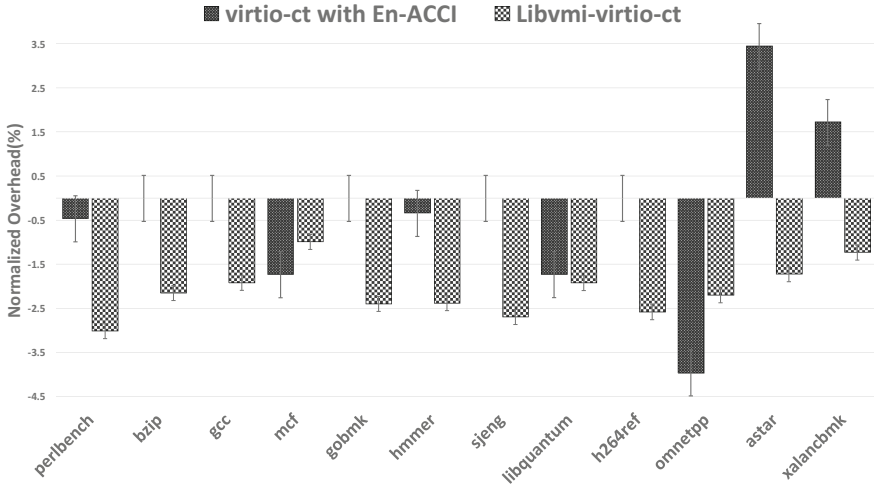


Fig. 6. SPEC INT 2006 perf. overhead.

of threads is 8), which is modest compared to 96.71% in LibVMI-virtio-ct (the thread number is 4).

We also ran SPECINT 2006 to evaluate the impact of En-ACCI to the other parallel processes. We set the performance of native virtio-ct [20] as the baseline, and compared En-ACCI with Libvmi-virtio-ct by running SPECINT 2006 when invoking the cryptographic service every 5 s with En-ACCI and Libvmi-virtio-ct integrated respectively. As shown in Fig. 6, the impact of En-ACCI to the other parallel processes is modest (less than 4.5%).

## 5.2 Security Analysis

En-ACCI raises the bar for the adversary to invoke the cryptographic cloud service from three aspects: 1. En-ACCI doesn't increase the risk of cryptographic key leakage itself, as it doesn't need the key for access control or audit. 2. En-ACCI enforces the access control policy specified by the tenant for each service invocation, which increases the difficulties of illegal invocation. 3. En-ACCI records the detailed information for each cryptographic cloud service invocation. This information is essential for causality analysis when the system is compromised.

For access control, in addition to identity and password used in KMS [2, 4, 10], En-ACCI leverages the approach described in Sect. 3.3 to extract inner VM state. En-ACCI checks user ID, user group ID and start time of the process to ensure the invoker process is created by the authorized user at the specified time. The attacker would fail to invoke the cryptographic service if the invoker process is started by a different account or is not started during the allowed time frames. En-ACCI compares the name of the process and digests of the process's code segment with that specified in the profile and hash library, to ensure only the

authorized executable program without integrity compromise can invoke the cryptographic service. The specified list of opened files and established network connections prevents the remote attacker who hijacks the victim process through file handles or network connections from invoking the cryptographic service.

In order to prevent kernel rootkit from manipulating the target memory region to fool En-ACCI, En-ACCI integrates existing rootkit detector. Once a rootkit is found, the cryptographic service is interrupted. Moreover, En-ACCI performs two steps for each service invocation; the first one prevents the attacker from abusing the cryptographic computation, while the second one ensures that the computation result is only returned to the legitimate invoker.

En-ACCI also has access to the detailed information about the inner VM state during the service invocation. Therefore, even if the access control mechanism is somehow bypassed (e.g., when the policy is incorrect), the cloud manager can identify the illegal invocation in time, analyze the malicious invocation thoroughly and modify the access control policy in time to avoid further malicious invocations.

The adversary who controls VM cannot compromise En-ACCI due to the isolation mechanism provided by the virtualization. Moreover, the source code of En-ACCI is only about 700 lines, which makes the formal analysis feasible.

### 5.3 Limitations

We discuss the limitations of the current En-ACCI prototype. En-ACCI relies on logical addresses of the kernel symbols and the kernel data structure to perform semantic analysis. However, as described in [12], the obtained semantic information may be incorrect when the kernel data structure is manipulated. In the current version, we rely on the followings assumptions to ensure the correctness of the obtained semantic information. (1) The guest OS is patched in time for known kernel-level vulnerabilities, which allow the attacker to hijack the control flow of the kernel. We admit that the attacker may still able to hijack the control flow using the zero-day vulnerabilities. (2) The integrity of the kernel is checked using existing rootkit detection tools. Therefore, our protection relies on the effectiveness of existing tools.

## 6 Related Works

In this section, we summarize existing work on secure service invocation and cryptographic key protection.

### 6.1 Cloud-Based Cryptographic Services

Virtio-ct emulates an HSM in VMM. Utilizing the isolation mechanism of the hypervisor, any code, including ring-0 malicious code in the guest OS cannot access cryptographic keys. To prevent the adversary from stealthily signing data, each time the key is accessed, virtio-ct drives the `pc-speaker` to make a sound

to notify the user. However, virtio-ct is designed for single PC scenarios instead of cloud.

For KMS [2, 4, 8, 10], cloud providers provide access control strategy. KMS can be invoked in three ways in the current commercial design: (1) the web-based console (2) the command line interface and (3) the cloud service API. Although a variety of user authentication and access control are provided, the security of KMS ultimately depends on the identity management and password-based authentication mechanism. Audit is also provided in several providers. For example, AWSCloudTrail [5] can record very basic information of the cryptographic invocations.

## 6.2 Cryptographic Keys Protection

Various schemes have been proposed to protect the confidentiality of the cryptographic key against memory disclosure attacks. New features [6, 16] provided by CPU manufacturer were adopted to protect the cryptographic keys. For example, Mimoso [21] only keeps plaintext of sensitive data in the transaction memory, which rolls back to the ciphertext once the memory is accessed by others. Intel Software Guard Extension (SGX) [6] allows user-level code to allocate private regions of memory, called enclave, which is isolated from the rest of the system, including OS and BIOS. With it, cryptographic services can be securely implemented.

As more and more services are being migrated to the cloud, the associated security problems emerge. To mitigate attacks from the inner VM, Virtio-ct [20] provides virtual cryptographic service while the corresponding key files are stored in the dedicated storage and the cryptographic program is executed in VMM. To prevent the attacks to VMM, TrustVisor [24] introduces a small hypervisor as TCB to enforce the data secrecy and program integrity, even if the OS is compromised.

En-ACCI adopts virtio-ct to prevent the attacks from the VM. It may also integrate the other works [6, 21] to avoid the memory disclosure attacks on the physical machine when VMM is deployed, and adopt the schemes proposed in [24] to reduce the size of TCB where the cryptographic computation is performed.

## 7 Conclusion

We propose En-ACCI, a VMI-based mechanism to add another line of defense for cryptographic cloud services. En-ACCI enforces the access control for the cryptographic cloud service based on the rich VM context, and provides better audit by recording the detailed information of the VM and invoker process. To achieve better performance, instead of adopting existing VMI tools (e.g., libvmi) directly, En-ACCI analyses the VM's memory based on the logical addresses and the kernel data structure of the guest OSes, and further identifies and parses the memory regions of the invoker process in VMM. To ensure the correctness of the semantical information, En-ACCI integrates existing rootkit detection

tools and checks the integrity of the invoker's code segments. The performance evaluation demonstrates that the performance overhead caused by En-ACCI is modest (about 17.77%).

**Acknowledgments.** This work was partially supported by National Natural Science Foundation of China (No. 61772518).

## References

1. Aliyun crypton service. <https://www.aliyun.com/product/hsm>
2. Aliyun Key Management Service. <https://www.aliyun.com/product/kms>
3. Amazon aws cloudhsm. <https://amazonaws-china.com/cloudhsm/>
4. Amazon aws key management service kms. <https://amazonaws-china.com/kms/>
5. Aws cloudtrail. <https://amazonaws-china.com/cn/cloudtrail>
6. Intel corporation, intel software guard extensions. <https://software.intel.com/en-us/sgx>
7. Kernel based virtual machine. [http://www.linux-kvm.org/page/Main\\_Page](http://www.linux-kvm.org/page/Main_Page)
8. Microsoft key vault. <https://www.azure.cn/home/features/key-vault/>
9. Qemu open source processor emulator. [http://wiki.qemu.org/Main\\_Page](http://wiki.qemu.org/Main_Page)
10. Tencentyun key management service kms. <https://cloud.tencent.com/product/kms>
11. The volatility framework. <https://code.google.com/archive/p/volatility/>
12. Bahram, S., Jiang, X., Wang, Z., Grace, M., Li, J., Srinivasan, D., Rhee, J., Xu, D.: DKSM: subverting virtual machine introspection for fun and profit. In: 2010 IEEE Symposium on Reliable Distributed Systems, pp. 82–91 (2010)
13. Biedermann, S., Katzenbeisser, S., Szefer, J.: Leveraging virtual machine introspection for hot-hardening of arbitrary cloud-user applications. In: 6th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2014, Philadelphia, PA, USA, 17–18 June 2014 (2014)
14. Chen, P., Xu, D., Mao, B.: Clouder: a framework for automatic software vulnerability location and patching in the cloud. In: 7th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2012, Seoul, Korea, 2–4 May 2012, p. 50 (2012)
15. TIS Committee et al.: Tool interface standard (tls) executable and linking format (elf) specification version 1.2. TIS Committee (1995)
16. Intel Corporation: Chapter 8: Intel transactional memory synchronization extensions. In: Intel Architecture Instruction Set Extensions Programming Reference (2013)
17. Garfinkel, T., Rosenblum, M.: A virtual machine introspection based architecture for intrusion detection. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA (2003)
18. Goyette, R.: A review of vTPM: virtualizing the trusted platform module. In: Proceedings of Network Security and Cryptography (2007)
19. Gu, Z., Deng, Z., Xu, D., Jiang, X.: Process implanting: a new active introspection framework for virtualization. In: 30th IEEE Symposium on Reliable Distributed Systems (SRDS 2011), Madrid, Spain, 4–7 October 2011, pp. 147–156 (2011)
20. Guan, L., Li, F., Jing, J., Wang, J., Ma, Z.: virtio-ct: a secure cryptographic token service in hypervisors. In: Tian, J., Jing, J., Srivatsa, M. (eds.) SecureComm 2014. LNICST, vol. 153, pp. 285–300. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-23802-9\\_22](https://doi.org/10.1007/978-3-319-23802-9_22)

21. Guan, L., Lin, J., Luo, B., Jing, J., Wang, J.: Protecting private keys against memory disclosure attacks using hardware transactional memory. In: 2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, 17–21 May 2015, pp. 3–19 (2015)
22. Jones, S.T., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H.: Antfarm: tracking processes in a virtual machine environment. In: Proceedings of the 2006 USENIX Annual Technical Conference, Boston, MA, USA, 30 May–3 June 2006, pp. 1–14 (2006)
23. Kivity, A.: kvm: the linux virtual machine monitor. In: Linux Symposium, Ottawa, Ontario (2007)
24. McCune, J.M., Li, Y., Qu, N., Zhou, Z., Datta, A., Gligor, V.D., Perrig, A.: Trustvisor: efficient TCB reduction and attestation. In: 31st IEEE Symposium on Security and Privacy, S&P 2010, 16–19 May 2010, Berkeley, Oakland, California, USA, pp. 143–158 (2010)
25. Seshadri, A., Luk, M., Qu, N., Perrig, A.: Secvisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity oses. In: Proceedings of the 21st ACM Symposium on Operating Systems Principles 2007, SOSP 2007, Stevenson, Washington, USA, 14–17 October 2007, pp. 335–350 (2007)
26. Vogl, S., Kilic, F., Schneider, C., Eckert, C.: X-TIER: kernel module injection. In: Lopez, J., Huang, X., Sandhu, R. (eds.) NSS 2013. LNCS, vol. 7873, pp. 192–205. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-38631-2\\_15](https://doi.org/10.1007/978-3-642-38631-2_15)



# Multi-authority Fast Data Cloud-Outsourcing for Mobile Devices

Yanting Zhang<sup>1,2</sup>, Jianwei Liu<sup>1</sup>, Zongyang Zhang<sup>1(✉)</sup>, and Yang Hu<sup>3</sup>

<sup>1</sup> School of Cyber Science and Technology, Beihang University, Beijing, China  
{yantingzhang, liujianwei, zongyangzhang}@buaa.edu.cn

<sup>2</sup> Shenyuan Honors College, Beihang University, Beijing, China

<sup>3</sup> School of Electronic and Information Engineering, Beihang University,  
Beijing, China

lucyyang@buaa.edu.cn

**Abstract.** We propose a multi-authority fast data cloud-outsourcing (MFDCO) scheme especially suitable for mobile devices. It is a multi-authority online/offline encapsulation scheme based on efficient large-universe ciphertext-policy attribute-based encryption, and supports fine-grained access control, dynamic revocation, and public validity test. Any party can become an authority to participate in the distribution of attribute credential and credential updating. Apart from the initial generation of public global parameters, there is no requirement for any coordination among distinct authorities. In addition, the MFDCO scheme allows data owners to enforce fine-grained access control through lightweight online operations, which is extremely friendly for mobile users. It is equipped with an efficient revocation mechanism to realize dynamic access credential revocations. It also allows public encapsulation validity test, thus preventing attackers from stuffing users' data storage accounts with invalid encapsulations, as well as achieving security against active attacks. Comprehensive analyses illustrate that the MFDCO scheme is suitable for commercial sensitive data cloud-outsourcing, especially in public cloud environment.

## 1 Introduction

Cloud computing is a promising distributed computation paradigm for large pool of shared resources, which can provide numerous benefits, e.g., broader data availability, reduced IT resource costs, better flexibility and increased collaboration. Data cloud-outsourcing, as an important part of cloud computing, has received considerable attention from both industries and academic organizations. By providing an on-demand, self-service, and pay-as-you-go business model [7], cloud outsourcing, especially outsourcing to public cloud, is an economic and convenient approach for efficient data sharing. Combined with online data management functionality, a data cloud-outsourcing system can provide flexible access control and almost unbounded resources in an on-demand fashion at low prices. In this way, enormous amount of commercial documents and



customer data can be outsourced to public cloud, so that corporations are shifted away from deploying expensive data storage systems and complicated data management systems. Meanwhile, employees can share data to the cloud and potential data consumers can access documents through public cloud conveniently.

While data cloud-outsourcing system provides numerous benefits to users, there are still various remaining problems. We will discuss three main challenges, which are untrusted cloud service providers (CSP), weak security models, and the difficulty of cross-domain data sharing.

First, an untrusted CSP may use corporations' sensitive data without authorization. To protect outsourced data against an untrusted CSP, a promising approach is to use cryptographic tools. Many up-to-date cryptographic access control countermeasures have been proposed to address this problem [11, 31, 32]. Comparing with traditional systems, cryptographic countermeasures allow corporation managers to enforce access control policies even when the CSP is untrusted. However, while mitigating this security risk, the introduction of cryptography incurs new problems such as key management, heavy computation and additional storage overheads. It is worth noting that existing proposals [13, 24, 30] involve numerous expensive asymmetric cryptographic operations, and the computation complexity increases with the complexity of the chosen access policy, making it unaffordable for mobile users to enforce fine-grained access control.

Second, current security models are too weak to capture realistic attacks in cloud storage scenarios. They mainly prevent passive attacks and collusion attacks. However, in public cloud outsourcing scenarios, for one thing, attackers may have the ability to collect data leaked by target users or even manipulate data to reveal partial sensitive information [2]. For another, as encapsulated data look random, attackers may easily stuff users' cloud storage accounts with junk data at a very low cost. Such attacks have not been fully considered previously. Few existing similar proposals, except for one [32] we previously proposed, can provide an efficient way to publicly verify encapsulation and filter junk data without sacrificing the efficiency of encapsulation procedure.

Third, in most existing access control proposals for cloud outsourcing, access credentials are managed by a central trusted authority. This centralization requirement results in a restriction that these systems can only be used within one trust domain or a specific organization. However, in commercial applications, data owners need to share documents across different organizations. A possible scenario is that both corporations issue access credentials as part of a joint project, where single-authority access control systems might be problematic.

Based on the above observations, we are inspired to design cross-domain access control scheme to achieve secure data cloud outsourcing.

## 1.1 Our Contributions

We propose a multi-authority fast data cloud-outsourcing (MFDCO) scheme especially suitable for mobile devices. It is a multi-authority online/offline encapsulation scheme based on efficient large-universe ciphertext-policy attribute-based encryption (CP-ABE), and supports fine-grained access control, dynamic

revocation, public encapsulation validity test. Compared with previous works [12, 18, 19], our scheme supports more functionalities, and is much faster in real-time data encapsulation. These features make our scheme a promising solution to secure data outsourcing in public cloud. In more detail, our MFDCO scheme satisfies the following properties:

- **Decentralization:** Any party can become an authority to distribute and update users’ access credentials, achieving decentralization.
- **Fine-grained access control:** Our scheme enforces fine-grained access control on the data record level for different users. Each data record is associated with an access policy. Only parties whose roles match the access policy specified by the data are able to decapsulate.
- **Dynamic credential revocation:** By publishing credential updating periodically, our scheme achieves dynamic access credential revocation without changing the global public parameters.
- **Public encapsulation validity test and junk encapsulation filtering:** With the help of a well-constructed verification term and the inner relationship of encapsulation components, an attacker sending random junk data can be easily detected and filtered. The cost of checking encapsulation validity is much less than that of generating well-formed encapsulations, which greatly mitigates the threat of junk data attack.
- **Online/offline encapsulation:** Most heavy computations are completed in the offline stage. In the online encapsulation stage, data owners only need to execute lightweight operations, which enables mobile users to realize real-time data outsourcing.
- **Security against active attacks:** An attacker can collude with CSP and all other users except the target user, and can adaptively know the decapsulated data except the target one.

## 1.2 High-Level Idea of Our Construction

To achieve decentralization in ABE, it is natural to think about equipping each authority with a different master secret key, and then sharing to data consumers through issuing corresponding part of access credentials. However, attackers may be able to control a set of corrupted authorities and make collusion attacks.

Recall in a single-authority ABE, a central authority chooses a unique set of random elements for each user to “tie” together different components of a specific private key, each component representing a different attribute. This method will be problematic in multi-authority settings, as these key components may be issued by different authorities. To resolve this problem, we import the Lewko and Waters (LW-11) multi-authority technique [16], which uses a unique global identity  $GID$  and applies a hash function over  $GID$  to “tie” together attribute credential components belonging to a specific user. The secret information can be extracted only when a data consumer’s whole access credential satisfies the policy of this encapsulation, so that attackers cannot combined their attribute credentials with other users’ to decapsulate unauthorized data.

The LW-11 decentralizing ABE works in composite order groups, where exponentiation operations are too slow for real-time data outsourcing. Besides, practical cloud-outsourcing applications call for expressive access policy, and each authority is usually responsible for several attributes. However, in the LW-11 scheme, attribute universe is restricted to polynomial size, each authority supports only one attribute and can be used only once when describing access policies, which obviously restricts its application. So, we further introduce the Rouselakis-Waters (RW-15) technique [20] to pare this decentralization mechanism down to prime order groups, and remove the above restrictions. A trade-off of the RW-15 technique is a static security model in the random oracle model, where key queries must be issued before parameters are published.

Considering revocation, we resort to the Boldyreva-Goyal-Kumar (BGK-08) revocation technique [4]. However, when combining the BGK-08 and the RW-15 technique in a straightforward way, the static model of RW-15 requires attackers to issue a target list of revocation on target time beforehand to allow a simulator to prepare public global parameters. This requirement weakens the security result of our MFDCO scheme. To remove revocation list requirement, we introduce the Seo-Emura (SE-13) revocation technique [22]. After dividing the master secret key into two parts, which are respectively contained in the attribute credential and credential updating, we construct the attribute credential part under the Rouselakia-Waters (RW-13) secret key form [19] and the credential update part under the Boneh-Boyen (BB-04) secret key form [5]. In this way, revocation list and revocation time are separated in access credential generation and revocation, therefore enhancing the security of our MFDCO scheme.

In order to achieve real-time encapsulation, we adopt the Hohenberger-Waters (HW-14) online/offline technique [12]. Without the knowledge of assigned access policy or actual data, we cover most heavy algebraic group operations in the offline stage. In the online stage, data owners decide the actual access policy and finish the encapsulation with lightweight operations. The main challenge comes from that the public verification term is related to specific access policy and data, but both are unknown in the offline stage. Here, we introduce chameleon hash functions. In the offline stage, data owners randomly choose an access policy and a data item, calling the chameleon hash function to compute a verification term. In the online stage, data owners utilize the chameleon hash trapdoor to replace random elements with actual ones without changing the verification item, i.e., change the pre-image of the hash to string of actual data and policy. In this way, our MFDCO system allows public encapsulation validity test and invalid encapsulation filtering, and further achieve security against active attacks.

### 1.3 Related Work

The concept of ABE was introduced by Sahai and Waters [21], since then many extensions of ABE for security [6, 14] and efficiency [26] have been proposed. Due to the capability of providing fine-grained access control over encrypted data, ABE is an ideal paradigm for data cloud outsourcing applications, but improving

its efficiency is still a challenge. Although techniques like proxy re-encryption [1] and security mediated certificateless signatures [29] techniques can be used to reduce the encapsulation cost, integrated schemes still require heavy operations during data encapsulation. Furthermore, few schemes support ill-formed encapsulation public filtering while keeping online encapsulation procedure efficient.

Very recently, we proposed a fast data cloud-outsourcing scheme with public validity test and flexible access control for mobile devices (FDCO) [32]. However, the FDCO scheme is limited to a single central authority, which greatly restricts data sharing domain. In this work, we utilize a two-step “splitting” technique and improve the FDCO scheme to a multi-authority data cloud-outsourcing system, supporting credentials from different organizations and enables cross-domain data sharing. Besides, lots of works have already considered multi-authority ABE schemes [6, 17], and multi-authority cloud storage environment [8, 13, 28].

## 2 Preliminaries

### 2.1 Notations

For  $a, b \in \mathbb{N}$  with  $a < b$ , we define  $[a, b] = \{a, a + 1, \dots, b\}$ . We write  $[a]$  as shorthand for  $[1, a]$ . By  $\{X_i\}_{i \in [n]}$ , we denote a sequence of elements  $X_1, X_2, \dots, X_n$ .

When  $\mathcal{S}$  is a set, the cardinality of  $\mathcal{S}$  is denoted by  $|\mathcal{S}|$ . By  $s_1, s_2, \dots, s_n \stackrel{R}{\leftarrow} \mathcal{S}$  with  $n \in \mathbb{N}$ , we denote that elements  $s_1, \dots, s_n$  are picked uniformly at random from  $\mathcal{S}$ . We define  $\mathbf{M} \in \mathbb{Z}_p^{m \times n}$  as a matrix of size  $m \times n$  with elements in  $\mathbb{Z}_p$ . Two special subsets of the matrix are the row vector  $\mathbb{Z}_p^{1 \times n}$  and the column vector  $\mathbb{Z}_p^{m \times 1}$ . We denote the  $i$ -th row of matrix  $\mathbf{M}$  by  $\mathbf{M}_i$  and the  $i$ -th entry in vector  $\mathbf{v}$  by  $v_i$ . For two vectors  $\mathbf{v}$  and  $\mathbf{w}$ , the inner product of them is denoted by  $\langle \mathbf{v}, \mathbf{w} \rangle$ . The operation  $(\cdot)^T$  denotes the transposed vector/matrix.

Denote a binary tree by  $BT$ . For a leaf node  $\eta$  in  $BT$ , we denote the set of nodes on the path from node  $\eta$  to the root node by  $\text{Path}(\eta)$  ( $\eta$  and the root node are also included in  $\text{Path}(\eta)$ ).

### 2.2 Linear Secret Sharing Scheme

**Definition 1 (LSSS [3, 20]).** Let  $p$  be a prime and  $\mathcal{U}$  be an attribute universe. A secret sharing scheme  $\Pi$  with domain of secrets  $\mathbb{Z}_p$  for realizing access policy on  $\mathcal{U}$  is linear if: (1) The shares of a secret  $z \in \mathbb{Z}_p$  for each attribute form a vector over  $\mathbb{Z}_p$ . (2) For each access policy  $\mathbb{A}$  on  $\mathcal{U}$ , there exists a share-generating matrix  $\mathbf{M} \in \mathbb{Z}_p^{\ell \times n}$  for  $\Pi$ . For each  $x \in [\ell]$ , we define function  $\delta(x)$  that labels  $\mathbf{M}_x$  with an attribute from  $\mathcal{U}$ , i.e.,  $\delta : [\ell] \rightarrow \mathcal{U}$ .

Considering the column vector  $\mathbf{v} = (z, r_2, r_3, \dots, r_n)^T$ ,  $\boldsymbol{\lambda} = \mathbf{M}\mathbf{v} \in \mathbb{Z}_p^{\ell \times 1}$  is the vector of  $\ell$  shares of the secret  $z$ , where  $r_2, \dots, r_n \stackrel{R}{\leftarrow} \mathbb{Z}_p$ . The  $x$ -th entry  $\lambda_x$  belongs to the attribute  $\delta(x)$  for  $x \in [\ell]$ .

We refer to the tuple  $(\mathbf{M}, \delta)$  as an access policy  $\mathbb{A}$  encoded by LSSS-policy. Let  $\mathcal{S}$  and  $\mathcal{S}'$  respectively denote an authorized set and an unauthorized set for

the LSSS policy  $(\mathbf{M}, \delta)$ , where  $\mathbf{M} \in \mathbb{Z}_p^{\ell \times n}$  and  $\delta : [\ell] \rightarrow \mathcal{U}$ . We define  $I \subseteq [\ell]$  as  $I = \{x | \delta(x) \in \mathcal{S}\}$ , and  $I' \subseteq [\ell]$  as  $I' = \{x | \delta(x) \in \mathcal{S}'\}$ .

Then we have the following two properties: (1) *Reconstruction property*: Any authorized set can reconstruct the shared secret efficiently. For any authorized set  $\mathcal{S}$ , there exists constants  $\{\omega_x \in \mathbb{Z}_p\}_{x \in I}$  such that for any valid shares  $\{\lambda_x = (\mathbf{M}\mathbf{v})_x\}_{x \in I}$  of a secret  $z$ , we have  $\sum_{x \in I} \omega_x \lambda_x = z$ . The constants  $\{\omega_x\}_{x \in I}$  can be generated in time polynomial in the size of  $\mathbf{M}$ ; (2) *Security property*: No unauthorized set can obtain any partial information about the shared secret. For any unauthorized set  $\mathcal{S}'$ , such constants  $\{\omega_x\}_{x \in I}$  do not exist. In this case, there exists a vector  $\mathbf{d} = (d_1, \dots, d_n) \in \mathbb{Z}_p^{1 \times n}$ , such that  $\langle \mathbf{M}_x, \mathbf{d} \rangle = 0$  for all  $x \in I'$  and  $d_1$  can be any non-zero element in  $\mathbb{Z}_p$ .

### 2.3 Access Policy

**Definition 2 (Access Policy [3]).** Let  $\mathcal{U}$  be a set of parties. An access policy  $\mathbb{A}$  on  $\mathcal{U}$  is a collection of non-empty subsets of  $\mathcal{U}$ , i.e.,  $\mathbb{A} \subseteq 2^{\mathcal{U}} \setminus \{\emptyset\}$ . A set in  $\mathbb{A}$  is called an authorized set, and a set not in  $\mathbb{A}$  is called an unauthorized set.

In our system, each stored data item is associated with an access policy. Only parties whose roles match the specified access policy are able to decapsulate. The roles of the parties are denoted by a set of attributes in the attribute universe  $\mathcal{U}$ . The access policy  $\mathbb{A}$  contains the authorized sets of attributes.

### 2.4 Chameleon Hash Functions

A chameleon hash function has a chameleon hash key/trapdoor pair  $(chk, td)$ . Using the chameleon hash key  $chk$ , anyone can efficiently compute the hash value of any given input, and it is hard to find a collision for any given input. However, with the trapdoor  $td$ , there exists an efficient algorithm which can find collisions for every given input [15]. Formally, it consists of three efficient algorithms:

- $(chk, td) \leftarrow \text{CHGen}(1^\lambda)$ . The key generation algorithm takes a security parameter  $\lambda \in \mathbb{N}$  as input, and outputs an hash key/trapdoor pair  $(chk, td)$ .
- $H_m \leftarrow \text{CHash}(chk, m, r_m)$ . The hash algorithm takes as inputs a chameleon hash key  $chk$ , a message  $m$ , and an auxiliary random parameter  $r_m$ . It outputs a hash value  $H_m$  of the given message  $m$ .
- $r_{m'} \leftarrow \text{Coll}(td, m, r_m, m')$ . The chameleon collision finding algorithm takes as inputs a trapdoor  $td$ , a message  $m$  and its auxiliary parameter  $r_m$  when previously computing its hash value  $H_m$ , and another message  $m' \neq m$  used for forging. It outputs another auxiliary parameter  $r_{m'}$  such that  $H_m = \text{CHash}(chk, m, r_m) = \text{CHash}(chk, m', r_{m'})$ .

### 2.5 Revocation Mechanism

The revocation mechanism is borrowed from [32]. There are four components: a binary tree  $BT$ , a revocation list  $RL$ , a time  $T$ , and an algorithm  $\text{CUNode}$ .

Each attribute set is assigned to a leaf node in  $BT$ . The revocation list  $RL$  stores all pairs of nodes assigned to revoked access credentials and their revocation time  $(\eta_i, T_i)$ . When an access credential is required to be revoked at time  $T$ , the system adds it to  $RL$ , runs algorithm  $CUNode$ , and refreshes credential updates.

The algorithm  $CUNode$  takes as inputs  $BT$ ,  $RL$ ,  $T$ , and outputs a minimal set of nodes whose credential updates need to be published. None of the nodes in  $RL$  with time  $t \leq T$  have any ancestor or themselves in this set, and all other leaf nodes have exactly one ancestor or themselves in this set.

## 2.6 Multi-authority Mapping

In the multi-authority ABE setting, each attribute is controlled by a specific authority  $i \in \mathcal{U}_\Theta$ , where  $\mathcal{U}_\Theta$  is the universe of all authorities. We define a public function  $T : \mathcal{U} \rightarrow \mathcal{U}_\Theta$  that maps each attribute to a unique authority. Recall that the function  $\delta : [\ell] \rightarrow \mathcal{U}$ , which maps each row of share-generating matrix  $\mathbf{M}$  to a specific attribute. We define a compound function  $\rho(\cdot) = T(\delta(\cdot))$ , which maps each row of  $\mathbf{M}$  to a specific authority, i.e.,  $\rho : [\ell] \rightarrow \mathcal{U}_\Theta$ .

For example, in the implementation of the RW-15 multi-authority CP-ABE scheme, both the attribute ID and the authority ID consist of alphanumeric strings. The full attributes are expressed in the form of “[attribute-id]@[authority-id]”, and the mapping  $T$  extracts the part after “@” of the full attribute string.

## 2.7 “Zero-Out” Technique

Let  $\mathbf{M} \in \mathbb{Z}_p^{\ell \times n}$  be the share-generating matrix of a linear secret sharing scheme for an access policy  $\mathbb{A}$ , and let  $\mathcal{C} \subseteq [\ell]$  be a unauthorized set of rows. Let  $c \in \mathbb{N}$  be the dimension of the space spanned by rows in  $\mathcal{C}$ .

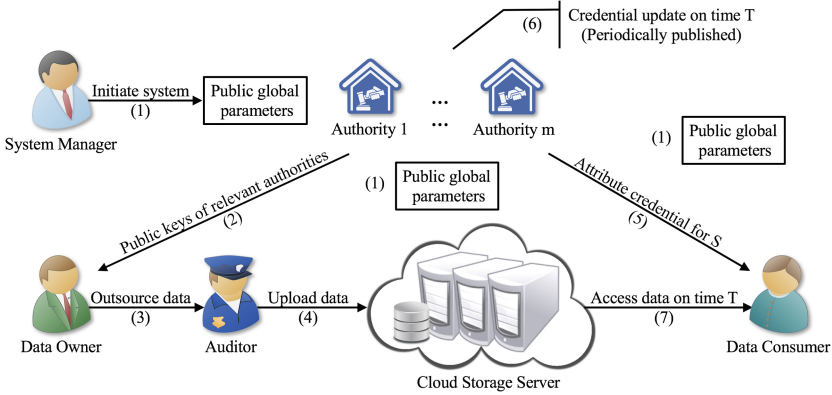
Then the distribution of the shares  $\{\lambda_x\}_{x \in [\ell]}$  sharing the secret  $z \in \mathbb{Z}_p$  generated with the matrix  $\mathbf{M}$  is the same as the distribution of the shares  $\{\lambda'_x\}_{\forall x \in [\ell]}$  sharing the same secret  $z$  generated with some matrix  $\hat{\mathbf{M}}$ , where  $\hat{M}_{x,j} = 0$  for all  $(x, j) \in \mathcal{C} \times [n - c]$ . We omit the detailed description and proof of the “Zero-Out” technique, and refer readers to RW-15 [20].

# 3 System Architecture and Security Model

## 3.1 System Architecture

The system architecture is shown in Fig. 1. Following the generic cloud storage system model [25, 30], our MFDCO scheme involves six types of parties: system manager, authority, data owner, auditor, cloud storage server and data consumer.

- *A system manager* is responsible for initiating the system.
- *Authorities* are responsible for issuing attribute credentials to recognized data consumers, and managing the revocation state by publishing credential updating of their responsible attributes.



**Fig. 1.** System architecture.

- *Data owners* encapsulate data with on-demand access policies, and upload to the cloud storage server to share with a specific set of data consumers.
- *An auditor* checks whether the data are correctly encapsulated according to the specified access policies before storing them in cloud.
- *A cloud storage server* maintains well-encapsulated data and responds to data retrieval requests.
- *Data consumers* get their attribute credentials and credential updating from relevant authorities, retrieve encapsulation from the cloud storage server, and decapsulate data that matching their attributes.

A MFDCO scheme consists of eight probabilistic polynomial-time algorithms.

- $\text{GlobalSetup}(1^\lambda, N) \rightarrow GP$ : Run by a system manager to initiate the system. This algorithm takes as inputs a security parameter  $\lambda$  and a maximal number of allowed attribute sets  $N$ , and outputs a public global parameter  $GP$ , including descriptions of an attribute universe  $\mathcal{U}$ , an authority universe  $\mathcal{U}_\Theta$ , a global identifier universe  $\mathcal{GID}$ , and a mapping  $\mathbb{T}$ .
- $\text{AuthoritySetup}(GP, i) \rightarrow \{PK_i, SK_i\}$ : Run by each authority  $i \in \mathcal{U}_\Theta$  to generate its own public/secret key pair. This algorithm takes as inputs a public global parameter  $GP$ , and outputs a public/secret key pair of each authority  $i$  as  $(PK_i, SK_i)$ .
- $\text{DataEnc}(GP, \{PK_i\}, \mathbb{A}, T, M) \rightarrow CT$ : Run by data owners to encapsulate data. This algorithm takes as inputs a public global parameter  $GP$ , a set of public key  $\{PK_i\}$ , an on-demand access policy  $\mathbb{A}$ , an encapsulation time  $T$  and data  $M$ , and outputs a corresponding encapsulation  $CT$ .
- $\text{EncFilt}(GP, CT) \rightarrow v$ : Run by auditors to verify and filter encapsulated data. This algorithm takes as inputs a public global parameter  $GP$  and an encapsulation  $CT$ , and outputs a bit  $v$ .  $v$  is set to 1 if  $CT$  is a valid encapsulation, and 0 otherwise.

- $\text{ACGen}(GP, GID, \mathcal{S}, \{SK_i\}) \rightarrow ac_{\mathcal{S}}$ : *Run by authorities to generate attribute credentials.* This algorithm takes as inputs a public global parameter  $GP$ , a global identifier  $GID$  of a user, a specified attribute set  $\mathcal{S} = \{A_1, A_2, \dots, A_{\kappa}\}$ , an authority's secret key  $SK_i$ . Each authority outputs its responsible set of attribute credential component  $ac_x$ , and finally we collect all  $ac_x$  to form the whole attribute credential  $ac_{\mathcal{S}}$ .
- $\text{CredUp}(GP, GID, \mathcal{S}, T, \{SK_i\}) \rightarrow cu_T$ : *Run by authorities for periodically credential updating to realize credential revocation.* The  $\text{CredUp}$  algorithm takes as inputs a public global parameter  $GP$ , a global identifier  $GID$  of a user, a specified attribute set  $\mathcal{S} = \{A_1, A_2, \dots, A_{\kappa}\}$ , a credential update time  $T$  and an authority's secret key  $SK_i$ . Each authority outputs its responsible set of credential updating component  $cu_{T,x}$ , and finally we collect all  $cu_{T,x}$  to form the whole credential updating  $cu_T$ .
- $\text{DataDec}(GP, ac_{\mathcal{S}}, cu_T, CT) \rightarrow M$ : *Run by data consumers to decapsulate data.* This algorithm takes as inputs a public global parameter  $GP$ , an attribute credential  $ac_{\mathcal{S}}$ , a credential update  $cu_T$  and an encapsulation  $CT$ , and outputs decapsulated data  $M$ .
- $\text{ACRevoke}(GP, \mathcal{S}, T) \rightarrow RL$ : *Run by authorities for user revocations.* The  $\text{ACRevoke}$  algorithm takes as inputs a public global parameter  $GP$ , a revoked attribute set  $\mathcal{S}$  and a revoked time  $T$ , and outputs a revocation list  $RL$ .

An MFDCO scheme must satisfy the following correctness property.

**Definition 3 (Correctness).** *An MFDCO scheme is correct if for any  $GP$  generated by the  $\text{GlobalSetup}$  algorithm, any set of key pairs  $\{PK_i, SK_i\}$  generated by the  $\text{AuthoritySetup}$  algorithm, any  $CT$  generated by the  $\text{DataEnc}$  algorithm using relevant authorities' public keys on any message  $M$  and access structure  $\mathbb{A}$ , any  $ac_{\mathcal{S}}$  generated by the  $\text{ACGen}$  algorithm and any  $cu_T$  generated by the  $\text{CredUp}$  algorithm using relevant authorities' secret keys for a user  $GID$  on any  $\mathbb{A}$ -authorized set of attributes, it holds  $\text{DataDec}(GP, ac_{\mathcal{S}}, cu_T, CT) = M$ .*

### 3.2 Security Model

We assume that after public global parameters being published, each non-corrupted authority honestly sets up itself and securely issues attribute credentials to legal data consumers. Non-corrupted authorities never reveal any private information to non-entitled parties. But an attacker can control a set of corrupted authorities. All other parties, including data owners, data consumers, auditors and the cloud storage server, are honest-but-curious. They correctly execute the required procedures, but may collude to get access to unauthorized information. In addition, some authorized data consumers may obviously leak their decapsulated data to attackers, but their attribute credentials are assumed to be kept securely.

### 3.3 Formal Security Definition

We formally define the security game between a challenger  $\mathcal{B}$  and an adversary  $\mathcal{A}$  following the general security model in Sect. 3.2.



**Goal of the Adversary:** To extract useful information from the challenge encapsulation of his/her choice, which is associated with the target access policy and target time that he/she previous committed to in the initiation stage.

**Ability of the Adversary:** The adversary can (1) choose a set of corrupted authorities that he can control before seeing the public global parameters; (2) collude with other legal unauthorized data consumers, and revoked authorized data consumers; (3) control the revocation state; and (4) adaptively know the recovered data, except the challenge one.

**Requirement:** The adversary cannot obtain any useful information about the challenge encapsulation.

Both adversary  $\mathcal{A}$  and challenger  $\mathcal{B}$  are given a security parameter  $\lambda$  as input. The security game is described as follows:

**Initialization.**  $\mathcal{A}$  commits to a challenge access policy  $(\mathbf{M}^*, \delta^*)$  and a challenge time  $T^*$ . It also needs to output a set of corrupted authorities  $\mathcal{C}_\Theta$ , a set of non-corrupted authorities  $\mathcal{N}_\Theta$ , a collection of all queried *GIDs* and their attributes sets  $\mathcal{Q} = \{(GID_i, S_i)\}$ .

**Setup.**  $\mathcal{B}$  generates *GP*, gives *GP* to  $\mathcal{A}$  and generates public keys of all non-corrupted authorities.

**Phase 1.**  $\mathcal{A}$  issues queries, and each query is one of: (1) *Attribute credential query* for attribute set  $\mathcal{S}$ .  $\mathcal{B}$  generates  $ac_{\mathcal{S}}$  for  $\mathcal{S}$  and gives it to  $\mathcal{A}$ ; (2) *Decapsulation query* for  $(hdr, en)$  with  $(\mathbf{M}, \delta)$  chosen by  $\mathcal{A}$ ;  $\mathcal{B}$  decapsulates the data and returns the result to  $\mathcal{A}$ . (3) *Revocation query* for attribute set  $\mathcal{S}$  and time  $T$ .  $\mathcal{B}$  runs *ACRevoke* to update revocation list *RL*; (4) *Credential update query* on  $T$ .  $\mathcal{B}$  generates and publishes credential updating according to *RL*.

**Challenge.** When  $\mathcal{A}$  decides that **Phase 1** is over, it outputs two equal-length data  $data_0$  and  $data_1$ .  $\mathcal{B}$  flips a coin  $b \xleftarrow{R} \{0, 1\}$ , encapsulates  $data_b$  with  $(\mathbf{M}^*, \delta^*)$  and  $T^*$ , and returns  $(hdr^*, en^*)$  to  $\mathcal{A}$ .

**Phase 2.**  $\mathcal{A}$  further issues queries, with an additional constraint that in the decapsulation query, the issued data satisfies that  $(hdr, en) \neq (hdr^*, en^*)$ .  $\mathcal{B}$  responds the same as in **Phase 1**.

**Guess.** Finally,  $\mathcal{A}$  outputs a guess  $b' \in \{0, 1\}$  and wins the game if  $b = b'$ .

Considering the running sequence of algorithms shown in Fig. 1, the above queries must satisfy the following conditions: (1) Revocation queries and credential update queries should be issued after all other queries, i.e., the time of revocation query and credential update query should be later than the time of all previous queries; (2) The revocation query on time  $T$  should be done before the credential update query on time  $T$ ; (3) If an attribute credential for  $\mathcal{S}$  satisfying  $(\mathbf{M}^*, \delta^*)$  is queried, then its access credential must be revoked for any  $t \leq T^*$ .

First, revocation query and credential update query will change the revocation state, thus determining the final access control state, i.e., whether a certain data consumer can decapsulate a certain encapsulation. So this operation need

to be executed at the end of all queries. Second, because revocation query will change the revocation list  $RL$ , which is an essential part in credential update procedure, so system updates the credentials at the end of the time period  $T$ , and all revoked procedures on that time must be done before credential updating. Third, if an attribute credential satisfying the target access policy  $(\mathbf{M}^*, \delta^*)$  is queried, but not revoked before the target time  $T^*$ , then adversary  $\mathcal{A}$  can use  $ac_S$  and  $cu_S$  to decapsulate the challenge encapsulation, in which situation system will be trivially broken.

The advantage of  $\mathcal{A}$  in attacking our system with security parameter  $\lambda$  is

$$Adv_{\mathcal{A}}(\lambda) = |\Pr[b' = b] - 1/2|$$

**Definition 4 (Security).** *An MFDCO scheme is secure against selectively chosen access policy & time and chosen ciphertext attack if for any probabilistic polynomial time adversary  $\mathcal{A}$ , the advantage of winning the security game defined above is negligible in  $\lambda$ .*

### 3.4 The $q$ -wDPBDHE2 Computational Assumption

The security of our MFDCO scheme cannot be directly reduced to the RW-13 CP-ABE or the RW-15 multi-authority scheme, since we additionally need to handle the decapsulation queries, the revocation queries and the credential update queries. We use a modified version of the  $q$ -Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption [27]. We refer to our assumption as  $q$ -wDPBDHE2 assumption for short. The difference between the  $q$ -wDPBDHE assumption of our previous FDCO scheme [32] and this  $q$ -wDPBDHE2 assumption is that, in the latter assumption, the  $\{g^{a^i/b_j}\}$  terms go up to  $i = 2q$  instead of  $q$ .

Let  $\mathcal{G}(1^\lambda)$  be an efficient group generator algorithm. Run  $(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$ , and sample  $g \xleftarrow{R} \mathbb{G}$ ,  $a, s, b_1, b_2, \dots, b_q \xleftarrow{R} \mathbb{Z}_p$ . The  $q$ -wDPBDHE2 problem is that given  $(D, G)$  as inputs, and then to determine whether  $G$  equals  $e(g, g)^{sa^{q+1}}$ , or is a random element in  $\mathbb{G}_T$ , where

$$D = \left( \begin{array}{l} g, g^s \\ g^{a^i}, g^{b_j}, g^{sb_j}, g^{a^i b_j}, \forall i \in [q], j \in [q] \\ g^{a^i \cdot b_j / b_{j'}^2}, \quad \forall i \in [2q], j \in [q], j' \in [q], i \neq q+1, j \neq j' \\ g^{a^i / b_j}, g^{a^i / b_j^2}, \quad \forall i \in [2q], j \in [q], i \neq q+1 \\ g^{sa^i b_j / b_{j'}}, g^{sa^i b_j / b_{j'}^2}, \forall i \in [q], j \in [q], j' \in [q], j \neq j' \end{array} \right).$$

The  $q$ -DPBDHE assumption was shown generically secure in [27], and following a similar proof, we can prove that the  $q$ -wDPBDHE2 assumption is also generically secure.

**Theorem 1.** For an algorithm  $\mathcal{A}$  that outputs  $b \in \{0, 1\}$ , its advantage in solving the  $q$ -wDPBDHE2 problem with the security parameter  $\lambda$  is defined as

$$Adv_{\mathcal{A}}(\lambda) = \left| \Pr \left[ \mathcal{A}(D, G = e(g, g)^{sa^{q+1}}) = 1 \right] - \Pr \left[ \mathcal{A}(D, G \stackrel{R}{\leftarrow} \mathbb{G}_T) = 1 \right] \right|,$$

where the probability is over random choices of an element  $g \in \mathbb{G}$ , exponents  $a, s, b_1, b_2, \dots, b_q \in \mathbb{Z}_p$ , and the random bits used by  $\mathcal{A}$ .

The  $q$ -wDPBDHE2 assumption holds if no polynomial time algorithm has non-negligible advantage in solving the  $q$ -wDPBDHE2 problem, i.e.,  $Adv_{\mathcal{A}}(\lambda) \leq \epsilon$ .

## 4 The Proposed MFDCO Scheme

### 4.1 Construction

$\text{GlobalSetup}(1^\lambda, N) \rightarrow GP$ : The setup algorithm takes as inputs a security parameter  $\lambda \in \mathbb{N}$  and the maximal number of allowed attribute sets  $N \in \mathbb{N}$ , which implies  $N$  different global identities ( $GIDs$ ) of users. Denote the attribute universe by  $\mathcal{U} = [0, \frac{p-1}{2}]$ , and the verification universe by  $\mathcal{V} = [\frac{p+1}{2}, p-1]$ . Denote the authority universe by  $\mathcal{U}_\Theta$ . Define a public function  $\mathsf{T} : \mathcal{U} \rightarrow \mathcal{U}_\Theta$ . The compound mapping  $\mathsf{T}$  is described in Sect. 2.6. Let  $\varepsilon_{\text{sym}} = (\text{SymEnc}, \text{SymDec})$  be a secure symmetric encryption scheme. Let  $\text{Hash} : \{0, 1\}^* \rightarrow [\frac{p+1}{2}, p-1]$  be a standard collision resistant hash function. The chosen chameleon hash function is  $\text{CHash} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ , with an auxiliary parameter universe  $\mathcal{R}$ . We define a hash function  $\text{H} : \mathcal{GID} \rightarrow \mathbb{G}$  to hash the identity of each user to an element in  $\mathbb{G}$ , and a function  $\text{F} : \mathbb{Z}_p \rightarrow \mathbb{G}$  to map an arbitrary string into an element in  $\mathbb{G}$ .

1. Obtain  $(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^\lambda)$ , where  $\mathbb{G}, \mathbb{G}_T$  are of prime order  $p$ , bilinear map  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ .
2. Sample  $g, h, u, v, h_r, u_r \stackrel{R}{\leftarrow} \mathbb{G}$ ,  $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p$  and compute  $g^\alpha$ .
3. Set  $RL = \emptyset$  and sample a binary tree  $BT$  with at least  $N$  leaf nodes.
4. Output the public global parameters

$$GP = \left( \begin{array}{c} \lambda, p, \mathbb{G}, \mathbb{G}_T, g^\alpha, \text{CHash}, \text{Hash}, \text{H}, \text{F}, \mathsf{T} \\ \mathcal{R}, \mathcal{U}, \mathcal{U}_\Theta, \mathcal{GID}, g, h, u, v, h_r, u_r, RL, BT \end{array} \right).$$

$\text{AuthoritySetup}(GP, i) \rightarrow \{PK_i, SK_i\}$ : Each authority  $i$  chooses two random exponents  $\alpha_i, \beta_i \stackrel{R}{\leftarrow} \mathbb{Z}_p$ , and publishes  $PK_i = \{e(g, g)^{\alpha_i}, g^{\beta_i}\}$  as its public key.  $SK_i = \{\alpha_i, \beta_i\}$  is set as secret key.

$\text{DataEnc}(GP, \{PK_i\}, (\mathbf{M}, \delta), T, data) \rightarrow (hdr, en)$ .

*A. Preparation:* Proceed without the knowledge of  $(\mathbf{M}, \delta), T$  or  $data$ .  $P$  is the given maximum number of rows in policy  $(\mathbf{M}, \delta)$ .

1. Sample  $z \xleftarrow{R} \mathbb{Z}_p$  and compute  $key = e(g, g)^{\alpha z}$ .
2. For each  $x \in [P]$ , sample  $\lambda'_x, \mu'_x, \alpha'_{\rho(x)}, \beta'_{\rho(x)}, T', \gamma_x, t_x \xleftarrow{R} \mathbb{Z}_p$ , and compute  $C_{R,1} = u_r^{T'} h_r$ ,  $C_{0,1,x} = e(g, g^\alpha)^{\lambda'_x} e(g, g)^{\alpha'_{\rho(x)} t_x} e(g, C_{R,1})^{t_x}$ ,  $C_{0,2,x} = g^{\alpha \mu'_x}$ ,  $C_{0,3,x} = F(\delta(x))^{t_x} v^{t_x}$ ,  $C_{0,4,x} = g^{\beta'_{\rho(x)} t_x}$ ,  $C_{x,1} = (u^{\gamma_x} h)^{-t_x}$ ,  $C_{x,2} = g^{t_x}$ .
3. Sample  $t_0 \xleftarrow{R} \mathbb{Z}_p$  and compute  $C_{v,1} = g^{t_0}$ .
4. Obtain  $(chk, td) \leftarrow \text{CHGen}(1^\lambda)$ , where  $(chk, td)$  denotes a pair of chameleon hash key and the corresponding trapdoor.
5. Sample  $r_{m'} \xleftarrow{R} \mathcal{R}$ ,  $m' \xleftarrow{R} \{0, 1\}^*$  and compute  $V = \text{Hash}(chk \parallel \text{CHash}(chk, m', r_{m'}))$ ,  $C_{v,2} = (u^V h)^{-t_0}$ .
6. Store the intermediate header as

$$ihdr = \left( \begin{array}{c} key, z, T', chk, td, r_{m'}, m', \{\lambda'_x, \mu'_x, \alpha'_{\rho(x)}, \beta'_{\rho(x)}, \gamma_x, t_x\}_{x \in [P]}, C_{R,1}, \\ C_{v,1}, C_{v,2}, \{C_{0,1,x}, C_{0,2,x}, C_{0,3,x}, C_{0,4,x}, C_{x,1}, C_{x,2}\}_{x \in [P]} \end{array} \right).$$

*B. Real-time encapsulation:* Take  $(\mathbf{M}, \delta)$ ,  $T$  and *data* as inputs, where the share-generating matrix  $\mathbf{M} \in \mathbb{Z}_p^{\ell \times n}$ ,  $\delta : [\ell] \rightarrow \mathcal{U}$ ,  $\ell \leq P$ . Considering the  $x$ -th row of matrix  $\mathbf{M}$ , its corresponding attribute is denoted by  $\delta(x)$ , which is managed by authority  $\rho(x)$ . Parse the public key of authority  $\rho(x)$  as  $PK_{\rho(x)} = \{e(g, g)^{\alpha_{\rho(x)}}, g^{\beta_{\rho(x)}}\}$ . Process items in *ihdr* and fulfill the encapsulation:

1. Sample  $y_2, \dots, y_n, w_2, \dots, w_n \xleftarrow{R} \mathbb{Z}_p$  and parse  $\mathbf{y} = (z, y_2, \dots, y_n)^T$ ,  $\mathbf{w} = (0, w_2, \dots, w_n)^T$ .
2. Parse  $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \dots, \lambda_\ell)^T = \mathbf{M}\mathbf{y}$  and  $\boldsymbol{\mu} = (\mu_1, \mu_2, \dots, \mu_\ell)^T = \mathbf{M}\mathbf{w}$ .
3. For each  $x \in [\ell]$ , compute  $C_{x,3} = \lambda_x - \lambda'_x$ ,  $C_{x,4} = -t_x(\delta(x) - \gamma_x)$ ,  $C_{x,5} = \mu_x - \mu'_x$ ,  $C_{x,6} = t_x(\beta_{\rho(x)} - \beta'_{\rho(x)})$ ,  $C_{x,7} = t_x(\alpha_{\rho(x)} - \alpha'_{\rho(x)})$ .
4. Compute  $C_{R,2} = T - T'$ ,  $en = \text{SymEnc}(key, data)$  and set  $m = en \parallel C_{v,1} \parallel C_{v,2} \parallel C_{1,1} \parallel \dots \parallel C_{1,7} \parallel \dots \parallel C_{\ell,1} \parallel \dots \parallel C_{\ell,7} \parallel C_{R,1} \parallel C_{R,2} \parallel (\mathbf{M}, \delta) \parallel T$ .
5. Utilize the collision finding algorithm to compute  $r_m = \text{Coll}(td, m', r_{m'}, m)$ .
6. Parse the stored data as  $(hdr, en)$ , where the header *hdr* is

$$hdr = \left( \begin{array}{c} (\mathbf{M}, \delta), T, chk, r_m, C_{v,1}, C_{v,2}, C_{R,1}, C_{R,2}, \\ \{C_{x,1}, C_{x,2}, \dots, C_{x,7}, C_{0,1,x}, \dots, C_{0,4,x}\}_{x \in [\ell]} \end{array} \right).$$

$\text{EncFilt}(GP, (hdr, en)) \rightarrow v$ : Detect and filter ill-formed encapsulation  $(hdr, en)$ :

1. Compute  $V = \text{Hash}(chk \parallel \text{CHash}(chk, m, r_m))$ .
2. For each  $x \in [\ell]$ , verify attributes  $e(g^{-1}, C_{x,1} u^{C_{x,4}}) \stackrel{?}{=} e(C_{x,2}, u^{\delta(x)} h)$ .
3. Verify encapsulation time  $C_{R,1} u_r^{C_{R,2}} \stackrel{?}{=} u_r^T h_r$ .
4. Verify verification term  $e(g^{-1}, C_{v,2}) \stackrel{?}{=} e(C_{v,1}, u^V h)$ .
5. If any equation does not hold, output  $v = 0$ ; otherwise output  $v = 1$ .

$\text{ACGen}(GP, GID, \mathcal{S}, \{SK_i\}) \rightarrow ac_S$ : To implement the revocation mechanism [4], we use a binary tree data structure and a credential updating node algorithm

**CUNode.** The algorithm **CUNode** outputs the minimal set of nodes needed to be updated, and its description is provided in the full version paper.

Take  $\mathcal{S} = \{A_1, A_2, \dots, A_\kappa\}$  as input. Let  $\hat{z}$  be the index of attribute  $\delta(x)$  in set  $\mathcal{S}$ , we have  $\delta(x) = A_{\hat{z}}$ . Parse the secret key of authority  $\mathsf{T}(A_{\hat{z}})$  as  $SK_{\rho(x)} = \{\alpha_{\rho(x)}, \beta_{\rho(x)}\}$ . Each authority does its own part of the following operations.

1. Sample an unassigned leaf node  $\eta$  from  $BT$  and store  $\mathcal{S}$  in node  $\eta$ .
2. For each node  $\theta \in \text{Path}(\eta)$ : If  $g_\theta$  is already stored in node  $\theta$ , retrieve it. Otherwise, sample  $g_\theta \xleftarrow{R} \mathbb{G}$  and store  $(g_\theta, \tilde{g}_\theta = g/g_\theta)$  in node  $\theta$ .
3. Sample  $r_\theta, r_1, r_2, \dots, r_\kappa \xleftarrow{R} \mathbb{Z}_p$ , and compute  $K_{\theta,1} = g^{r_\theta}$ , for each  $x \in [\kappa]$ ,  $K_{\theta,x,1} = g_\theta^{\alpha_{\rho(x)}} \mathsf{H}(GID)^{\beta_{\rho(x)}} \mathsf{F}(\delta(x))^{r_\theta}$ ,  $K_{\theta,x,2} = g^{r_x}$ ,  $K_{\theta,x,3} = (u^{\delta(x)}h)^{r_x} v^{-r_\theta}$ .
4. Output an attribute credential as

$$ac_{\mathcal{S}} = (\mathcal{S}, \{(K_{\theta,1}, \{K_{\theta,x,1}, K_{\theta,x,2}, K_{\theta,x,3}\}_{x \in [\kappa]})\}_{\theta \in \text{Path}(\eta)}).$$

**CredUp**( $GP, GID, \mathcal{S}, T, \{SK_i\}$ )  $\rightarrow cu_T$ : Take  $\mathcal{S} = \{A_1, A_2, \dots, A_\kappa\}$  as input, where  $\kappa = |\mathcal{S}|$ . The corresponding authority of attribute  $A_{\hat{z}}$  is  $\mathsf{T}(A_{\hat{z}})$ . Let  $\hat{z}$  be the row index of the attribute  $\delta(x)$  in set  $\mathcal{S}$ , we have  $\delta(x) = A_{\hat{z}}$ . Parse the secret key of authority  $\mathsf{T}(A_{\hat{z}})$  as  $SK_{\rho(x)} = \{\alpha_{\rho(x)}, \beta_{\rho(x)}\}$ . Each authority does its own part of the following operations.

1. Invoke the **CUNode** algorithm. For each node  $\theta \in \text{CUNode}(BT, RL, T)$ , retrieve  $\tilde{g}_\theta$  from node  $\theta$ .
2. For each  $x \in [\kappa]$ , compute  $\tilde{K}_{\theta,x,1} = \tilde{g}_\theta^{\alpha_{\rho(x)}} \mathsf{H}(GID)^{\beta_{\rho(x)}} (u_r^T h_r)$ .
3. Publish the credential updating as  $cu_T = \{(\theta, \{\tilde{K}_{\theta,x,1}\}_{x \in [\kappa]})\}_{\theta \in \text{CUNode}(BT, RL, T)}$

**DataDec**( $GP, ac_{\mathcal{S}}, cu_T, (hdr, en)$ )  $\rightarrow data$ : Take as inputs attribute credential  $ac_{\mathcal{S}}$  and credential updating  $cu_T$ :  $ac_{\mathcal{S}} = (\mathcal{S}, \{(K_{\theta,1}, \{K_{\theta,x,1}, K_{\theta,x,2}, K_{\theta,x,3}\}_{x \in [\kappa]})\}_{\theta \in I'})$ ,  $cu_T = \{(\theta, \{\tilde{K}_{\theta,x,1}\}_{x \in [l]})\}_{\theta \in J'}$ , where  $I' = \text{Path}(\eta)$  and  $J' = \text{CUNode}(BT, RL, T)$ . Check  $I' \cap J'$ : If  $I' \cap J' = \emptyset$ , no ancestor node of  $ac_{\mathcal{S}}$  is updated, which means the access credential for  $\mathcal{S}$  is revoked, so output  $\perp$ . Otherwise, choose  $\theta \in I' \cap J'$ :

1. Compute  $K_{x,0} = K_{\theta,x,1} \tilde{K}_{\theta,x,1} = g^{\alpha_{\rho(x)}} u_r^T h_r \mathsf{H}(GID)^{2\beta_{\rho(x)}} \mathsf{F}(\delta(x))^{r_\theta}$ .
2. Parse  $I = \{x | \delta(x) \in \mathcal{S}\}$  and compute  $\{\omega_x \in \mathbb{Z}_p\}_{x \in I}$  such that  $\sum_{x \in I} \omega_x \mathbf{M}_x = (1, 0, \dots, 0)$ , and we have  $\sum_{x \in I} \omega_x \lambda_x = z$  and  $\sum_{x \in I} \omega_x \mu_x = 0$ .
3. Compute

$$\begin{aligned} & \frac{C_{0,1,x} e(g, g^\alpha)^{C_{x,3}} e(g, g)^{C_{x,7}} e(\mathsf{H}(GID), C_{0,4,x} g^{C_{x,6}})^2 e(\mathsf{H}(GID), C_{0,2,x} g^{\alpha C_{x,5}})}{e(C_{0,3,x}, K_{\theta,1})^{-1} e(C_{x,1} u^{C_{x,4}}, K_{\theta,\hat{z},2})^{-1} e(C_{x,2}, K_{\theta,\hat{z},3})^{-1} e(C_{x,2}, u_r^{-C_{R,2}} K_{x,0})} \\ & = e(g, g^\alpha)^{\lambda_x} e(\mathsf{H}(GID), g^\alpha)^{\mu_x} \end{aligned}$$

and output an encapsulated key as  $key = \prod_{x \in I} (e(g, g^\alpha)^{\lambda_x} e(\mathsf{H}(GID), g^\alpha)^{\mu_x})^{\omega_x}$ ,

where  $\hat{z}$  is the index of the attribute  $\delta(x)$  in  $\mathcal{S}$ .

**ACRevoke**( $GP, \mathcal{S}, T$ )  $\rightarrow RL$ : If there is a leaf node  $\eta$  associated with  $\mathcal{S}$ , then update  $RL \leftarrow RL \cup \{(\eta, T)\}$  and publish it. Otherwise, output  $\perp$ .

## 4.2 Correctness

Next we show the stored data is correctly encapsulated by the data owner. For all  $x \in I$ , we have:  $e(g^{-1}, C_{x,1}u^{C_{x,4}}) = e(C_{x,3}, u^{\rho(x)}h)$  and  $C_{R,1}u_r^{C_{R,2}} = g, u_r^T h_r$ . Since  $V = \text{Hash}(chk \parallel \text{CHash}(chk, m, r_m))$ , we have  $e(g^{-1}, C_{v,2}) = e(C_{v,1}, u^V h)$ .

If an attribute set  $\mathcal{S}$  satisfies the access policy  $(\mathbf{M}, \delta)$ , then we have  $\sum_{x \in I} \omega_x \lambda_x = s$  and  $\sum_{x \in I} \omega_x \mu_x = 0$ . Since  $\hat{z}$  is the index of attribute  $\delta(x)$  in set  $\mathcal{S}$ , we have  $\delta(x) = A_{\hat{z}}$ . Therefore, we have  $key = \prod_{x \in I} (e(g, g^\alpha)^{\lambda_x} eH(GID), g^\alpha)^{\mu_x} \omega_x = e(g, g)^{\alpha z}$ .

Hence, a data consumer can correctly extract the encapsulation key  $key$  to recover the data.

## 4.3 Formal Security Result

We proceed our security reduction in the random oracle model under a modified  $q$ -wDPBDHE computational assumption defined in Sect. 3.4, which is called  $q$ -wDPBDHE2 assumption. More formally,

**Theorem 2.** *If the  $q$ -wDPBDHE2 assumption holds, then all probabilistic polynomial time adversaries, with maximal number of  $q_s$  attribute credential queries, maximal number of  $q_d$  decapsulation queries, have only a negligible advantage in breaking our MFDCO scheme in the random oracle model.*

We prove the security of our MFDCO scheme by constructing a sequence of games [23], in which *Game 0* is identical to the game defined in Sect. 3.2 while other games are similar to *Game 0* in their overall structure, and only differ from *Game 0* in terms of how the simulator works. All of these games are viewed as operating on the same underlying probability space. We show that if adversary  $\mathcal{A}$  can only solve the  $q$ -wDPBDHE2 problem with negligible advantage, then he can only break our MFDCO scheme in the security model defined in Sect. 3.3 with negligible advantage. The fact that no efficient algorithm can solve the  $q$ -wDPBDHE2 problem with non-negligible advantage in polynomial time assures the security of our MFDCO scheme. Next we show the high level idea of the proof. The details of the proof is provided in the full version paper.

In our MFDCO scheme, the security and privacy of the stored data is protected by the chosen symmetric encryption scheme  $\varepsilon_{\text{sym}}$ , e.g., AES [10], with a secret key  $key$ . The key  $key$  is encapsulated in  $hdr$  and can only be decapsulated by authorized data consumers. Even if an attacker colludes with all unauthorized users and the CSP, and adaptively gets all keys except the target key, he can get no useful information about the target data unless he can break the well-designed symmetric encryption scheme.

We adopt a two-step “splitting” technique to realize fine-grained access control over the protected  $key$  with multi-authority and revocation functionality. (1) “*Splitting-1*”: we split hidden parameters (i.e., part of secret key information) into two parts: During the generation of authorities’ public keys, the elements

in the first column of the secret sharing matrix are programmed into  $e(g, g)^{\alpha_i}$  component, while the remaining in  $g^{\beta_i}$ . During the generation of the challenge ciphertext, the secret sharing vector  $\lambda$  (corresponding to  $e(g, g)^{\alpha_i}$ ) will hold the secret  $z = sa^{q+1}$  on the first position, and the zero sharing vector  $\mu$  (corresponding to  $g^{\beta_i}$ ) will hold terms  $sa^q, sa^{q-1}, \dots, sa^2$  on all other positions except the first one. (2) “*Splitting-2*”: we present the attribute credential (corresponding to data consumers’ attribute set  $\mathcal{S}$ ) under the RW-13 key form and the credential updating (corresponding to revocations on time  $T$ ) under the BB-04 key form.

When generating data consumers’ attribute credentials or credential updating, all terms in “splitting-1” procedure are “recombined” to give a full series of  $q$  terms. Once an authorized data consumer gets his/her attribute credential and credential updating, he/she can recover the key and decapsulate data.

In order to isolate the influence of corrupted authorities and achieve security against active attacks, we reconstruct the challenge access policy by two steps in the security reduction. First, we utilize the “Zero-Out” technique [20] to substitute the secret sharing matrix  $\mathbf{M}^*$  with a new matrix  $\hat{\mathbf{M}}^*$ . It allows a simulator to isolate an unauthorized set of rows and ignore it in the remaining of the security proof. Then, utilizing verification term  $V$ , we introduce the notion of “Virtual Attribute”, and use  $V^*$  (the verification term of the challenge ciphertext) as an special virtual attribute to substitute matrix  $\hat{\mathbf{M}}^*$  with another new matrix  $\widetilde{\mathbf{M}}^*$ . The access policy is transformed into *the original policy OR*  $V^*$ . As  $V \notin [0, \frac{p-1}{2}]$ ,  $V$  is not a valid attribute in our MFDCO scheme, whereas  $V$  is a valid attribute in the RW-13 CP-ABE scheme. A simulator can correctly generate an attribute credential for the attribute set  $\mathcal{S} = \{V\}$ . After these two steps, from the view of  $\mathcal{A}$ , the security game remains indistinguishable with the origin one as if simulator used  $\mathbf{M}^*$ .

## 5 Efficiency Analysis

Table 1 presents the complexity of each procedure in our MFDCO scheme. We assume that an access credential  $ac_{\mathcal{S}}$  is associated with attribute set  $\mathcal{S}$  for  $\kappa = |\mathcal{S}|$ , and the access policy specified in encapsulation is  $(\mathbf{M}, \delta)$  with  $\mathbf{M} \in \mathbb{Z}_p^{\ell \times n}$ . For simplicity, each authority is responsible of one attribute. The maximal number of attributes involved in attribute sets and access policies are  $P$ . The number of attributes involved in the decapsulation procedure is denoted by  $|I| \leq P$ . For revocation procedure, we assume that  $R$  of  $N$  access credentials are revoked.

From Table 1, we conclude that our scheme is efficient, especially in real-time encapsulation, where no multiplication, exponent or pairing operation in  $\mathbb{G}$  and  $\mathbb{G}_T$  is involved. The proposal provides better user experience for mobile devices.

**Table 1.** Complexity of MFDCO.  $\tau_e$  denotes the time to compute one exponentiation in  $\mathbb{G}$  or  $\mathbb{G}_T$ .  $\tau_m$  denotes the time of one multiplication in  $\mathbb{G}$ .  $\tau_p$  denotes the time of one pairing operation.  $P$  denotes the maximum number of rows in  $\mathbf{M}$ .  $|I|$  denotes the number of attributes involved in **DataDec**.  $\ell$  denotes the number of attributes involved in access policy  $\mathbb{A}$ .  $\kappa$  denotes the total number of attributes allowed in a instantiation.  $R$  of  $N$  access credentials are revoked in **CredUp**.

Procedures		Time cost
<b>Setup</b>		$\tau_p + \tau_e$
<b>AuthoritySetup</b>		$\tau_p + 2\tau_e$
<b>DataEnc</b>	<i>Preparation</i>	$(9P + 5)\tau_e + (3P + 2)\tau_m + (P + 1)\tau_p$
	<i>Real-time</i>	none
<b>EncFilt</b>		$(2\ell + 3)\tau_e + (2\ell + 3)\tau_m + (2\ell + 2)\tau_p$
<b>ACGen</b>		$\log N \cdot [(7\kappa + 1)\tau_e + 4\kappa\tau_m]$
<b>CredUp</b>		$R \log(N/R)(3\kappa\tau_e + 3\kappa\tau_m)$ if $1 < R \leq N/2$ $(N - R)(3\kappa\tau_e + 3\kappa\tau_m)$ if $N/2 < R \leq N$
<b>DataDec</b>		$6 I \tau_e + (11 I  - 1)\tau_m + 7 I \tau_p$

## 6 Conclusion

We propose an efficient MFDCO scheme that offers a fast and secure cross-domain public cloud outsourcing approach, especially appealing to mobile devices, and achieve security against active attackers in the random oracle model. The MFDCO supports fine-grained access control, dynamic credential revocation, public encapsulation validity test and junk encapsulation filtering, and achieves decentralization and fast online encapsulation. As a future work, we will consider implementing our MFDCO scheme in embedded platform to evaluate its practicality, and further introduce the technique of certificateless encryption and timed-release encryption [9] to see if it brings about exciting practical result.

**Acknowledgment.** The work is partly supported by the National Key R&D Program of China (2017YFB1400700), Beijing Natural Science Foundation (4182033), the fund of the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences (No.2017-MS-02), and Beihang University Innovation & Practice Fund for Graduate (No.YCSJ-02-2018-03).

## References

1. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. *TISSEC* **9**(1), 1–30 (2006)
2. Bardou, R., et al.: Efficient padding oracle attacks on cryptographic hardware. In: Safavi-Naini, R., Canetti, R. (eds.) *CRYPTO 2012*. LNCS, vol. 7417, pp. 608–625. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-32009-5\\_36](https://doi.org/10.1007/978-3-642-32009-5_36)
3. Beimel, A.: Secure schemes for secret sharing and key distribution. Ph.D. thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)



4. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: CCS 2008, pp. 417–426 (2008)
5. 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). [https://doi.org/10.1007/978-3-540-24676-3\\_14](https://doi.org/10.1007/978-3-540-24676-3_14)
6. Chase, M., Chow, S.S.M.: Improving privacy and security in multi-authority attribute-based encryption. In: CCS 2009, pp. 121–130 (2009)
7. Choo, K.R., Domingo-Ferrer, J., Zhang, L.: Cloud cryptography: theory, practice and future research directions. FGCS **62**, 51–53 (2016)
8. Chow, S.S.M.: A framework of multi-authority attribute-based encryption with outsourcing and revocation. In: SACMAT 2016, pp. 215–226 (2016)
9. Chow, S.S.M., Roth, V., Rieffel, E.G.: General certificateless encryption and timed-release encryption. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 126–143. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-85855-3\\_9](https://doi.org/10.1007/978-3-540-85855-3_9)
10. Daemen, J., Vincent, R.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002). <https://doi.org/10.1007/978-3-662-04722-4>
11. Domingo-Ferrer, J., Ricci, S., Domingo-Enrich, C.: Outsourcing scalar products and matrix products on privacy-protected unencrypted data stored in untrusted clouds. Inf. Sci. **436–437**, 320–342 (2018)
12. Hohenberger, S., Waters, B.: Online/offline attribute-based encryption. In: PKC 2014, pp. 293–310 (2014)
13. Hung, T., Li, X., Wan, Z., Wan, M.: Privacy preserving cloud data access with multi-authorities. In: INFOCOM 2013, pp. 2625–2633 (2013)
14. Itkis, G., Shen, E., Varia, M., Wilson, D., Yerukhimovich, A.: Bounded-collusion attribute-based encryption from minimal assumptions. In: Fehr, S. (ed.) PKC 2017. LNCS, vol. 10175, pp. 67–87. Springer, Heidelberg (2017). [https://doi.org/10.1007/978-3-662-54388-7\\_3](https://doi.org/10.1007/978-3-662-54388-7_3)
15. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS 2000 (2000)
16. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-20465-4\\_31](https://doi.org/10.1007/978-3-642-20465-4_31)
17. Li, J., Huang, Q., Chen, X., Chow, S.S.M., Wong, D.S., Xie, D.: Multi-authority ciphertext-policy attribute-based encryption with accountability. In: ASIACCS 2011, pp. 386–390 (2011)
18. Li, M., Yu, S., Zheng, Y., Ren, K., Lou, W.: Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. TPDS **24**(1), 131–143 (2013)
19. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: CCS 2013, pp. 463–474 (2013)
20. Rouselakis, Y., Waters, B.: Efficient statically-secure large-universe multi-authority attribute-based encryption. In: FC 2015, pp. 315–332 (2015)
21. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). [https://doi.org/10.1007/114266639\\_27](https://doi.org/10.1007/114266639_27)
22. Seo, J.H., Emura, K.: Revocable identity-based encryption revisited: security model and construction. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 216–234. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-36362-7\\_14](https://doi.org/10.1007/978-3-642-36362-7_14)

23. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. IACR Cryptology ePrint Archive (2006)
24. Wan, Z., Liu, J., Deng, R.H.: Hasbe: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing. *TIFS* **7**(2), 743–754 (2012)
25. Wang, C., Wang, Q., Ren, K.: Privacy-preserving public auditing for data storage security in cloud computing. In: *INFOCOM 2010*, pp. 1–9 (2010)
26. Wang, S., Liang, K., Liu, J.K., Chen, J., Yu, J., Xie, W.: Attribute-based data sharing scheme revisited in cloud computing. *TIFS* **11**(8), 1661–1673 (2017)
27. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: *PKC 2011*, pp. 53–70 (2011)
28. Yang, K., Jia, X., Ren, K., Zhang, B.: DAC-MACS: effective data access control for multi-authority cloud storage systems. In: *INFOCOM 2010*, pp. 2895–2903 (2013)
29. Yap, W.-S., Chow, S.S.M., Heng, S.-H., Goi, B.-M.: Security mediated certificate-less signatures. In: Katz, J., Yung, M. (eds.) *ACNS 2007*. LNCS, vol. 4521, pp. 459–477. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-72738-5\\_30](https://doi.org/10.1007/978-3-540-72738-5_30)
30. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: *INFOCOM 2010*, pp. 1–9 (2010)
31. Yu, Y., Miyaji, A., Au, M.H., Susilo, W.: Cloud computing security and privacy: standards and regulations. *Comput. Stand. Interfaces* **54**, 1–2 (2017)
32. Zhang, Y., Liu, J., Zhang, Z., Liu, W.: Fast data cloud-outsourcing with public filtering and flexible access control for mobile devices (2018, submitted)



# Hide the Modulus: A Secure Non-Interactive Fully Verifiable Delegation Scheme for Modular Exponentiations via CRT

Osmanbey Uzunkol<sup>1</sup>(✉), Jothi Rangasamy<sup>2</sup>, and Lakshmi Kuppusamy<sup>2</sup>

<sup>1</sup> Faculty of Mathematics and Computer Science, FernUniversität in Hagen, Hagen, Germany

osmanbey.uzunkol@gmail.com

<sup>2</sup> Society for Electronic Transactions and Security (SETS), Chennai, India  
{jothiram,lakshdev}@setsindia.net

**Abstract.** Security protocols using public-key cryptography often requires large number of costly modular exponentiations (**MEs**). With the proliferation of resource-constrained (mobile) devices and advancements in cloud computing, delegation of such *expensive* computations to powerful server providers has gained lots of attention. In this paper, we address the problem of verifiably secure delegation of **MEs** using two servers, where at most one of which is assumed to be malicious (the OMTUP-model). We first show verifiability issues of two recent schemes: We show that a scheme from IndoCrypt 2016 does not offer full verifiability, and that a scheme for  $n$  simultaneous **MEs** from AsiaCCS 2016 is verifiable only with a probability 0.5909 instead of the author's claim with a probability 0.9955 for  $n = 10$ . Then, we propose the first *non-interactive fully verifiable* secure delegation scheme by *hiding the modulus* via Chinese Remainder Theorem (CRT). Our scheme improves also the computational efficiency of the previous schemes considerably. Hence, we provide a *lightweight delegation* enabling weak clients to securely and verifiably delegate **MEs** without any expensive local computation (neither online nor offline). The proposed scheme is highly useful for devices having (a) only ultra-lightweight memory, and (b) limited computational power (e.g. sensor nodes, RFID tags).

**Keywords:** Verifiable and secure delegation  
Modular exponentiations · Cloud security · Applied cryptography  
Lightweight cryptography

## 1 Introduction

Recent advances in mobile computing, internet of things (IoT), and cloud computing makes delegating heavy computational tasks from computationally weak units, devices, or components to a powerful third party servers (also programs and applications) feasible and viable. This enables weak mobile clients with

limited memory and computational capabilities (e.g. sensor nodes, smart cards and RFID tags) to be able to utilize several applications of these technologies, which otherwise is difficult and often impossible because of underlying resource-intensive operations and consumption of considerable amount of energy.

Unlike fully homomorphic encryption, *secure delegation* of expensive cryptographic operations (like **MEs** modulo a prime number  $p$ ) is the most practical option along with its little computational costs and applications for critical security applications. However, delegating **MEs** of the form  $u^a \bmod p$  to untrusted servers while ensuring the desired security and privacy properties is highly challenging; i.e. either  $u$  or  $a$ , or even both (in most privacy enhancing applications), contain sensitive informations, thence required to be properly protected from untrusted servers. Beside these challenges, ensuring the verifiability of the delegated computation is very important. As also pointed out in [8,11], failure in the verification of a delegated computation has severe consequences especially if the delegated **MEs** are the core parts of authentication or signature schemes.

**Related Work.** After the introduction of wallets with observers by Chaum and Pedersen [4], Hohenberger and Lysyanskaya [7] provided the first secure delegation scheme for group exponentiations (**GEs**) with a verifiability probability  $1/2$  using two servers, where at most one of them is assumed to be malicious (the OMTUP-model). They also gave the first formal simulation-based security notions for the delegation of **GEs** in the presence of malicious powerful servers. In ESORICS 2012, Chen *et al.* [5] improved both the verifiability probability (to  $2/3$ ) and the computational overhead of [7]. A secure delegation scheme for two simultaneous **GEs** with a verifiability probability  $1/2$  is also introduced in [5]. In ESORICS 2014, for the first time Wang *et al.* [13] proposes a delegation scheme for **GEs** using a *single untrusted server* with a verifiability probability  $1/2$ . This scheme involves an *online* group exponentiation of a *small* exponent by the delegator; the choice of such a small exponent is subsequently shown to be insecure by Chevalier *et al.* [6] in ESORICS 2016. Furthermore, it is also shown in [6] essentially that a secure *non-interactive* (i.e. single-round) delegation with a single untrusted server requires at least an online computation of a **GE** even without any verifiability if the modulus  $p$  is known to the server. Kiraz and Uzunkol [8] introduce the first *two-round* secure delegation scheme for **GEs** using a single untrusted server having an *adjustable* verifiability probability requiring however a huge number of queries to the server. They also provide a delegation scheme for  $n$  simultaneous **GEs** with an adjustable verifiability probability. Cavallo *et al.* [2] propose subsequently another delegation scheme with a verifiability probability  $1/2$  again by using a single untrusted server under the assumption that pairs of the form  $(u, u^x)$  are granted at the precomputation for variable base elements  $u$ . However, realizing this assumption is difficult (mostly impossible) for resource-constrained devices. In AsiaCCS 2016, Ren *et al.*[11] proposed the *first fully verifiable* (with a verifiability probability 1) secure delegation scheme for **GEs** in the OMTUP-model at the expense of an *additional round* of communication. They also provide a two-round secure delegation scheme for

$n \in \mathbb{Z}^{>1}$  simultaneous **GEs** which is claimed to have a verifiability probability  $1 - \frac{1}{2n(n+1)}$ .

Kuppasamy and Rangasamy use in INDOCRYPT 2016 [9] for the first time the special *ring structure* of  $\mathbb{Z}_p$  with the aim of eliminating the second round of communication and providing full verifiability simultaneously. They propose a *non-interactive* efficient secure delegation scheme for **MEs** using Chinese Remainder Theorem (CRT) in the OMTUP-model which is claimed to satisfy full-verifiability under the intractability of the factorization problem. This approach is also used very recently by Zhou *et al.* [14] together with *disguising* the modulus  $p$  itself, also assuming the intractability of the factorization problem. They proposed an efficient delegation scheme with an adjustable verifiability probability using a single untrusted server. However, the scheme in [14] does not achieve the desired security properties.

**Our Contribution.** This paper has the following major goals:

1. We analyze two delegation schemes recently proposed at INDOCRYPT 2016 [9] and at AsiaCCS 2016 [11]:
  - (a) We show that the scheme in [9] is unfortunately *totally* unverifiable, i.e. a malicious server can always cheat the delegator without being noticed, instead of the author's claim of satisfying the full verifiability.
  - (b) We show that the scheme for  $n$  simultaneous **MEs** in [11] does not achieve the claimed verifiability guarantees; instead of having the verifiability probability  $1 - \frac{1}{2n(n+1)}$ , it only has the verifiability probability at most  $1 - \frac{n-1}{2(n+1)}$ . For instance, it offers a verifiability probability at most  $\approx 0.5909$  instead of the author's claim in [11] offering a verifiability probability  $\approx 0.9955$  for  $n = 10$ .
2. We propose the first *non-interactive fully verifiable* secure delegation scheme HideP for **MEs** in the OMTUP-model by disguising the prime number  $p$  via CRT. HideP is not only computationally much more efficient than the previous schemes but requires also no interactive round, whence substantially reduces the *communication overhead*. In particular, hiding  $p$  enables the delegator to achieve both *non-interactivity and full verifiability* at the same time efficiently.
 

Note that the delegator of **MEs** hides the prime modulus  $p$  from the servers, and *not* from a party intended to be communicated (i.e. a weak device (delegator) does not hide  $p$  with whom it wants to run a cryptographic protocol). In other words, it *solely* hides  $p$  from the third-party servers to which the computation of **MEs** is delegated.
3. We apply HideP to speed-up blinded Nyberg-Rueppel signature scheme [10].

We refer the readers to the full version of the paper [12] which provide a delegated preprocessing technique Rand. It eliminates the large *memory requirement* and reduces substantially the *computational cost* of the precomputation step. The overall delegation mechanism (i.e. HideP together with Rand) offers a *complete solution* for delegating the expensive **MEs** with full verifiability and security,

whence distinguish our mechanism as a highly usable secure delegation primitive for resource-constrained devices.

## 2 Preliminaries and Security Model

In this section, we first revisit the definitions and the basic notations related to the delegation of **MEs**. We then give a *formal* security model by adapting the previous security models of Hohenberger and Lysyanskaya [7] and Cavallo *et al.* [2]. Lastly, an overview for the requirements of the delegation of a **ME**<sup>1</sup> is given.

### 2.1 Preliminaries

We denote by  $\mathbb{Z}_m$  the quotient ring  $\mathbb{Z}/m\mathbb{Z}$  for a natural number  $m \in \mathbb{N}$  with  $m > 1$ . Similarly,  $\mathbb{Z}_m^*$  denotes the multiplicative group of  $\mathbb{Z}_m$ .

Let  $\sigma$  be a global security parameter given in a unary representation (e.g.  $1^\sigma$ ). Let further  $p$  and  $q$  be prime numbers with  $q \mid (p - 1)$  of lengths  $\sigma_1$  and  $\sigma_2$ , respectively. The values  $\sigma_1$  and  $\sigma_2$  are calculated at the setup of a cryptographic protocol on the input of  $\sigma$ . Let  $\mathbb{G} = \langle g \rangle$  denote the multiplicative subgroup of  $\mathbb{Z}_p^*$  of order  $q$  with a fixed generator  $g \in \mathbb{G}$ .

The process of running a probabilistic algorithm  $A$ , which accepts  $x_1, x_2, \dots$  as inputs, and produces an output  $y$ , is denoted by  $y \leftarrow A(x_1, x_2, \dots)$ . Let  $(z_A, z_B, \text{tr}) \leftarrow (A(x_1, x_2, \dots), B(y_1, y_2, \dots))$  denote the process of running an interactive protocol between an algorithm  $A$  and an algorithm  $B$ , where  $A$  accepts  $x_1, x_2, \dots$ , and  $B$  accepts  $y_1, y_2, \dots$  as inputs (possibly together with some *random* coins) to produce the final output  $z_A$  and  $z_B$ , respectively. We use the expression  $\text{tr}$  to represent the sequence of messages exchanged by  $A$  and  $B$  during protocol execution. By abuse of notation, the expression  $y \leftarrow x$  also denotes assigning the value of  $x$  to a variable  $y$ .

**Delegation Mechanism and Protocol Definition.** We assume that a delegation mechanism consists of two types of parties called as the *client* (or *delegator*)  $\mathcal{C}$  (*trusted* but resource-constrained part) and *servers*  $\mathcal{U}$  (potentially *untrusted* but powerful part), where  $\mathcal{U}$  can consist of one or more parties. Hence, the scenario raises if  $\mathcal{C}$  is willing to *delegate* (or *outsource*) the computation of certain functions to  $\mathcal{U}$ . For a given  $\sigma$ , let  $F : \text{Dom}(F) \rightarrow \text{CoDom}(F)$  be a function, where  $F$ 's domain is denoted by  $\text{Dom}(F)$  and  $F$ 's co-domain is denoted by  $\text{CoDom}(F)$ .  $\text{desc}(F)$  denotes the description of  $F$ . We have two cases for  $\text{desc}(F)$ :

1.  $\text{desc}(F)$  is known to both  $\mathcal{C}$  and  $\mathcal{U}$ , or
2.  $\text{desc}(F)$  is known to  $\mathcal{C}$ , and another description  $\text{desc}(F')$  is given to  $\mathcal{U}$  such that the function  $F$  can only be obtained from  $F'$  if a *trapdoor* information  $\tau$  is given. By abuse of notation, we sometimes write  $\tau(F) = F'$ .

---

<sup>1</sup> In this paper, we introduce a *special* delegation scheme by working with a subgroup  $\mathbb{G}$  of the group  $\mathbb{Z}_p^*$  of prime order  $q$ .

From now on, we concentrate on the second case since we propose a delegation scheme in this scenario. A client-server protocol for the delegated computation of  $F$  is defined as a multiparty communication protocol between  $\mathcal{C}$  and  $\mathcal{U}$  and denoted by  $(\mathcal{C}(1^\sigma, \text{desc}(F), x, \tau), \mathcal{U}(1^\sigma, \text{desc}(F')))$ , where the input  $x$  and the trapdoor  $\tau$  are known *only* by  $\mathcal{C}$ . A delegated computation of the value  $y = F(x)$ , denoted by

$$(y_{\mathcal{C}}, y_{\mathcal{U}}, \text{tr}) \leftarrow (\mathcal{C}(1^\sigma, \text{desc}(F), x, \tau), \mathcal{U}(1^\sigma, \text{desc}(F'))),$$

which is an execution of the above client-server protocol using independently chosen random bits for  $\mathcal{C}$  and  $\mathcal{U}$ . At the end of this execution,  $\mathcal{C}$  learns  $y_{\mathcal{C}} = y$ ,  $\mathcal{U}$  learns  $y_{\mathcal{U}}$ ; and  $\text{tr}$  is the sequence of messages exchanged by  $A$  and  $B$ . Note that the execution may happen sequentially or concurrently. In the case of the delegation of **MEs**, the aim is to always have  $y_{\mathcal{U}} = \emptyset$ .

*Factorization Problem.* We prove some security properties of the proposed scheme later by using the intractability of the factorization problem<sup>2</sup>: Given a composite integer  $n$ , where  $n$  is a product of two distinct primes  $p$  and  $q$ , the *factorization problem* asks to compute  $p$  or  $q$ . The formal definition is as follows:

**Definition 1.** (*Factorization Problem*) Let  $\sigma$  be a security parameter given in unary representation. Let further  $\mathcal{A}$  be a probabilistic polynomial-time algorithm. Let further the primes  $p$  and  $q$ ,  $p \neq q$ , are obtained by running a modulus generation algorithm **PrimeGen** on the input of  $\sigma$  with  $n = pq$ . Run  $\mathcal{A}$  with the input  $n$ . The adversary  $\mathcal{A}$  wins the experiment if it outputs either  $p$  or  $q$ . We define the advantage of  $\mathcal{A}$  as

$$\text{Adv}_{\mathcal{A}}^{\text{Fact}}(\sigma) = \text{Prob}[x = p \text{ or } x = q : (n, p, q) \leftarrow \text{PrimeGen}(1^\sigma), x \leftarrow \mathcal{A}(n)].$$

## 2.2 Security Model

Hohenberger and Lysyanskaya provided first formal simulation-based security notions for secure and verifiable delegation of cryptographic computations in the presence of malicious powerful servers [7]. Different security assumptions for delegation of **MEs** can be summarized according to [7] as follows:

- One-Untrusted Program (OUP): There exists a single malicious program  $\mathcal{U}$  performing the delegated **MEs**.
- One-Malicious version of a Two-Untrusted Program (OMTUP): There exist two untrusted programs  $\mathcal{U}_1$  and  $\mathcal{U}_2$  performing the delegated **MEs** but only one of them behaves maliciously.
- Two-Untrusted Program (TUP): There exist two untrusted programs  $\mathcal{U}_1$  and  $\mathcal{U}_2$  performing the delegated **MEs** and both of them may simultaneously behave maliciously, but they do not maliciously collude.

<sup>2</sup> We assume here that the prime numbers  $p$  and  $q$  are chosen suitably that the factorization of  $n = pq$  is intractable.

Cavallo *et al.* [2] gave a formal definition for delegation schemes by *relaxing* the security definitions first given in [7]. Although the simulation-based security definitions [7] intuitively include (whatever can be efficiently computed about secret values with the protocol’s view can also be efficiently computed without this view [6]) the most direct way of guaranteeing the desired secrecy and verifiability, its formalization is unfortunately highly complex and subtle. Therefore, simpler indistinguishability-based security definitions have been recently used both in [2] and in [6], which, in particular, include the fact that an untrusted server is unable to distinguish which inputs the other parties use.

In this section, we adapt the security definitions of [2] for our security requirements to the OMTUP-model of [7], i.e. the adversary is modeled by a pair of algorithms  $\mathcal{A} = (\mathcal{E}, \mathcal{U}')$ , where  $\mathcal{E}$  denotes the adversarial environment and  $\mathcal{U}' = (\mathcal{U}'_1, \mathcal{U}'_2)$  is a malicious adversarial software in place of  $\mathcal{U} = (\mathcal{U}_1, \mathcal{U}_2)$ , where exactly one of  $(\mathcal{U}'_1, \mathcal{U}'_2)$  is assumed to be malicious. In the OMTUP-model we have the fundamental assumption that after interacting with  $\mathcal{C}$ , any communication between  $\mathcal{E}$  and  $\mathcal{U}'_1$  or between  $\mathcal{E}$  and  $\mathcal{U}'_2$  pass solely through the delegator  $\mathcal{C}$  [7].

**Completeness.** If the parties  $(\mathcal{C}, \mathcal{U}_1$  and  $\mathcal{U}_2)$  executing the scheme follow the scheme specifications, then  $\mathcal{C}$ ’s output obtained at the end of the execution would be equal to the output obtained by evaluating the function  $F$  on  $\mathcal{C}$ . The following is the formal definition for completeness:

**Definition 2.** For the security parameter  $\sigma$ , let  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  be a client-server protocol for the delegated computation of a function  $F$ . We say that  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  satisfies completeness if for any  $x$  in the domain of  $F$ , it holds that

$$\text{Prob}[(y_C, y_S, \text{tr}) \leftarrow (\mathcal{C}(1^\sigma, \text{desc}(F), x), \mathcal{U}_i(1^\sigma, \text{desc}(F')) : y_C = F(x)] = 1.$$

**Verifiability.** Verifiability means informally that if  $\mathcal{C}$  follows the protocol, then the malicious adversary  $\mathcal{A} = (\mathcal{E}, \mathcal{U}'_i)$ ,  $i = 1$  or  $i = 2$ , cannot convince  $\mathcal{C}$  to obtain some output  $y'$  different from the actual output  $y$  at the end of the protocol. The model let further the adversary choose  $\mathcal{C}$ ’s trapdoored input  $\tau(F(x))$  and take part in exponential/polynomial number of protocol executions before it attempts to convince  $\mathcal{C}$  with incorrect output values (corresponding to the environmental adversary  $\mathcal{E}$ ).

**Definition 3.** Let  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  be a client-server protocol for the delegated computation of a function  $F$  and  $\mathcal{U}' = (\mathcal{U}'_1, \mathcal{U}'_2)$  be a malicious adversarial software in place of  $\mathcal{U} = (\mathcal{U}_1, \mathcal{U}_2)$ . We say that  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  satisfies  $(t_v, \epsilon_v)$ -verifiability against a malicious adversary if for any  $\mathcal{A} = (\mathcal{E}, \mathcal{U}'_i)$ , either  $i = 1$  or  $i = 2$ , running in time  $t_v$ , it holds that

$$\text{Prob}[out \leftarrow \text{VerExp}_{F', \mathcal{A}}(1^\sigma) : out = 1] \leq \epsilon_v,$$

for small  $\epsilon_v$ , where experiment  $\text{VerExp}$  is defined as follows:

1.  $i = 1$ .
2.  $(a, \tau(F(x_1)), \text{aux}) \leftarrow \mathcal{A}(1^\sigma, \text{desc}(F'))$



3. *While*  $a \neq \text{attack}$  *do*  
 $(y_i, (a, \tau(F(x_{i+1})), \text{aux}), \text{tr}_i) \leftarrow (\mathcal{C}(\tau(F(x_i))), \mathcal{A}(\text{aux}))$   
 $i \leftarrow i + 1$
4.  $\tau(F(x)) \leftarrow \mathcal{A}(\text{aux})$
5.  $(y', \text{aux}, \text{tr}_i) \leftarrow (\mathcal{C}(\tau(F(x))), \mathcal{A}(\text{aux}))$
6. *return*: 1 *if*  $y' \neq \perp$  *and*  $y' \neq F(x)$
7. *return*: 0 *if*  $y' = \perp$  *or*  $y' = F(x)$ .

If  $\epsilon_v$  is negligibly small for any algorithm  $A$  running in time  $t_v$ , then  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  is said to satisfy full verifiability.

**Security.** Security means informally that if  $\mathcal{C}$  follows the protocol, then the malicious adversary  $\mathcal{A} = (\mathcal{E}, \mathcal{U}'_i)$ ,  $i = 1$  or  $i = 2$ , cannot obtain any information about  $\mathcal{C}$ 's input  $x$ . The model let further the adversary choose  $\mathcal{C}$ 's trapdoored input  $\tau(F(x))$  and take part in exponential/polynomial number of protocol executions before it attempts to obtain useful information about  $\mathcal{C}$ 's input (corresponding to the environmental adversary  $\mathcal{E}$ ).

**Definition 4.** Let  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  be a client-server protocol for the delegated computation of a function  $F$  and  $\mathcal{U}' = (\mathcal{U}'_1, \mathcal{U}'_2)$  be a malicious adversarial software in place of  $\mathcal{U} = (\mathcal{U}_1, \mathcal{U}_2)$ . We say that  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  satisfies  $(t_s, \epsilon_s)$ -security against a malicious adversary if for any  $\mathcal{A} = (\mathcal{E}, \mathcal{U}'_i)$ , either  $i = 1$  or  $i = 2$ , running in time  $t_s$ , it holds that

$$\text{Prob}[out \leftarrow \text{SecExp}_{F', \mathcal{A}}(1^\sigma) : out = 1] \leq \epsilon_s,$$

for negligibly small  $\epsilon_s$  for any algorithm  $A$  running in time  $t_s$ , where experiment  $\text{SecExp}$  is defined as follows:

1.  $(a, \tau(F(x_1)), \text{aux}) \leftarrow \mathcal{A}(1^\sigma, \text{desc}(F'))$
2. *While*  $a \neq \text{attack}$  *do*  
 $(y_i, (a, \tau(F(x_{i+1})), \text{aux}), \cdot) \leftarrow (\mathcal{C}(\tau(F(x_i))), \mathcal{A}(\text{aux}))$   
 $i \leftarrow i + 1$
3.  $(\tau(F(x_0)), \tau(F(x_1)), \text{aux}) \leftarrow \mathcal{A}(\text{aux})$
4.  $b \leftarrow 0, 1$
5.  $(y', b', \text{tr}) \leftarrow (\mathcal{C}(\tau(F(x_b))), \mathcal{A}(\text{aux}))$
6. *return*: 1 *if*  $b = b'$
7. *return*: 0 *if*  $b \neq b'$ .

*Remark 1.* We emphasize that the above security definition corresponds to the OMTUP-model of [7]. As in [7], the adversary  $\mathcal{A}$  corresponds to both  $\mathcal{E}$  and  $\mathcal{U}'$ , and can only interact each other over  $\mathcal{C}$  after they once begin interacting with  $\mathcal{C}$ . The behavior of both parts ( $\mathcal{E}$  and  $\mathcal{U}'$ ) is modeled as a single adversary  $\mathcal{A}$  by letting the adversary  $\mathcal{A}$  submit its own inputs to  $\mathcal{C}$  and see/take part in multiple executions of  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$ .

**Efficiency Metrics.**  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  has efficiency parameters

$$(t_F, t_{m_C}, t_C, t_{U_1}, t_{U_2}, cc, mc)$$

where  $F$  can be computed using  $t_F(\sigma)$  atomic operations, requires  $t_{mc}(\sigma)$  atomic storage for  $\mathcal{C}$ ,  $\mathcal{C}$  computes  $t_{\mathcal{C}}(\sigma)$  atomic operations,  $\mathcal{U}_i$  can be run using  $t_{\mathcal{U}_i}(\sigma)$  atomic operations,  $\mathcal{C}$  and  $\mathcal{U}_i$  exchange a total of at most  $mc$  messages of total length at most  $cc$  for  $i = 1, 2$ .<sup>3</sup>

### 2.3 Steps of a Delegation Scheme

Let  $p$  and  $q$  be distinct prime numbers. We now give four main steps of a delegation of  $u^a \bmod p$  under the OMTUP-model, where  $u \in \mathbb{G}$ ,  $a \in \mathbb{Z}_q^*$  and  $\mathbb{G}$  is a subgroup of  $\mathbb{Z}_p^*$  of order  $q$ .

1. **Precomputation: Invocation of the subroutine Rand:** A preprocessing subroutine `Rand` is required to randomize  $u$  and  $a$  and to generate the trapdoor information  $\tau$ , see the paper's full version for the details [12].
2. **Randomizing  $a \in \mathbb{Z}_q^*$  and  $u \in \mathbb{G}$ .** The base  $u$  and the exponent  $a$  are both randomized by  $\mathcal{C}$  by performing only modular multiplications (MMs) in  $\mathbb{Z}_q^*$  and  $\mathbb{G}$  with the values from `Rand` using the trapdoor information  $\tau$ .
3. **Delegation to servers.** The randomized elements are queried to the servers  $\mathcal{U}_1$  and  $\mathcal{U}_2$  by using  $\tau$ . For  $i = 1, 2$ ,  $U_i(\tau(\alpha), \tau(h))$  denotes the delegation of  $h^\alpha \bmod p$  with  $\alpha \in \mathbb{Z}_q^*$ ,  $h \in \mathbb{G}$  using the trapdoor information  $\tau$  in order to *disguise* the parameters  $p, q$ , whence the concrete description of  $\mathbb{G}$ .
4. **Verification of the delegated computation.** Upon receiving the outputs of  $\mathcal{U}_1$  and  $\mathcal{U}_2$ , the validity of the delegated computation is verified by comparing the received data with some elements from `Rand`. If the verification fails, an error message  $\perp$  is returned.
5. **Derandomizing outputs and computing  $u^a \bmod p$ .** If the verification is successful, then  $u^a \bmod p$  is computed by  $\mathcal{C}$  by performing only MMs.

## 3 Verifiability Issues in Two Recent Delegation Schemes

In this section, we show two verifiability issues for recently proposed delegation schemes appeared in INDOCRYPT 2016 [9] and AsiaCCS 2016 [11].

### 3.1 An Attack on the Verifiability of Kuppusamy and Rangasamy's Scheme from INDOCRYPT 2016

Using CRT, Kuppusamy and Rangasamy proposed a highly efficient secure delegation scheme for MEs in subgroups of  $\mathbb{Z}_p^*$  [9]. We now show that the scheme is unfortunately totally unverifiable.

---

<sup>3</sup> We here only consider the group operations like group multiplications, modular reduction, inversions and exponentiations as atomic operations, and neglect any lower-order operations such as congruence testing, equality testing, and modular additions.

**Attack:** Let the notation be as in [9]. Assume first that the server  $\mathcal{U}_1$  is malicious and  $\mathcal{U}_2$  is honest. Since the prime  $p$  is public,  $\mathcal{U}_1$  can compute  $r_1 r_2 = n/p$ , and return the bogus values

$$Y_{11} := D_{11} + r_1 r_2 \bmod n, \text{ and } Y_{12} := D_{12} + r_1 r_2 \bmod n. \tag{1}$$

Now,  $\mathcal{U}_1$  can successfully distinguish  $D_{11}$  and  $D_{12}$  from  $D_{13}$  with probability 1 since the first component of  $D_{13}$  is an element of  $\mathbb{G}$  whereas the first components of  $D_{11}$  and  $D_{12}$  are elements of  $\mathbb{Z}_n$ . Afterwards, by the choices of the distinct primes  $p, r_1$  and  $r_2$ , and the properties  $Y_{12} \equiv D_{12} \bmod r_2$  and  $Y_{11} \equiv D_{11} \bmod r_2$ ,  $\mathcal{U}_1$  can pass the verification step with  $Y_{11}$  and  $Y_{12}$  instead of using  $D_{11}$  and  $D_{12}$ , respectively. This leads to the bogus final output

$$Y_{12} \cdot D_{21} \cdot D_{13}$$

instead of the actual output  $u^a = D_{12} \cdot D_{13} \cdot D_{22}$  given in [9].

Similarly, a malicious  $\mathcal{U}_2$  can successfully distinguish  $D_{21}$  from  $D_{22}$  with probability 1 since the first component of  $D_{22}$  is an element of  $\mathbb{G}$  whereas the first component of  $D_{21}$  is an element of  $\mathbb{Z}_n$ . Then,  $\mathcal{U}_2$  can act as the untrusted server by computing

$$Y_{21} \equiv D_{21} + r_1 r_2 \bmod r_2. \tag{2}$$

Afterwards, by the choices of the distinct primes  $p, r_1$  and  $r_2$  and the property  $Y_{21} \equiv D_{21} \bmod r_2$ ,  $\mathcal{U}_2$  can pass the verification step with the bogus value  $Y_{21}$ . This results in the output

$$D_{12} \cdot Y_{21} \cdot D_{13}$$

instead of  $u^a = D_{12} \cdot D_{13} \cdot D_{21}$  given in [9]. Hence, the scheme in [9] is unfortunately totally unverifiable and the claim regarding full verifiability [9, Thm. 2, pp. 90] does not hold.

### 3.2 An Attack on the Verifiability of Ren *et al.*'s Simultaneous Delegation Scheme from AsiaCCS 2016

Ren *et al.* proposed the first fully verifiable two-round secure delegation scheme for **GEs** together with a delegation scheme of  $n$  simultaneous **MEs** [11]. We now show that the author's claim [11, Thm. 4.2, pp. 298] does not hold.

**Attack:** Let the notation be as in [11]. Assume without loss of generality that the server  $\mathcal{U}_2$  is malicious and  $\mathcal{U}_1$  is honest. Then,  $\mathcal{U}_2$  chooses a random  $\theta \in \mathbb{G}$  and sends the bogus value

$$T_{212} \equiv D_{212} \cdot \theta$$

instead of  $D_{212}$  after correctly distinguishing  $D_{212}$  from  $D_{211}$  with probability at least  $1/2$ . Then,  $\mathcal{C}$  computes

$$\Theta := \theta \left( \prod_{j=1, i \neq j}^n w_j \right)^{c/t_1} \equiv T_{212} g^{-c}.$$

In order to pass the verification step with  $\Theta \cdot T$  instead of  $T$ ,  $\mathcal{U}_2$  requires to find an output  $T_{23j}$  with  $T_{23j} \notin \{D_{22}, D_{23i}\}$ , i.e.  $T_{23j} \equiv D_{23j} \pmod n$  for some  $i \neq j$ , and sends  $\theta \cdot T_{23j}$  instead of  $T_{23j}$ . Now, since there are  $n(n+1)/2$  pairs from the set

$$D := \{D_{22}, D_{231}, \dots, D_{23n}\}$$

we totally have  $n(n-1)$  possibilities for  $T_{23j}$  corresponding to a single component of such a pair. If  $(\Theta_1, \Theta_2)$  is a pair from the set  $D$ . Then,

1. there exists 2 values for  $T_{23j}$  which can be detected by  $\mathcal{C}$  corresponding to the single pair with  $(\Theta_1, \Theta_2) \equiv (D_{22}, D_{23i}) \pmod p$ ,
2. there exists  $n-1$  values of  $T_{23j}$  which can be detected by  $\mathcal{C}$  corresponding to the pairs of the form  $(\Theta_1, \Theta_2)$  with  $T_1 = \Theta_1 \equiv D_{22}$  and  $\Theta_2 \not\equiv D_{23i}$ ,
3. there exists  $n-1$  values of  $T_{23j}$  which can be detected by  $\mathcal{C}$  corresponding to the pairs of the form  $(\Theta_1, \Theta_2)$  with  $T_{23j} = \Theta_1 \equiv D_{23i}$  and  $\Theta_2 \not\equiv D_{12}$ .

Therefore, there exist

$$n(n+1) - 2 - (n-1) - (n-1) = n(n+1) - 2n = n(n-1)$$

possible values for  $T_{23j}$  with  $T_{23j} \notin \{D_{22}, D_{23i}\}$ . Combining with the probability of correctly guessing the position of  $D_{232}$ , the server  $\mathcal{U}_2$  can cheat  $\mathcal{C}$  with a probability at least  $\frac{n(n-1)}{2n(n+1)} = \frac{n-1}{2(n+1)}$ . Hence, the scheme is verifiable with a probability at most  $1 - \frac{n-1}{2(n+1)}$  instead of the author's claim that the scheme would be verifiable with a probability  $1 - \frac{1}{2n(n+1)}$ . Thereby it also leads to a bogus output  $\theta u_1^{a_1} \dots u_n^{a_n}$ .

For example with  $n = 10$  and  $n = 100$ , the scheme is verifiable only with probabilities at most  $13/22 \approx 0.5909$  and  $103/202 \approx 0.5099$  instead of the claims with probabilities  $219/220 \approx 0.9955$  and  $20199/20200 \approx 0.9999$ , respectively. Clearly, the verification probability becomes  $1/2$  if  $n$  tends to infinity.

## 4 HideP: A Secure Fully Verifiable One-Round Delegation Scheme for Modular Exponentiations

In this section, we introduce our secure delegation scheme HideP in the OMTUP-model.

Let  $\mathbb{G} = \langle g \rangle$  denote the multiplicative subgroup of  $\mathbb{Z}_p^*$  of prime order  $q$  with a fixed generator  $g \in \mathbb{G}$ . Our scheme HideP uses another prime  $r \neq p$  of length  $\sigma_1$  (e.g.  $p$  and  $r$  are of about the same size) such that  $\mathbb{G}_1$  is a subgroup of prime order  $q_1$  of length  $\sigma_2$  (e.g.  $q$  and  $q_1$  are of about the same size). We set  $n := p \cdot r$  and  $m := q_1 \cdot q$ . Note that HideP uses the prime number  $p$  as a trapdoor information, i.e.  $p$  must be *kept secret* to both  $\mathcal{U}_1$  and  $\mathcal{U}_2$ .

Throughout the section  $\mathcal{U}_i(\alpha, h)$  denotes that  $\mathcal{U}_i$  takes  $(\alpha, h) \in \mathbb{Z}_m^* \times \mathbb{Z}_n^*$  as inputs, and outputs  $h^\alpha \pmod n$  for  $i = 1, 2$ , as described in Sect. (2).

### 4.1 HideP: A Secure Fully Verifiable One-Round Delegation Scheme

Our aim is to delegate  $u^a \bmod p$  with  $a \in \mathbb{Z}_q^*$  and  $u \in \mathbb{G}$ .

We now describe our scheme HideP. Public and private parameters of HideP are given as follows:

**Public parameter:**  $n$ ,

**Private parameters:** Prime numbers  $p, r, q$ , and  $q_1$ , description of the subgroup

$\mathbb{G}$  of  $\mathbb{Z}_p^*$  of order  $q$ ,  $u \in \mathbb{G}$ ,  $a \in \mathbb{Z}_q^*$ .<sup>4</sup>

Additionally, the *static values*

$$Q_r := r \cdot (r^{-1} \bmod p) \bmod n, \quad Q_p := p \cdot (p^{-1} \bmod r) \bmod n, \quad (3)$$

$$Q_{q_1} := q_1 \cdot (q_1^{-1} \bmod q) \bmod m, \quad Q_q := q \cdot (q^{-1} \bmod q_1) \bmod m, \quad (4)$$

and

$$R := g \cdot Q_r + g_1 \cdot Q_p \bmod n \quad (5)$$

are calculated at the initialization of HideP.

**Precomputation.** Using the existing preprocessing technique or a delegated version Rand as described in [12],  $\mathcal{C}$  first outputs

$$(G_t \equiv g^t Q_r \bmod n, \quad G_{\gamma t} \equiv g^{\gamma t} Q_r \bmod n, \quad H_{\gamma t} \equiv g_1^{\gamma t} Q_p \bmod n),$$

$$(H_{t_1} \equiv g_1^{t_1} Q_p \bmod n, \quad H_{t_2} \equiv g_1^{t_2} Q_p \bmod n, \quad g_1^t \bmod r),$$

and

$$(\gamma^{-1} \bmod m, \quad T_1 \equiv t_1 Q_q \bmod m, \quad T_2 \equiv t_2 Q_q \bmod m)$$

for random elements  $t_1, t_2, t \in \mathbb{Z}_m^*$  with  $t = t_1 + t_2$ .

**Masking.** The base  $u$  is randomized by  $\mathcal{C}$  with

$$x_1 \equiv u \cdot G_t + H_{t_1} \bmod n, \quad (6)$$

$$x_2 \equiv u G_t + H_{t_2} \bmod n, \quad (7)$$

$$y \equiv G_{\gamma t} + H_{\gamma t} \bmod n. \quad (8)$$

Note that by CRT we have

$$x_1 \equiv x_2 \equiv u g^t \bmod p, \quad y \equiv g^{\gamma t} \bmod p,$$

---

<sup>4</sup> More precisely, hiding  $p$  enables the delegator to *achieve the full verifiability in a single round* unlike the fully verifiable scheme in [11] which requires an additional round of communication. The reason is that it is possible for  $\mathcal{C}$  to send the randomized base and the exponent by a system of simultaneous congruences, and recover/verify the actual outputs by performing modular reductions (once modulo  $p$  for *recovery*, and once modulo  $r$  for *verification*) in a *single round*. Note that for a given  $p$  each client  $\mathcal{C}$  is required to use the same prime number  $r$  since otherwise  $p$  can be found by taking gcd's of different moduli.

and

$$x_1 \equiv g_1^{t_1} \pmod r, x_2 \equiv g_1^{t_2} \pmod r, y \equiv g_1^{\gamma t} \pmod r.$$

Then, the exponent  $a$  is first written as the sum of two randomly chosen elements  $a_1, a_2 \in \mathbb{Z}_m^*$  with  $a = a_1 + a_2$ . Then, the following randomizations are also computed by  $\mathcal{C}$

$$\alpha_1 \equiv a_1 \cdot Q_{q_1} + T_1 \pmod m, \quad (9)$$

$$\alpha_2 \equiv a_2 \cdot Q_{q_1} + T_2 \pmod m, \quad (10)$$

$$\alpha_3 \equiv -a \cdot \gamma^{-1} \pmod m. \quad (11)$$

**Query to  $\mathcal{U}_1$ .**  $\mathcal{C}$  sends the following queries in random order to  $\mathcal{U}_1$ :

1.  $\mathcal{U}_1(\alpha_1, x_1) \leftarrow X_1 \equiv x_1^{\alpha_1} \pmod n$ ,
2.  $\mathcal{U}_1(\alpha_3, y) \leftarrow Y_1 \equiv y^{\alpha_3} \pmod n$ .

**Query to  $\mathcal{U}_2$ .** Similarly,  $\mathcal{C}$  sends the following queries in random order to  $\mathcal{U}_2$ :

1.  $\mathcal{U}_2(\alpha_2, x_2) \leftarrow X_2 \equiv x_2^{\alpha_2} \pmod n$ ,
2.  $\mathcal{U}_2(\alpha_3, y) \leftarrow Y_2 \equiv y^{\alpha_3} \pmod n$ .

**Verifying the Correctness of the Outputs of  $\{\mathcal{U}_1, \mathcal{U}_2\}$ .** Upon receiving the queries  $X_1$  and  $Y_1$  from  $\mathcal{U}_1$ , and  $X_2$  and  $Y_2$  from  $\mathcal{U}_2$ , respectively,  $\mathcal{C}$  verifies

$$(X_1 \pmod r) \cdot (X_2 \pmod r) \stackrel{?}{\equiv} g_1^t \quad (12)$$

and

$$Y_1 \stackrel{?}{\equiv} Y_2 \pmod n. \quad (13)$$

**Recovering  $u^a$ .** If Congruences (12) and (13) hold simultaneously, then  $\mathcal{C}$  believes that the values  $X_1$ ,  $X_2$ ,  $Y_1$  and  $Y_2$  have been computed correctly. It outputs

$$u^a \equiv (X_1 \pmod p) \cdot (X_2 \pmod p) \cdot (Y_1 \pmod p). \quad (14)$$

If the verification step fails, then  $\mathcal{C}$  outputs  $\perp$ .

## 5 Security and Efficiency Analysis

In this section, we give the security analysis of HideP and give a detailed comparison with the previous schemes.

### 5.1 Security Analysis

**Theorem 1.** *Let  $F'$  be given by the exponentiation modulo  $n = pr$ , where the trapdoor information  $\tau$  is given by the primes  $p$  and  $r$ ,  $p \neq r$ . Let further  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  be a one-client, two-server, one-round delegation protocol implementation of HideP. Let the adversary be given as  $\mathcal{A} = (\mathcal{U}', \mathcal{E})$  in the OMTUP-model*

(i.e.  $\mathcal{U}' = (\mathcal{U}'_1, \mathcal{U}'_2)$  and at most one of  $\mathcal{U}'_i$  is malicious with  $i = 1$  or  $i = 2$ ). Then, in the OMTUP-model, the protocol  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  satisfies

1. completeness for HideP,
2. security for the exponent  $a$  and the exponentiation  $u^a$  against any (computationally unrestricted) malicious adversary  $\mathcal{A}$ , i.e.  $\epsilon_s = 0$ , and security for the base  $u$  with  $t_s = \text{poly}(\sigma)$  and  $\epsilon_s = \text{Adv}_{\mathcal{A}'}^{\text{Fact}}(\sigma)$ ,
3. full verifiability for any malicious adversary  $\mathcal{A}$ , where  $t_v = \text{poly}(\sigma)$  and  $\epsilon_v = \text{Adv}_{\mathcal{A}}^{\text{Fact}}(\sigma)$ , and verifiability for any computationally unrestricted malicious adversary  $\mathcal{A}$  with  $\epsilon_v = 1/2 + \epsilon$ , where  $\epsilon$  is negligibly small in  $\sigma$ ,
4. efficiency with parameters where  $(t_F, t_{mc}, t_C, t_{\mathcal{U}_1}, t_{\mathcal{U}_2}, cc, mc)$ , where
  - $F$  can be computed by performing  $t_F = 1$  exponentiation modulo  $p$
  - $C$ 's memory requirement is  $t_{mc}$  consists of 1 output of the Rand scheme,
  - $C$  can be run by expending  $t_C$  atomic operations consisting of 7 modular multiplications and 5 modular reductions (2 multiplications modulo  $p$ , 1 multiplication modulo  $r$ , 3 multiplications modulo  $m$ , 1 multiplication modulo  $n$ , 2 reductions modulo  $r$ , and 3 reductions modulo  $p$ ),
  - $\mathcal{U}_i, i = 1, 2$  computes  $t_{\mathcal{U}_i} = 2$  exponentiations modulo  $n$  for each  $i = 1, 2$ ,
  - $C$  and  $\mathcal{U}_i$  exchange a total of at most  $mc = 4$  messages of total length  $cc$  consisting of 2 elements modulo  $m$  and 2 elements modulo  $n$  for  $i = 1, 2$ .

**Proof.** We first note that the efficiency results can easily be verified by inspecting the description of HideP for the efficiency parameters given above. Throughout the rest of the proof we assume without loss of generality that  $\mathcal{U}_1$  is a malicious server, i.e. adversary is given as  $\mathcal{A} = (\mathcal{U}_1, \mathcal{E})$ .

*Completeness.* We first prove the completeness of the verification step. Since the same base  $y$  and the exponent  $\alpha_3$  are delegated to both  $\mathcal{U}_1$  and  $\mathcal{U}_2$ , the congruence  $Y_1 \equiv Y_2 \equiv y^{\alpha_3}$  holds by the OMTUP assumption. Furthermore, by the choice of  $T_1 \equiv t_1 Q_q, T_2 \equiv t_2 Q_q$ , we have the congruences

$$a_1 Q_{q_1} + T_1 \equiv t_1 \pmod{q_1}, \quad a_2 Q_{q_1} + T_2 \equiv t_2 \pmod{q_1}.$$

Then, together with the equality  $t = t_1 + t_2$  the following congruence holds:

$$\begin{aligned} (X_1 \pmod{r}) \cdot (X_2 \pmod{r}) &\equiv (x_1^{\alpha_1} \pmod{r}) \cdot (x_2^{\alpha_2} \pmod{r}) \\ &\equiv g_1^{t_1} \cdot g_1^{t_2} \pmod{r} \\ &\equiv g_1^{t_1+t_2} \pmod{r} \\ &\equiv g_1^t \pmod{r}. \end{aligned}$$

Hence, the result follows for the verification step. Then, the result follows by the congruences

$$a_1 Q_{q_1} + T_1 \equiv a_1 \pmod{q}, \quad a_2 Q_{q_1} + T_2 \equiv a_2 \pmod{q},$$

the equality  $a = a_1 + a_2$  and Lagrange's theorem

$$\begin{aligned}
 (X_1 \cdot X_2 \bmod p) \cdot (Y_1 \bmod r) &\equiv (x_1^{\alpha_1} \cdot x_2^{\alpha_2} \bmod p) \cdot (y^{\alpha_3} \bmod p) \\
 &\equiv (ug^t)^{a_1} \cdot (ug^t)^{a_2} \cdot g^{-at\gamma\gamma^{-1}} \bmod p \\
 &\equiv (ug^t)^{a_1+a_2} \cdot g^{-at} \bmod p \\
 &\equiv (ug^t)^a \cdot g^{-at} \bmod p \\
 &\equiv u^a \cdot g^{at} \cdot g^{-at} \bmod p \\
 &\equiv u^a \cdot g^{at-at} \bmod p \\
 &\equiv u^a \bmod p.
 \end{aligned}$$

*Security.* We argue that HideP satisfies security under the OMTUP-model due to the following observations:

1. On a single execution of  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  the input  $(\alpha, x)$  in the query sent by  $\mathcal{C}$  to the adversary  $\mathcal{A} = (\mathcal{U}_1, \mathcal{E})$  does not leak any information about  $u$ ,  $a$  and  $u^a$ . The reason is that
  - $u$  is randomized by multiplying with  $g^t$  which is random. Hence, the adversary  $\mathcal{A}$  cannot obtain any useful information about  $u$  even if the factors  $p, r$  of  $n$  are known,
  - $a$  is randomized by  $a_1$  and  $a_2$  and  $a\gamma$ . Hence  $\mathcal{A}$  cannot obtain any useful information about  $a$  by obtaining  $a_1$  through  $x_1$  and  $a\gamma \bmod p$  even if it knows the factors  $p, r$  and  $q, q_1$  of  $n$  and  $m$ , respectively.
  - To obtain useful information about  $u^a$ ,  $\mathcal{A}$  requires to know  $x_2$  which is random and not known by the OMTUP assumption.
2. Even if the adversary  $\mathcal{A}$  sees multiple executions of  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  wherein the inputs of  $\mathcal{C}$  are adversarially chosen,  $\mathcal{A}$  cannot obtain any useful information about the exponent  $a$  chosen by  $\mathcal{C}$ , and the desired exponentiation  $u^a$  in a new execution since logical divisions of  $a = a_1 + a_2$  at each execution involve freshly generated random elements. This implies that  $\epsilon_s = 0$  for the exponent  $a$  and the output  $u^a \bmod p$ . Assume that  $\mathcal{A}$  can break the secrecy of the base  $u$  with a non-negligible probability. In particular, it can obtain useful information about both elements  $u \cdot G_T$  and  $ug^t$  with a non-negligible probability, where  $G_T \equiv g^t \bmod p$  for some  $t$ . Then,  $\mathcal{A}$  can obtain  $\gcd((uG_T - ug^t), n)$ . This gives the factors  $p$  and  $r$  of  $n$  with a non-negligible probability as  $u \cdot G_T \equiv ug^t \bmod p$  holds. This implies that  $t_s = \text{poly}(\sigma)$  and  $\epsilon_s$  is at most  $\text{Adv}_{\mathcal{A}}^{\text{Fact}}(\sigma)$ , i.e.  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  is a secure implementation of HideP if the factorization problem is intractable .

In particular, these arguments show that  $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$  provides unconditional security for the exponent  $a$  and the output  $u^a$  against any (computationally unrestricted) adversary and security for the base  $u$  against any polynomially bounded adversary.



*Verifiability.* Since  $Y_1$  and  $Y_2$  both have the same base and exponent elements,  $\mathcal{U}_1$  cannot cheat the delegator  $\mathcal{C}$  by manipulating  $Y_1$  by the OMTUP assumption. This means that  $\mathcal{U}_1$  can only pass the verification step by manipulating the output  $X_1$ . Hence, the result  $\epsilon_p = 1/2 + \epsilon$  (where  $\epsilon$  is negligibly small in the security parameter  $\sigma$ ) holds for any adversary  $\mathcal{U}_1$  since  $\mathcal{U}_1$  needs to know the correct position of  $x_1$  which has at most  $1/2$ . We now show that if there exists an adversary  $\mathcal{A}$  that breaks the verifiability property with a non-negligible probability, then  $\mathcal{A}$  can be used to effectively solve the factorization problem. Assume now that  $\mathcal{U}_1$  as a malicious server passes the verification step with a bogus output  $Z_1$  (instead of  $X_1 = x_1^{\alpha_1}$ ) with a non-negligible probability. Then, the following congruence must hold for any arbitrary output  $X_2$  of the honest server  $\mathcal{U}_2$

$$Z_1 X_2 \equiv X_1 \cdot X_2 \equiv g_1^t \pmod r \tag{15}$$

with a non-negligible probability. This implies that  $\mathcal{U}_1$  can decide whether the congruence  $Z_1 \equiv X_1 \pmod r$  holds with a non-negligible probability. We note that  $Z_1 \not\equiv 0 \pmod r$  as otherwise Congruence 15 cannot hold with  $g_1^t \not\equiv 0 \pmod r$ . This implies that  $Z_1 - X_1 \equiv 0 \pmod r$  and that  $Z_1 \not\equiv 0 \pmod r$ . From the inequality  $Z_1 - X_1 < n$  (when the representatives are considered as integers), it follows that  $\mathcal{U}_1$  can compute  $\text{gcd}(Z_1 - X_1, n) = r$  with a non-negligible probability. Hence,  $\mathcal{U}_1$  can obtain information about both the factors  $p$  and  $r$  of  $n$  with a non-negligible probability. This implies that  $t_v = \text{poly}(\sigma)$  and  $\epsilon_v$  is at most  $\text{Adv}_{\mathcal{A}}^{\text{Fact}}(\sigma)$ .  $\square$

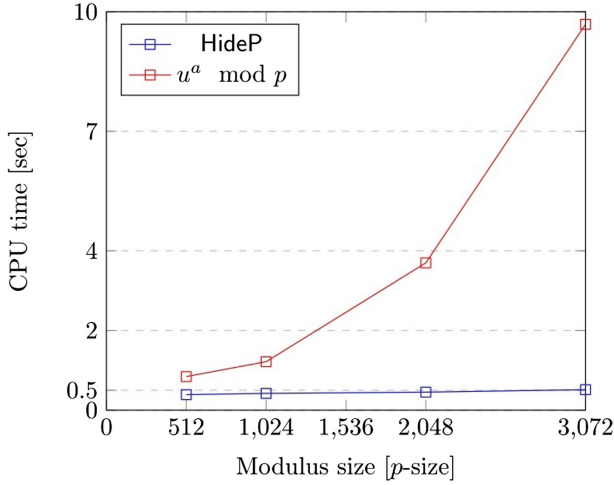
**Table 1.** Comparison of computational and communication costs for  $\mathcal{C}$ .

	Secret $p$	# MMs	# Servers	# Rounds	# Queries	Verifiability
[7] TC'05	no	509	2	1	8	1/2
[5] ESORICS'12	no	307	2	1	6	2/3
[13] ESORICS'14 ( $\chi = 2^{64}$ )	no	508	1	1	4	1/2
[8] IJIS'16 ( $c = 4$ )	no	200	1	2	60	9/10
[11] AsiaCCS'16	no	512	2	2	6	1
[9] INDOCRYPT'16	no	27	2	1	5	0
[14] IEEE'17 ( $b = 16$ )	yes	69	1	1	4	31/32
HideP	yes	24	2	1	4	1

## 5.2 Comparison

We now compare HideP with the previous delegation schemes for **MEs**. We denote by **MM** a modular multiplication, **MI** a modular inversion, and **MR** a modular reduction. Throughout the comparison we make the following assumptions:

- we regard 1 **MM** modulo  $n$  as  $\approx 4$  **MMs** modulo  $p$ ,
- 1 **MM** modulo  $p$  and 1 **MM** modulo  $r$  cost approximately the same amount of computation,



**Fig. 1.** CPU time: HideP vs. Computation

**Table 2.** CPU cost: HideP vs. Computation

p-size	CPU cost for $u^a \bmod p$		
	Delegation cost(ms)	Computing cost(ms)	Gain factor
512-bit	390	843	$\approx 2.16$
1024-bit	421	1216	$\approx 2.89$
2048-bit	452	3697	$\approx 8.18$
3072-bit	515	9684	$\approx 18.80$

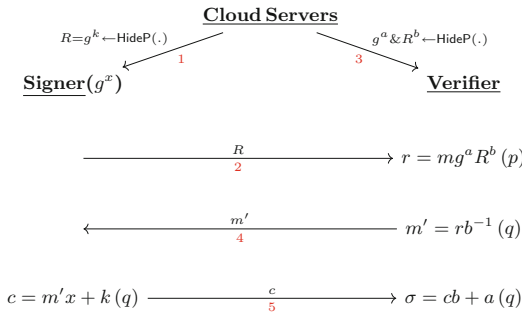
Experiments were conducted on a laptop with an Intel Core i5 2.6 GHz processor and 4 GB RAM. Results presented were taken out of 1000 iterations. The comparison is between the CPU time for HideP’s 24 MMs and local computation of a ME

- 1 MI is at worst 100 times slower than 1 MM (see [8]),
- we regard 1 MR costs approximately 1 MM (e.g. by means of Barret’s or Montgomery’s modular reduction techniques).

We give the delegator’s computational workload in Table 1 by considering the approximate number of MMs modulo  $p$ . In particular, Table 1 compares computational cost and communication overhead of HideP with the previous schemes. It shows that HideP has not only the best computational cost but requires also only a single round with 4 queries (instead of 2 rounds and 6 queries when compared with the only scheme in the literature satisfying full verifiability [11]) (Table 1).

## 6 Application: Verifiably Delegated Blind Signatures

Blind signatures were introduced by Chaum [3] and allow a user to obtain the signature of another user in such a way that the signer do not see the actual message to be signed and the user without having knowledge of the signing key is able to get the message signed with that key. Blind signatures are useful in privacy preserving protocols. For example, in e-cash scenario, a bank needs to sign blindly the coins withdrawn by its users. Normally, in blind signature protocols, both the signer and the verifier have to compute **MEs** using private and public keys, respectively. As an example, delegation of **MMs** in blinded Nyberg-Rueppel signature scheme [1,10] using HideP is depicted in Fig. 2. It is also evidenced from Fig.1 that the time taken by HideP is much smaller than that of directly computing  $u^a \bmod p$ , and this gain in CPU time increases rapidly with the size of the modulus. Hence, HideP becomes more attractive for resource-constrained scenario such as mobile environment when we go for higher security levels.



**Fig. 2.** Delegating blinded Nyberg-Rueppel signature

## 7 Conclusion

In this work, we addressed the problem of secure and verifiable delegation of **MEs**. We observed that two recent schemes [9,11] do not satisfy the claimed verifiability probabilities. We presented an efficient non-interactive fully verifiable secure delegation scheme HideP in the OMTUP-model by disguising the modulus  $p$  using CRT. In particular, HideP is the first non-interactive fully verifiable and the most efficient delegation scheme for modular exponentiations leveraging the properties of  $\mathbb{Z}_p$  via CRT. As future works, proposing an efficient fully verifiable delegation scheme without any requirement of online or offline computation of **MEs** by the delegator (or its impossibility) under the TUP/OUP assumptions could be highly interesting.

**Acknowledgement.** We thank the anonymous reviewers for their helpful comments on the previous version of the paper which led to improvements in the presentation of the paper.

## References

1. Asghar, N.: A survey on blind digital signatures. Technical report (2011)
2. Cavallo, B., Di Crescenzo, G., Kahrobaei, D., Shpilrain, V.: Efficient and secure delegation of group exponentiation to a single server. In: Mangard, S., Schaumont, P. (eds.) RFIDSec 2015. LNCS, vol. 9440, pp. 156–173. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-24837-0\\_10](https://doi.org/10.1007/978-3-319-24837-0_10)
3. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology, pp. 199–203. Springer, Boston, MA (1983). [https://doi.org/10.1007/978-1-4757-0602-4\\_18](https://doi.org/10.1007/978-1-4757-0602-4_18)
4. 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). [https://doi.org/10.1007/3-540-48071-4\\_7](https://doi.org/10.1007/3-540-48071-4_7)
5. Chen, X., Li, J., Ma, J., Tang, Q., Lou, W.: New algorithms for secure outsourcing of modular exponentiations. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 541–556. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-33167-1\\_31](https://doi.org/10.1007/978-3-642-33167-1_31)
6. Chevalier, C., Laguillaumie, F., Vergnaud, D.: Privately outsourcing exponentiation to a single server: cryptanalysis and optimal constructions. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9878, pp. 261–278. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-45744-4\\_13](https://doi.org/10.1007/978-3-319-45744-4_13)
7. Hohenberger, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 264–282. Springer, Heidelberg (2005). [https://doi.org/10.1007/978-3-540-30576-7\\_15](https://doi.org/10.1007/978-3-540-30576-7_15)
8. Kiraz, M.S., Uzunkol, O.: Efficient and verifiable algorithms for secure outsourcing of cryptographic computations. *Int. J. Inf. Sec.* **15**(5), 519–537 (2016). <https://doi.org/10.1007/s10207-015-0308-7>
9. Kuppusamy, L., Rangasamy, J.: CRT-based outsourcing algorithms for modular exponentiations. In: Dunkelman, O., Sanadhya, S.K. (eds.) INDOCRYPT 2016. LNCS, vol. 10095, pp. 81–98. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-49890-4\\_5](https://doi.org/10.1007/978-3-319-49890-4_5)
10. Nyberg, K., Rueppel, R.A.: Message recovery for signature schemes based on the discrete logarithm problem. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 182–193. Springer, Heidelberg (1995). <https://doi.org/10.1007/BFb0053434>
11. Ren, Y., Ding, N., Zhang, X., Lu, H., Gu, D.: Verifiable outsourcing algorithms for modular exponentiations with improved checkability. In: AsiaCCS 2016, pp. 293–303. ACM, New York (2016). <https://doi.org/10.1145/2897845.2897881>
12. Uzunkol, O., Rangasamy, J., Kuppusamy, L.: Hide The Modulus: a secure non-interactive fully verifiable delegation scheme for modular exponentiations via CRT (full version). IACR Cryptology ePrint Archive, Report 2018 (2018). <https://eprint.iacr.org/2018/644>
13. Wang, Y., Wu, Q., Wong, D.S., Qin, B., Chow, S.S.M., Liu, Z., Tan, X.: Securely outsourcing exponentiations with single untrusted program for cloud storage. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 326–343. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-11203-9\\_19](https://doi.org/10.1007/978-3-319-11203-9_19)
14. Zhou, K., Affi, M.H., Ren, J.: ExpSOS: secure and verifiable outsourcing of exponentiation operations for mobile cloud computing. *IEEE Trans. Inf. Forensics Sec.* **12**(11), 2518–2531 (2017). <https://doi.org/10.1109/TIFS.2017.2710941>



# Offline Assisted Group Key Exchange

Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang<sup>(✉)</sup>

NTNU, Norwegian University of Science and Technology, Trondheim, Norway  
{colin.boyd,gareth.davies,kristian.gjosteen,yao.jiang}@ntnu.no

**Abstract.** We design a group key exchange protocol with forward secrecy where most of the participants remain offline until they wish to compute the key. This is well suited to a cloud storage environment where users are often offline, but have online access to the server which can assist in key exchange. We define and instantiate a new primitive, a blinded KEM, which we show can be used in a natural way as part of our generic protocol construction. Our new protocol has a security proof based on a well-known model for group key exchange. Our protocol is efficient, requiring Diffie–Hellman with a handful of standard public key operations per user in our concrete instantiation.

**Keywords:** Authenticated key exchange  
Group key exchange · Forward secrecy · Cloud storage

## 1 Introduction

We consider the following collaboration scenario. Isabel would like to use a cloud storage provider to share some files with her collaborators Robin and Rolf. While Isabel and her collaborators have some level of trust in the cloud storage provider, they do not want the provider to be able to see the contents of their files. In other words, Isabel needs to share some secret key material with Robin and Rolf. This paper addresses the problem of sharing this secret key material.

There are a number of possible solutions. The simplest is for Isabel to encrypt the key material using public key encryption and send the ciphertexts to Robin and Rolf, who can then decrypt. However, this solution does not provide *forward secrecy*. If either Robin or Rolf’s decryption keys are compromised at any point in the future, the confidentiality of the key material is also compromised.

*Group key exchange* (GKE) can give us forward secrecy. However, Isabel and her collaborators will not be online all the time, and the time spent offline is non-trivial. If Isabel and her collaborators want to use a traditional GKE, then Isabel cannot share her files until every collaborator has been online. Likewise, the individual collaborators cannot look at the shared files until every other collaborator has been online. This is impractical, and no system that has interactions between the initiator and the responders can be practical in this setting.

In this paper, we propose a GKE protocol that provides forward secrecy and is non-interactive with respect to the sharing parties, hence suitable for our

collaboration scenario: Isabel comes online, runs her part of the GKE protocol, receives the key material and shares the files. As the individual collaborators come online, they run their part of the GKE protocol, receive the key material and get access to the shared files.

### 1.1 Secure Sharing and Forward Secrecy

The users in our collaboration scenario will be content to trust their cloud storage provider (CSP) to make their data available. Some users will be content to trust their CSP to use simple access control to prevent unauthorized access or modification. However, for many users such a convenient trust assumption regarding confidentiality or integrity is either unreasonable, legally impossible or otherwise undesirable. For this reason, many CSPs support (in addition to access control) the obvious solution of user-side encryption of data, where the CSP does not know the key material used for encryption and decryption<sup>1</sup>.

The use of encryption means that groups of users must establish shared key material in order to share data. This suggests group key exchange. However, group key exchange protocols are usually interactive, while in our collaboration scenario, Isabel's collaborators may not all be online at the same time, so completing the group key exchange would take too long, and until the key material was agreed upon, no work could be done.

We therefore desire *non-interactive* solutions that allow the initiator to complete their actions before any recipients come online, and do not require any interaction between the recipients. This rules out traditional group key exchange protocols [2, 3, 6, 14, 19].

The natural non-interactive solution is to use public key encryption (or perhaps other similar primitives, such as broadcast encryption). However, in the outsourced storage scenario, *forward secrecy* – compromise of long-term keys does not compromise previously completed sessions – is important. Forward secrecy is typically achieved through the use of interaction with Diffie–Hellman or other ephemeral keys. Using ephemeral keys for confidentiality and long-term keys only for authentication ensures that later release of long-term secrets does not reveal the session key.

Forward secrecy presents an inherent conflict with our requirement to have a non-interactive solution. Indeed, a simple generic argument implies that forward secrecy without interaction is impossible: without interaction the recipient cannot provide an ephemeral input and therefore the recipient's long-term key alone must be sufficient to recover the session key. Recent proposals have attempted to work around this argument in different ways. The first line of work, including the X3DH [24] and ART [9] protocols, insists that recipients upload some pre-keys to the CSP at some point before the initiator begins their activity. These pre-keys are then used as if they were ephemeral, however if one recipient never comes online then they could sit on the server indefinitely: this is a re-definition of ephemeral and long-term keys, as used by standard key exchange

---

<sup>1</sup> This practice is confusingly often called *zero knowledge* in commercial circles.

Protocol	Forward Secrecy	Non-Interactive	Security Proof	Efficient	Parties
X3DH [24]	✓ <sup>a</sup>	✗ <sup>d</sup>	✗	✓	2
ART [9]	✓ <sup>a</sup>	✗ <sup>d</sup>	✓	✓	$N$
ORTT KE [15,17,13]	✓ <sup>b</sup>	✓	✓	✗	2
GKE [6,2,19,3,14]	✓	✗	✓	✓	$N$
Mona [23], Tresorit [21,22]	✗	✓	✓	✓	$N$
Chu et al. [8]	✗	✓	✓	✓	$N$
This work	✓ <sup>c</sup>	✓	✓	✓	$N$

**Fig. 1.** Comparison of secure sharing protocols. <sup>a</sup> Re-defined ‘ephemeral keys’; <sup>b</sup> Re-defined ‘long-term keys’; <sup>c</sup> If the server honestly deletes all ephemeral data; <sup>d</sup> Users must upload pre-keys.

security models. Another approach, taken by Green and Miers [15] and further developed by Günther et al. [17] and Derler et al. [13], concerns so-called zero-round-trip-time (ORTT) key exchange. In this model, the long-term decryption key is updated (punctured) once the recipient comes online, in such a way that the crucial ciphertexts can no longer be decrypted by that (long-term) key. Thus the long-term key is no longer static but evolves over time. Forward secrecy with puncturable encryption relies crucially on the assumption that the protocol (single) message arrives at the receiver. Until that happens the receiver private key is not updated and so the encrypted data is vulnerable to receiver compromise. In addition we note that these works rely on less efficient cryptographic primitives and require increased storage and secure deletion properties at the receiver. Figure 1 summarizes selected existing literature on file-sharing protocols.

## 1.2 Contributions

In this paper, we run into two major obstacles. We need a group key exchange protocol that is non-interactive with respect to the initiator and the responders, and that at the same time provides forward secrecy.

We overcome these obstacles by noting that the cloud server is online at all times, and use ephemeral values provided by the cloud server to give us forward secrecy. This allows us to achieve the best possible level of forward secrecy in our collaboration scenario, without trusting the cloud server. Our protocol is simple and relies only on standard assumptions.

We regard the following as the main contributions of this paper.

- We propose a novel practical group key exchange protocol suitable for use in cloud storage. Our protocol is described in Sect. 5.
- We include a formal security analysis of our protocol in a strong security model with trust assumptions suited to the cloud scenario. The proof is in a security model which is detailed in Sect. 3.
- We introduce definitions and constructions for a new cryptographic primitive, blinded KEMs, which may find other applications. We describe this primitive and provide two secure constructions in Sect. 4.

## 2 Preliminaries

For a set  $S$ , denote  $x \stackrel{\$}{\leftarrow} S$  to mean choosing  $x$  uniformly at random from  $S$ . We write **return**  $b' \stackrel{?}{=} b$  as shorthand for **if**  $b' = b$  **then return** 1; **else return** 0, with an output of 1 indicating successful adversarial behavior.

### 2.1 Public-Key Encryption

A public-key encryption scheme  $\text{PKE} = (\text{KG}_{\text{pke}}, \text{Enc}, \text{Dec})$  with message space  $\mathcal{M}$  is defined as follows.  $\text{KG}_{\text{pke}}$  takes as input some security parameter(s), if any, and outputs a public encryption key  $pk$  and a secret decryption key  $sk$ .  $\text{Enc}$  takes a message  $m$  and produces a ciphertext  $c$  using  $pk$ :  $c \leftarrow \text{Enc}_{pk}(m)$ .  $\text{Dec}$  decrypts a ciphertext  $c$  using  $sk$  to recover  $m$  or in the case of failure a symbol  $\perp$ :  $m/\perp \leftarrow \text{Dec}_{sk}(c)$ . Correctness requires that  $m \leftarrow \text{Dec}_{sk}(\text{Enc}_{pk}(m))$  for all  $m \in \mathcal{M}$ .

We denote the usual advantage of an adaptive chosen ciphertext adversary  $\mathcal{A}$  against real-or-random security for the public-key encryption scheme by  $\text{Adv}_{\text{PKE}}^{\text{ror-cca2}}(\mathcal{A})$ . In our protocol's security proof, it is actually convenient to use a generalization of this notion, which we discuss in the full version [4].

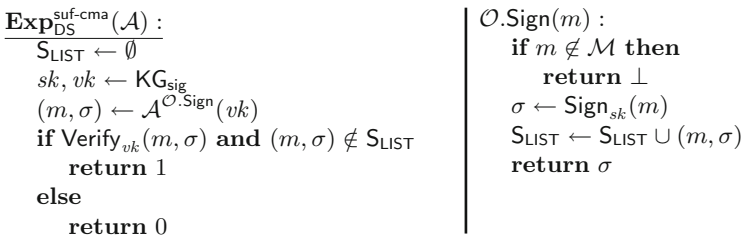
### 2.2 Digital Signatures

A signature scheme  $\text{DS} = (\text{KG}_{\text{sig}}, \text{Sign}, \text{Verify})$  with message space  $\mathcal{M}$  is defined as follows.  $\text{KG}_{\text{sig}}$  takes as input some security parameter(s), if any, and outputs a signing key  $sk$  and a public verification key  $vk$ .  $\text{Sign}$  creates a signature  $\sigma$  on a message  $m$ :  $\sigma \leftarrow \text{Sign}_{sk}(m)$ .  $\text{Verify}$  verifies that the signature on the message is in fact valid:  $0/1 \leftarrow \text{Verify}_{vk}(m, \sigma)$ , with 1 indicating successful verification. Correctness requires that  $\text{Verify}_{vk}(m, \text{Sign}_{sk}(m)) = 1$  for all  $m \in \mathcal{M}$ .

**Definition 1.** Let  $\text{DS} = (\text{KG}_{\text{sig}}, \text{Sign}, \text{Verify})$  be a signature scheme. Then the *suf-cma advantage* of an adversary  $\mathcal{A}$  against  $\text{DS}$  is defined as

$$\text{Adv}_{\text{DS}}^{\text{suf-cma}}(\mathcal{A}) = \Pr[\text{Exp}_{\text{DS}}^{\text{suf-cma}}(\mathcal{A}) = 1].$$

where the experiment  $\text{Exp}_{\text{DS}}^{\text{suf-cma}}(\mathcal{A})$  is given in Fig. 2.



**Fig. 2.** The experiment defining *suf-cma* security for signature schemes.



Note that in the existential unforgeability under chosen message attack (**euf-cma**) game the list  $S_{LIST}$  only keeps track of the messages queried by the adversary during the **Sign** queries phase, so  $\mathcal{A}$  is not allowed to output  $(m, \sigma_2)$  if she sent  $m$  to  $\mathcal{O}.\text{Sign}$  and received  $\sigma_1$ .

### 2.3 Hardness Assumptions

**Definition 2.** Fix a cyclic group  $\mathbb{G}$  of prime order  $q$  with generator  $g$ . The advantage of an algorithm  $\mathcal{A}$  solving the Decision Diffie-Hellman (DDH) problem for  $\mathbb{G}$  and  $g$  is

$$\text{Adv}_{\mathbb{G}}^{\text{DDH}}(\mathcal{A}) = 2 \left| \Pr[\text{Exp}_{\mathbb{G}}^{\text{DDH}}(\mathcal{A}) = 1] - \frac{1}{2} \right|$$

where the experiment  $\text{Exp}_{\mathbb{G}}^{\text{DDH}}(\mathcal{A})$  is given in Fig. 3.

$\text{Exp}_{\mathbb{G}}^{\text{DDH}}(\mathcal{A}) :$

```

 $b \xleftarrow{\$} \{0, 1\}$ 
 $x, y, z \xleftarrow{\$} \mathbb{Z}_q$ 
if  $b = 1$ 
     $c \leftarrow g^{xy}$ 
else
     $c \leftarrow g^z$ 
 $b' \leftarrow \mathcal{A}(g^x, g^y, c)$ 
return  $b' \stackrel{?}{=} b$ 
    
```

**Fig. 3.** DDH experiment.

$\text{Exp}_{\mathcal{F}}^{\text{CR}}(\mathcal{A}) :$

```

 $f \xleftarrow{\$} \mathcal{F}$ 
 $x, y \leftarrow \mathcal{A}(f)$ 
if  $x \neq y \wedge f(x) = f(y)$  then
    return 1
else
    return 0
    
```

**Fig. 4.** Collision resistance experiment.

**Definition 3.** Let  $\mathcal{F}$  be a family of functions. The collision resistance advantage of an adversary  $\mathcal{A}$  running in time  $t$  is

$$\text{Adv}_{\mathcal{F}}^{\text{CR}}(\mathcal{A}) = \left| \Pr[\text{Exp}_{\mathcal{F}}^{\text{CR}}(\mathcal{A}) = 1] \right|$$

where the experiment  $\text{Exp}_{\mathcal{F}}^{\text{CR}}(\mathcal{A})$  is given in Fig. 4.

Note that in an abuse of notation, we sometimes write  $\text{Adv}_f^{\text{CR}}(\mathcal{A})$ , with the understanding that the function family  $\mathcal{F}$  exists and that the choice of a function  $f$  is done at some point.

## 3 GKE Protocol Model

The model described in this section is based on previous models for group key exchange such as those of Katz and Yung [19] and Bresson and Manulis [5]. This includes game-based security definitions.

### 3.1 Communication Model

A GKE protocol  $\mathcal{P}$  is a collection of probabilistic algorithms that determines how oracles of the principals behave in response to signals (messages) from their environment.

*Protocol Participants and Long-Lived Keys.* Each principal  $V$  in the protocol is either a user  $U$  or a server  $S$ . In every session, each user may act as either an initiator  $I$  or a responder  $R$ . Each principal  $V$  holds long-term secret keys, and corresponding public keys of all principals are known to all.

*Session Identifiers and Partner Identifiers.* Protocol principals maintain multiple instances, or sessions, that may be run simultaneously and we denote a session of principal  $V$  by the oracle  $\prod_V^\alpha$  with  $\alpha \in \mathbb{N}$ .

Each oracle  $\prod_V^\alpha$  is associated with the variables  $\text{status}_V^\alpha$ ,  $\text{role}_V^\alpha$ ,  $\text{pid}_V^\alpha$ ,  $\text{sid}_V^\alpha$ ,  $k_V^\alpha$  as follows:

- $\text{status}_V^\alpha$  takes a value from  $\{\text{unused}, \text{ready}, \text{accepted}, \text{rejected}\}$ .
- $\text{role}_V^\alpha$  takes a value from:  $S, I, R$ .
- $\text{pid}_V^\alpha$  contains a set of principals.
- $\text{sid}_V^\alpha$  contains a string defined by the protocol.
- $k_V^\alpha$  the agreed session key (if any).

A session identifier, denoted  $\text{sid}$ , is a protocol-defined value stored at a principal intended to provide a link to other sessions in the same protocol run. A set of partner identifiers, denoted  $\text{pid}$ , contains the identities of all intended users in a session.

Each oracle  $\prod_V^\alpha$  is *unused* until initialization, by which it is told to act as a server or a user together with the long term secret keys. During initialization all oracles begin with  $\text{status}_V^\alpha = \text{ready}$  and  $\text{role}_V^\alpha$ ,  $\text{pid}_V^\alpha$ ,  $\text{sid}_V^\alpha$  and  $k_V^\alpha$  all equal to  $\perp$ .

*Executing the Protocol.* After the protocol starts, each oracle  $\prod_V^\alpha$  learns its partner identifier  $\text{pid}_V^\alpha$  and sends, receives and processes messages.

If the protocol at oracle  $\prod_V^\alpha$  *fails*, for example if signature verification or key confirmation fails, then the oracle changes its state to *rejected* and no longer responds to protocol messages. Otherwise, if  $V$  is a user, after computing  $k_V^\alpha$  oracle  $\prod_V^\alpha$  changes its state to *accepted* and no longer responds to protocol messages, and if  $V$  is the server, oracle  $\prod_V^\alpha$  accepts after all responder oracles get their messages or expiration.

### 3.2 Security Notions

*Adversarial Model.* An efficient adversary  $\mathcal{A}$  interacts with sessions by using the set of queries defined below. This models the ability of  $\mathcal{A}$  to completely control the network, deciding which instances run and obtaining access to other useful information. The **Test** query can only be asked once by  $\mathcal{A}$  and is only used to measure adversary's success; it does not correspond to any actual adversary's ability.

- **Execute**( $\mathcal{S}$ ): Input a set of unused oracles  $\mathcal{S}$  which execute an honest run of the protocol. The oracles compute what the protocol specifies and returns the output messages.
- **Send**( $\Pi_V^\alpha, m$ ): Sends message  $m$  to oracle  $\Pi_V^\alpha$ . The oracle computes what the protocol defines, and sends back the output message (if any), together with the status of  $\Pi_V^\alpha$ .
- **Corrupt**( $V$ ): Outputs principal  $V$ 's long-term secret key.
- **Reveal**( $\Pi_V^\alpha$ ): Outputs session key  $k_V^\alpha$  if oracle  $\Pi_V^\alpha$  has accepted and holds some session key  $k_V^\alpha$ .
- **Test**( $\Pi_V^\alpha$ ): If oracle  $\Pi_V^\alpha$  has status *accepted*, holding a session key  $k_V^\alpha$ , then a bit  $b$  is randomly chosen and this query outputs the session key  $k_V^\alpha$  if  $b = 1$ , or a random string from the session key space if  $b = 0$ .

*Partnering.* A secure GKE protocol should ensure that the session key established in an oracle  $\Pi_V^\alpha$  is independent of session keys established in other sessions, except for the partners of  $\Pi_V^\alpha$ . This is modeled by allowing the adversary to reveal any session key except the one in the **Test** session and its partners. Informally, partnering is defined in such a way that oracles who are supposed to agree on the shared session key are partners.

**Definition 4.** *Two oracles  $\Pi_V^\alpha$  and  $\Pi_W^\beta$  are partners if  $pid_V^\alpha = pid_W^\beta$  and  $sid_V^\alpha = sid_W^\beta$ .*

*Freshness.* The notion of freshness models the conditions on the adversary's behaviour that are required to prevent trivial wins.

**Definition 5.** *An oracle  $\Pi_V^\alpha$  is fresh if neither this oracle nor any of its partnered oracles have been asked a **Reveal** query, and either*

- *no server player nor any player in  $pid_V^\alpha$  was corrupted before every partnered oracle reached status *accepted*; or*
- *no player in  $pid_V^\alpha$  is ever corrupted.*

*Security Game.* Bringing together everything we have introduced so far, we can describe the game that allows us to measure the advantage of an adversary against a GKE protocol.

**Definition 6.** *Let  $P$  be a GKE protocol. The game  $\mathbf{Exp}_P^{\text{ake}}(\mathcal{A})$  consists of the following three phases:*

- **Initialization.** *Each principal  $V$  runs the key generation algorithm to generate long-term key pairs. The secret keys are only known to the principal, while public keys are revealed to every principal and the adversary.*
- **Queries.** *The adversary  $\mathcal{A}$  is allowed to make **Execute**, **Send**, **Reveal**, **Corrupt**, and **Test** queries. During this phase,  $\mathcal{A}$  is only allowed to ask only one **Test** query to a fresh oracle, which should remain fresh until the end of this phase.*
- **Guessing.**  *$\mathcal{A}$  outputs its guess  $b'$ .*

The output of the game is 1 if  $b = b'$ , otherwise 0.

The advantage of the adversary  $\mathcal{A}$  against the  $\text{ake}$ -security of  $\mathsf{P}$  is

$$\text{Adv}_{\mathsf{P}}^{\text{ake}}(\mathcal{A}) = 2 \left| \Pr[\text{Exp}_{\mathsf{P}}^{\text{ake}}(\mathcal{A}) = 1] - 1/2 \right|.$$

## 4 Blinded KEM

The concept of using public-key encryption to transport keys for use in symmetric encryption is by now well studied [1, 10–12, 18, 20]. This primitive is known as a *key encapsulation mechanism* (KEM) and is used in conjunction with a *data encapsulation mechanism* (DEM) that models some symmetric encryption scheme. This KEM-DEM framework is widely deployed in internet protocols, however – as we mentioned earlier – it does not provide any forward secrecy. The cloud scenario allows the initiator to store the encapsulated key and the DEM ciphertext in some repository for the recipient to later retrieve, but we ask: can the (untrusted) cloud give us some notion of forward secrecy of the key that the initiator wishes to transport?

It is well known how to turn a KEM into a key exchange protocol. We shall introduce a new primitive, which we call *blinded KEM*, and in the next section we will explain how to turn such a primitive into a group key exchange protocol suitable for our purposes.

Compared to a traditional KEM, a blinded KEM has two additional algorithms: a blinding algorithm takes some encapsulation<sup>2</sup> and adds a blinding value, and an unblinding algorithm (that requires an unblinding key created by the blinding algorithm) removes this blinding value from the blinded key. Note that this construction does not generalize existing KEMs since our decapsulation procedure works on blinded encapsulations rather than encapsulations.

The point of this new idealized primitive is to allow parties to safely outsource decapsulation by creating a blinded encapsulation, having someone else decapsulate and then unblinding the result. With careful key management, this idea will give us forward secrecy in our cloud scenario. We will develop this idea into a group key exchange protocol in the next section.

The concept of blinding is best known in the context of blind signatures, but have been used extensively in many areas of cryptography. It has also been used in the context of blind decryption [16, 25], and some of the schemes are quite similar to our constructions, even though they have very different applications in mind and also different security requirements.

After providing a definition of this primitive’s algorithms, we give two natural constructions (based on DH and RSA).

**Definition 7.** A blinded key encapsulation mechanism (blinded KEM) BKEM consists of five algorithms ( $\text{KG}_{\text{BKEM}}$ , Encap, Blind, Decap, Unblind). The key generation algorithm  $\text{KG}_{\text{BKEM}}$  outputs an encapsulation key  $ek$  and a decapsulation

<sup>2</sup> We abuse nomenclature throughout the rest of the paper and use ‘encapsulation’ to refer to a key encapsulation that is yet to be blinded.

key  $dk$ . The encapsulation algorithm **Encap** takes as input an encapsulation key and outputs an encapsulation  $C$  and a key  $k \in \mathcal{G}$ . The blinding algorithm takes as input an encapsulation key and an encapsulation and outputs a blinded encapsulation  $\tilde{C}$  and an unblinding key  $uk$ . The decapsulation algorithm **Decap** takes a decapsulation key and a (blinded) encapsulation as input and outputs a (blinded) key  $k$ . The unblinding algorithm takes as input an unblinding key and a blinded key and outputs a key.

The algorithms satisfy the correct decapsulation requirement: When  $(ek, dk) \leftarrow \text{KG}_{\text{BKEM}}, (C, k) \leftarrow \text{Encap}_{ek}, (\tilde{C}, uk) \leftarrow \text{Blind}_{ek}(C)$  and  $\tilde{k} \leftarrow \text{Decap}_{dk}(\tilde{C})$ , then

$$\text{Unblind}_{uk}(\tilde{k}) = k.$$

**Definition 8.** Let  $\text{BKEM} = (\text{KG}_{\text{BKEM}}, \text{Encap}, \text{Blind}, \text{Decap}, \text{Unblind})$  be a blinded KEM. The distinguishing advantage of any adversary  $\mathcal{A}$  against BKEM getting  $r$  blinded decapsulation samples is

$$\text{Adv}_{\text{BKEM}}^{\text{ind}}(\mathcal{A}, r) = 2 \left| \Pr[\text{Exp}_{\text{BKEM}}^{\text{ind}}(\mathcal{A}, r) = 1] - 1/2 \right|,$$

where the experiment  $\text{Exp}_{\text{BKEM}}^{\text{ind}}(\mathcal{A}, r)$  is given in Fig. 5.

```

ExpBKEMind( $\mathcal{A}, r$ ) :
   $b \xleftarrow{\$} \{0, 1\}$ 
   $(ek, dk) \leftarrow \text{KG}_{\text{BKEM}}$ 
   $(C, k_1) \leftarrow \text{Encap}_{ek}$ 
   $k_0 \xleftarrow{\$} \mathcal{G}$ 
  for  $j \in \{1, \dots, r\}$  do
     $(\tilde{C}_j, uk_j) \leftarrow \text{Blind}_{ek}(C)$ 
     $\tilde{k}_j \leftarrow \text{Decap}_{dk}(\tilde{C}_j)$ 
   $b' \leftarrow \mathcal{A}(ek, C, k_b, \{(\tilde{C}_j, \tilde{k}_j)\}_{1 \leq j \leq r})$ 
  return  $b' \stackrel{?}{=} b$ 
    
```

**Fig. 5.** Indistinguishability experiment  $\text{Exp}_{\text{BKEM}}^{\text{ind}}(\mathcal{A}, r)$  for a blinded KEM.

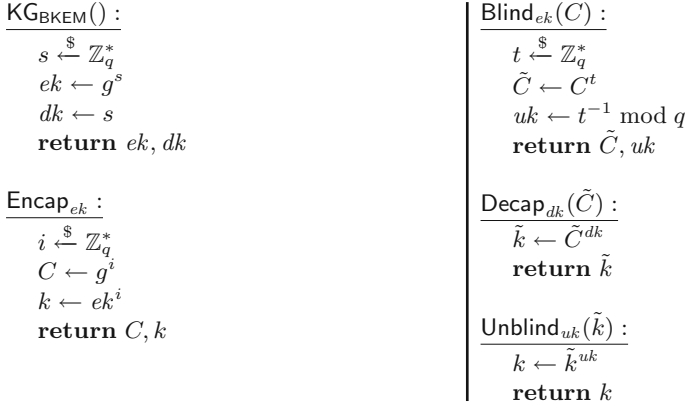
**Definition 9.** Let  $ek$  be any public key and let  $C_0$  and  $C_1$  be two encapsulations. Define  $X_0$  and  $X_1$  to be the statistical distribution of the blinded encapsulation output by  $\text{Blind}_{ek}(C_0)$  and  $\text{Blind}_{ek}(C_1)$ , respectively. We say that the blinded KEM is  $\epsilon$ -blind if the statistical distance of  $X_0$  and  $X_1$  is at most  $\epsilon$ .

**Definition 10.** Let  $ek$  be any public key and let  $C$  be an encapsulation of the key  $k$ . Let  $\tilde{C}$  be a blinded encapsulation of  $C$  with corresponding unblinding key  $uk$ . We say that the blinded KEM is rigid if there is exactly one  $\tilde{k}$  such that  $\text{Unblind}_{uk}(\tilde{k}) = k$ .

We now present two instantiations of blinded KEMs based on well-known hardness assumptions, namely DDH and the RSA problem.

### 4.1 Construction I: DH-Based

We consider the following Diffie-Hellman-based blinded KEM (DH-BKEM). Let  $\mathbb{G}$  be a group of prime order  $q$  with generator  $g$  and define DH-BKEM in Fig. 6.



**Fig. 6.** Diffie-Hellman-based blinded KEM (DH-BKEM).

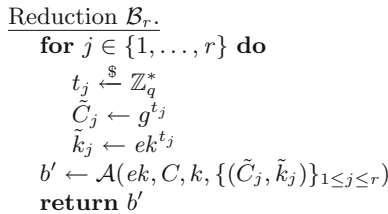
**Theorem 1.** DH-BKEM is a 0-blind BKEM and is rigid. Furthermore, let  $\mathcal{A}$  be any adversary against the above construction getting  $r$  blinded decapsulation samples. Then there exists an adversary  $\mathcal{B}_r$  against DDH such that

$$\text{Adv}_{\text{DH-BKEM}}^{\text{ind}}(\mathcal{A}, r) \leq \text{Adv}_{\mathbb{G}}^{\text{DDH}}(\mathcal{B}_r).$$

The running time of  $\mathcal{B}_r$  is essentially the same as the running time of  $\mathcal{A}$ .

*Proof.* For any encapsulation, since  $t$  is a random number, the blinded encapsulation  $\tilde{C}$  output by Blind is uniformly distributed on  $\mathbb{G}$ . It follows that the construction is 0-blind. In a similar vein, the unblinding procedure is a permutation on the keyspace so the construction is rigid.

Next, consider a tuple  $(ek, C, k)$ . The reduction  $\mathcal{B}_r$  is given in Fig. 7. In the event that  $(ek, C, k)$  is a DDH tuple, then  $\mathcal{B}_r$  perfectly simulates the input of  $\mathcal{A}$  in  $\text{Exp}_{\text{DH-BKEM}}^{\text{ind}}(\mathcal{A}, r)$  when  $b = 1$ . Otherwise,  $\mathcal{B}_r$  perfectly simulates the input of  $\mathcal{A}$  in  $\text{Exp}_{\text{DH-BKEM}}^{\text{ind}}(\mathcal{A}, r)$  when  $b = 0$ . The claim follows.



**Fig. 7.** DDH adversary  $\mathcal{B}_r$  playing  $\text{Exp}_{\mathbb{G}}^{\text{DDH}}(\mathcal{B}_r)$ , used in the proof of Theorem 1.

## 4.2 Construction II: RSA-Based

We consider the following RSA-based blinded KEM (RSA-BKEM). Unlike the above DH-based blinded KEM, this is less suitable for use in key exchange, since generating RSA keys is quite expensive. The scheme needs a hash function  $H_{\text{RSA-BKEM}}$ , and is detailed in Fig. 8.

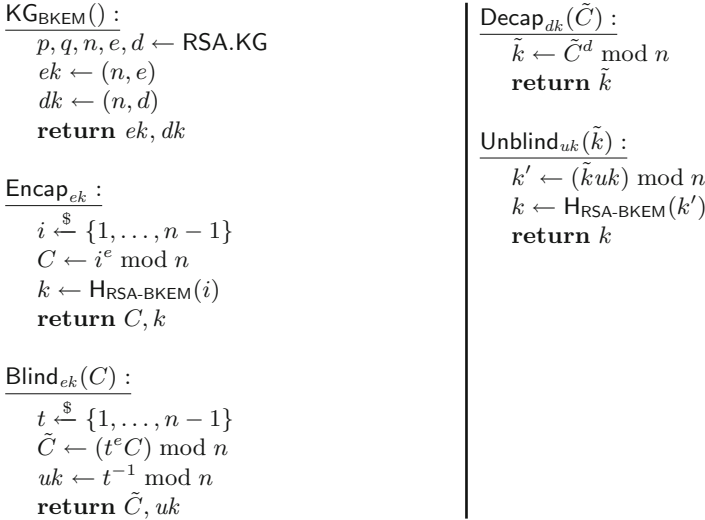


Fig. 8. RSA-based blinded KEM (RSA-BKEM).

Just like for the DH-based construction, this scheme is a blinded KEM, it is 0-blind and any adversary against indistinguishability in the random oracle model can be turned into an adversary against the RSA problem, in a straightforward way. We omit the proof. Note that this construction is not rigid since any hash collision provides two different values that map to the same  $k$ . (Dealing with this would complicate the security proof for little gain.)

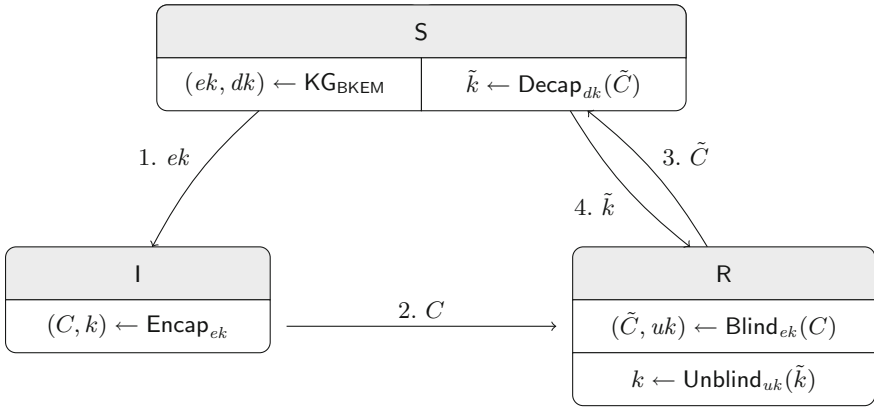
## 5 Offline Assisted Group Key Exchange Protocol

We now describe a generic protocol for cloud-assisted group key exchange using a blinded KEM, and then give a concrete instantiation using our DH-based blinded KEM from Sect. 4.1. Our scenario consists of the following participants:

- The *initiator* wants to establish a shared key  $k$  with a set of responders. First, the initiator  $I$  interacts with the server, then the initiator generates a key and “invitation messages” for the responders  $R_1, \dots, R_n$ .
- Each *responder* wants to allow the initiator to establish a shared key with him. When responder  $R_i$  gets their “invitation message” from the initiator, they will interact with the server to decrypt the shared key.

- The *server* temporarily stores information assisting in the computation of the shared secret key  $k$ , until every responder has gotten the key.

A conceptual overview of our construction is given in Fig. 9: the numbering indicates the order in which the phases of the protocol are done. A more diagrammatic overview is provided for the single-responder case in Fig. 10, and the general case is presented in Fig. 11. In these figures and for the rest of this section we will reduce notational overload by writing  $\text{Sign}_{R_j}$  instead of  $\text{Sign}_{sk_{R_j}}$  (and  $\text{Enc}_{R_j}$  instead of  $\text{Enc}_{pk_{R_j}}$  etc.), and allow the reader to infer which type of key is being used from the algorithm in use.



**Fig. 9.** Diagram describing how the group key exchange protocol uses the blinded KEM to do key exchange in the single responder case. For clarity, identities, nonces, session identifiers, key confirmation, public key encryption and digital signatures are omitted. Figure 10 contains a more detailed message sequence chart for the single responder case.

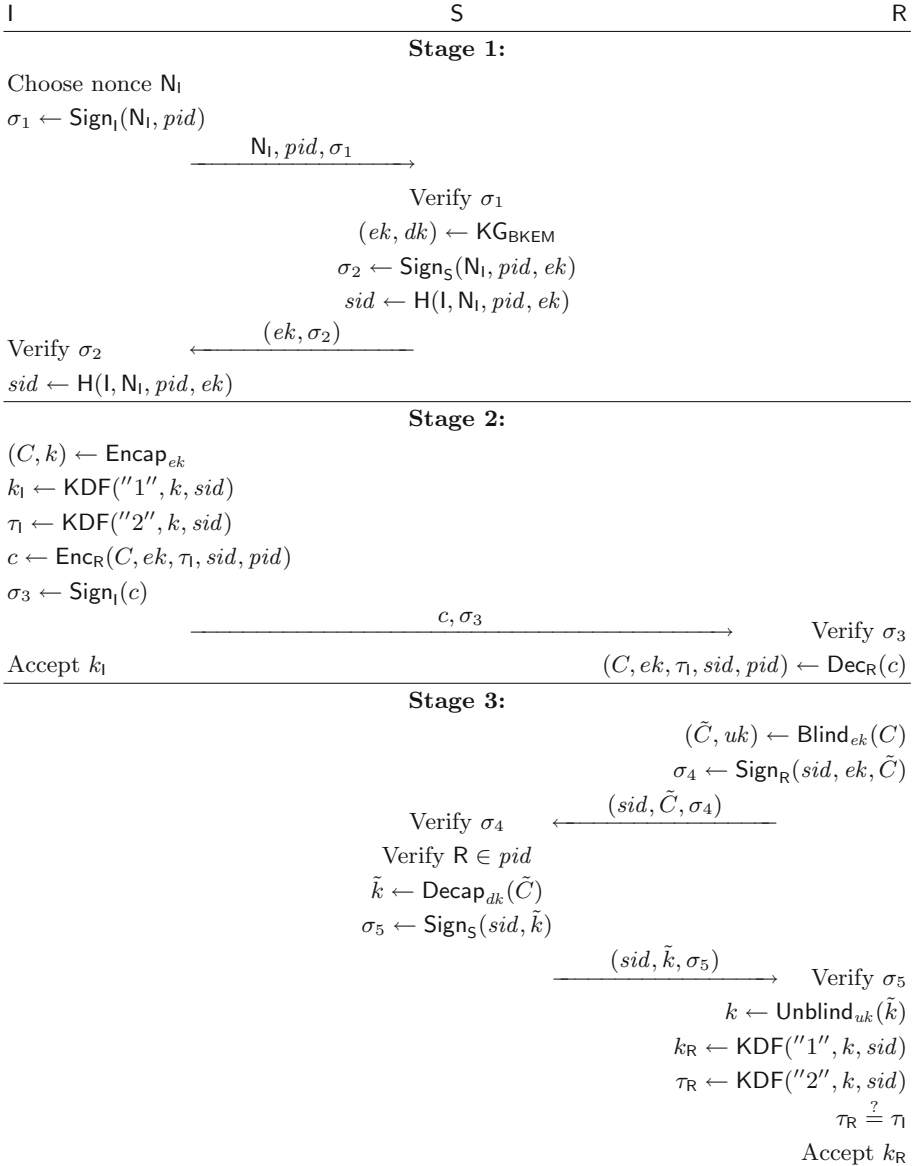
**Definition 11.** An *Offline Assisted Group Key Exchange Protocol (OAGK)* is defined in Fig. 11 and is parameterized by the following components. Let

- $\text{BKEM} = (\text{KG}_{\text{BKEM}}, \text{Encap}, \text{Blind}, \text{Decap}, \text{Unblind})$  be a blinded KEM,
- $\text{DS} = (\text{KG}_{\text{sig}}, \text{Sign}, \text{Verify})$  be a signature scheme,
- $\text{PKE} = (\text{KG}_{\text{pke}}, \text{Enc}, \text{Dec})$  be a public-key encryption scheme,
- $H$  be a hash function,
- $\text{KDF}$  be a key derivation function.

Note that in our model, we do not have a reveal state query, so there is no need to explicitly erase state information. In a real implementation, making sure that ephemeral and medium-term key material is erased at appropriate times is vital.

In order to break our protocol an adversary must compromise both the server and one of the users. The server stores a medium-term key which is deleted after the protocol run is complete (or after a time-out) after which compromise of the server is allowed. We note that it would not be difficult to enhance our protocol with *forward secure encryption* [7] if receiver compromise is deemed a likely risk.





**Fig. 10.** Message sequence chart for the OAGK protocol with a single responder R. Figure 11 contains a complete protocol description for the multi-responder case.

## 5.1 Efficiency

There are different ways to measure the efficiency of group key exchange protocols, including the number of protocol messages, the number of rounds of parallel messages, and the (average) computation per user. There exist theoretically

I running oracle  $\prod_I^\alpha$  as initiator on input  $pid$ :

1. Choose random  $N_1$ .
2.  $\sigma_1 \leftarrow \text{Sign}_I(N_1, pid)$ .
3. Send  $(N_1, pid, \sigma_1)$  to S.
10. Get  $(ek, \sigma_2)$  from S.
11. Verify that  $\sigma_2$  is S's signature on  $(N_1, pid, ek)$ .
12.  $sid \leftarrow H(l, N_1, pid, ek)$ .
13.  $(C, k) \leftarrow \text{Encap}_{ek}$ .
14. Session key  $k_1^\alpha \leftarrow \text{KDF}("1", k, sid)$
15. Key confirmation:  
 $\tau_1^\alpha \leftarrow \text{KDF}("2", k, sid)$
16. For every responder  $R_j$  in  $pid$ , do:
  - (a)  $c_j \leftarrow \text{Enc}_{R_j}(C, ek, \tau_1^\alpha, sid, pid)$ .
  - (b)  $\sigma_{3,j} \leftarrow \text{Sign}_I(c_j)$ .
  - (c) Send  $(c_j, \sigma_{3,j})$  to  $R_j$ .
17. Output  $k_1^\alpha$ .

$R_j$  running oracle  $\prod_{R_j}^\nu$  as responder on message  $(c_j, \sigma_{3,j})$  from I:

18. Verify that  $c_j$  is I's signature on  $\sigma_{3,j}$ .
19.  $(C, ek, \tau_1^\alpha, sid, pid) \leftarrow \text{Dec}_{R_j}(c_j)$ .
20.  $(\tilde{C}_j, uk_j) \leftarrow \text{Blind}_{ek}(C)$ .
21.  $\sigma_4 \leftarrow \text{Sign}_{R_j}(sid, ek, \tilde{C}_j)$ .
22. Send  $(sid, \tilde{C}_j, \sigma_4)$  to S.
32. Get  $(sid, \tilde{k}_j, \sigma_5)$  from S.
33. Verify that  $\sigma_5$  is S's signature on  $(sid, \tilde{k}_j)$ .
34.  $k_j \leftarrow \text{Unblind}_{uk_j}(\tilde{k}_j)$ .
35. Session key:  $k_{R_j}^\nu \leftarrow \text{KDF}("1", k_j, sid)$
36. Key confirmation:
 

$\tau_{R_j}^\nu \leftarrow \text{KDF}("2", k_j, sid)$

**If**  $\tau_{R_j}^\nu = \tau_1^\alpha$  **then**  
 Accept and output  $k_{R_j}^\nu$ .

**else**  
 Reject.

Phase I of S running oracle  $\prod_S^\beta$  as server on message  $(N_1, pid, \sigma_1)$  from I:

4. Verify that  $\sigma_1$  is I's signature on  $(N_1, pid)$ .
5.  $(ek, dk) \leftarrow \text{KGBKEM}$ .
6.  $\sigma_2 \leftarrow \text{Sign}_S(N_1, pid, ek)$ .
7.  $sid \leftarrow H(l, N_1, pid, ek)$ .
8. Store  $(sid, l, pid, dk, \emptyset)$ .
9. Send  $(ek, \sigma_2)$  to I.

Phase II of S running oracle  $\prod_S^\beta$  as server on message  $(sid, \tilde{C}_j, \sigma_4)$  from  $R_j$ , with stored state  $(sid, l, pid, dk, T)$ :

23. Lock the state  $(sid, \dots)$  until done.
24. Verify that  $\sigma_4$  is  $R_j$ 's signature on  $(sid, ek, \tilde{C}_j)$ .
25. Verify that  $R_j \in pid$ .
26. Verify that  $R_j \notin T$ .
27.  $\tilde{k}_j \leftarrow \text{Decap}_{dk}(\tilde{C}_j)$ .
28.  $\sigma_5 \leftarrow \text{Sign}_S(sid, \tilde{k}_j)$ .
29. Send  $(sid, \tilde{k}_j, \sigma_5)$  to  $R_j$ .
30. Let  $T' = T \cup \{R_j\}$ .
31. Update the state  $(sid, \dots, T)$  to  $(sid, \dots, T')$ .

**Fig. 11.** The three roles of the group key exchange protocol. Suppose  $\{R_j\}_{j \in J}$  are the identities of users that I wishes to share a common session key with ( $pid = l|\{R_j\}_{j \in J}$ ). Note that the line numbering indicates the order in which the lines of the various roles are reached during a protocol execution.

efficient examples [2,3] but most practical protocols employ a generalisation of the Diffie–Hellman protocol. One such generalisation is the well-known scheme of Burmester and Desmedt [6] which requires 2 rounds of communication and 3 exponentiations per user in its unauthenticated version.

An example of a modern optimised protocol is that of Gao et al. [14] which adds signatures to all messages and requires users to verify the signature on broadcast messages from all other users. In comparison our requirements are relatively modest. We require 3 rounds but do not use broadcast messages at all. The protocol participants perform 5 public key operations each, consisting of signature generation/verification, public key encryption/decryption and key encapsulation/decapsulation. As mentioned, the non-interactive nature of our scenario means that we wish for the initiator to be able to do all of their interaction during some initial phase.

## 5.2 Protocol Security

An adversary against the GKE protocol OAGK plays the game defined in Sect. 3.2. We need to give a useful bound for its advantage.

**Theorem 2.** *Consider an adversary  $\mathcal{A}$  against the GKE protocol OAGK running with  $n$  users, having at most  $s$  sessions, each involving at most  $r$  responders. Then adversaries  $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$  and  $\mathcal{B}_4$  exist, running in essentially the same time as  $\mathcal{A}$ , such that*

$$\begin{aligned} \mathbf{Adv}_{\text{OAGK}}^{\text{ake}}(\mathcal{A}) &\leq \mathbf{Adv}_{\text{H}}^{\text{CR}}(\mathcal{B}_0) + (n+1)\mathbf{Adv}_{\text{DS}}^{\text{suf-cma}}(\mathcal{B}_1) \\ &\quad + snr\mathbf{Adv}_{\text{PKE}}^{\text{ror-cca2}}(\mathcal{B}_2) + s\mathbf{Adv}_{\text{KDF}}^{\text{CR}}(\mathcal{B}_3) + sr\epsilon \\ &\quad + s\mathbf{Adv}_{\text{BKEM}}^{\text{ind}}(\mathcal{B}_4, r) \\ &\quad + \text{negligible terms.} \end{aligned}$$

We sketch the ideas used in the proof. We need to guess which session the adversary is going to issue the **Test** query for. If we guess correctly, the game proceeds unchanged. If we guess incorrectly, the game immediately stops, we flip a coin  $b'$  and pretend that the adversary output  $b'$ . It is clear that the adversary's advantage in this game is now  $1/s$  times the original advantage.

We must also handle the situation where the adversary issues a corruption query that would render our chosen session non-fresh. In this case, the game immediately stops, we flip a coin  $b'$  and pretend that the adversary output  $b'$ . Observe that if we stop for this reason, the adversary could not issue a test query (and our chosen session is now the only session a test query could be issued for), so the adversary would have no information about  $b$ . The probability that the adversary guesses  $b$  correctly is therefore unchanged.

Depending on when the server is corrupted (if it is corrupted at all), we need to bound the adversary's advantage in slightly different ways. An upper bound on the adversary's advantage will then be the sum of the two different bounds.

If we suppose that every partnered oracle in our session reached status *accepted* before the server or any player running a partnered oracle is corrupted.

In this case, thanks to the signatures and the nonces, the adversary sees at most a blinded KEM public encapsulation key, an encapsulation of a session key, at most  $r$  blinded encapsulations of the same session key with corresponding blinded decapsulations. By indistinguishability for the blinded KEM, it follows that the adversary cannot distinguish between the actual encapsulated key and a randomly chosen key, so the adversary has no information about  $b$ .

Next, suppose no responder player is ever corrupted. In this case, the adversary (in the worst case) chooses the keys for the blinded KEM, but the public key encryption ensures that the adversary cannot see the actual encapsulation of the key. In other words, the adversary only sees blinded encapsulations of an unknown encapsulation, which reveals little information about the encapsulated key by  $\epsilon$ -blindness of the blinded KEM. Furthermore, the rigidity of the blinded KEM ensures that every responder can detect an incorrect server response, unless a collision in the key derivation function occurs.

The complete proof is given in the full version [4].

### 5.3 Instantiating the Protocol with the DH Blinded KEM

We instantiate the above offline assisted group key exchange protocol OAGK with the DH-based blinded KEM from Sect. 4.1, the protocol denoted by DH-OAGK. In this instantiation, we choose the nonce  $N_I$  from the group  $\mathbb{G}$ .

In Fig. 12, we present the core of the resulting protocol (without identities, nonces, session identifiers, key confirmation, authentication and encryption) similar to Fig. 9. We only show one responder.

Theorems 1 and 2 show that this instantiation is secure.

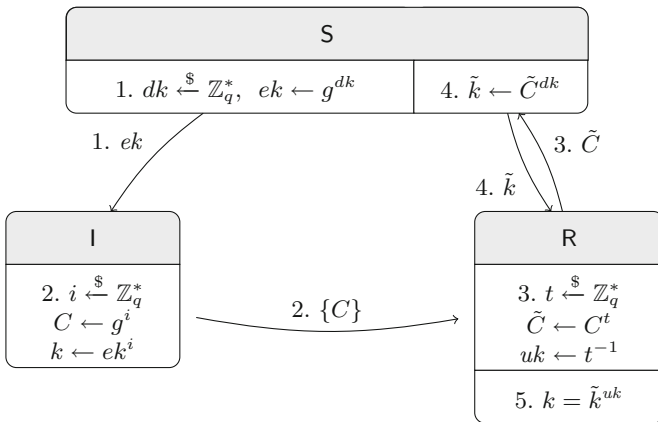


Fig. 12. Running protocol DH-OAGK with one responder, where  $\{C\} = \text{Enc}_R(C, \dots)$ .

## References

1. Abe, M., Gennaro, R., Kurosawa, K., Shoup, V.: Tag-KEM/DEM: a new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 128–146. Springer, Heidelberg (2005). [https://doi.org/10.1007/11426639\\_8](https://doi.org/10.1007/11426639_8)
2. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. IACR Cryptology ePrint Archive, 2002:80 (2002)
3. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 480–499. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44371-2\\_27](https://doi.org/10.1007/978-3-662-44371-2_27)
4. Boyd, C., Davies, G.T., Gjøsteen, K., Jiang, Y.: Offline assisted group key exchange. Cryptology ePrint Archive, Report 2018/114 (2018). <https://eprint.iacr.org/2018/114>
5. Bresson, E., Manulis, M.: Securing group key exchange against strong corruptions. In: Abe, M., Gligor, V.D. (eds.) Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, pp. 249–260. ACM (2008)
6. Burmester, M., Desmedt, Y.: A secure and efficient conference key distribution system. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995). <https://doi.org/10.1007/BFb0053443>
7. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. *J. Cryptol.* **20**(3), 265–294 (2007)
8. Chu, C.-K., Chow, S.S.M., Tzeng, W.-G., Zhou, J., Deng, R.H.: Key-aggregate cryptosystem for scalable data sharing in cloud storage. *IEEE Trans. Parallel Distrib. Syst.* **25**(2), 468–477 (2014)
9. Cohn-Gordon, K., Cremers, C., Garratt, L., Millican, J., Milner, K.: On ends-to-ends encryption: asynchronous group messaging with strong security guarantees. Cryptology ePrint Archive, Report 2017/666 (2017). <https://eprint.iacr.org/2017/666>
10. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. IACR Cryptology ePrint Archive, 2001:108 (2001)
11. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.* **33**(1), 167–226 (2003)
12. Dent, A.W.: A designer’s guide to KEMs. In: Paterson, K.G. (ed.) Cryptography and Coding 2003. LNCS, vol. 2898, pp. 133–151. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-40974-8\\_12](https://doi.org/10.1007/978-3-540-40974-8_12)
13. Derler, D., Jager, T., Slamanig, D., Striecks, C.: Bloom filter encryption and applications to efficient forward-secret 0-RTT key exchange. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 425–455. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-78372-7\\_14](https://doi.org/10.1007/978-3-319-78372-7_14)
14. Gao, W., Neupane, K., Steinwandt, R.: Tuning a two-round group key agreement. *Int. J. Inf. Sec.* **13**(5), 467–476 (2014)
15. Green, M.D., Miers, I.: Forward secure asynchronous messaging from puncturable encryption. In: 2015 IEEE Symposium on Security and Privacy, pp. 305–320, May 2015
16. Green, M.: Secure blind decryption. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 265–282. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-19379-8\\_16](https://doi.org/10.1007/978-3-642-19379-8_16)

17. 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, Part III. LNCS, vol. 10212, pp. 519–548. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56617-7\\_18](https://doi.org/10.1007/978-3-319-56617-7_18)
18. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-74143-5\\_31](https://doi.org/10.1007/978-3-540-74143-5_31)
19. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-45146-4\\_7](https://doi.org/10.1007/978-3-540-45146-4_7)
20. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-28628-8\\_26](https://doi.org/10.1007/978-3-540-28628-8_26)
21. Lám, I., Szebeni, S., Buttyán, L.: Invitation-oriented TGDH: key management for dynamic groups in an asynchronous communication model. In: 41st International Conference on Parallel Processing Workshops, ICPPW 2012, pp. 269–276. IEEE Computer Society (2012)
22. Lám, I., Szebeni, S., Buttyán, L.: Tresorium: cryptographic file system for dynamic groups over untrusted cloud storage. In: 41st International Conference on Parallel Processing Workshops, ICPPW 2012, pp. 296–303. IEEE Computer Society (2012)
23. Liu, X., Zhang, Y., Wang, B., Yan, J.: Mona: secure multi-owner data sharing for dynamic groups in the cloud. *IEEE Trans. Parallel Distrib. Syst.* **24**(6), 1182–1191 (2013)
24. Marlinspike, M., Perrin, T.: The X3DH key agreement protocol, November 2016. <https://signal.org/docs/specifications/x3dh/>
25. Sakurai, K., Yamane, Y.: Blind decoding, blind undeniable signatures, and their applications to privacy protection. In: Anderson, R. (ed.) IH 1996. LNCS, vol. 1174, pp. 257–264. Springer, Heidelberg (1996). [https://doi.org/10.1007/3-540-61996-8\\_45](https://doi.org/10.1007/3-540-61996-8_45)

# **Advanced Encryption**



# Function-Dependent Commitments for Verifiable Multi-party Computation

Lucas Schabhüser<sup>(✉)</sup>, Denis Butin, Denise Demirel, and Johannes Buchmann

Technische Universität Darmstadt, Darmstadt, Germany  
lschabhueser@cdc.informatik.tu-darmstadt.de

**Abstract.** In cloud computing, delegated computing raises the security issue of guaranteeing data authenticity during a remote computation. Existing solutions do not simultaneously provide fast correctness verification, strong security properties, and information-theoretic confidentiality. We introduce a novel approach, in the form of function-dependent commitments, that combines these strengths. We also provide an instantiation of function-dependent commitments for linear functions that is unconditionally, i.e. information-theoretically, hiding and relies on standard hardness assumptions. This powerful construction can for instance be used to build verifiable computing schemes providing information-theoretic confidentiality. As an example, we introduce a verifiable multi-party computation scheme for shared data providing public verifiability and unconditional privacy towards the servers and parties verifying the correctness of the result. Our scheme can be used to perform verifiable computations on secret shares while requiring only a single party to compute the audit data for verification. Furthermore, our verification procedure is asymptotically even more efficient than performing operations locally on the shared data. Thus, our solution improves the state of the art for authenticated computing, verifiable computing and multi-party computation.

**Keywords:** Commitments · Homomorphic cryptography · MPC

## 1 Introduction

Today, it is common practice to outsource time-consuming computations to the cloud. Such infrastructures attractively offer cost savings and dynamic computing resource allocation. In such a situation, it is desirable to be able to verify the outsourced computation. The verification must be *efficient*, by which we mean that the verification procedure is significantly faster than verified computation itself. Otherwise, the verifier could as well carry out the computation by himself, negating the advantage of outsourcing.

Often, not only the data owner is interested in the correctness of a computation; but also third parties, like insurance companies in the case of medical data. For such third party verifiers, another desired property for verification



procedures is *proof of origin*: evidence linking the result of the outsourced computation to their input. This additional guarantee is required because proofs of correct computation usually do not explicitly include input values. It is especially important if the verifier of the correctness proof is a third party who does not trust the cloud provider to provide the correct input to the computation. Together, these two pieces of evidence guarantee that the output received by the provider was indeed correctly computed from the initially provided input.

In addition, there are scenarios in which computations are performed over sensitive data. For instance, a cloud server may collect health data of individuals and compute their averages. So the challenge arises to design efficient verification procedures for outsourced computing that are privacy-preserving. Growing amounts of data are sensitive enough to require *long-term* protection. Electronic health records, voting records, or tax data require protection periods exceeding the lifetime of an individual. Over such a long time, complexity-based confidentiality protection is unsuitable because algorithmic progress is unpredictable. In contrast, *information-theoretic* confidentiality protection is not threatened by algorithmic progress and supports long-term security.

Two categories of solutions simultaneously address verifiability, proof of origin, and confidentiality:

- Homomorphic authenticators [1], which sometimes allow for efficient verification, keeping the computational effort of the verifier low. They do, however, not provide information-theoretic privacy, i.e. they are not long-term secure. Schemes like the one presented in [11] offer context hiding security, i.e. authenticators to the output of a computation do not leak information about the input. In this work, we consider a privacy notion which is even stronger than context hiding. In our case, no information is leaked; in particular, not even about the output.
- Homomorphic commitments [6, 25, 27] can be used for auditing. Authenticity is typically achieved by using a secure bulletin board [16]. In particular, Pedersen commitments [25] provide long-term security: they achieve information-theoretic privacy for the input values to arbitrary linear functions. Homomorphic commitments however, feature computationally costly correctness verification.

For a more detailed comparison to related work, see Sect. 5.

## 1.1 Contributions

In this paper, we solve the problem of providing efficient verification and proof of origin with information-theoretic privacy for linear functions. To achieve this, we introduce a novel generic construction that combines information-theoretic privacy with strong unforgeability and fast verification. We call this construction *function-dependent commitments* (FDCs). In addition to this main contribution, we provide a concrete, unconditionally hiding instantiation of FDCs for linear functions using pairings, demonstrating that our generic construction can be

realized in the standard model. In terms of hardness assumptions, only a variant of the Diffie–Hellman problem [11] is required. Our instantiation achieves succinctness and efficient verification. Finally, we showcase a verifiable multi-party computation scheme based on the concrete instantiation. This scheme makes it possible to verify whether the reconstructed result has been computed correctly by computing additional audit data on a *single* storage server. Previous proposals require *all* storage servers to perform computations to check correctness. Our scheme provides unconditional input-output privacy towards the servers and parties verifying computational correctness.

## 1.2 Outline

The remainder of this paper is organized as follows. We first introduce our framework for FDC schemes (Sect. 2). We then present a concrete instantiation of an FDC using pairings, and prove its properties (Sect. 3). A sketch of how this instantiation can be used to build a verifiable computing scheme for shared data is presented next (Sect. 4). Finally, we compare our contribution with related work (Sect. 5) and conclude (Sect. 6).

## 2 Function-Dependent Commitments

In this section, we present our novel FDC scheme and define its relevant properties. We define the classical properties of commitments, binding and hiding, in the context of FDCs. Furthermore we provide definitions for evaluation correctness and unforgeability. In terms of performance properties, we consider succinctness and amortized efficiency.

Like in the case of homomorphic commitments or authenticators, a function-dependent commitment can be used to derive new commitments by its homomorphic properties. It is necessary that the homomorphic property cannot be abused to create forgeries. In the context of homomorphic authenticators, the notions of labeled and multi-labeled programs (see e.g. [4]) are introduced to provide meaningful security definitions.

Evaluating a function can be modeled as performing a program on a set of labeled inputs that belong to a given dataset. On a high level, a message is uniquely identified by two identifiers: one input identifier  $\tau$ , and one dataset identifier  $\Delta$ . One can think of a dataset as an array of message, and of the input identifiers as pointers to specific positions within this array.

This enables a precise description of homomorphic properties. For authenticators, it is usually required that only authenticators created under the same dataset identifier are used for homomorphic evaluation. We now formally describe labeled and multi-labeled programs, in the vein of Backes et al. [4].

A *labeled program*  $\mathcal{P}$  consists of a tuple  $(f, \tau_1, \dots, \tau_k)$ , where  $f : \mathcal{M}^k \rightarrow \mathcal{M}$  is a function with  $k$  inputs and  $\tau_i \in \chi$  is a label for the  $i$ -th input of  $f$  from some set  $\chi$ . Given a set of labeled programs  $\mathcal{P}_1, \dots, \mathcal{P}_t$  and a function  $g : \mathcal{M}^t \rightarrow \mathcal{M}$ , they can be composed by evaluating  $g$  over the labeled programs, i.e.  $\mathcal{P}^* = g(\mathcal{P}_1, \dots, \mathcal{P}_t)$ .

The identity program with label  $\tau$  is given by  $\mathcal{I}_\tau = (f_{id}, \tau)$ , where  $f_{id} : \mathcal{M} \rightarrow \mathcal{M}$  is the identity function. Program  $\mathcal{P} = (f, \tau_1, \dots, \tau_k)$  can be expressed as the composition of  $k$  identity programs  $\mathcal{P} = f(\mathcal{I}_{\tau_1}, \dots, \mathcal{I}_{\tau_k})$ .

A *multi-labeled program*  $\mathcal{P}_\Delta$  is a pair  $(\mathcal{P}, \Delta)$  of the labeled program  $\mathcal{P}$  and a dataset identifier  $\Delta$ . Given a set of  $t$  multi-labeled programs with the same data set identifier  $\Delta$ , i.e.  $(\mathcal{P}_1, \Delta), \dots, (\mathcal{P}_t, \Delta)$ , and a function  $g : \mathcal{M}^t \rightarrow \mathcal{M}$ , a composed multi-label program  $\mathcal{P}_\Delta^*$  can be computed, consisting of the pair  $(\mathcal{P}^*, \Delta)$ , where  $\mathcal{P}^* = g(\mathcal{P}_1, \dots, \mathcal{P}_t)$ . Analogously to the identity program for labeled programs, we refer to a multi-labeled identity program by  $\mathcal{I}_{(\Delta, \tau)} = ((f_{id}, \tau), \Delta)$ .

Using the formalism of multi-labeled programs, we now define FDCs.

**Definition 1.** *A FDC scheme is a tuple of algorithms (Setup, KeyGen, PublicCommit, PrivateCommit, FunctionCommit, Eval, FunctionVerify, PublicDecommit):*

$\text{Setup}(1^\lambda)$  takes as input the security parameter  $\lambda$  and outputs public parameters  $\text{pp}$ . We implicitly assume that every algorithm uses these public parameters, leaving them out of the notation.

$\text{KeyGen}(1^\lambda)$  takes the security parameter  $\lambda$  as input and outputs a secret-public key pair  $(\text{sk}, \text{pk})$ .

$\text{PublicCommit}(\text{m}, \text{r})$  takes as input a message  $\text{m}$  and randomness  $\text{r}$  and outputs commitment  $\text{C}$ .

$\text{PrivateCommit}(\text{sk}, \text{m}, \text{r}, \Delta, \tau)$  takes as input the secret key  $\text{sk}$ , a message  $\text{m}$ , randomness  $\text{r}$ , a dataset  $\Delta$ , and an identifier  $\tau$  and outputs an authenticator  $\text{A}$  for the tuple  $(\text{m}, \text{r}, \Delta, \tau)$ .

$\text{FunctionCommit}(\text{pk}, \mathcal{P})$  takes as input the public key  $\text{pk}$  and a labeled program  $\mathcal{P}$  and outputs a function commitment  $\text{F}$  to  $\mathcal{P}$ .

$\text{Eval}(\mathcal{P}_\Delta, \text{A}_1, \dots, \text{A}_n)$  takes as input a multi-labeled program  $\mathcal{P}_\Delta = ((f, \tau_1, \dots, \tau_n), \Delta)$  and a set of authenticators  $\text{A}_1, \dots, \text{A}_n$ , where  $\text{A}_i$  is an authenticator for  $(\text{m}_i, \text{r}_i, \Delta, \tau_i)$ , for  $i = 1, \dots, n$ . It computes an authenticator  $\text{A}^*$  to the tuple  $(f(\text{m}_1, \dots, \text{m}_n), f(\text{r}_1, \dots, \text{r}_n), \Delta, (\tau_1, \dots, \tau_n))$  using  $\text{A}_1, \dots, \text{A}_n$  and outputs  $\text{A}^*$ .

$\text{FunctionVerify}(\text{pk}, \text{A}, \text{C}, \text{F}, \Delta)$  takes as input a public key  $\text{pk}$ , a FDC containing an authenticator  $\text{A}$  and a commitment  $\text{C}$ , a function commitment  $\text{F}$ , as well as a dataset identifier  $\Delta$ . It outputs either 1 (accept) or 0 (reject).

$\text{PublicDecommit}(\text{m}, \text{r}, \text{C})$  takes as input message  $\text{m}$ , randomness  $\text{r}$ , and commitment  $\text{C}$ . It outputs either 1 (accept) or 0 (reject).

$\text{FunctionVerify}$  only verifies whether the pair  $(\text{C}, \text{A})$  is a correct FDC to  $\text{F}$  while  $\text{PublicDecommit}$  allows to check that  $\text{C}$  opens to a specific pair of opening values  $(\text{m}, \text{r})$ .

## 2.1 Properties of Function-Dependent Commitments

As for classical commitments we want our schemes to be *binding* (see e.g. [30]). That is, after committing to a message, it should be infeasible to open the commitment to a different message. For a formal definition we refer to the full version.

Another important notion, targeting privacy, is the hiding property. Commitments are intended to not leak information about the messages they contain. This is not to be confused with the context hiding property, where homomorphic authenticators to the output of a computation do not leak information about the inputs to the computation. They do however leak information about the output.

**Definition 2 (Hiding).** *A FDC is called computationally hiding if the sets of commitments  $\{\text{PublicCommit}(m, r) \mid r \xleftarrow{\$} \mathcal{R}\}$  and  $\{\text{PublicCommit}(m', r') \mid r' \xleftarrow{\$} \mathcal{R}\}$  as well as  $\{\text{PrivateCommit}(\text{sk}, m, r, \Delta, \tau) \mid r \xleftarrow{\$} \mathcal{R}\}$  and  $\{\text{PrivateCommit}(\text{sk}, m', r', \Delta, \tau) \mid r' \xleftarrow{\$} \mathcal{R}\}$  have distributions that are indistinguishable for any probabilistic polynomial-time (PPT) adversary  $\mathcal{A}$  for all  $m \neq m' \in \mathcal{M}$ .*

*A FDC is called unconditionally hiding if these sets have the same distribution respectively for all  $m \neq m' \in \mathcal{M}$ .*

An obvious requirement for an FDC is to be *correct*, i.e. if messages are authenticated properly and evaluation is performed honestly, the resulting commitment should be verified. This is formalized in the following definition.

**Definition 3 (Evaluation Correctness).** *A FDC achieves evaluation correctness if for any set of messages,  $m_1, \dots, m_n \subset \mathcal{M}$ , any set of randomness  $r_1, \dots, r_n \subset \mathcal{R}$ , any set of identifiers  $\tau_1, \dots, \tau_n \subset \mathcal{X}$ , and any multi-labeled program  $\mathcal{P}_\Delta = ((f, \tau_1, \dots, \tau_n), \Delta)$  we have  $\text{FunctionVerify}(\text{pk}, A, C, F, \Delta) = 1$ , where  $A_i = \text{PrivateCommit}(\text{sk}, m_i, r_i, \Delta, \tau_i)$  for  $i \in [n]$ ,  $A = \text{Eval}(\mathcal{P}_\Delta, A_1, \dots, A_n)$ ,  $C = \text{PublicCommit}(f(m_1, \dots, m_n), f(r_1, \dots, r_n))$ , and  $F = \text{FunctionCommit}(\text{pk}, \mathcal{P})$ .*

For the security notion of FDCs, we first provide a definition for *well defined programs* and *forgeries* on these programs. Then, we introduce an experiment the attacker can run in order to generate a successful forgery and present a definition for unforgeability based on this experiment.

**Definition 4 (Well Defined Program).** *A labeled program  $\mathcal{P} = (f, \tau_1, \dots, \tau_n)$  is well defined with respect to a list  $L_\Delta$  if one of the two following cases holds:*

1. *There are messages  $m_1, \dots, m_n$  such that  $(\tau_i, m_i) \in L_\Delta \forall i = 1, \dots, n$ .*
2. *There is an  $i \in \{1, \dots, n\}$  such that  $(\tau_i, \cdot) \notin L_\Delta$  and  $f(\{m_j\}_{(\tau_j, m_j) \in L_\Delta} \cup \{\tilde{m}_k\}_{(\tau_k, \cdot) \notin L_\Delta})$  does not depend on the choice of  $\tilde{m}_k \in \mathcal{M}$ .*

**Definition 5 (Forgery).** *A forgery is a tuple  $(\mathcal{P}_\Delta, C, A)$  such that*

$$\text{FunctionVerify}(\text{pk}, A, C, \text{FunctionCommit}(\text{pk}, \mathcal{P}), \Delta) = 1$$

*holds and one of the following conditions is met:*

**Type 1:** *The list  $L_\Delta$  was not initialized during the game, i.e. no message was ever committed under the data set identifier  $\Delta$ .*

**Type 2:**  *$\mathcal{P}_\Delta$  is well defined with respect to list  $L_\Delta$  and  $C \neq \text{PublicCommit}(f(\{m_j\}_{(\tau_j, m_j) \in L_\Delta}), f(\{r_j\}_{(\tau_j, r_j) \in L_\Delta}))$ , i.e.  $C$  is not the correct commitment to the output of the computation.*

**Type 3:**  $\mathcal{P}_\Delta$  is not well defined with respect to  $L_\Delta$ .

This definition of forgeries is consistent with existing definitions, e.g. [11]. It is an immediate corollary of [18, Theorem 5.1] that if  $\mathcal{P}$  contains a linear function, then any adversary who outputs a Type 3 forgery can be converted into one that outputs a Type 2 forgery. To define unforgeability, we first describe the experiment  $\mathbf{EXP}_{\mathcal{A}, \text{Com}}^{UF-CMA}(\lambda)$  between an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$ .

$\mathbf{EXP}_{\mathcal{A}, \text{Com}}^{UF-CMA}(\lambda)$  :

**Setup**  $\mathcal{C}$  calls  $\text{pp} \xleftarrow{\$} \text{Setup}(1^\lambda)$  and gives  $\text{pp}$  to  $\mathcal{A}$ .

**Key Generation**  $\mathcal{C}$  calls  $(\text{pk}, \text{sk}) \xleftarrow{\$} \text{KeyGen}(1^\lambda)$  and gives  $\text{pk}$  to  $\mathcal{A}$ .

**Queries**  $\mathcal{A}$  adaptively submits queries for  $(\Delta, \tau, m, r)$  where  $\Delta$  is a dataset,  $\tau$  is an identifier,  $m$  is a message, and  $r$  is a random value.  $\mathcal{C}$  proceeds as follows:

- If  $(\Delta, \tau, m, r)$  is the first query with dataset identifier  $\Delta$ , it initializes an empty list  $L_\Delta = \emptyset$  for  $\Delta$ .
- If  $L_\Delta$  does not contain a tuple  $(\tau, \cdot, \cdot)$ , i.e.  $\mathcal{A}$  never queried  $(\Delta, \tau, \cdot, \cdot)$ ,  $\mathcal{C}$  calls  $A \leftarrow \text{PrivateCommit}(\text{sk}, m, r, \Delta, \tau)$ , updates the list  $L_\Delta = L_\Delta \cup (\tau, m, r)$ , and gives  $A$  to  $\mathcal{A}$ .
- If  $(\tau, m, r) \in L_\Delta$ , then  $\mathcal{C}$  returns the same authenticator  $A$  as before.
- If  $L_\Delta$  already contains a tuple  $(\tau, m', r')$  for  $(m, r) \neq (m', r')$ ,  $\mathcal{C}$  returns  $\perp$ .

**Forgery**  $\mathcal{A}$  outputs a tuple  $(\mathcal{P}_\Delta, m, r, A)$ .

$\mathbf{EXP}_{\mathcal{A}, \text{Com}}^{UF-CMA}(\lambda)$  outputs 1 if the tuple returned by  $\mathcal{A}$  is a forgery as defined before in Definition 5.

**Definition 6 (Unforgeability).** A FDC is unforgeable if for any PPT adversary  $\mathcal{A}$  we have

$$\Pr[\mathbf{EXP}_{\mathcal{A}, \text{Com}}^{UF-CMA}(\lambda) = 1] = \text{negl}(\lambda),$$

where  $\text{negl}(\lambda)$  denotes any function negligible in the security parameter  $\lambda$ .

Regarding performance, we consider additional properties. *Succinctness* specifies a limit on the size of the FDCs, thus keeping the required bandwidth low, when using FDCs to verify the correctness of an outsourced computation.

**Definition 7 (Succinctness).** A FDC is succinct if, for a fixed security parameter  $\lambda$ , the size of the authenticators depends at most logarithmically on the dataset size  $n$ .

*Amortized efficiency* specifies a bound on the computational effort required to perform verifications.

**Definition 8 (Amortized Efficiency).** Let  $\mathcal{P}_\Delta = ((f, \tau_1, \dots, \tau_n), \Delta)$  be a multi-labeled program,  $m_1, \dots, m_n \in \mathcal{M}$  a set of messages,  $r_1, \dots, r_n \in \mathcal{R}$  a set of randomness, and  $t(n)$  be the time required to compute  $f(m_1, \dots, m_n)$ . A FDC achieves amortized efficiency if for given authenticator  $A$  and function commitment  $F$  the time required for the computation of  $\text{FunctionVerify}(\text{pk}, A, \text{PublicCommit}(m, r), F, \Delta)$  is  $t' = o(t(n))$ .

Analogous definitions for amortized efficiency haven been given in [4, 11, 14, 31]. The usual one-time pre-computation is captured by our algorithm `FunctionCommit`. In the case of reuse of the same function over multiple datasets, this property enables an improvement in terms of runtime.

### 3 A Pairing-Based FDC Instantiation

In this section, we present an instantiation of a FDC scheme based on pairings. Our construction uses asymmetric bilinear groups. It can be use to verify the correct evaluation of linear functions. In the following, we analyze our scheme with regard to their hiding and binding property, as well as correctness, unforgeability, succinctness and amortized efficiency.

**Definition 9.** Let  $\mathcal{G}$  be a generator of cyclic groups of order  $p$  and let  $\mathbb{G} \xleftarrow{\$} \mathcal{G}(1^\lambda)$ . We say the Discrete Logarithm assumption (DL) holds in  $\mathbb{G}$  if there exists no PPT adversary  $\mathcal{A}$  that, given  $(g, g^a)$  for a random generator  $g \in \mathbb{G}$  and random  $a \in \mathbb{Z}_p$ , can output  $a$  with more than negligible probability, i.e. if  $\Pr[a \leftarrow \mathcal{A}(g, g^a) \mid g \xleftarrow{\$} \mathbb{G}, a \xleftarrow{\$} \mathbb{Z}_p] = \text{negl}(\lambda)$ .

**Definition 10 (Asymmetric bilinear groups [8]).** An asymmetric bilinear group is a tuple  $\text{bgrp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ , such that:

- $\mathbb{G}_1, \mathbb{G}_2$ , and  $\mathbb{G}_T$  are cyclic groups of prime order  $p$ ,
- $g_1 \in \mathbb{G}_1$  and  $g_2 \in \mathbb{G}_2$  are generators for their respective groups,
- the DL assumption holds in  $\mathbb{G}_1, \mathbb{G}_2$ , and  $\mathbb{G}_T$ ,
- $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  is bilinear, i.e.  $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$  holds for all  $a, b \in \mathbb{Z}$ ,
- $e$  is non-degenerate, i.e.  $e(g_1, g_2) \neq 1$ , and
- $e$  is efficiently computable.

We write  $g_t = e(g_1, g_2)$ .

The security of our pairing-based instantiation relies on a hardness assumption previously introduced by Catalano et al. [11], the Flexible Diffie-Hellman Inversion (FDHI) problem. It was shown by these authors that the FDHI holds in the generic group model.

**Definition 11 ([11]).** Let  $\mathcal{G}$  be a generator of asymmetric bilinear groups and let  $\text{bgrp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\$} \mathcal{G}(1^\lambda)$ . We say the Flexible Diffie-Hellman Inversion (FDHI) assumption holds in  $\text{bgrp}$  if for every PPT adversary  $\mathcal{A}$

$$\Pr[W \in \mathbb{G}_1 \setminus \{1_{\mathbb{G}_1}\} \wedge W' = W^{\frac{1}{z}} : (W, W') \leftarrow \mathcal{A}(g_1, g_2, g_2^z, g_2^v, g_1^{\frac{z}{v}}, g_1^r, g_1^{\frac{r}{v}}) \mid z, r, v \xleftarrow{\$} \mathbb{F}_p] = \text{negl}(\lambda).$$

### 3.1 Construction

We are now ready to describe the algorithms making up our FDC. We use a signature scheme  $\Sigma = (\text{SigKeyGen}, \text{Sign}, \text{SigVerify})$  and a pseudorandom function  $F : \mathcal{K} \times \{0, 1\}^* \rightarrow \mathbb{F}_p$ . For a set of possibly different messages  $\mathbf{m}_1, \dots, \mathbf{m}_n$ , we denote by  $\mathbf{m}_i$  the  $i$ -th message. Since our messages are vectors, i.e.  $\mathbf{m} \in \mathbb{F}_p^T$ , we write  $\mathbf{m}[j]$  to indicate the  $j$ -th entry of message vector  $\mathbf{m}$ . Therefore  $\mathbf{m}_i[j]$  denotes the  $j$ -th entry of the  $i$ -th message. Given a linear function  $f$ , its  $i$ -th coefficient is denoted by  $f_i$ , i.e.  $f(\mathbf{m}_1, \dots, \mathbf{m}_n) = \sum_{i=1}^n f_i \mathbf{m}_i$ .

**Setup** takes as input the security parameter  $\lambda$ . It defines the parameters  $n, T \in \mathbb{N}$ . Then, it first chooses a bilinear map  $\text{bgrp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$  with  $e(g_1, g_2) = g_t$ . Afterwards, it chooses  $a_0, \dots, a_T \in \mathbb{F}_p$  uniformly at random. It checks whether  $a_j \neq a_i$  for all  $j \neq i$ . If it fails it chooses a new  $a_j$ . Then, for all  $j = 0, \dots, T$  it computes  $H_j = g_1^{a_j}$ . It outputs the public parameters  $\text{pp} = (n, T, \text{bgrp}, H_0, \dots, H_T)$ .

**KeyGen** takes as input the security parameter  $\lambda$ , and the public parameters  $\text{pp}$ . Then it chooses  $y \in \mathbb{F}_p$  uniformly at random and computes  $Y = g_2^y$ . Additionally it chooses  $b_1, \dots, b_n \in \mathbb{F}_p$  uniformly at random and checks whether  $b_i \neq b_j$  for all  $i \neq j$ . If it fails it chooses a new  $b_i$ . Then, for all  $i = 1, \dots, n$  it computes  $\hat{H}_i = g_1^{b_i}$  and  $\hat{h}_i = g_1^{b_i}$ . Then, the algorithm chooses random seeds  $K, K' \in \mathcal{K}$  for a pseudorandom function  $F : \mathcal{K} \times \{0, 1\}^* \rightarrow \mathbb{F}_p$ . Finally, it generates keys for the signature scheme by calling  $(\text{sk}_{\text{Sig}}, \text{pk}_{\text{Sig}}) \leftarrow \text{SigKeyGen}(1^\lambda)$  and outputs public key  $\text{pk} = (\text{pk}_{\text{Sig}}, Y, \hat{h}_1, \dots, \hat{h}_n)$  and secret key  $\text{sk} = (\text{sk}_{\text{Sig}}, y, \hat{H}_1, \dots, \hat{H}_n, K, K')$ .

**PublicCommit** takes as input the public parameters  $\text{pp}$ , a message  $\mathbf{m} \in \mathbb{F}_p^T$ , and randomness  $r \in \mathbb{F}_p$ . It computes  $\mathbf{C} = H_0^r \cdot \prod_{j=1}^T H_j^{m[j]}$ , where  $m[j]$  is the  $j$ -th entry of message vector  $\mathbf{m} \in \mathbb{F}_p^T$ , and outputs commitment  $\mathbf{C}$ .

**PrivateCommit** takes as input the secret key  $\text{sk}$ , a message  $\mathbf{m} \in \mathbb{F}_p^T$ , randomness  $r \in \mathbb{F}_p$ , a dataset  $\Delta \in \{0, 1\}^*$ , and an identifier  $\tau \in [n]$ . It first computes  $z = F_K(\Delta)$  with the pseudorandom function  $F$  and calculates  $Z = g_2^z$ . Then, it binds  $Z$  to the dataset identifier  $\Delta$  by signing their concatenation, i.e.  $\sigma_\Delta = \text{Sign}(\text{sk}_{\text{Sig}}, \Delta \parallel Z)$ . Then, it computes  $u = F_{K'}(\Delta \parallel \tau)$ ,  $U = g_1^u$ , and  $V = (U \cdot \hat{H}_\tau \cdot H_0^{yr} \cdot \prod_{j=1}^T H_j^{ym[j]})^{\frac{1}{z}}$ . It returns authenticator  $\mathbf{A} = (\sigma_\Delta, Z, U, V)$ .

**FunctionCommit** takes as input the public key  $\text{pk}$  and a labeled program  $\mathcal{P} = (f, \tau_1, \dots, \tau_n)$ . It computes  $\mathbf{F} = \prod_{i=1}^n \hat{h}_i^{f_i}$ , where  $f_i$  denotes the  $i$ -th coefficient of  $f$ , and outputs function commitment  $\mathbf{F}$ .

**Eval** takes as input a linear function  $f$  and authenticators  $\mathbf{A}_1, \dots, \mathbf{A}_n$  where  $\mathbf{A}_i = (\sigma_{\Delta, i}, Z_i, U_i, V_i)$ . It sets  $\sigma_\Delta = \sigma_{\Delta, 1}$ ,  $Z = Z_1$  and computes  $U = \prod_{i=1}^n U_i^{f_i}$  and  $V = \prod_{i=1}^n V_i^{f_i}$  and outputs authenticator  $\mathbf{A} = (\sigma_\Delta, Z, U, V)$ .

**FunctionVerify** takes as input the public key  $\text{pk}$ , an authenticator  $\mathbf{A} = (\sigma_\Delta, Z, U, V)$ , a commitment  $\mathbf{C}$ , a function commitment  $\mathbf{F}$ , and a dataset identifier  $\Delta$ . It checks whether  $\text{SigVerify}(\text{pk}_{\text{Sig}}, \sigma_\Delta, \Delta \parallel Z) = 1$  holds. If not it outputs 0, otherwise it checks whether  $e(V, Z) = e(U, g_2) \cdot \mathbf{F} \cdot e(\mathbf{C}, Y)$  holds. If it does, it outputs 1; otherwise it outputs 0.

**PublicDecommit** takes as input the public parameters  $\mathbf{pp}$ , a message  $m \in \mathbb{F}_p^T$ , randomness  $r \in \mathbb{F}_p$ , and a commitment  $C$ . It outputs 1 if  $C = \mathbf{PublicCommit}(\mathbf{pp}, m, r)$  and 0 otherwise.

Our construction can also be used to provide authenticity in the form of unconditionally hiding authenticators, similarly to signatures. First, algorithm **KeyGen** is called. To authenticate a message  $m$ , the owner of the corresponding secret key  $\mathbf{sk}$  generates a random value  $r$  and computes an authenticator  $A$  with algorithm **PrivateCommit**. The authenticator  $A$  serves as a signature for  $m$ . To verify the authenticity of  $m$ , the verifier first requests  $m$  and  $r$  from the data owner. It next computes commitment  $C$  by calling **PublicCommit** with  $m$ ,  $r$ , and the public key  $\mathbf{pk}$ . Then, it uses  $\mathbf{pk}$  and algorithm **FunctionCommit** to generate a function commitment  $F_{id}$  to the identity function of  $m$ . Finally, it calls algorithm **FunctionVerify** to check whether the tuple  $C, A, F_{id}, \Delta$  is valid.

### 3.2 Properties

In the following, we first prove that our concrete scheme is indeed correct in the sense of Definition 3. We then prove that it satisfies the classical commitment properties — binding and hiding. With respect to efficiency, we next show succinctness and amortized efficiency. Finally, we reduce the security of our scheme to the hardness of the FDHI assumption.

**Theorem 1.** *Our construction is a correct FDC (Definition 3).*

*Proof.* Let  $m_1, \dots, m_n \in \mathbb{F}_p^T$  be a set of messages,  $r_1, \dots, r_n \in \mathbb{F}_p$  a set of randomness,  $\tau_1, \dots, \tau_n \in \chi$  set of identifiers, and  $\mathcal{P}_\Delta = ((f, \tau_1, \dots, \tau_n), \Delta)$  a linear multi-labeled program. We set  $m = f(m_1, \dots, m_n)$ ,  $r = f(r_1, \dots, r_n)$ ,  $A_i = (\sigma_{\Delta, i}, Z_i, U_i, V_i) \leftarrow \mathbf{PrivateCommit}(\mathbf{sk}, m_i, r_i, \Delta, \tau_i)$ , for  $i = 1, \dots, n$ , as well as  $F \leftarrow \mathbf{FunctionCommit}(\mathbf{pk}, \mathcal{P}_\Delta)$  and  $C \leftarrow \mathbf{PublicCommit}(m, r)$ .

Let  $A = (\sigma_\Delta, Z, U, V) \leftarrow \mathbf{Eval}(\mathcal{P}_\Delta, \mathbf{PrivateCommit}(\mathbf{sk}, m_1, r_1, \Delta, \tau_1), \dots, \mathbf{PrivateCommit}(\mathbf{sk}, m_n, r_n, \Delta, \tau_n))$ . By construction, we have  $\sigma_\Delta = \sigma_{\Delta, 1}$  which is correctly verified as long as the underlying signature scheme is correct.

Furthermore, consider  $(f_{id}, \tau_i)$  the identity function on the  $i$ -th input and let  $F_i \leftarrow \mathbf{FunctionCommit}(\mathbf{pk}, ((f_{id}, \tau_i), \Delta)) = F_i = \hat{h}_i$ . Since  $z_i = F_K(\Delta)$  and  $Z_i = g_2^{z_i}$  we have  $Z_i = Z, \forall i \in [n]$  and therefore  $e(V_i, Z_i) = e(V_i, Z) = e\left((U_i \cdot \hat{H}_{\tau_i} \cdot H_0^{y r_i} \cdot \prod_{j=1}^T H_j^{y m_i[j]})^{\frac{1}{z}}, Z\right) = e\left(U_i \cdot \hat{H}_{\tau_i} \cdot H_0^{y r_i} \cdot \prod_{j=1}^T H_j^{y m_i[j]}, g_2\right)$

$$= e(U_i, g_2) \cdot e\left(\hat{H}_{\tau_i}, g_2\right) \cdot e\left(H_0^{y r_i} \cdot \prod_{j=1}^T H_j^{y m_i[j]}, g_2\right)$$

$$= e(U_i, g_2) \cdot \hat{h}_{\tau_i} \cdot e\left(H_0^{r_i} \cdot \prod_{j=1}^T H_j^{m_i[j]}, g_2^y\right).$$

Hence  $e(V, Z) = e\left(\prod_{i=1}^n V_i^{f_i}, Z\right)$

$$= e\left(\left(\prod_{i=1}^n U_i^{f_i} \cdot \hat{H}_{\tau_i}^{f_i} \cdot H_0^{y r_i f_i} \cdot \prod_{j=1}^T H_j^{f_i \cdot y m_i[j]}\right)^{\frac{1}{z}}, Z\right)$$

$$= e\left(\prod_{i=1}^n U_i^{f_i}, g_2\right) \cdot \left(\prod_{i=1}^n \hat{h}_{\tau_i}^{f_i}\right) \cdot e\left(H_0^{\sum_{i=1}^n f_i \cdot r_i} \cdot \left(\prod_{j=1}^T H_j^{\sum_{i=1}^n f_i \cdot m_i[j]}\right), g_2^y\right)$$

$$= e(U, g_2) \cdot F \cdot e\left(H_0^r \cdot \prod_{j=1}^T H_j^{m[j]}, Y\right) = e(U, g_2) \cdot F \cdot e(C, Y)$$

This shows the correctness of our scheme.



**Theorem 2.** *Our construction is a binding FDC scheme as long as the discrete logarithm problem in  $\mathbb{G}_1$  is hard.*

*Proof.* Assume we have access to an oracle  $\mathcal{O}(\cdot)$  that on input  $\text{pk}$  outputs  $(\mathbf{m}, r) \neq (\mathbf{m}', r')$  such that  $\text{PublicCommit}(\mathbf{m}, r) = \text{PublicCommit}(\mathbf{m}', r')$ . Given  $g_1 \in \mathbb{G}_1$  from  $\text{bgp}$  we show how to use  $\mathcal{O}(\cdot)$  to solve the discrete logarithm problem in  $\mathbb{G}_1$ , i.e. computing  $x$  for  $g_1^x = g_1^x$ , where  $g_1 \in \mathbb{G}_1$ . During the generation of public key  $\text{pk}$  in  $\text{KeyGen}$ , we choose random  $\alpha_0, \dots, \alpha_T \in \mathbb{F}_p^*$  and compute  $H_0 = g_1^{\alpha_0}$  and  $H_i = g_1^{\alpha_i}$  for  $i = 1, \dots, T$ .

Afterwards, we query  $\mathcal{O}(\text{pk})$ , and receive  $(\mathbf{m}, r) \neq (\mathbf{m}', r')$ . If  $r = r'$  we submit a new query. Otherwise we have  $H_0^r \cdot \prod_{j=1}^T H_j^{m_j} = H_0^{r'} \cdot \prod_{j=1}^T H_j^{m'_j} \Leftrightarrow g_1^{\alpha_0 r} \cdot \prod_{j=1}^T g_1^{\alpha_j m_j} = g_1^{\alpha_0 r'} \cdot \prod_{j=1}^T g_1^{\alpha_j m'_j} \Leftrightarrow g_1^{x \cdot \alpha_0 (r - r') + \sum_{j=1}^T \alpha_j (m_j - m'_j)} = g_1^0$   
 $\Leftrightarrow x \cdot \alpha_0 (r - r') + \sum_{j=1}^T \alpha_j (m_j - m'_j) = 0 \Leftrightarrow x = \frac{1}{\alpha_0 (r - r')} \sum_{j=1}^T \alpha_j (m'_j - m_j)$   
 and found the discrete logarithm  $g_1^x = g_1^x$ . The binding property of algorithm  $\text{FunctionCommit}$  can be proven completely analogously.

**Theorem 3.** *Our construction is an unconditionally hiding FDC (Definition 2).*

*Proof.* If  $r \xleftarrow{\$} \mathbb{F}_p$  is chosen uniformly at random then  $\{H_0^r \mid r \xleftarrow{\$} \mathbb{F}_p\}$  is uniformly distributed over  $\mathbb{G}_1$ . Therefore the set  $\{H_0^r \cdot \prod_{j=1}^T H_j^{m_j} \mid r \xleftarrow{\$} \mathbb{F}_p\}$  is uniformly distributed over  $\mathbb{G}_1$ . So  $\{\text{PublicCommit}(\mathbf{m}, r) \mid r \in \mathbb{F}_p\}$  and  $\{\text{PublicCommit}(\mathbf{m}', r') \mid r' \in \mathbb{F}_p\}$  have the same distribution  $\forall \mathbf{m}, \mathbf{m}' \in \mathbb{F}_p^T$ . The output of  $\text{PrivateCommit}$  is an authenticator  $\mathbf{A} = (\sigma_\Delta, Z, U, V)$ . By construction  $\sigma_\Delta, Z, U$  are all independent of  $\mathbf{m}$ . Considering the  $V$  component, we have  $V = \left( U \cdot \hat{H}_\tau \cdot (H_0^r \cdot \prod_{j=1}^T H_j^{m_j})^y \right)^{\frac{1}{z}}$ . This is uniformly distributed over  $\mathbb{G}_1$  if and only if the set  $\{H_0^r \cdot \prod_{j=1}^T H_j^{m_j} \mid r \xleftarrow{\$} \mathbb{F}_p\}$  is. As we have shown this to be true  $\{\text{PrivateCommit}(\text{sk}, \mathbf{m}, r, \Delta, \tau) \mid r \in \mathbb{F}_p\}$  and  $\{\text{PrivateCommit}(\text{sk}, \mathbf{m}', r', \Delta, \tau) \mid r' \in \mathbb{F}_p\}$  have the same distribution for all  $\mathbf{m}, \mathbf{m}' \in \mathbb{F}_p^T$ .

**Theorem 4.** *Our construction is succinct (Definition 7).*

*Proof.* The number of elements contained in an authenticator  $\text{PrivateCommit}$  is constant and does therefore not depend on  $n$ , the size of the dataset.

**Theorem 5.** *Our construction achieves amortized efficiency (Definition 8).*

*Proof.*  $\text{PublicCommit}$  is independent of  $n$ .  $\text{FunctionVerify}$  consists of a signature verification, two pairing evaluations, and two group operations. Thus their combined running time is independent of  $n$  whereas an evaluation of  $\mathbf{f}$  is  $\geq O(n)$ . Therefore, our construction achieves amortized efficiency for suitably large  $n$ .

**Theorem 6.** *Our construction is an unforgeable FDC scheme (Definition 6) if  $\Sigma$  is an unforgeable signature scheme,  $F$  is a pseudorandom function and the FDHI assumption (see Definition 11) holds in  $\text{bgp}$ .*

*Proof.* This proof follows the structure of [11, Theorem 8]. A major difference is that, in our security reduction, the actual outcome of the computation function  $f$  is never required. In particular [12, Lemmata 5 and 7], knowledge of the forged outcome of the computation is a crucial part of the security reductions that prove indistinguishability between games. We present a new indistinguishability reduction that only uses group elements.

To prove Theorem 6, we define a series of games with the adversary  $\mathcal{A}$  and we show that the adversary  $\mathcal{A}$  wins, i.e. the game outputs 1, only with negligible probability. Following the notation of [11], we write  $G_i(\mathcal{A})$  to denote that a run of game  $i$  with adversary  $\mathcal{A}$  returns 1. We use flag values  $\text{bad}_i$ , initially set to false. If at the end of the game any of these flags is set to true, the game simply outputs 0. Let  $\text{Bad}_i$  denote the event that  $\text{bad}_i$  is set to true during game  $i$ .

Due to Theorem 5.1 in [18], any adversary who outputs a Type 3 forgery (see Definition 5) can be converted into one that outputs a Type 2 forgery. Therefore we only have to deal with Type 1 and Type 2 forgeries.

**Game 1** is the security experiment  $\text{EXP}_{\mathcal{A}, \text{Com}}^{UF-CMA}(\lambda)$  between an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$ , where  $\mathcal{A}$  only outputs Type 1 or Type 2 forgeries.

**Game 2** is defined as Game 1, except for the following change. Whenever  $\mathcal{A}$  returns a forgery  $(\mathcal{P}_{\Delta^*}^*, m^*, r^*, A^*)$  with  $A^* = (\sigma_{\Delta^*}^*, Z^*, U^*, V^*)$  and  $Z^*$  has not been generated by the challenger during the queries, then Game 2 sets  $\text{bad}_2 = \text{true}$ . It is worth noticing that after this change the game never outputs 1 if  $\mathcal{A}$  returns a Type 1 forgery.

**Game 3** is defined as Game 2, except that the pseudorandom function  $F$  is replaced by a random function  $\mathcal{R} : \{0, 1\}^* \rightarrow \mathbb{F}_p$ .

**Game 4** is defined as Game 3, except for the following change. At the beginning  $\mathcal{C}$  chooses  $\mu \in [Q]$  uniformly at random, with  $Q = \text{poly}(\lambda)$  the number of queries made by  $\mathcal{A}$  during the game. Let  $\Delta_1, \dots, \Delta_Q$  be all the datasets queried by  $\mathcal{A}$ . Then if in the forgery  $\Delta^* \neq \Delta_\mu$ , set  $\text{bad}_4 = \text{true}$ .

**Game 5** is defined as Game 4, except for the following change. At the beginning,  $\mathcal{C}$  chooses  $z_\mu \in \mathbb{F}_p$  at random and computes  $Z_\mu = g_2^{z_\mu}$ . It uses  $Z_\mu$  whenever queried for dataset  $\Delta_\mu$ . It chooses  $b_i, s_i \in \mathbb{F}_p$  uniformly at random for  $i = 1, \dots, n$  and sets  $\hat{H}_i = g_1^{b_i + z_\mu s_i}$  as well as  $\hat{h}_i = g_t^{b_i + z_\mu s_i}$ . If  $k = \mu$ , simulator  $\mathcal{S}$  sets the component  $U_\tau = g_1^{-b_\tau - a_0 y r - \sum_{j=1}^T a_j y m[j]}$ .

**Game 6** is defined as Game 5, except for the following change. The challenger runs an additional check. It computes  $\hat{m} = \mathcal{P}(m_1, \dots, m_n), \hat{r} = \mathcal{P}(r_1, \dots, r_n)$ , as well as  $\hat{A} = \text{Eval}(\mathcal{P}_{\Delta^*}^*, A_1, \dots, A_n)$ , i.e. it runs an honest computation over the messages, randomness and authenticators in dataset  $\Delta_\mu$ . If

$$\text{FunctionVerify}(\text{pk}, A^*, C^*, \text{FunctionCommit}(\text{pk}, \mathcal{P}^*), \Delta^*) = 1$$

and  $U^* = \hat{U}$ , then  $\mathcal{C}$  sets  $\text{bad}_6 = \text{true}$ .

**Game 7** is defined as Game 6, except for the following change. During a query for  $(\Delta_\mu, \tau, m, r)$ , the challenger sets  $U_\tau = g_1^{-b_\tau}$ .

- Any noticeable difference between Games 1 and 2 can be reduced to producing a forgery for the signature scheme. If  $\text{Bad}_2$  occurs, then  $\mathcal{A}$  produced a valid

signature  $\sigma_{\Delta^*}^*$  for  $(\Delta^* \mid Z^*)$  despite never having queried a signature on any  $(\Delta^* \mid \cdot)$ . This is a forgery on the signature scheme.

- Under the assumption that  $F$  is pseudorandom, Games 2 and 3 are computationally indistinguishable.
- By definition,  $\Pr[G_3(\mathcal{A})] = Q \cdot \Pr[G_4(\mathcal{A})]$ .
- $\Pr[G_4(\mathcal{A})] = \Pr[G_5(\mathcal{A})]$ , since the public keys are perfectly indistinguishable.
- Clearly,  $|\Pr[G_5(\mathcal{A})] - \Pr[G_6(\mathcal{A})]| \leq \Pr[\text{Bad}_6]$ . This occurs only with negligible probability if the FDHI assumption holds. For a proof of this statement, we refer to the full version.
- Since the  $b_i$  were chosen uniformly at random, Game 7 is perfectly indistinguishable from Game 6. After these modifications, Game 7 can only output 1 if  $\mathcal{A}$  produces a forgery  $(\mathcal{P}_{\Delta^*}^*, m^*, r^*, A^*)$  with  $A^* = (\sigma_{\Delta^*}^*, Z^*, U^*, V^*)$  s.t.

FunctionVerify(pk,  $A^*$ , PublicCommit( $\hat{m}, \hat{r}$ ), FunctionCommit(pk,  $\mathcal{P}^*$ ),  $\Delta^*$ ) = 1

and  $(\hat{m}, \hat{r}) \neq (m^*, r^*)$ ,  $\hat{U} \neq U^*$ , and  $\hat{V} \neq V^*$ . This only occurs with negligible probability if the FDHI assumption holds. For a corresponding proof, we refer to the full version.

## 4 Verifiable Computing on Shared Data from Our FDC

We now show how to build a verifiable multi-party computation scheme for linear functions using our pairing-based FDC construction from Sect. 3. A trivial solution would be to use a linearly homomorphic authenticator on each set of shares, running the homomorphic evaluation multiple times in parallel. Our construction only requires a single evaluation over the authenticators. We first recall the algorithms making up a verifiable computing scheme. We then briefly list relevant properties, and then present our construction. Finally, we sketch proofs for the properties of our verifiable computing scheme, notably input and output privacy.

**Definition 12 (Verifiable Computing Scheme).** *A Verifiable Computing Scheme  $\mathcal{VC}$  is a tuple of the following PPT algorithms ([19]):*

$\text{VKeyGen}(1^\lambda, f)$  : *The probabilistic key generation algorithm takes a security parameter  $\lambda$  and the description of a function  $f$ . It generates a secret key  $\text{sk}$ , a corresponding verification key  $\text{vk}$ , and a public evaluation key  $\text{ek}$  (that encodes the target function  $f$ ) and returns all these keys.*

$\text{ProbGen}(\text{sk}, x)$  : *The problem generation algorithm takes a secret key  $\text{sk}$  and data  $x$ . It outputs a decoding value  $\rho_x$  and a public value  $\sigma_x$  which encodes  $x$ .*

$\text{Compute}(\text{ek}, \sigma_x)$  : *The computation algorithm takes the evaluation key  $\text{ek}$  and the encoded input  $\sigma_x$ . It outputs an encoded version  $\sigma_y$  of the function's output  $y = f(x)$ .*

$\text{Verify}(\text{vk}, \rho_x, \sigma_y)$  : *The verification algorithm obtains a verification key  $\text{vk}$  and the decoding value  $\rho_x$ . It converts the encoded output  $\sigma_y$  into the output of the function  $y$ . If  $y = f(x)$  holds, it returns  $y$  or outputs  $\perp$  indicating that  $\sigma_y$  does not represent a valid output of  $f$  on  $x$ .*

Relevant properties for *verifiable computing schemes* are *correctness*, *publicly verifiability*, and *security*. For formal definitions, see [17]. Further privacy properties are *input privacy w.r.t. the servers*, *output privacy w.r.t. the servers*, *input privacy w.r.t. the verifier*, and *output privacy w.r.t. the verifier* (see [17]). These computationally secure versions can naturally be extended to information-theoretically secure versions; for more details we refer to the full version.

#### 4.1 Construction

Our instantiation of a FDC can be used to build a verifiable computing scheme for shared data supporting linear functions. Secure multi-party computation performed on shared data is realized using a secret sharing scheme, e.g. Shamir secret sharing [30], which we briefly describe. To share a secret  $\mathbf{m} \in \mathbb{F}_p$ , the client chooses random  $a_1, \dots, a_{t-1} \in \mathbb{F}_p$  and computes the polynomial  $P(x) = \mathbf{m} + a_1x + \dots + a_{t-1}x^{t-1}$ . By evaluating  $P(j)$  for  $j = 1, \dots, k$  it creates  $k$  shares which are given to  $k$  shareholders. Since a polynomial of degree  $t - 1$  is uniquely determined by  $t$  points  $(j, P(j))$  one can recover the secret by requesting  $t$  shares. At the same time, even a computationally unbounded adversary cannot learn anything about  $\mathbf{m}$  from  $t - 1$  shares or less (see [30]). Shamir secret sharing is linearly homomorphic, i.e.  $\alpha P(j) + \beta P'(j) = (\alpha P + \beta P')(j)$  for any two polynomials  $P, P' \in \mathbb{F}_p[x]$  and constants  $\alpha, \beta \in \mathbb{F}_p$ . Linear functions can thus be evaluated locally on the shares.

Verifiable computing for shared data can be performed as follows. For `VKeyGen`, the client runs `Setup`, `Gen`, and `FunctionCommit` of our construction (see Sect. 3). In our construction, the verification key consists of the public key and the function commitment, i.e.  $\text{vk} = (\text{pk}, \text{F})$  and the evaluation key  $\text{ek}$  is just the multi-labeled program  $\mathcal{P}_\Delta$ . Assume the client outsourced its secret data to a distributed storage system, i.e. it computed for each secret  $\mathbf{m}_i$  a polynomial  $P_i(x)$  and sent  $\phi_j(\mathbf{m}_i) = P_i(j)$  to shareholder  $j$ . To allow the shareholders to perform operations on this data, for each secret  $\mathbf{m}_i$  for  $i = 1, \dots, n$  it first chooses a random value  $r_i$  and sends a corresponding share  $\phi'_j(r_i) = P'_i(j)$  to shareholder  $j$ .

For `ProbGen`, the secret key  $\text{sk}$  is used by the client to run `PrivateCommit` of our construction computing a public value  $\sigma_{\mathbf{m}_i}$  in form of an authenticator  $\mathbf{A}_i$  for the pair  $(\mathbf{m}_i, r_i)$ . Then, the client sends this value to a dedicated shareholder. The authenticators are unconditionally hiding, i.e. they reveal no information about secrets  $\mathbf{m}_i$  nor randomness  $r_i$  even to a computationally unbounded attacker. The share  $P_i(j)$  and the authenticator  $\mathbf{A}_i$  is in our construction the encoding required by `ProbGen` with no decoding value needed.

For `Compute`, a distinct principal (or the client) gives program  $\mathcal{P}_\Delta = ((\mathbf{f}, \tau_1, \dots, \tau_n), \Delta)$  to the shareholders. Since a majority of storage servers is assumed to be honest (this is a common assumption), privacy-violating functions can be denied. Each shareholder  $j$  performs program  $\mathcal{P}_\Delta$  by evaluating  $\mathbf{f}$  on its shares, i.e. it computes shares  $\phi_j(\mathbf{m}) = \mathbf{f}(\phi_j(\mathbf{m}_1), \dots, \phi_j(\mathbf{m}_n))$  and  $\phi'_j(\mathbf{r}) = \mathbf{f}(\phi'_j(r_1), \dots, \phi'_j(r_n))$ . Furthermore, the dedicated shareholder computes an authenticator for the result by performing `Eval` of our construction on  $\mathbf{A}_1, \dots, \mathbf{A}_n$ . The shareholders then use their shares to reconstruct [30] the

claimed outcome of the function evaluation  $m$  and  $r$ , and call `PublicCommit` with  $m$  and  $r$  to obtain a corresponding commitment  $C$ . These commitments are linearly homomorphic, and the shareholders can construct  $C$  by reconstructing  $m$  and  $r$  in the exponent.

For `Verify`, anyone can run `FunctionVerify` with  $C$ ,  $A$ ,  $F$ , and  $\Delta$  as input, in order to prove that  $C$  is a commitment to the correct solution. If output privacy is not desired,  $m$  and  $r$  can be made public; this allows checking that these are opening values to  $C$  by calling `PublicDecommit` of our construction.

## 4.2 Properties

**Theorem 7.** *The verifiable computing scheme for shared data presented above provides correctness, public verifiability, security, input privacy w.r.t. the servers, output privacy w.r.t the servers, input privacy w.r.t. the verifier, and output privacy w.r.t. the verifier.*

*Proof.* These properties mainly follow from the properties of our FDC:

**Correctness** follows directly from the evaluation correctness of FDCs.

**Public verifiability** follows from the construction of algorithm `Gen` of FDCs.

The output of this algorithm are two keys, a secret key to generate the authenticators and a public key to generate commitments to messages and functions.

**Security** follows from the unforgeability of our FDC.

**Input privacy w.r.t. the servers** follows from using multi-party computation, where each shareholder independently computes the function on its shares. Our scheme offers input privacy against an adversary actively corrupting at most  $t - 1$  shareholders, while unforgeability holds even if the adversary can actively corrupt *all* shareholders. We prove privacy w.r.t. the servers by showing that a simulator  $\mathcal{S}$  can simulate the protocol without needing to know any input values. Assume that  $s_j(m) = (s_j(m[1]), \dots, s_j(m[T]))$  and the simulator  $\mathcal{S}$  stores each  $x_j$ , where  $H_j = g_1^{x_j}$ . Then, the simulator chooses  $r_i \in \mathbb{F}_p$  uniformly at random for  $i = 1, \dots, n$  and gives `PrivateCommit`( $sk, 0, r_i, \Delta, \tau$ ),  $s_j(0), s_j(r_i)$  to the adversary  $\mathcal{A}$  for  $i = 1, \dots, n, j = 1, \dots, t - 1$ . Afterwards,  $\mathcal{A}$  outputs  $A^*$  and  $t - 1$  shares of the result  $(\hat{m}, \hat{r})$ .  $\mathcal{S}$  can produce shares that reconstruct to  $\hat{m}$  and use its knowledge of each  $x_j$  to find an opening  $(\hat{m}, \hat{r})$  such that `PublicCommit`( $\hat{m}, \hat{r}$ ) satisfies `FunctionVerify`( $pk, \text{Eval}(\mathcal{P}_\Delta, \text{PrivatCommit}(sk, m_1, r_1, \Delta, \tau_1), \dots, \text{PrivatCommit}(sk, m_n, r_n, \Delta, \tau_n)), \text{PublicCommit}(\hat{m}, \hat{r}), \text{FunctionCommit}(pk, \mathcal{P}), \Delta) = 1$ .

**Output privacy w.r.t. the servers** follows directly from the input privacy w.r.t. the servers and the unconditional hiding property of the public commitments.

**Input privacy w.r.t. the verifier** is derived as follows. We show that a simulator  $\mathcal{S}$  given access to the secret key  $sk$ , a message  $\hat{m}$ , and randomness  $\hat{r}$  can compute the authenticator  $\hat{A} = (\sigma_\Delta, Z, \hat{U}, \hat{V})$  to the outcome  $(\hat{m}, \hat{r})$  of a computation  $\mathcal{P}_\Delta = ((f, \tau_1, \dots, \tau_n), \Delta)$  without needing to know any valid

input values. It first computes  $z = F_K(\Delta)$  with the pseudorandom function  $F$  and calculates  $Z = g_2^z$ . Then, it binds  $Z$  to the dataset identifier  $\Delta$  by concatenating both and signing it, i.e.  $\sigma_\Delta = \text{Sign}(\text{sk}_{\text{Sig}}, \Delta \parallel Z)$ . Then, it computes  $u_i = F_{K'}(\Delta \parallel i)$ , and  $U_i = g_1^{u_i}$  for  $i = 1, \dots, n$ . Afterwards it computes  $\hat{U} = \prod_{i=1}^n U_i^{f_i}$ . It sets  $\hat{V} = (\hat{U} \cdot \prod_{i=1}^n \hat{H}_i^{f_i} \cdot H_0^{y \hat{r}} \cdot \prod_{j=1}^T H_j^{y \hat{m}[j]})^{\frac{1}{z}}$ . It returns authenticator  $\hat{A} = (\sigma_\Delta, Z, \hat{U}, \hat{V})$ . By construction, this is the authenticator created by `Eval` for any authenticators to valid input values  $(m_j, r_j)$ . Therefore an authenticator created by `Eval` hides the input values perfectly, i.e. even against a computationally unbounded adversary.

**Output privacy w.r.t. the verifier** follows directly from the input privacy w.r.t. the verifier and the unconditional hiding property of the public commitments.

## 5 Related Work

*Commitments:* Commitment schemes are a convenient tool to add verifiability to various processes, such as secret sharing [25], multi-party computation [6], or e-voting [22]. The most well-known and widely used commitment schemes used to provide verifiability are Pedersen’s commitments [25]. However, this work presents the first *function-dependent* commitment scheme. Unlike previous commitment schemes, our solution provides succinctness and amortized efficiency. Furthermore, function-dependent commitments support messages stored in datasets and thus enables a much more expressive notion of public verifiability and more rigorous definition of forgery. Besides, a secure bulletin board is not required for our solution. In [21], the notion of *functional commitments* is introduced. Their notion of function bindingness, however, is strictly weaker than our notion of adaptive unforgeability. The instantiation proposed supports linear functions on field elements, i.e. vectors of length 1, while we support vectors of arbitrary polynomial length. Furthermore, notions such as amortized efficiency and succinctness are not considered. In commitment based audit schemes authenticity is typically achieved by using a secure bulletin board [16], for which finding secure instantiations has been challenging so far.

*Homomorphic authenticators:* Homomorphic authenticators have been proposed both in the secret key setting, as homomorphic MACs (e.g. [1, 4, 10, 31]), and in the public key setting as homomorphic signatures (e.g. [2, 11, 13, 14, 26]). In contrast, our approach additionally considers information-theoretic privacy. Most existing constructions do not consider output privacy. In [26] a solution is proposed in the random oracle model for computational output privacy. Our construction is the first to achieve this in an information-theoretic sense.

*Catalano–Fiore–Nizzardo homomorphic signature scheme* [11]: Both our FDC and the homomorphic signature scheme presented in [11] are based on the FDHI assumption, and indeed our FDC builds on this homomorphic signature scheme. The Catalano–Fiore–Nizzardo construction is context hiding, i.e. signatures to

the output of a function do not leak information about the inputs to the function beyond knowledge of the output to a third party verifier. By contrast, our FDC achieves an even stronger privacy property: information-theoretic input–output privacy with respect to both verifiers and servers. A freshly signed signature in the case of [11] still reveals information about the message to an adversary corrupting a server. Our unconditionally hiding FDC, however, does not. Furthermore, our verification algorithm `FunctionVerify` only requires a commitment to the output of a computation, enabling output privacy, while verification in [11] requires the output itself. This called for a novel strategy in our security reduction.

*Functional cryptography:* Functional dependencies in general were, until now, only available for primitives required to be either hiding, i.e. functional encryption, or binding, i.e. functional signatures. Our notion of FDCs is both binding and (depending on the instantiation, even unconditionally) hiding. Functional encryption and functional signatures have been used to build verifiable computing schemes. However, the functional-encryption-based solution proposed by Parno et al. [24] does not provide privacy nor public verifiability. The solution by Barbosa and Farshim [5] makes use of additional primitives, such as predicate encryption schemes for general predicates for which no efficient construction is available. Furthermore, this solution only provides computational input privacy with respect to the verifier. For functional signatures, only one verifiable computing scheme has been proposed by Boyle et al. [9]; it does not provide any privacy.

*Verifiable computation:* There are many more verifiable computing schemes using proof- and argument-based systems, or based on fully homomorphic encryption. However, there are only few approaches that address public verifiability and input privacy. There are also argument-based verifiable computing schemes available that provide public verifiability and statistical input privacy towards the verifier [3, 7, 15, 23, 29]. All of these are based on strong, so called non-falsifiable assumptions [20]. However, these verifiable computing schemes are not tailored to perform computations on secret shares. Schoenmakers and Veeningen show [28] how to achieve this, but they demand that every shareholder performs computations in order to allow for verification and thereby produces a significant overhead. Our verifiable computing scheme for shared data allows to process secret shares while only one storage server has to compute the audit data. Furthermore, since our solution only makes use of FDC and signatures we are able to provide a concrete instantiation of this approach using our FDC construction. The resulting scheme provides public verifiability, unconditional input privacy towards the servers and the verifier, and relies on standard assumptions.

*Multi-party computation:* Regarding multi-party computation, only two schemes enable a publicly verifiable audit trail. They have been proposed by Baum et al. [6] and Schabhüser et al. [27]. Unlike our approach, they achieve public verifiability for arbitrary arithmetic circuits. However, our approach is the

first with amortized efficiency. The client incurs setup costs, but setup is only performed once. Afterwards, multiple functions can be evaluated and verified. The verification process itself is more efficient than performing the operations locally for a suitably large number of inputs  $n$ .

## 6 Conclusion

In this paper, we introduced a novel approach to guarantee data authenticity in delegated computing settings. Our function-dependent commitments enable fast correctness verification, proof of origin, and information-theoretic input-output privacy. We also provided a concrete instantiation of this generic construction for linear functions. Using this instantiation, we introduced a verifiable computing scheme for shared data with unconditional privacy both towards the server and the verifier. Furthermore, this scheme only requires a single party to perform the computationally more costly computation of the authenticators. Our instantiation does not require revealing a computation's outcome, but merely a commitment to it, thus also satisfying information-theoretic output privacy.

*Future Work:* Our FDC instantiation adds verifiability and authenticity to applications while maintaining privacy — even unconditionally. We intend to further analyse this impact of FDCs on applications processing sensitive data. To this end, we plan to examine the composability of FDCs and investigate black-box constructions of FDC-based verifiable multi-party computation schemes.

**Acknowledgments.** This work has been co-funded by the DFG as part of project “Long-Term Secure Archiving” within CRC 1119 CROSSING. It has also received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement 644962.

## References

1. Agrawal, S., Boneh, D.: Homomorphic MACs: MAC-based integrity for network coding. In: Abdalla, M., Pointcheval, D., Fouque, P.-A., Vergnaud, D. (eds.) ACNS 2009. LNCS, vol. 5536, pp. 292–305. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-01957-9\\_18](https://doi.org/10.1007/978-3-642-01957-9_18)
2. Attrapadung, N., Libert, B.: Homomorphic network coding signatures in the standard model. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 17–34. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-19379-8\\_2](https://doi.org/10.1007/978-3-642-19379-8_2)
3. Backes, M., Barbosa, M., Fiore, D., Reischuk, R.M.: ADSNARK: nearly practical and privacy-preserving proofs on authenticated data. In: SP 2015, pp. 271–286. IEEE Computer Society (2015)
4. Backes, M., Fiore, D., Reischuk, R.M.: Verifiable delegation of computation on outsourced data. In: CCS 2013, pp. 863–874. ACM (2013)
5. Barbosa, M., Farshim, P.: Delegatable homomorphic encryption with applications to secure outsourcing of computation. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 296–312. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-27954-6\\_19](https://doi.org/10.1007/978-3-642-27954-6_19)



6. Baum, C., Damgård, I., Orlandi, C.: Publicly auditable secure multi-party computation. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 175–196. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10879-7\\_11](https://doi.org/10.1007/978-3-319-10879-7_11)
7. 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). [https://doi.org/10.1007/978-3-642-40084-1\\_6](https://doi.org/10.1007/978-3-642-40084-1_6)
8. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. *SIAM J. Comput.* **32**(3), 586–615 (2003)
9. 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). [https://doi.org/10.1007/978-3-642-54631-0\\_29](https://doi.org/10.1007/978-3-642-54631-0_29)
10. Catalano, D., Fiore, D., Gennaro, R., Nizzardo, L.: Generalizing homomorphic MACs for arithmetic circuits. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 538–555. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-54631-0\\_31](https://doi.org/10.1007/978-3-642-54631-0_31)
11. Catalano, D., Fiore, D., Nizzardo, L.: Programmable hash functions go private: constructions and applications to (homomorphic) signatures with shorter public keys. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 254–274. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48000-7\\_13](https://doi.org/10.1007/978-3-662-48000-7_13)
12. Catalano, D., Fiore, D., Nizzardo, L.: Programmable Hash Functions go Private: Constructions and Applications to (Homomorphic) Signatures with Shorter Public Keys. *IACR Cryptology ePrint Archive* 2015, 826 (2015)
13. Catalano, D., Fiore, D., Warinschi, B.: Efficient network coding signatures in the standard model. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 680–696. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-30057-8\\_40](https://doi.org/10.1007/978-3-642-30057-8_40)
14. Catalano, D., Fiore, D., Warinschi, B.: Homomorphic signatures with efficient verification for polynomial functions. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 371–389. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44371-2\\_21](https://doi.org/10.1007/978-3-662-44371-2_21)
15. Costello, C., Fournet, C., Howell, J., Kohlweiss, M., Kreuter, B., Naehrig, M., Parno, B., Zahur, S.: Geppetto: versatile verifiable computation. In: SP 2015, pp. 253–270. IEEE Computer Society (2015)
16. Culnane, C., Schneider, S.A.: A peered bulletin board for robust use in verifiable voting systems. In: CSF, pp. 169–183. IEEE Computer Society (2014)
17. Demirel, D., Schabhüser, L., Buchmann, J.A.: Privately and Publicly Verifiable Computing Techniques: A Survey. *Springer Briefs in Computer Science*. Springer, Heidelberg (2017). <https://doi.org/10.1007/978-3-319-53798-6>
18. Freeman, D.M.: Improved security for linearly homomorphic signatures: a generic framework. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 697–714. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-30057-8\\_41](https://doi.org/10.1007/978-3-642-30057-8_41)
19. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-14623-7\\_25](https://doi.org/10.1007/978-3-642-14623-7_25)
20. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: STOC, pp. 99–108. ACM (2011)

21. Libert, B., Ramanna, S.C., Yung, M.: Functional commitment schemes: from polynomial commitments to pairing-based accumulators from simple assumptions. In: ICALP 2016. LIPIcs, vol. 55, pp. 30:1–30:14. Schloss Dagstuhl (2016)
22. Moran, T., Naor, M.: Receipt-free universally-verifiable voting with everlasting privacy. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 373–392. Springer, Heidelberg (2006). [https://doi.org/10.1007/11818175\\_22](https://doi.org/10.1007/11818175_22)
23. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: nearly practical verifiable computation. In: SP 2013, pp. 238–252. IEEE Computer Society (2013)
24. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: verifiable computation from attribute-based encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-28914-9\\_24](https://doi.org/10.1007/978-3-642-28914-9_24)
25. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). [https://doi.org/10.1007/3-540-46766-1\\_9](https://doi.org/10.1007/3-540-46766-1_9)
26. Schabhüser, L., Buchmann, J., Struck, P.: A linearly homomorphic signature scheme from weaker assumptions. In: O’Neill, M. (ed.) IMACC 2017. LNCS, vol. 10655, pp. 261–279. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-71045-7\\_14](https://doi.org/10.1007/978-3-319-71045-7_14)
27. Schabhüser, L., Demirel, D., Buchmann, J.A.: An unconditionally hiding auditing procedure for computations over distributed data. In: CNS 2016, pp. 552–560. IEEE (2016)
28. 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). [https://doi.org/10.1007/978-3-319-28166-7\\_1](https://doi.org/10.1007/978-3-319-28166-7_1)
29. Schoenmakers, B., Veeningen, M., de Vreede, N.: Trinocchio: privacy-preserving outsourcing by distributed verifiable computation. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) ACNS 2016. LNCS, vol. 9696, pp. 346–366. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-39555-5\\_19](https://doi.org/10.1007/978-3-319-39555-5_19)
30. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
31. Zhang, L.F., Safavi-Naini, R.: Generalized homomorphic MACs with efficient verification. In: ASIAPKC 2014, pp. 3–12. ACM (2014)



# On Constructing Pairing-Free Identity-Based Encryptions

Xin Wang<sup>1,2</sup>, Bei Liang<sup>3</sup>, Shimin Li<sup>1,2</sup>, and Rui Xue<sup>1,2</sup>(✉)

<sup>1</sup> State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China  
{wangxin9076,lishimin,xuerui}@iie.ac.cn

<sup>2</sup> School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

<sup>3</sup> Chalmers University of Technology, Gothenburg, Sweden  
lbei@chalmers.se

**Abstract.** In this paper, we focus on constructing IBE from hardness assumptions without pairings. Especially, we propose two IBE schemes that are provably secure under new number theoretic assumptions over the group  $\mathbb{Z}_{N^2}^*$ , in the Random Oracle (RO) model. We essentially take advantage of the underlying algebraic structure to overcome the difficulties in devising an IBE scheme.

More precisely, our contributions are two-fold and can be summarised as follows: (i) We give two concrete pairing-free constructions of IBE based on a variant of DDH assumption and Paillier's DCR assumption respectively over the group  $\mathbb{Z}_{N^2}^*$ . These schemes are quite efficient and easily to be proven IND-ID-CPA in the random oracle model. (ii) We also provide a generic construction of selectively secure IBE from DDH group with a DL-solvable subgroup in the standard model by employing puncturable PRFs and indistinguishability obfuscation.

**Keywords:** Identity-based encryption · Pairing  
Number-theoretic assumption · Random Oracle  
Quadratic residuosity · Diffie-Hellman

## 1 Introduction

Identity-based encryption (IBE) initiated by Shamir [21] is a public-key mechanism, where given short public parameters arbitrary strings could serve as valid public keys. The first practical IBE scheme was realized by Boneh and Franklin [6] based on bilinear maps and the security of this construction was argued in the random oracle model [2]. Soon after, a widespread interest in realizing IBE schemes based on various assumptions has been aroused. Roughly, according to the based-on assumptions of existing constructions of IBE this body of work can be divided into three directions.

The first aims to build on various assumptions on groups with a bilinear map (pairing), e.g., [4–6, 9, 22, 23]. However, the implementation of pairing itself

is much more expensive than traditional number theoretic operations, e.g., modular exponentiation.

The second pursues on constructing IBE based on learning-with-errors (LWE) assumption instead of bilinear pairing, which is started from the work of Gentry et al. [15]. Since the first LWE-based IBE scheme in [15] is shown to be secure in the random oracle model, several other LWE-based IBE schemes meeting the security in standard model are proposed subsequently [1, 10]. Lattice-based IBE schemes behave better in encryption and decryption procedures than pairing-based ones, and have prominent feature of resisting quantum attacks. However, the somewhat large public parameter size and private key size substantially restrict their deployment in space-sensitive applications, e.g., mobile devices.

The third direction seeks to construct on a more traditional number theoretic problem, namely the Quadratic Residuosity (QR) assumption (in the random oracle model). Cocks [12] provided the first elegant QR-based IBE scheme of which the security proof also relies on the random oracle model. However, the encryption of an  $\ell$ -bit message is of size  $2\ell \log_2 N$ , which implies Cocks's IBE scheme is not space efficient. From then on, constructing an IBE scheme with short ciphertext avoiding pairings turns into an interesting problem. In the following work, Boneh et al. [7] proposed a more efficient QR-based IBE scheme in the RO model, which reduces the length of ciphertext in Cocks's scheme by expanding an  $\ell$ -bit message to a ciphertext of size  $\ell + \log_2 N$ . Very recently, constructions of IBE based on the hardness of the (Computational) Diffie-Hellman (CDH) Problem and Factoring have also be obtained. The great work of Döttling and Garg [13] proposed a candidate construction of IBE scheme from the CDH assumption by a novel tree based approach. Their constructions bypass the known impossibility results [3] through delicately embedding chameleon encryptions and public key encryptions into several garblings of circuits. However, the non-black-box use of underlying primitives inside the garbled circuits makes the ciphertext size really large, also the encryption and decryption procedures thereby inevitably become a main bottleneck to implement in practice. They leave the problem of constructing an efficient IBE scheme from the Diffie-Hellman Assumption as an open problem.

In this work, we proceed the study on the third direction. Since we have observed the drawbacks above and hardness of constructing *efficient* QR-based (in the RO model) and DH-based IBE schemes (in the standard model), we ask the following question:

*Can we build efficient IBE schemes on other traditional pairing-free number theoretic problems, such as Decisional Composite Residuosity problem or Decisional Diffie-Hellman problem in the random oracle model?*

**Our Contributions.** In this paper, we give positive answer to the above question. We give two concrete constructions of IBE schemes based on a variant of DDH assumption [8] and Paillier's DCR [18] assumption respectively over the group  $\mathbb{Z}_{N^2}^*$ . These schemes are quite efficient and easily to be proven IND-ID-CPA

in the random oracle model. We also provide a generic construction of selectively secure IBE from DDH group with a DL-solvable subgroup in the standard model.

In Table 1, we compare our concrete constructions with existing high performance pairing-based IBE schemes. Pairing-based schemes are equipped with a bilinear group pair  $(G, G)$  where a bilinear map  $e : G \times G \rightarrow G_T$  exists. The second column shows the underlying assumptions. The third to fifth columns list the public parameter size, plaintext length and ciphertext length respectively which are measured by numbers of specific group elements consumed by each schemes. The notation  $|\cdot|$  where  $\cdot$  is one of  $(G, G_T, \mathbb{Z}_{N^2}^*)$  stands for a single element in a concrete group. The sixth and seventh columns show the time of basic operations in encryption and decryption. The basic operations including exponentiation (Exp) and pairing which occur on bilinear group, modular exponentiation (Mod Exp) which occurs on group  $\mathbb{Z}_{N^2}^*$ . As can be seen from the table, our schemes are efficient in that the encryption only takes 2 modular exponentiations and the decryption takes one, whereas the pairing-based schemes not only require a number of exponentiation operations (w.r.t.  $G$  or  $G_T$ ), but more or less depend on several pairing operations as well.

**Table 1.** Comparison with existing pairing-based IBEs

	Assumption	Parameter size	Plaintext length	Ciphertext length	Encryption	Decryption	Random Oracle
BF01 [6]	BDH	$2 G $	$ G_T $	$ G  +  G_T $	2 Exp, 1 Pairing	1 Exp, 1 Pairing	✓
BB04 [4]	BDH	$3 G  +  G_T $	$ G_T $	$2 G  +  G_T $	3 Exp, 1 Pairing	2 Pairing	✗
This paper	sDDH	$\log_2 N +  \mathbb{Z}_{N^2}^* $	$ \mathbb{Z}_{N^2}^* $	$2 \mathbb{Z}_{N^2}^* $	2 Mod Exp	1 Mod Exp	✓
	DCR	$\log_2 N +  \mathbb{Z}_{N^2}^* $	$\log_2 N$	$2 \mathbb{Z}_{N^2}^* $	2 Mod Exp	1 Mod Exp	✓

**Overview of our Techniques.** Here we briefly present the roadmap to achieve our pairing-free IBE construction (in the RO model). The core idea of constructing an IBE scheme is to look for a mechanism that a private key corresponding to any identity could be derived by a trusted third party, usually known as private key generator (PKG). Consider the classical ElGamal encryption:  $(g^r, h^r \cdot m)$  where the public key is a group element  $h = g^a$  for a secret exponent  $a$ . Given different secret key  $a_i$  the public/secret key pair  $(g^{a_i}, a_i)$  is different, which in turn could be regarded as an identity (public key) and its corresponding user key (secret key). Thus, if we are able to merge exponentially many public/secret key pairs into one system that could be managed by a PKG (using master secret key), then a IBE system which performs as efficiently as the original separated encryption systems is achieved. However, starting from this intuitive point of view, when we set  $g^a$  as the hash value for identity ID, say  $H(\text{ID})$ , an intractable problem is how to extract the secret key  $a$  from  $H(\text{ID})$  which afterwards will be used as the decryption key. Since solving the discrete logarithm problem is hard in a cyclic group of prime order, we seek to find a set that makes it possible to extract discrete logarithm.

We require two ingredients to establish our IBE cryptosystem. The first one is a large set which is embedded in a DDH hard cyclic group and allows for extracting exponentially many discrete logarithms *privately* with the aid of a trapdoor<sup>1</sup>. We refer to this set as the DL-*solvable* set. It is essential to not rely on any algorithms that are designed to resolve the longstanding discrete logarithm problem as they are time-consuming and not suitable for our IBE scenario. In other words, this “extracting” process should be accomplished *efficiently* by the PKG who owns a master secret key. The second one is a secure encoding method or hash function that could transform any fixed-length identity from a bitstring into a member of the DL-solvable set. Based on these two groundwork, thereafter we could successfully implement our IBE constructions.

Our IBE constructions make use of a cryptographic group  $\mathbb{Z}_{N^2}^*$ , which has been well investigated because of the elegant work of Paillier [18]. The central idea of realizing the IBE scheme in the  $\mathbb{Z}_{N^2}^*$  group is that we can solve partial discrete logarithm in this group if we know the factorization of  $N$ . More precisely, if an ElGamal type encryption scheme is built on a cyclic subgroup that is set up in  $\mathbb{Z}_{N^2}^*$ , then the ciphertext is composed of  $([g^r]_{N^2}, [h^r \cdot M]_{N^2})$ . Applying our intuitive idea of setting the public/secret key pair, the hash value  $H(\text{ID})$  for an identity ID plays the role of value  $h$ , and the private key  $d_{\text{ID}}$  for ID is set as the exponent value of  $H(\text{ID})$  by solving the discrete logarithm using the factorization of  $N$ . In consequence, the IBE scheme resulting from the ElGamal scheme is as following:  $C = ([g^r]_{N^2}, [H(\text{ID})^r \cdot M]_{N^2})$  and  $d_{\text{ID}} = x$  where  $g^x = H(\text{ID})$ . In this case, the IBE scheme can be easily designed and the security of the scheme can be proven by the DDH assumption.

However, in order to design an IBE scheme as described above, it should be possible to design hash function  $H(\text{ID})$  and solve the discrete logarithm of  $H(\text{ID})$  using trapdoor information. To settle this problem, we define a DL-solvable set  $G_s := \{g^1, g^2, \dots, g^{N-1}\}$  and embed the value  $x$  into a new exponent where  $x$  can be seen as a partial discrete logarithm. Using the helpful property in the set  $G_s$  that the partial discrete logarithm of the subgroup  $\langle 1 + N \rangle$  is inherited by using  $[g^{\lambda(N)}]_{N^2} = 1 + N$ , where  $\lambda(N)$  stands for the Carmichael’s function, we could derive the private key  $d_{\text{ID}}$ . One may wonder whether a possible way of mapping into the subset  $G_s$  even exists since it seems hard to restrict the image of hash functions within the sets generated by  $g$ . We demonstrate the possibility of mapping identity  $\text{ID} \in \{0, 1\}^*$  to an element in set  $G_s$  by designing a group hash function, namely  $\mathcal{H} : \{0, 1\}^* \rightarrow G_s$  in the group  $\mathbb{Z}_{N^2}^*$ . The group hash function is highly structured, and may not have strong properties of simulating a random oracle. Unfortunately, we can only prove the security of our scheme assuming “some” hash function could be served as a random oracle. Removing the random oracle would be a challenging work.

Our second construction is based on the Decisional Composite Residuosity (DCR) assumption. Recall that DCR assumption states that uniform

---

<sup>1</sup> Of course, this cannot be realized in a prime order group due to the hardness of discrete logarithm problem, instead we can choose a composite order group with unknown order.

distribution over the group  $\mathbb{Z}_{N^2}^*$  is computationally indistinguishable from that over the subgroup of  $N$ -th power residues. As none of these two indistinguishable groups is cyclic, it's hard to adapt our previous technique to the DCR-based scheme in the same way. Fortunately, we can further take advantage of the underlying  $\mathbb{Z}_{N^2}^*$  group structure in the case that the factors of  $N$  are safe primes. Here we denote by  $\mathbb{G}$  and  $\mathbb{G}_N$  the quadratic residues of  $\mathbb{Z}_{N^2}^*$  and  $2N$ -th power residues of  $\mathbb{Z}_{N^2}^*$  respectively, both of which are cyclic subgroups of  $\mathbb{Z}_{N^2}^*$ . Under this circumstance, the DCR assumption indicates that to distinguish the uniform distribution over  $\mathbb{G}$  from the uniform distribution over  $\mathbb{G}_N$  is computationally infeasible [16]. To construct an IBE scheme, we first design a PKE scheme where the ciphertext is  $(\lceil g^r \rceil_{N^2}, \lceil h^r \cdot (1+N)^M \rceil_{N^2})$ . Now the element  $g$  is a generator of the subgroup  $\mathbb{G}_N$  instead of  $\mathbb{G}$ . If  $h$  is a uniform and random element in  $\mathbb{G}_N$ , the scheme is proven secure by DCR assumption. The resulted IBE scheme is as:  $C = (\lceil g^r \rceil_{N^2}, \lceil H(\text{ID})^r \cdot (1+N)^M \rceil_{N^2})$  and  $d_{\text{ID}} = x$  where  $g^x = H(\text{ID})$ . But the problem arises that it seems not possible to solve the discrete logarithm of  $H(\text{ID})$  with a trapdoor if we simply hash onto the cyclic group  $\mathbb{G}_N$ , since the order of this group is a multiplication of two primes, which ensures that discrete logarithm problem is evidently hard. So as to potentially use the underlying structure of group itself to help us reverse the secret  $x$ , we define a DL-solvable set  $G_s := \{[g(1+N)]^1, [g(1+N)]^2, \dots, [g(1+N)]^{\frac{N-1}{4}}\}$  where  $\lceil g(1+N) \rceil_{N^2}$  is a generator of  $\mathbb{G}$ . First observe that it is solvable by the master secret key owner as he/she can cancel out the  $g^\alpha$  term. Furthermore, if we lift this set up to  $N$ -th power, the  $1+N$  part of each element vanishes. We hence give a IBE construction:  $(\lceil g^{Nr} \rceil_{N^2}, \lceil H(\text{ID})^{Nr} \cdot (1+N)^M \rceil_{N^2})$  assuming the existence of hash function  $H : \{0, 1\}^* \rightarrow G_s$ .

Both of our constructions over the group  $\mathbb{Z}_{N^2}^*$  rely on a security proof in the random oracle model. In an IBE security game, the simulator should answer the key extraction queries without the factorization of the modulus. We mainly take advantage of the programmability of RO to resolve this problem.

We also give a generic way to construct a selectively secure IBE starting with DDH group with a DL-solvable subgroup [11] in the standard model assuming the existence of  $i\mathcal{O}$  and puncturable PRF. The new aspect of our  $i\mathcal{O}$ -based construction is our instantiation of the hash function for any identity ID where a PRF is applied to a fixed-length identity to get a pseudorandomness value which is corresponding to a secret exponent of a public element in the DL-solvable set.

## 2 Preliminaries

In what follows we will denote with  $\kappa \in \mathbb{N}$  a security parameter. We denote by  $\text{negl}(\cdot)$  a function negligible in some parameter. We write  $x \leftarrow X$  for sampling  $x$  from the set  $X$  uniformly at random.  $[N]$  denotes a set  $\{1, 2, \dots, N\}$  where  $N$  is a positive integer and  $\lceil \cdot \rceil_n$  stands for an integer modulo  $n$ . We denote by  $U(S)$  the uniform distribution on a finite set  $S$ . If two distributions  $X, Y$  are taken values from a same finite set  $\Omega$ , the statistical distance  $\Delta(X, Y)$  of distributions



$X$  and  $Y$  is defined as:

$$\Delta(X, Y) = \frac{1}{2} \sum_{\omega \in \Omega} |\Pr_{x \leftarrow X}[x = \omega] - \Pr_{y \leftarrow Y}[y = \omega]|.$$

Below we recall the notions of some cryptographic primitives. For lack of space, we leave the definition of indistinguishability obfuscation and puncturable pseudorandom functions in Appendix A.

### 2.1 DDH Group with a DL-Solvable Subgroup

The formal definition of DDH group with an easy DL subgroup was initiated by Castagnos and Laguillaumie [11]. We slightly modify their definition below by adding a new assumption named sDDH which was already considered in the context of the group  $\mathbb{Z}_{N^2}^*$  [8]. Please refer to [11] for more details.

**Definition 1.** We define a DDH group with a DL-solvable subgroup as a pair of algorithms (Gen, Solve). The Gen algorithm is a group generator which takes as input two parameters  $\lambda$  and  $\mu$  and outputs a tuple  $(B, n, p, s, g, f, G, F)$ . The integers  $B, n, p$  and  $s$  are such that  $s$  is a  $\lambda$ -bit integer,  $p$  is a  $\mu$ -bit integer,  $\gcd(p, s) = 1$ ,  $n = p \cdot s$  and  $B$  is an upper bound for  $s$ . The set  $(G, \cdot)$  is a cyclic group of order  $n$  generated by  $g$ , and  $F \subset G$  is the subgroup of  $G$  of order  $p$  and  $f$  is a generator of  $F$ . The upper bound  $B$  is chosen such that the distribution induced by  $\{g^r, r \leftarrow \{0, \dots, Bp-1\}\}$  is statistically indistinguishable from  $U(G)$ . We assume that: The DL problem is easy in  $F$ . The Solve algorithm is a deterministic polynomial time algorithm that solves the discrete logarithm problem in  $F$ . Especially, the small DDH (sDDH) problem is hard in  $G$  given access to the Solve algorithm. That is:

$$\Pr \left[ \begin{array}{l} (B, n, p, s, g, f, G, F) \leftarrow \text{Gen}(1^\lambda, 1^\mu), x, z \leftarrow \mathbb{Z}_n, y \leftarrow \mathbb{Z}_p, \\ b = b^* : X = g^x, Y = g^y, b \leftarrow \{0, 1\}, Z_0 = g^z, Z_1 = g^{xy}, \\ b^* \leftarrow \mathcal{A}(B, p, g, f, G, F, X, Y, Z_b, \text{Solve}(\cdot)) \end{array} \right] - \frac{1}{2}$$

is negligible for all probabilistic polynomial time adversary  $\mathcal{A}$ .

### 2.2 Identity-Based Encryption

**Definition 2 (Identity-Based Encryption (IBE) [6]).** An identity-based encryption scheme with message space  $\mathcal{M}$  consists of four algorithms (Setup, Extract, Enc, Dec):

- Setup( $1^\kappa$ ): outputs public parameters  $\text{params}$  and a master secret key  $\text{msk}$ .
- Extract( $\text{msk}, \text{ID}$ ): outputs the secret key  $d_{\text{ID}}$  for identity  $\text{ID}$ .
- Enc( $\text{params}, \text{ID}, M$ ): outputs a ciphertext  $C$ .
- Dec( $C, d_{\text{ID}}$ ): outputs message  $M$  encrypted in  $C$ .



The correctness of the scheme requires that when  $d_{ID} = \text{Extract}(\text{msk}, ID)$ , then the following holds:

$$\forall M \in \mathcal{M} : \text{Dec}(\text{Enc}(\text{params}, ID, M), d_{ID}) = M$$

**Definition 3 (IND-ID-CPA Security).** We say that an IBE scheme  $\Pi$  is semantically secure against an chosen plaintext attack (IND-ID-CPA) if every PPT adversary  $\mathcal{A}$  has an advantage negligible in  $\kappa$  in the following game:

**Setup:** A challenger runs the Setup algorithm. It gives  $\mathcal{A}$  the resulting public parameters  $\text{params}$  and keeps the corresponding master secret key  $\text{msk}$ .

**Phase 1:**  $\mathcal{A}$  issues private key extraction queries  $ID_1, \dots, ID_m$ . The challenger responds by running algorithm Extract to generate the private key  $d_i$  corresponding to identity  $ID_i$ . It sends  $d_i$  to  $\mathcal{A}$ . These queries may be asked adaptively.

**Challenge:**  $\mathcal{A}$  outputs two equal length messages  $M_0, M_1$  and an identity  $ID^*$  which has not been queried in Phase 1. The challenger picks a random bit  $b$  and returns to  $\mathcal{A}$  the challenge ciphertext  $C^* = \text{Enc}(\text{params}, ID^*, M_b)$ .

**Phase 2:**  $\mathcal{A}$  issues more extraction queries  $ID_{m+1}, \dots, ID_n$  where  $ID_i \neq ID^*$ . The challenger responds as in Phase 1.

**Guess:** Finally,  $\mathcal{A}$  outputs a guess  $b' \in \{0, 1\}$  and wins the experiment if  $b' = b$ .

A selective-identity (IND-sID-CPA security) game is defined by modifying the above definition to force the adversary to announce the challenge identity  $ID^*$  before the Setup stage. We define the advantage of  $\mathcal{A}$  against the IBE scheme  $\Pi$  as  $|\Pr[b' = b] - \frac{1}{2}|$ .

### 3 Background on $\mathbb{Z}_{N^2}^*$

The group  $\mathbb{Z}_{N^2}^*$  was introduced by Paillier to construct efficient public key encryption schemes and digital signatures. In our setting, we need the modulus to be a little different, i.e., the prime factors of  $N$  are safe primes. In this section, we will mostly follow the notation and description from [8] except some minor changes.

Let  $N = pq$  be an RSA modulus, and  $p, q$  are safe primes, i.e.  $p = 2p' + 1, q = 2q' + 1$ , where  $p', q'$  are also primes. Such a modulus will be called a strong modulus in the remaining part. We will denote by  $\lambda(N)$  the Carmichael's function, i.e.  $\lambda(N) = \text{lcm}(p - 1, q - 1)$ . We denote by  $\mathcal{SP}(\kappa)$  the sets of safe primes of length  $\kappa$ . Consider the subgroup  $\mathbb{QR}_{N^2}$  of  $\mathbb{Z}_{N^2}^*$ , it's the cyclic group of quadratic residues modulo  $N^2$ . From now on, we denote this subgroup by  $\mathbb{G}$ . We denote by  $\mathbb{G}_N$  the subgroup of  $N$ -th power residues of  $\mathbb{G}$ . We have that  $\lambda(N) = 2p'q'$ , and the order of group  $\mathbb{G}$  is  $|\mathbb{G}| = \lambda(N^2)/2 = N\lambda(N)/2 = ppq'q'$ , the order of its subgroup  $\mathbb{G}_N$  is  $|\mathbb{G}_N| = p'q'$ .

#### 3.1 The Partial Discrete Logarithm Problem

Let  $g$  be a generator of  $\mathbb{G}$ . Suppose that  $\lceil g^{\lambda(N)} \rceil_{N^2} = (1 + N)$ . We will assume such a generator  $g$  w.l.o.g for computational simplicity in the remaining article. Such a  $g$  can be obtained with the help of the factorization of  $N$ .

**Proposition 1 (Existence of  $g$ ).** *For a strong modulus  $N$ , there exists a generator  $g$  of  $\mathbb{G}$  with  $\lceil g^{\lambda(N)} \rceil_{N^2} = (1 + N)$ .*

*Proof.* We can sample  $g$  as  $g'^a \cdot (1 + N)^b$  where  $g'$  is a generator of  $\mathbb{G}_N$ ,  $a \leftarrow \lceil p'q' \rceil$  and  $b \in \mathbb{Z}_N^*$  is computed as  $b = \lceil \lambda(N)^{-1} \rceil_N = \lceil (2p'q')^{-1} \rceil_N$  known the factorization of  $N$ . Specifically,  $g'$  can be sampled by  $x \leftarrow \mathbb{Z}_{N^2}^*$  and raising up to  $\lceil x^{2N} \rceil_{N^2}$ . The so obtained  $g$  satisfies:  $\lceil g^{\lambda(N)} \rceil_{N^2} = \lceil (1 + N)^{b\lambda(N)} \rceil_{N^2} = (1 + N)^{\lceil \lambda(N)^{-1} \cdot \lambda(N) \rceil_N} = 1 + N$ .  $\square$

The following assumption posits the hardness of solving the *partial discrete logarithm* [19] of an element in  $\mathbb{G}$  without knowing the factorization of a strong modulus  $N$ :

**Definition 4 (Partial Discrete Logarithm (PDL) over  $\mathbb{Z}_{N^2}^*$ ).** *For every probabilistic polynomial time algorithm  $\mathcal{A}$ , there exists a negligible function  $\text{negl}(\cdot)$  such that:*

$$\Pr \left[ \mathcal{A}(N, g, h) = \lceil a \rceil_N \left| \begin{array}{l} p, q \leftarrow \mathcal{SP}(\kappa); N = pq; \\ g \leftarrow \mathbb{G}; a \leftarrow \llbracket \mathbb{G} \rrbracket; \\ h = \lceil g^a \rceil_{N^2} \end{array} \right. \right] = \text{negl}(\kappa).$$

Paillier illustrated a way of solving the PDL problem if the factorization of  $N$  is provided:

Suppose we have  $h = g^a$ , to extract  $\lceil a \rceil_N$  from  $h$ , we can do:

1. Compute  $C = \lceil h^{\lambda(N)} \rceil_{N^2} = \lceil (1 + N)^a \rceil_{N^2} = \lceil (1 + aN) \rceil_{N^2}$ ;
2. Return the integer  $(C - 1)/N$ .

We will use this algorithm to extract private keys in our IBE schemes.

### 3.2 The Decisional Diffie-Hellman Problem over $\mathbb{Z}_{N^2}^*$

We state the *Decisional Diffie-Hellman Assumption* (DDH) over group  $\mathbb{Z}_{N^2}^*$ .

**Definition 5 (DDH Assumption over  $\mathbb{Z}_{N^2}^*$ ).** *For every probabilistic polynomial time algorithm  $\mathcal{A}$ , there exists a negligible function  $\text{negl}(\cdot)$  such that:*

$$\Pr \left[ \mathcal{A}(N, g, X, Y, Z_b) = b \left| \begin{array}{l} p, q \leftarrow \mathcal{SP}(\kappa); N = pq; \\ g \leftarrow \mathbb{G}; x, y, z \leftarrow \llbracket \mathbb{G} \rrbracket; \\ X = \lceil g^x \rceil_{N^2}; Y = \lceil g^y \rceil_{N^2}; \\ Z_0 = \lceil g^z \rceil_{N^2}; Z_1 = \lceil g^{xy} \rceil_{N^2}; \\ b \leftarrow \{0, 1\}; \end{array} \right. \right] - \frac{1}{2} = \text{negl}(\kappa).$$

### 3.3 The Small Diffie-Hellman Problem over $\mathbb{Z}_{N^2}^*$

Actually, we will use a variant of the Diffie-Hellman Problem proposed by Bresson et al. [8]. That is, given a pair  $(A, B) = (g^a, g^b)$  when  $b$  is relatively small ( $b \in \mathbb{Z}_N$ ), to distinguish  $C = [g^{ab}]_{N^2}$  from a random element in  $\mathbb{G}$ .

The *Small Decisional Diffie-Hellman Assumption* (sDDH) is formalized as follows:

**Definition 6 (Small-DDH Assumption over  $\mathbb{Z}_{N^2}^*$ ).** *For every probabilistic polynomial time algorithm  $\mathcal{A}$ , there exists a negligible function  $\text{negl}(\cdot)$  such that:*

$$\Pr \left[ \begin{array}{c} \mathcal{A}(N, g, X, Y, \\ Z_b) = b \end{array} \middle| \begin{array}{l} p, q \leftarrow \mathcal{SP}(\kappa); N = pq; \\ g \leftarrow \mathbb{G}; x, z \leftarrow [|\mathbb{G}|]; y \leftarrow \mathbb{Z}_N; \\ X = [g^x]_{N^2}; Y = [g^y]_{N^2}; \\ Z_0 = [g^z]_{N^2}; Z_1 = [g^{xy}]_{N^2}; \\ b \leftarrow \{0, 1\}; \end{array} \right] - \frac{1}{2} = \text{negl}(\kappa).$$

### 3.4 DCR Assumption and Underlying Group Structure

**Definition 7 (The Decisional Composite Residuosity (DCR) assumption).** *Let  $N = pq$  be a strong modulus. Consider the finite sets:*

$$P := \{y = [x^N]_{N^2} \mid x \leftarrow \mathbb{Z}_N^*\}$$

$$\mathbb{Z}_{N^2}^* = \{z = [(1 + N)^y x^N]_{N^2} \mid x \leftarrow \mathbb{Z}_N^*, y \leftarrow \mathbb{Z}_N\}$$

The DCR assumption posits that for any probabilistic polynomial time adversary  $\mathcal{A}$ , the advantage

$$\text{Adv}_{\mathcal{A}}^{\text{DCR}} = |\Pr[\mathcal{A}(x, N) = 1 \mid x \leftarrow P] - \Pr[\mathcal{A}(x, N) = 1 \mid x \leftarrow \mathbb{Z}_{N^2}^*]|$$

is negligible.

If  $N$  is a strong modulus, we can decompose  $\mathbb{Z}_{N^2}^*$  as an internal direct product:

$$\mathbb{Z}_{N^2}^* \cong \mathbf{G}_N \cdot \mathbf{G}_{N'} \cdot \mathbf{G}_2 \cdot \mathbf{T}$$

where each group  $\mathbf{G}_\tau$  is a cyclic group of order  $\tau$ , and  $\mathbf{T}$  is the subgroup  $\{-1, 1\}$ . This decomposition is unique, except for the choice of  $\mathbf{G}_2$ . Note that the element  $(1 + N)$  has order  $N$  in  $\mathbb{Z}_{N^2}^*$ , i.e. it generates  $\mathbf{G}_N$ , and that  $[(1 + N)^a]_{N^2} = 1 + aN$  for  $0 \leq a < N$ . Observe that  $P = \mathbf{G}_{N'} \mathbf{G}_2 \mathbf{T}$ ,  $\mathbb{G} = \mathbf{G}_N \mathbf{G}_{N'}$  and  $\mathbb{G}_N = \mathbf{G}_{N'}$ . So  $\mathbb{G} = \mathbb{G}_N \times \langle 1 + N \rangle$ .

Note that in the setting above, the DCR assumption implies that the uniform distributions over  $\mathbb{G}_N$  and  $\mathbb{G}$ , are computationally indistinguishable. This is exactly what we base the security of our second IBE scheme on.

## 4 Constructions of Identity-Based Encryption

### 4.1 IBE Under sDDH Assumption Over $\mathbb{Z}_{N^2}^*$

Our first scheme can be viewed as an IBE variant of the classical ElGamal cryptosystem. Let  $g$  be a generator of  $\mathbb{G}$ . In the scheme, the encryptor needs to sample a uniform randomness from the set  $[|\mathbb{G}|]$ , but we cannot publish  $|\mathbb{G}|$  as it gives a way to factor modulus  $N$ . Fortunately, we could sample uniformly from another set  $[N^2/4]$ . It's easy to see that  $\Delta(U([\mathbb{G}]), U([N^2/4])) = 1 - 4p'q'/pq = (p + q - 1)/pq < 1/p + 1/q = \mathcal{O}(1/2^\kappa)$ . We denote  $G_s$  as the set  $\{g^1, g^2, \dots, g^{N-1}\}$ . We use hash function  $\mathcal{H} : \{0, 1\}^* \rightarrow G_s$  that maps identities to the subset  $G_s$  of quadratic residues. The message space is  $\mathbb{G}$ .

- Setup( $1^\kappa$ )  $\rightarrow$  params
  1.  $p, q \leftarrow \mathcal{SP}(\kappa)$ ,  $N = pq$ ,  $\text{msk} = \lambda(N)$ .
  2. Output  $\text{params} = \{N, g, \mathcal{H}\}$ , where  $\mathcal{H} : \text{ID}_{\text{space}} \rightarrow G_s$  is a hash function.
- Extract( $\text{msk}, \text{ID}$ )  $\rightarrow d_{\text{ID}}$ 
  1. Output  $d_{\text{ID}} = \frac{\lceil \mathcal{H}(\text{ID})^{\lambda(N)} \rceil_{N^2-1}}{N}$ .
- Enc( $\text{params}, \text{ID}, M$ )  $\rightarrow C$ 
  1. Randomly pick  $r \leftarrow [N^2/4]$ .
  2. Compute  $C = (\lceil g^r \rceil_{N^2}, \lceil \mathcal{H}(\text{ID})^r \cdot M \rceil_{N^2})$ .
- Dec( $C, d_{\text{ID}}$ )  $\rightarrow M$ 
  1. Parse  $C$  as  $(C_1, C_2)$ .
  2. Output  $M = C_2/C_1^{d_{\text{ID}}}$ .

*Possibility of Hashing into  $G_s$ .* The main difficulty to implement the scheme resides in hashing an arbitrary length bitstring into the set  $G_s$ . If the output value of the hash function deviates from this set, the decryption algorithm of the scheme won't work. Here we show a possible way to hash into the target set.

Firstly, a collision-resistant hash function  $H : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$  is applied to the identity to map into an intended length  $\ell$ , which is a polynomial of the security parameter  $\kappa$ . Then we can embed in the Setup algorithm the following process: pick  $a_{i,0}, a_{i,1} \leftarrow \mathbb{Z}_{\lfloor N/\ell \rfloor}$  for  $i = 1, \dots, \ell$  and then publish  $g^{a_{i,0}}, g^{a_{i,1}}$  as part of the description<sup>2</sup> of the hash function  $\mathcal{H}$ . The hash function takes as input an arbitrary identity and is defined as:  $\mathcal{H}(\text{ID}) = \prod_{i=1}^\ell g^{a_{i,H(\text{ID})_i}}$ , where  $H(\text{ID})_i$  denotes the  $i$ -th bit of an  $\ell$ -bit string  $H(\text{ID})$ .

Note that the above hash function perfectly maps an identity into our target subset  $G_s$ . Also, we only give a way to hash into the target subset of the whole group, rather than propose a candidate hash function for RO. Indeed, random oracle model is only a heuristic way to prove the security of a scheme and it may cause security concerns to instantiate a random oracle in the real world using any concrete hash function [17].

Next, we study the security of this scheme.

<sup>2</sup> Note that since  $\ell$  is a polynomial of the security parameter  $\kappa$ , but  $N$  is exponentially large, a brute force may not be possible to retrieve  $a_i \in \mathbb{Z}_{\lfloor N/\ell \rfloor}$  from  $g^{a_i}$ . For instance, practically  $\ell = 80$ ,  $N = 2^{1024}$ .

**Theorem 1.** *The scheme above is IND-ID-CPA under the Small Decisional Diffie-Hellman Assumption in  $\mathbb{Z}_{N^2}^*$ , in the random oracle model.*

*Proof.* Let  $\mathcal{H} : \{0, 1\}^* \rightarrow G_s$  be a hash function viewed as a random oracle. Suppose  $\mathcal{A}$  is an adversary attacking our scheme and has success probability  $\epsilon$ . We construct an algorithm  $\mathcal{B}$  that solves the sDDH problem in  $\mathbb{Z}_{N^2}^*$ . Initially,  $\mathcal{B}$  is given a random tuple  $(N, g, X = g^x, Y = g^y, Z)$  where  $x \leftarrow [|\mathbb{G}|], y \leftarrow \mathbb{Z}_N, Z = [g^{xy}]_{N^2}$  or  $Z \leftarrow \mathbb{G}$ .  $\mathcal{B}$  aims to output 1 if  $Z = [g^{xy}]_{N^2}$  and 0 otherwise.

1.  $\mathcal{B}$  sets  $\text{params} = \{N, g, \mathcal{H}\}$  and gives it to  $\mathcal{A}^{\text{Extract}_{\text{msk}}(\cdot), \mathcal{H}(\cdot)}$ . That is,  $\mathcal{A}$  has oracle access to  $\text{Extract}_{\text{msk}}(\cdot)$  and  $\mathcal{H}(\cdot)$ , it may issue private key extraction and hash queries, after what it decides a target identity  $\text{ID}^*$  to attack, and outputs two equal-length messages  $M_0, M_1 \in \mathbb{G}$ .
2. Upon receiving  $\text{ID}^*$  from  $\mathcal{A}$ :
  - if  $\mathcal{H}(\text{ID}^*) = Y$ , then
    - (a) Choose  $b \leftarrow \{0, 1\}$ , compute  $C^* = (X, [Z \cdot M_b]_{N^2})$  and send it to  $\mathcal{A}$ .
    - (b)  $\mathcal{A}^{\text{Extract}_{\text{msk}}(\cdot), \mathcal{H}(\cdot)}$  may issue more queries, with the restriction that it cannot ask for  $\text{Extract}_{\text{msk}}(\text{ID}^*)$ , after what it returns its guess  $b'$ .
    - (c) if  $b' = b$ , return 1; otherwise return 0.
  - if  $\mathcal{H}(\text{ID}^*) \neq Y$  then return a random bit and abort.

Next, we show how  $\mathcal{B}$  simulates answers to  $\mathcal{A}$ 's oracle queries.  $\mathcal{B}$  maintains a list  $\text{L}[\mathcal{H}]$  with a tuple of the form  $(\cdot, \cdot, \cdot)$  per row. It is initialized to  $\emptyset$ . Let  $q_H$  be the number of hash queries.  $\mathcal{B}$  chooses a random  $i^* \in \{1, 2, \dots, q_H\}$ .

**Hash Queries.** When  $\mathcal{A}$ 's  $j$ -th query on some ID is requested,  $\mathcal{B}$  checks if there is an entry of the form  $(\text{ID}, h, a)$  in  $\text{L}[\mathcal{H}]$ ; If so, it returns  $h$ . Otherwise, if  $j = i^*$ , define  $h_j = Y$  and add entry  $(\text{ID}, h_j, \perp)$  to  $\text{L}[\mathcal{H}]$ . Else,  $\mathcal{B}$  chooses random  $a_j \in \mathbb{Z}_N$ , and computes  $h = [g^{a_j}]_{N^2}$ , then it appends  $(\text{ID}, h_j, a_j)$  to  $\text{L}[\mathcal{H}]$  and returns  $h_j$  to  $\mathcal{A}$ .

**Extraction Queries.** When  $\mathcal{A}$  queries key extraction oracle on some ID,  $\mathcal{B}$  checks if there is an entry of the form  $(\text{ID}, h, a)$  in the list. If not, it calls  $\mathcal{H}(\text{ID})$  so that there is an entry. If  $a \neq \perp$ , then  $\mathcal{B}$  returns  $a$ . Otherwise, it returns  $\perp$ .

We establish two equations to analyze the success probability of  $\mathcal{B}$  in solving the sDDH challenge.

We have that:

$$\begin{aligned} & \Pr[ \mathcal{B}(N, g, X, Y, Z) = 1 \mid Z = [g^{xy}]_{N^2} ] \\ &= \Pr[ Y = \mathcal{H}(\text{ID}^*) ] \cdot \Pr[ \mathcal{B}(N, g, X, Y, Z) = 1 \mid Z = [g^{xy}]_{N^2} \wedge Y = \mathcal{H}(\text{ID}^*) ] \\ & \quad + \Pr[ Y \neq \mathcal{H}(\text{ID}^*) ] \cdot \Pr[ \mathcal{B}(N, g, X, Y, Z) = 1 \mid Z = [g^{xy}]_{N^2} \wedge Y \neq \mathcal{H}(\text{ID}^*) ] \\ &= \frac{1}{q_H} \cdot \epsilon + \left(1 - \frac{1}{q_H}\right) \cdot \frac{1}{2} \end{aligned}$$

similarly,

$$\begin{aligned} & \Pr[\mathcal{B}(N, g, X, Y, Z) = 1 \mid Z \leftarrow \mathbb{G}] \\ &= \Pr[Y = \mathcal{H}(\text{ID}^*)] \cdot \Pr[\mathcal{B}(N, g, X, Y, Z) = 1 \mid Z \leftarrow \mathbb{G} \wedge Y = \mathcal{H}(\text{ID}^*)] \\ &\quad + \Pr[Y \neq \mathcal{H}(\text{ID}^*)] \cdot \Pr[\mathcal{B}(N, g, X, Y, Z) = 1 \mid Z \leftarrow \mathbb{G} \wedge Y \neq \mathcal{H}(\text{ID}^*)] \\ &= \frac{1}{q_H} \cdot \frac{1}{2} + \left(1 - \frac{1}{q_H}\right) \cdot \frac{1}{2} \end{aligned}$$

Thus we have:

$$\begin{aligned} & \left| \Pr[\mathcal{B}(N, g, X, Y, Z) = 1 \mid Z = [g^{xy}]_{N^2}] - \Pr[\mathcal{B}(N, g, X, Y, Z) = 1 \mid Z \leftarrow \mathbb{G}] \right| \\ &= \frac{1}{q_H} \left| \epsilon - \frac{1}{2} \right| \end{aligned}$$

By the sDDH assumption in  $\mathbb{Z}_{N^2}^*$ , it must be negligible. Therefore  $|\epsilon - \frac{1}{2}|$  is negligible too, which shows that the scheme is IND-ID-CPA.  $\square$

## 4.2 IBE Under DCR Assumption

Next we show a more delicate scheme which can be proved secure under Paillier’s *Decisional Composite Residuosity* (DCR) assumption. As far as we are concerned, it is the first IBE scheme yielded by DCR assumption.

– **Setup**( $1^\kappa$ )  $\rightarrow$  **params**

First sample  $x \leftarrow \mathbb{Z}_{N^2}^*$  and compute<sup>3</sup>  $g = [x^{2N}]_{N^2}$ . Let  $h = [g(1+N)]_{N^2}$ .

We denote  $G_s$  as the set  $\{h^1, h^2, \dots, h^{\frac{N-1}{4}}\}$ . We use a hash function  $\mathcal{H} : \{0, 1\}^* \rightarrow G_s$  that maps identities to the subset  $G_s$  of quadratic residues. Define  $\mathcal{H}'(\text{ID}) := [\mathcal{H}(\text{ID})^N]_{N^2}$ . The public parameters and the master secret key are given by:

**params** =  $\{N, g, \mathcal{H}'\}$ , **msk** =  $\{\lambda(N)\}$ . The message space is  $\mathbb{Z}_N$ .

– **Extract**(**msk**, **ID**)  $\rightarrow d_{\text{ID}}$

Suppose  $h_{\text{ID}} = H(\text{ID}) = [g(1+N)]^a$  ( $a \in [\frac{N-1}{4}]$ ).

1. Compute  $a' = h_{\text{ID}}^{\lambda(N)} = [(1+N)^{a\lambda(N)}]_{N^2}$ .

2.  $d_{\text{ID}} = \frac{a'-1}{N} \cdot [\lambda^{-1}(N)]_N$ .

That is, if  $\mathcal{H}(\text{ID}) = [[g(1+N)]_{N^2}]^a$  for some  $a \in [\frac{N-1}{4}]$ , we extract  $a$  as  $d_{\text{ID}}$ .

– **Enc**(**params**, **ID**,  $M$ )  $\rightarrow C$

1. Randomly pick  $r \leftarrow [\frac{N-1}{4}]$ .

2. Compute  $C = ([g^{Nr}]_{N^2}, [\mathcal{H}'(\text{ID})^r \cdot (1+N)^M]_{N^2})$ .

– **Dec**( $C, d_{\text{ID}}$ )  $\rightarrow M$

1. Parse  $C$  as  $(C_1, C_2)$ .

2. Output  $M = \left( [C_2/C_1^{d_{\text{ID}}}]_{N^2} - 1 \right) / N$ .

<sup>3</sup> Observe that  $g$  generates the  $2N$ -th power residue subgroup of  $\mathbb{Z}_{N^2}^*$ , namely  $\mathbb{G}_N$  w.h.p: the probability that  $g$  is not a generator is  $\frac{p'+q'-1}{p'q'} \leq \frac{1}{p'} + \frac{1}{q'}$ .

*Possibility of Hashing into  $G_s$ .* Like our first IBE scheme, it seems hard to even map into the intended set  $G_s$ . We give a similar function to resolve this problem.

First, apply a collision-resistant hash function  $H : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$  to the input identity. We embed in the Setup algorithm the following process:

Let  $g$  be a generator of  $\mathbb{G}_N$  and  $h = [g(1+N)]_{N^2}$ . Randomly pick  $a_{i,0}, a_{i,1} \leftarrow \mathbb{Z}_{\lfloor (N-1)/4\ell \rfloor}$  for  $i = 1, \dots, \ell$  and then publish  $h^{a_{i,0}}, h^{a_{i,1}}$  as part of the description of the hash function  $\mathcal{H}$ . The hash function takes as input an identity and is defined as:  $\mathcal{H}(\text{ID}) = \prod_{i=1}^\ell h^{a_{i,H(\text{ID})_i}}$ , where  $H(\text{ID})_i$  denotes the  $i$ -th bit of an  $\ell$ -bit string  $H(\text{ID})$ .

**Theorem 2.** *The scheme above is IND-ID-CPA under the Decisional Composite Residuosity Assumption, in the random oracle model.*

*Proof.* Let  $\mathcal{H}' : \{0, 1\}^* \rightarrow G_s^N$  be a hash function viewed as a random oracle. Suppose  $\mathcal{A}$  is an adversary attacking our scheme and has success probability  $\epsilon$ . We construct an algorithm  $\mathcal{B}(N, Y)$  that solves the DCR problem in  $\mathbb{Z}_{N^2}^*$ .  $\mathcal{B}$ 's goal is to output 1 if  $Y \leftarrow \mathbb{G}_N$  and 0 if  $Y \leftarrow \mathbb{G}$ .

1.  $\mathcal{B}$  chooses a random element  $x \in \mathbb{Z}_{N^2}^*$  and computes  $g = [x^{2N}]_{N^2}$ . It is easy to see that  $g$  generates  $\mathbb{G}_N$  with all but a negligible probability. It sets  $\text{params} = \{N, g, \mathcal{H}'\}$  and gives it to  $\mathcal{A}^{\text{Extract}_{\text{msk}}(\cdot), \mathcal{H}'(\cdot)}$ . That is,  $\mathcal{A}$  has oracle access to  $\text{Extract}_{\text{msk}}(\cdot)$  and  $\mathcal{H}'(\cdot)$ , it may issue extraction and hash queries, after what it decides a target identity  $\text{ID}^*$  to attack, and outputs two equal-length messages  $M_0, M_1 \in \mathbb{Z}_N$ .
2. Upon receiving  $\text{ID}^*$  from  $\mathcal{A}$ :
  - if  $\mathcal{H}'(\text{ID}^*) = Y$ , then
    - (a) Choose a random bit  $b, r \leftarrow [\frac{N-1}{4}]$ , compute  $C^* = ([g^{Nr}]_{N^2}, [Y^r \cdot (1+N)^{M_b}]_{N^2})$  and send it to  $\mathcal{A}$ .
    - (b)  $\mathcal{A}^{\text{Extract}_{\text{msk}}(\cdot), \mathcal{H}'(\cdot)}$  may issue more extraction and hash queries, with the restriction that it cannot ask for  $\text{Extract}_{\text{msk}}(\text{ID}^*)$ , after what it returns its guess  $b'$ .
    - (c) if  $b' = b$ , return 1; otherwise return 0.
  - if  $\mathcal{H}'(\text{ID}^*) \neq Y$  then return a random bit and abort.

Next, we show how  $\mathcal{B}$  simulates answers to  $\mathcal{A}$ 's oracle queries.  $\mathcal{B}$  maintains a list  $L[\mathcal{H}']$  with a tuple of the form  $(\cdot, \cdot, \cdot)$  per row. It is initialized to  $\emptyset$ . Let  $q_H$  be the number of hash queries.  $\mathcal{B}$  chooses a random  $i^* \in \{1, 2, \dots, q_H\}$ .

**Hash Queries.** When  $\mathcal{A}$ 's  $j$ -th query on some ID is requested,  $\mathcal{B}$  checks if there is an entry of the form  $(\text{ID}, h, a)$  in  $L[\mathcal{H}']$ , especially with  $a \neq \perp$ ; If so, it returns  $h$ . Otherwise, if  $j = i^*$ , define  $h_j = Y$  and add entry  $(\text{ID}, h_j, \perp)$  to  $L[\mathcal{H}']$ . Else,  $\mathcal{B}$  chooses  $a_j \leftarrow [\frac{N-1}{4}]$ , and computes  $h_j = [g^{a_j}]_{N^2}$ , then it appends  $(\text{ID}, h_j, a_j)$  to  $L[\mathcal{H}']$  and returns  $h_j$  to  $\mathcal{A}$ .

**Extraction Queries.** When  $\mathcal{A}$  queries key extraction oracle on some ID,  $\mathcal{B}$  checks if there is an entry of the form  $(\text{ID}, \cdot, \cdot)$  in the list. If not, it calls  $\mathcal{H}'(\text{ID})$  so that there is an entry. If  $a_j \neq \perp$  in the corresponding entry, then  $\mathcal{B}$  returns  $a_j$ . Otherwise, it returns  $\perp$ .

We analyze the success probability of  $\mathcal{B}$  in solving the DCR challenge. There are three cases here (we omit the modulo operation for visually comfort):

- **case 1:**  $Y = \mathcal{H}'(\text{ID}^*) \in \mathbb{G}_N$ . It's implicit that in this case  $\text{ID}^*$  is exactly  $\mathcal{A}$ 's  $i^*$ -th query to  $\mathcal{H}'$ . Moreover, the challenge ciphertext  $C^* = (g^{Nr}, Y^r \cdot (1+N)^{M_b})$  is perfectly simulated because  $\mathcal{H}'(\text{ID}^*) \in \mathbb{G}_N$ . To summarize,  $\mathcal{B}$  returns 1 when  $\mathcal{A}$  wins in the IND-ID-CPA game, as long as no abort happens. That is,  $\Pr[\mathcal{B}(N, Y) = 1 \mid Y \in \mathbb{G}_N \wedge Y = \mathcal{H}'(\text{ID}^*)] = \epsilon$ .
- **case 2:**  $Y = \mathcal{H}'(\text{ID}^*) \in \mathbb{G}$ . In this case, we can represent  $Y$  as  $g^{r'}(1+N)^{r''}$  where  $r' \leftarrow [p'q']$  and  $r'' \leftarrow \mathbb{Z}_N$  and  $C^* := (g^{Nr}, g^{r'r'}(1+N)^{r'r''+M_b})$ . Apparently, the second part of  $C^*$  acts like an “one-time pad” of the message  $M_b$ , so the whole ciphertext is independent of the choice of  $b$ .  $\Pr[\mathcal{B}(N, Y) = 1 \mid Y \in \mathbb{G} \wedge Y = \mathcal{H}'(\text{ID}^*)] = \frac{1}{2}$ .
- **case 3:**  $Y \neq \mathcal{H}'(\text{ID}^*)$ . In this case,  $\mathcal{B}$  guesses the wrong ID that  $\mathcal{A}$  attempts to attack and returns a random bit. Thus,  $\Pr[\mathcal{B}(N, Y) = 1 \mid Y \in \mathbb{G}_N \wedge Y \neq \mathcal{H}'(\text{ID}^*)] = \Pr[\mathcal{B}(N, Y) = 1 \mid Y \in \mathbb{G} \wedge Y \neq \mathcal{H}'(\text{ID}^*)] = \frac{1}{2}$ .

Therefore, we have that:

$$\begin{aligned} & \Pr[\mathcal{B}(N, Y) = 1 \mid Y \in \mathbb{G}_N] \\ &= \Pr[Y = \mathcal{H}'(\text{ID}^*)] \cdot \Pr[\mathcal{B}(N, Y) = 1 \mid Y \in \mathbb{G}_N \wedge Y = \mathcal{H}'(\text{ID}^*)] \\ & \quad + \Pr[Y \neq \mathcal{H}'(\text{ID}^*)] \cdot \Pr[\mathcal{B}(N, Y) = 1 \mid Y \in \mathbb{G}_N \wedge Y \neq \mathcal{H}'(\text{ID}^*)] \\ &= \frac{1}{q_H} \cdot \epsilon + (1 - \frac{1}{q_H}) \cdot \frac{1}{2} \end{aligned}$$

similarly,

$$\Pr[\mathcal{B}(N, Y) = 1 \mid Y \in \mathbb{G}] = \frac{1}{q_H} \cdot \frac{1}{2} + (1 - \frac{1}{q_H}) \cdot \frac{1}{2}$$

Thus we have:  $\left| \Pr[\mathcal{B}(N, Y) = 1 \mid Y \in \mathbb{G}_N] - \Pr[\mathcal{B}(N, Y) = 1 \mid Y \in \mathbb{G}] \right| = \frac{1}{q_H} \left| \epsilon - \frac{1}{2} \right|$ , which is negligible due to the DCR assumption. This indicates that  $\left| \epsilon - \frac{1}{2} \right|$  is negligible too. So the scheme is IND-ID-CPA under the DCR assumption.  $\square$

## 5 Generic Construction of Selectively Secure IBE

Starting with a DDH group with a DL-solvable subgroup, we can build a generic IBE scheme with the help of  $i\mathcal{O}$  and PPRF.

- **Setup**( $1^\kappa$ ):
  1. Run  $\text{Gen}(1^\lambda, 1^\mu) \rightarrow (B, n, p, s, g, f, G, F)$ .  $\text{msk} := s$ .
  2. Choose a puncturable PRF key  $K$  for  $F$  where  $F(K, \cdot) : \{0, 1\}^{\ell(\kappa)} \rightarrow \mathbb{Z}_p$ . Then create an obfuscation of the program  $G_1$  (Fig. 1). The size of the program is padded to be  $\max\{|G_1|, |G_2|\}$ . We refer to the obfuscated program as the hash function  $\mathcal{H} : \{0, 1\}^\ell \rightarrow G_s$ . Define the DL-solvable set as  $G_s := \{g^1, g^2, \dots, g^{p-1}\}$ .
  3. Output  $\text{params} = \{B, p, g, f, G, F, \mathcal{H}\}$ , where  $\mathcal{H} : \text{ID}_{\text{space}} \rightarrow G_s$  is a hash function.



**Input:** Identity ID  
**Constants:** PRF key  $K, g$

1. Compute  $a = F(K, \text{ID})$ .
2. Output  $g^a$ .

**Fig. 1.** Program  $G_1$

- Extract(msk, ID):
  1. First run  $\text{Solve}(B, p, g, f, G, F, \mathcal{H}(\text{ID})^s)$  and denote by  $a$  the output.
  2. Then run  $\text{Solve}(B, p, g, f, G, F, g^s)$  and the output is saved as  $b$ .
  3. Output  $d_{\text{ID}} = a \cdot [b^{-1}]_p$ .
- Enc(params, ID,  $M$ ):
  1. Randomly pick  $r \leftarrow \{0, \dots, Bp - 1\}$ .
  2. Compute  $C = (g^r, \mathcal{H}(\text{ID})^r \cdot M)$ .
- Dec( $C, d_{\text{ID}}$ ):
  1. Parse  $C$  as  $(C_1, C_2)$ .
  2. Output  $M = C_2 / C_1^{d_{\text{ID}}}$ .

**Correctness.** The decryption correctness relies on the functionality of Extract algorithm. This follows by the fact that the hash function maps an identity into the DL-solvable set and retrieving the discrete logarithm of  $\mathcal{H}(\text{ID})$  through the algorithm Extract is correct with  $s$  as an input. To elaborate, first note that for all  $h \in G_s, h^s \in F$ . By the correctness of  $\text{Solve}(B, p, g, f, G, F, g^s)$  we get a  $b \in \mathbb{Z}_p$  through the equation  $g^s = f^b$ . Suppose  $\mathcal{H}(\text{ID}) = g^t$  for an integer  $t < p$ . Similarly, we can get  $a$  by another execution of  $\text{Solve}(B, p, g, f, G, F, \mathcal{H}(\text{ID})^s)$  such that  $g^{ts} = f^a$ . The above indicates that  $f^{bt} = f^a$ , then  $t \equiv a \cdot [b^{-1}]_p$ .

**Input:** Identity ID  
**Constants:** Punctured PRF key  $K(\{\text{ID}^*\})$ ,  $\text{ID}^* \in \{0, 1\}^\ell, g, g^{y^*}$  where  $y^* \in \mathbb{Z}_p$

1. If  $\text{ID} = \text{ID}^*$ , output  $g^{y^*}$ .
2. Else output  $g^a$  where  $a = F(K(\{\text{ID}^*\}), \text{ID})$ .

**Fig. 2.** Program  $G_2$

**Theorem 3.** *If the obfuscation scheme is indistinguishably secure,  $F$  is a secure puncturable PRF, and the Small Decisional Diffie-Hellman assumption holds in  $G$ , the IBE scheme above is IND-sID-CPA.*

*Proof.* We give a formal proof in Appendix B. □

## 6 Conclusion

In this paper, we provide two efficient IBE schemes which rely on pairing-free number-theoretic assumptions. The ciphertext is comprised of two group elements and the main cost of an encryption operation is only 2 modular exponentiations. Although the security proof relies on the random oracle techniques, the new constructions provide a fresh idea of utilizing specific algebraic structure. We believe that similar structures could be found in further investigations. Moreover, we provide a generic construction of selectively secure IBE in the standard model.

**Acknowledgments.** The authors would like to thank anonymous reviewers for their helpful comments and suggestions. This work was supported by National Natural Science Foundation of China (Grants 61472414,61772514,61602061), and National Key R&D Program of China (2017YFB1400700).

## A Preliminaries (Cont'd.)

### A.1 Indistinguishability Obfuscation

We present the formal definition following the syntax of Garg et al. [14]:

**Definition 8 (Indistinguishability Obfuscation ( $i\mathcal{O}$ )).** A uniform PPT machine  $i\mathcal{O}$  is called an indistinguishability obfuscator for a circuit class  $\{\mathcal{C}_\kappa\}$  if the following holds:

- (**Correctness:**) For all security parameters  $\kappa \in \mathbb{N}$ ,  $C \in \mathcal{C}_\kappa$ , and inputs  $x$ :

$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\kappa, C)] = 1.$$

- (**Indistinguishability:**) For any (not necessarily uniform) PPT distinguisher  $(\text{Samp}, \mathcal{D})$ , there exists a negligible function  $\text{negl}$  such that the following holds: if  $\Pr[\forall x, C_0(x) = C_1(x); (C_0, C_1, \sigma) \leftarrow \text{Samp}(1^\kappa)] \geq 1 - \text{negl}(\kappa)$ , then:

$$\begin{aligned} &|\Pr[\mathcal{D}(\sigma, i\mathcal{O}(\kappa, C_0)) = 1 : (C_0, C_1, \sigma) \leftarrow \text{Samp}(1^\kappa)] \\ &- \Pr[\mathcal{D}(\sigma, i\mathcal{O}(\kappa, C_1)) = 1 : (C_0, C_1, \sigma) \leftarrow \text{Samp}(1^\kappa)]| \leq \text{negl}(\kappa). \end{aligned}$$

### A.2 Puncturable Pseudorandom Functions

Below we recall the definition of puncturable PRFs, as given by Sahai et al. [20]:

**Definition 9.** A puncturable family of PRFs  $F$  is given by a triple of Turing machines  $\text{Key}, \text{Puncture}, \text{Eval}$ , and a pair of computable functions  $n(\cdot)$  and  $m(\cdot)$ , satisfying the following conditions:

- (**Functionality preserved under puncturing**). For every PPT adversary  $\mathcal{A}$  such that  $\mathcal{A}(1^\lambda)$  outputs a set  $S \subseteq \{0, 1\}^{n(\kappa)}$ , then for all  $x \in \{0, 1\}^{n(\kappa)}$  where  $x \notin S$ , we have that:

$$\Pr[\text{Eval}(K, x) = \text{Eval}(K_S, x) : K \leftarrow \text{Key}(1^\kappa), K_S = \text{Puncture}(K, S)] = 1.$$

- (**Pseudorandom at punctured points**). For every PPT adversary  $(\mathcal{A}_1, \mathcal{A}_2)$  such that  $\mathcal{A}_1(1^\kappa)$  outputs a set  $S \subseteq \{0, 1\}^{n(\kappa)}$  and  $x \in S$ , consider an experiment where  $K \leftarrow \text{Key}(1^\kappa)$  and  $K_S = \text{Puncture}(K, S)$ . Then we have

$$|\Pr[\mathcal{A}_2(K_S, x, \text{Eval}(K, x)) = 1] - \Pr[\mathcal{A}_2(K_S, x, U_{m(\kappa)}) = 1]| \leq \text{negl}(\kappa),$$

where  $U_{m(\kappa)}$  denotes the uniform distribution over  $m(\kappa)$  bits.

## B Proof of Theorem 3

We begin by given a sequence of games played between a challenger and an adversary.

- Game<sub>0</sub>:
  1. The adversary selectively gives the challenger the identity  $\text{ID}^*$ .
  2. The public parameters  $\text{params}$  are chosen by the challenger invoking  $\text{Gen}(1^\kappa)$ .
  3.  $K$  is chosen as a key for the PPRF.
  4. The hash function  $\mathcal{H}(\cdot)$  is created as an obfuscation of the program  $G_1$ .
  5. The adversary queries the key extraction oracle a polynomial number of times on  $\text{ID} \neq \text{ID}^*$ . It receives back  $F(K, \text{ID})$ . Once this phase is end, the adversary gives two equal length messages  $m_0, m_1$ .
  6. The challenger chooses a random bit  $b \in \{0, 1\}$ ,  $r \leftarrow \{0, \dots, Bp - 1\}$  and outputs  $C^* = (g^r, \mathcal{H}(\text{ID}^*)^r \cdot m_b)$ .
  7. The adversary receives  $C^*$  and could still issue key extraction queries for polynomial times with the same restriction that  $\text{ID} \neq \text{ID}^*$ , finally it outputs  $b'$  as its guess of  $b$ .
  8. If  $b' = b$ , the game outputs 1, else outputs 0.
- Game<sub>1</sub>: Is the same as Game<sub>0</sub> except that  $y^* = F(K, \text{ID}^*)$  and the hash function  $\mathcal{H}(\cdot)$  is replaced by an obfuscation of the program  $G_2$  (Fig. 2).
- Game<sub>2</sub>: Is the same as Game<sub>1</sub> except that  $y^* \leftarrow \mathbb{Z}_p$ .
- Game<sub>3</sub>: Is the same as Game<sub>2</sub> except that the challenge ciphertext  $C^*$  is computed as  $(g^r, g^{r'} \cdot m_b)$  where  $r' \leftarrow \{0, \dots, Bp - 1\}$  is chosen independently of  $r$ .

We establish the following lemmas and they together yield Theorem 3 that the so obtained IBE scheme is selectively secure.

**Lemma 1.** *If the obfuscation scheme is indistinguishability secure, then the advantage of any PPT adversary is negligibly close between Game<sub>0</sub> and Game<sub>1</sub>.*

*Proof.* We set up two algorithms  $\text{Samp}$  and  $\mathcal{D}$ :

$\text{Samp}(1^\kappa)$  runs the adversary to obtain  $\text{ID}^*$  and its state  $\tau'$ . It then invokes  $\text{Gen}(1^\kappa)$  to obtain  $\text{params}$  and  $\text{msk}$ . It chooses  $K$  as the key for PPRF. It sets  $y^* = F(K, \text{ID}^*)$  and  $\tau = (\text{ID}^*, \text{params}, \text{msk}, K, \tau')$ . It builds  $C_1$  as the program for  $G_1$ , and  $C_2$  as the program for  $G_2$ .

$\mathcal{D}$  takes as input  $\tau$  and an obfuscation of a circuit  $C_1$  or  $C_2$ . When the adversary makes a key extraction query on  $\text{ID} \neq \text{ID}^*$ ,  $\mathcal{D}$  use the  $K$  within  $\tau$  to return  $F(K, \text{ID})$ . Once the adversary gives two equal length messages  $m_0, m_1$ ,  $\mathcal{D}$  chooses a random bit  $b$  and constructs challenge ciphertext  $C^* = (g^r, \mathcal{H}(\text{ID}^*)^r \cdot m_b)$ . Eventually, the adversary sends a bit  $b'$  and wins the game if  $b' = b$ .  $\mathcal{D}$  outputs 1 if the adversary wins.

Observe that if  $\mathcal{D}$  receives an obfuscation of  $C_1$ , the probability  $\mathcal{D}$  outputs 1 is equal to the probability of the adversary winning in  $\text{Game}_0$ . And if  $\mathcal{D}$  receives an obfuscation of  $C_2$ , the probability  $\mathcal{D}$  outputs 1 is equal to the probability of the adversary winning in  $\text{Game}_1$ . Then the lemma follows.  $\square$

**Lemma 2.** *If the punctured PRF is secure, then the advantage of any PPT adversary is negligibly close between  $\text{Game}_1$  and  $\text{Game}_2$ .*

*Proof.* In order to reduce this lemma to the property of PPRF’s pseudorandomness at the punctured points, we give the algorithms  $\mathcal{A}_1$  and  $\mathcal{A}_2$ .

$\mathcal{A}_1(1^\kappa)$  runs the adversary to obtain  $\text{ID}^*$  and its state  $\tau'$ , then it outputs the set  $S = \{\text{ID}^*\}$ .

$\mathcal{A}_2$  obtains  $S = \{\text{ID}^*\}$ ,  $K(\{\text{ID}^*\}) = \text{Puncture}(K, \text{ID}^*)$ , and either a value  $y^* = F(K, \text{ID}^*)$  or a uniformly random  $y^* \in \mathbb{Z}_p$ .  $\mathcal{A}_2$  runs  $\text{Gen}(1^\kappa)$  to obtain  $\text{params}$ , then it can get  $g^{y^*}$ . This value corresponds to exactly the  $g^{y^*}$  value in  $\text{Game}_1$  if  $y^* = F(K, \text{ID}^*)$  or in  $\text{Game}_2$  if  $y^* \leftarrow \mathbb{Z}_p$ .  $\mathcal{A}_2$  can then obfuscate the program  $G_2$  and answer the key extraction queries from the adversary since it knows  $K(\{\text{ID}^*\})$ . The obfuscated program is modeled as a hash function  $\mathcal{H}$ . When the adversary gives two equal length messages  $m_0, m_1$ ,  $\mathcal{A}_2$  chooses a bit  $b$  uniformly at random and constructs challenge ciphertext  $C^* = (g^r, \mathcal{H}(\text{ID}^*)^r \cdot m_b)$ . The adversary may issue more key extraction queries of  $\text{ID} \neq \text{ID}^*$ , and  $\mathcal{A}_2$  answers them in a similar way. Eventually, the adversary sends a bit  $b'$  and wins the game if  $b' = b$ .  $\mathcal{A}_2$  outputs 1 if the adversary wins.

By our construction, the lemma follows.  $\square$

**Lemma 3.** *If sDDH assumption holds in group  $G$ , then the advantage of any PPT adversary is negligibly close between  $\text{Game}_2$  and  $\text{Game}_3$ .*

*Proof.* To prove this lemma, we establish a distinguisher  $\mathcal{D}$ .  $\mathcal{D}$  takes as input a tuple  $(B, p, g, f, G, F, X, Y, Z, \text{Solve}(\cdot))$  where  $X = g^x, Y = g^y$  for  $x \leftarrow \mathbb{Z}_n, y \leftarrow \mathbb{Z}_p$ . The target of  $\mathcal{D}$  is to output 1 if  $Z = g^{xy}$  or 0 if  $Z = g^z$ , for  $z \leftarrow \mathbb{Z}_n$ .

Next the distinguisher  $\mathcal{D}$  invokes the adversary to get  $\text{ID}^*$ , chooses a PRF key  $K$  and computes  $K(\{\text{ID}^*\})$  itself. It then can obfuscate the program  $G_2$  through its knowledge of  $K(\{\text{ID}^*\}), \text{ID}^*, g, Y$ . The obfuscated program is modeled as a hash function  $\mathcal{H}$ . It sends  $\text{params} = \{B, p, g, f, G, F, \mathcal{H}\}$  to the adversary. When the adversary issues a key extraction query of  $\text{ID} \neq \text{ID}^*$ , it uses  $K(\{\text{ID}^*\})$  to

compute  $F(K(\{\text{ID}^*\}), \text{ID})$  and returns this value back. Once the adversary sends a pair of equal length message  $m_0, m_1$ , it creates the ciphertext  $C^* = (X, Z \cdot m_b)$  after flipping a random coin  $b$  and sends to the adversary. The adversary may issue more key extraction queries of  $\text{ID} \neq \text{ID}^*$ , and  $\mathcal{D}$  answers them in a similar way. By construction, there are two cases. If  $Z = g^{xy}$ , the probability of  $\mathcal{D}$  outputting 1 is exactly the probability that the adversary succeeds in  $\text{Game}_2$ . On the other hand, if  $Z = g^z$  for a uniformly random  $z \in \mathbb{Z}_n$ , the probability of  $\mathcal{D}$  outputting 1 is the probability that the adversary succeeds in  $\text{Game}_3$ .

By sDDH assumption, the difference between these two probabilities must be negligible.  $\square$

**Lemma 4.** *The advantage of any PPT adversary in  $\text{Game}_3$  is negligible.*

*Proof.* In  $\text{Game}_3$ , the challenge ciphertext perfectly “hides” the message, especially regardless of choice of the bit  $b$ . The lemma follows.  $\square$

## References

1. 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). [https://doi.org/10.1007/978-3-642-13190-5\\_28](https://doi.org/10.1007/978-3-642-13190-5_28)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: CCS (1993). <https://doi.org/10.1145/168588.168596>
3. Boneh, D., Papakonstantinou, P., Rackoff, C., Vahlis, Y.: On the impossibility of basing identity based encryption on trapdoor permutations. In: FOCS (2008)
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). [https://doi.org/10.1007/978-3-540-24676-3\\_14](https://doi.org/10.1007/978-3-540-24676-3_14)
5. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-28628-8\\_27](https://doi.org/10.1007/978-3-540-28628-8_27)
6. 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). [https://doi.org/10.1007/3-540-44647-8\\_13](https://doi.org/10.1007/3-540-44647-8_13)
7. Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption without pairings. In: FOCS (2007). <https://doi.org/10.1109/focs.2007.50>
8. Bresson, E., Catalano, D., Pointcheval, D.: A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In: Lai, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 37–54. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-40061-5\\_3](https://doi.org/10.1007/978-3-540-40061-5_3)
9. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24676-3\\_13](https://doi.org/10.1007/978-3-540-24676-3_13)
10. 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). [https://doi.org/10.1007/978-3-642-13190-5\\_27](https://doi.org/10.1007/978-3-642-13190-5_27)

11. Castagnos, G., Laguillaumie, F.: Linearly homomorphic encryption from DDH. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 487–505. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-16715-2\\_26](https://doi.org/10.1007/978-3-319-16715-2_26)
12. 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). [https://doi.org/10.1007/3-540-45325-3\\_32](https://doi.org/10.1007/3-540-45325-3_32)
13. Döttling, N., Garg, S.: Identity-based encryption from the diffie-hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 537–569. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63688-7\\_18](https://doi.org/10.1007/978-3-319-63688-7_18)
14. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)
15. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC (2008). <https://doi.org/10.1145/1374376.1374407>
16. Kiayias, A., Tsiounis, Y., Yung, M.: Group encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 181–199. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-76900-2\\_11](https://doi.org/10.1007/978-3-540-76900-2_11)
17. Kobitz, N., Menezes, A.J.: The random oracle model: a twenty-year retrospective. *Des. Codes Crypt.* **77**(23), 587–610 (2015). <https://doi.org/10.1007/s10623-015-0094-2>
18. 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). [https://doi.org/10.1007/3-540-48910-X\\_16](https://doi.org/10.1007/3-540-48910-X_16)
19. Paillier, P.: Public-key cryptosystems based on discrete logarithms residues. In: EUROCRYPT (1999)
20. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: STOC (2014)
21. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). [https://doi.org/10.1007/3-540-39568-7\\_5](https://doi.org/10.1007/3-540-39568-7_5)
22. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-03356-8\\_36](https://doi.org/10.1007/978-3-642-03356-8_36)
23. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005). [https://doi.org/10.1007/11426639\\_7](https://doi.org/10.1007/11426639_7)



# Multi-key Homomorphic Proxy Re-Encryption

Satoshi Yasuda<sup>(✉)</sup>, Yoshihiro Koseki, Ryo Hiromasa, and Yutaka Kawai

Mitsubishi Electric, Kamakura, Kanagawa, Japan  
Yasuda.Satoshi@ea.MitsubishiElectric.co.jp,  
Koseki.Yoshihiro@ak.MitsubishiElectric.co.jp,  
Hiromasa.Ryo@aj.MitsubishiElectric.co.jp,  
Kawai.Yutaka@da.MitsubishiElectric.co.jp

**Abstract.** In this paper, we propose a new notion of *multi-key homomorphic proxy re-encryption* (MH-PRE) in which inputs of homomorphic evaluation are encrypted by different public keys and the evaluated ciphertext is decrypted by a single secret key. We obtain it by adding the re-encryption property of proxy re-encryption to multi-key homomorphic encryption (MHE). MHE, firstly proposed by López-Alt, Tromer and Vaikuntanathan (STOC 2012), can perform homomorphic evaluations on ciphertexts from different keys, but decrypting the output ciphertext of the homomorphic evaluation requires all the secret keys associated to the input ciphertexts. In order to decrypt the output ciphertext with a single secret key, we introduce the notion of the re-encryption to MHE. In particular, we construct an MH-PRE scheme by applying the *key switching technique* to the MHE scheme of Peikert and Shiehian (TCC 2016).

**Keywords:** Fully homomorphic encryption  
Multi-key homomorphic encryption · Proxy re-encryption  
Homomorphic Proxy Re-Encryption

## 1 Introduction

Fully Homomorphic Encryption (FHE) allows users to perform an arbitrary computation on encrypted data only with public information. By using FHE, the confidentiality of the users' data can be maintained even if computations are performed on an untrusted server. Such computations on encrypted data are called homomorphic evaluation. After the breakthrough work of Gentry's blueprint [Gen09], many various FHE schemes have been proposed [DGHV10, BV11a, BV11b, BGV12, GSW13]. Because of its characteristics, FHE is a useful cryptographic concept for constructing cryptographic protocols such as outsourcing computations to cloud servers without compromising privacy.

We consider the following typical situation of using FHE to outsource computations to a cloud server. Data providers upload to a cloud server their own data encrypted by FHE. The cloud server performs a homomorphic evaluation for a

certain function on the received data, and passes the result of the evaluation to a receiver. The receiver decrypts the ciphertext to obtain the result of the function evaluation on the data from the data providers. This seems to be able to protect the data providers' private information. However, if the receiver obtains the input ciphertexts (e.g., by colluding with or stealing the data from the server), he can obtain the private information of the data providers by decrypting the obtained ciphertexts. This is a serious problem in terms of confidentiality of the providers' data. Moreover, if the receiver's secret key leaks out, all the data encrypted by the data providers also leaks out since all the input ciphertexts to the homomorphic evaluation must be under the same public key in ordinary FHE schemes. To prevent these problems, we require an FHE scheme satisfying the following two properties: (1) Each provider can encrypt his data under his own public key. Even if secret keys of some providers are exposed, this property keeps the data of other providers secret. (2) The receiver cannot decrypt providers' original ciphertexts. This property ensures that providers' original data are kept secret even though the receiver obtains the providers' original ciphertext.

Multi-key Homomorphic Encryption (MHE) is first proposed by López-Alt et al. [LTV12], and various schemes have been proposed [CM15, MW16, BP16, PS16, CZW17]. MHE can perform homomorphic evaluations on ciphertexts under different public keys. By using MHE in the situation described above, each provider can encrypt his data with his own public key, which satisfies the property (1), and the server can perform homomorphic evaluations on their ciphertexts. However, to decrypt the evaluated ciphertext of MHE, we need to know all the secret keys for the involved ciphertexts in the homomorphic evaluation. Specifically, considering a ciphertext  $c_1$  and  $c_2$  under public key  $\text{pk}_1$  and  $\text{pk}_2$  respectively, decrypting the resulting ciphertext  $c'$  requires  $\text{sk}_1$  and  $\text{sk}_2$  corresponding to  $\text{pk}_1$  and  $\text{pk}_2$ . To allow a receiver to know the evaluation result, the data providers should share their secret keys with the receiver, or interactively decrypt the evaluated ciphertext as threshold decryption protocols. As in the case of using ordinary FHE, it is not desirable to share the secret keys since they enable the receiver to decrypt the ciphertexts before the evaluation. In the threshold decryption [LTV12, CM15, MW16, CZW17, BHP17], each user partially decrypts every evaluated ciphertext using his own secret key and then the receiver sums up the partial decryption results, so this increases computations and interactions of the data providers and the receiver.

## 1.1 Our Results

In this paper, we propose a new notion of MHE where a resulting ciphertext from homomorphic evaluation can be decrypted without secret keys for the involved ciphertext in the evaluation. We call the proposed notion *Multi-key Homomorphic Proxy Re-Encryption (MH-PRE)*, and obtain by adding to MHE the re-encryption property of Proxy Re-Encryption (PRE) [ID03, AFGH06, PRSV17]. PRE can allow a proxy to re-encrypt a ciphertext encrypted under Alice's public key into one that can be decrypted by Bob's secret key. The re-encryption



requires a so-called re-encryption key generated from Alice’s secret information and Bob’s public information.

As well as MHE, the proposed MH-PRE allows the providers to encrypt his data with his own public key while enabling homomorphic evaluations on the ciphertexts under the different public keys. Furthermore, by applying a re-encryption to the evaluated ciphertext, the receiver can decrypt it only with his own secret key. The re-encryption is done with re-encryption keys generated by the providers and sent to the server in advance. This achieves the property (2) described above, that is, the receiver cannot decrypt the providers original ciphertexts since he does not know the secret keys for their ciphertexts. In addition, since the providers only generate the re-encryption keys and send them to the cloud server, there is no interactive communication between the providers and the receiver such as the threshold decryption.

Furthermore, we show an instantiation of MH-PRE by extending the MHE scheme proposed by Peikert and Shiehian [PS16] and prove the IND-CPA security defined in [PRSV17] for the proposed scheme. The extension is done by applying the *key switching technique* of [BGV12], which allows us to change a key required for decrypting a ciphertext. However, just applying the key switching technique to the [PS16] scheme leads the flawed scheme that leaks information on the providers’ secret keys. To avoid this problem, we add a modification to a re-encryption algorithm so as not to leak the providers’ secret keys using the property of decryption algorithm of [GSW13, PS16].

## 1.2 Technical Overview

In order to describe our scheme, consider the following situation. Let providers  $i$  and  $j$  and a receiver  $k$  have their own key pair  $(\mathbf{pk}_i, \mathbf{sk}_i)$ ,  $(\mathbf{pk}_j, \mathbf{sk}_j)$  and  $(\mathbf{pk}_k, \mathbf{sk}_k)$ , respectively. The providers  $i$  and  $j$  encrypt their messages and send the ciphertexts to a cloud server, which performs homomorphic evaluations on the ciphertexts and re-encrypts the evaluated ciphertext to the receiver  $k$ . Before the re-encryption begins, the provider  $i$  (resp. the provider  $j$ ) generates a re-encryption key  $\mathbf{rk}_{i \rightarrow k}$  (resp.  $\mathbf{rk}_{j \rightarrow k}$ ) from his own secret key  $\mathbf{sk}_i$  (resp.  $\mathbf{sk}_j$ ) and the receiver’s public key  $\mathbf{pk}_k$ , and sends the re-encryption key to the cloud server. Then, the receiver  $k$  can decrypt the received ciphertext from the cloud server using only his own secret key  $\mathbf{sk}_k$ . The starting point of our MH-PRE scheme is the MHE scheme of [PS16].

**The MHE Scheme of [PS16].** A secret key  $\mathbf{sk}$  of their scheme is  $\mathbf{t} = (-\bar{\mathbf{t}}\|\mathbf{1}) \in \mathbb{Z}_q^n$  and a public key  $\mathbf{pk}$  is  $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$  s.t.  $\mathbf{tB} \approx \mathbf{0}$ . A ciphertext is a triple of matrices  $(\mathbf{C}, \mathbf{F}, \mathbf{D})$ , which satisfies for the secret key  $\mathbf{t}$

$$\mathbf{tC} \approx \mu(\mathbf{t} \otimes \mathbf{g}), \quad (1)$$

where  $\mu$  is the plaintext and  $\mathbf{g} := (1, 2, 4, \dots, 2^{\lceil \log q \rceil - 1})$ . In order to perform homomorphic evaluations on ciphertexts under different public keys,  $\mathbf{C}$  is extended to  $\mathbf{C}'$  that satisfies Eq. (1) under  $\mathbf{t}' = (\mathbf{t}\|\mathbf{t}^*) \in \mathbb{Z}_q^{2n}$  for an additional

secret key  $\mathbf{t}^*$  without changing the plaintext. Here,  $(\mathbf{t} \parallel \mathbf{t}^*)$  means a concatenation of  $\mathbf{t}$  and  $\mathbf{t}^*$ . In the considered situation, the ciphertexts  $\mathbf{C}_i$  and  $\mathbf{C}_j$  of providers  $i$  and  $j$  are extended to  $\mathbf{C}'_i$  and  $\mathbf{C}'_j$  that satisfies Eq. (1) for the concatenated secret key  $\mathbf{t}' = (\mathbf{t}_i \parallel \mathbf{t}_j)$ . Then, the cloud server performs the homomorphic evaluations on these extended ciphertexts and obtains the resulting ciphertext  $\mathbf{C}_{ij}$ , which satisfies Eq. (1) for  $\mathbf{t}' = (\mathbf{t}_i \parallel \mathbf{t}_j)$ .

**Our MH-PRE Scheme.** The provider  $i$  (resp. the provider  $j$ ) generates a re-encryption key  $\text{rk}_{i \rightarrow k}$  (resp.  $\text{rk}_{j \rightarrow k}$ ) from his own secret key  $\text{sk}_i = \mathbf{t}_i$  (resp.  $\text{sk}_j = \mathbf{t}_j$ ) and receiver's public key  $\text{pk}_k = \mathbf{B}_k$  as follows:

$$\text{rk}_{i \rightarrow k} := \mathbf{B}_k \mathbf{X}_i + \begin{pmatrix} \mathbf{0}_{(n-1) \times n\ell} \\ \mathbf{t}_i \cdot (\mathbf{I}_n \otimes \mathbf{g}) \end{pmatrix}.$$

The re-encryption key generation is the same as the generation of a key switching key in [BGV12]. If  $\mathbf{X}$  is a random binary matrix, the re-encryption key is considered as a ciphertext which encrypts the secret key  $\mathbf{t}_i$  under the public key  $\mathbf{B}_k$ . The following equation holds for the re-encryption key  $\text{rk}_{i \rightarrow k}$  and the receiver's secret key  $\text{sk}_k = \mathbf{t}_k$ :

$$\mathbf{t}_k \cdot \text{rk}_{i \rightarrow k} \approx \mathbf{t}_i \cdot (\mathbf{I}_n \otimes \mathbf{g}). \quad (2)$$

This follows from the fact that the secret key satisfies  $\mathbf{t}_k \mathbf{B}_k \approx \mathbf{0}$ , and its last element is 1.

The cloud server uses  $\text{rk}_{i \rightarrow k}$  and  $\text{rk}_{j \rightarrow k}$  received from each provider to re-encrypt  $\mathbf{C}_{ij}$  into a ciphertext that can be decrypted only with  $\text{sk}_k$ . First, the cloud concatenates the re-encryption keys to  $(\text{rk}_{i \rightarrow k} \parallel \text{rk}_{j \rightarrow k})$  so as to correspond to the secret key  $\mathbf{t}' = (\mathbf{t}_i \parallel \mathbf{t}_j)$ , then the following holds for  $(\text{rk}_{i \rightarrow k} \parallel \text{rk}_{j \rightarrow k})$  and  $\mathbf{t}_k$ :

$$\begin{aligned} \mathbf{t}_k (\text{rk}_{i \rightarrow k} \parallel \text{rk}_{j \rightarrow k}) &= (\mathbf{t}_k \text{rk}_{i \rightarrow k} \parallel \mathbf{t}_k \text{rk}_{j \rightarrow k}) \\ &\approx (\mathbf{t}_i \cdot (\mathbf{I}_n \otimes \mathbf{g}) \parallel \mathbf{t}_j \cdot (\mathbf{I}_n \otimes \mathbf{g})) \quad (\text{from Equation (2)}) \\ &= (\mathbf{t}_i \parallel \mathbf{t}_j) \cdot (\mathbf{I}_{2n} \otimes \mathbf{g}). \end{aligned}$$

The re-encryption is performed as follows:

$$\mathbf{C}^* := (\text{rk}_{i \rightarrow k} \parallel \text{rk}_{j \rightarrow k}) \cdot (\mathbf{I}_{2n} \otimes \mathbf{g}^{-1}) [\mathbf{C}_{ij}] \in \mathbb{Z}_q^{n \times 2n\ell}.$$

Here,  $(\mathbf{I}_{2n} \otimes \mathbf{g}^{-1})$  means an operation that satisfies  $(\mathbf{I}_{2n} \otimes \mathbf{g}) \cdot (\mathbf{I}_{2n} \otimes \mathbf{g}^{-1}) [\mathbf{C}_{ij}] = \mathbf{C}_{ij}$ . We see that  $\mathbf{C}^*$  is decrypted correctly only with  $\text{sk}_k = \mathbf{t}_k$  as:

$$\begin{aligned} \mathbf{t}_k \mathbf{C}^* &= (\mathbf{t}_i \parallel \mathbf{t}_j) \cdot (\mathbf{I}_{2n} \otimes \mathbf{g}) \cdot (\mathbf{I}_{2n} \otimes \mathbf{g}^{-1}) [\mathbf{C}_{ij}] \\ &= (\mathbf{t}_i \parallel \mathbf{t}_j) \cdot \mathbf{C}_{ij} \\ &\approx \mu \cdot ((\mathbf{t}_i \parallel \mathbf{t}_j) \otimes \mathbf{g}) \quad (\text{from Equation (1) holds for } \mathbf{t}' \text{ and } \mathbf{C}_{ij}). \end{aligned}$$

However, this equation shows that the re-encrypted ciphertext  $\mathbf{C}^*$  leaks the information on  $\mathbf{t}_i$  and  $\mathbf{t}_j$  to the receiver. In order to prevent the leakage on the

secret keys, the server outputs the penultimate column  $\mathbf{c}^*$  of  $\mathbf{C}^*$  instead of all columns of  $\mathbf{C}^*$ . The MHE scheme [PS16] on which our scheme based uses only a particular column (i.e. a penultimate column) of the ciphertext matrices to decrypt, so it is obvious that we can decrypt correctly even if our re-encryption algorithm only outputs  $\mathbf{c}^*$ . By outputting only  $\mathbf{c}^*$  as a re-encrypted ciphertext, it is not possible to re-encrypt any more. Therefore, our scheme can be seen as a single-hop PRE scheme. Also for the same reason, it is not possible to perform any homomorphic evaluation on the re-encrypted ciphertexts while it can be performed any times before the re-encryption. However, our MH-PRE scheme has enough functionality for the above situation since the receiver dose not perform homomorphic evaluations on the received ciphertexts.

### 1.3 Related Work

Homomorphic Proxy Re-Encryption (H-PRE) [MLO16,DRS17] is a similar notion to our MH-PRE. Our MH-PRE can be seen as the extend notion of H-PRE. The extension is done by adding the multi-key property of MHE to H-PRE. H-PRE also allows users to encrypt their data with their own key and a receiver to decrypt the resulting ciphertexts from homomorphic evaluations only with his secret key. It is obtained by applying PRE to FHE as well as our scheme except we use MHE. When performing homomorphic evaluations, H-PRE first re-encrypts each ciphertext under different public keys to the receiver’s public key. Then, we can perform the homomorphic evaluation on the re-encrypted ciphertexts since the input ciphertexts of the homomorphic evaluation are encrypted under the same public key now. In H-PRE, however, there is a problem that the re-encryption must be done before the homomorphic evaluations. If we want to send a ciphertext of a computation result to multiple receivers, we have to perform the same homomorphic evaluation on the re-encrypted ciphertexts under each receiver’s public key.

## 2 Preliminaries

**Notations.** Let  $S$  be a set and  $\mathcal{P}$  be a distribution over  $S$ . Then,  $a \leftarrow S$  represents that  $a \in S$  is chosen uniformly at random from  $S$ , and  $b \leftarrow \mathcal{P}$  represents that  $b \in S$  is sampled from  $\mathcal{P}$ . For a function  $f$ , we say  $f(\lambda) = \text{negl}(\lambda)$  if  $f$  is negligible in  $\lambda$ , and  $f(\lambda) = \text{poly}(\lambda)$  if  $f$  is polynomial in  $\lambda$ . PPT stands for probabilistic polynomial time.  $G = (V, E)$  represents a graph comprising a set  $V$  of vertices and a set  $E$  of edges.

In this paper, we use bold lower-case letters (e.g.,  $\mathbf{a}$ ) to denote row vectors, and bold upper-case letters (e.g.,  $\mathbf{A}$ ) to denote matrices. We use the notation  $(\mathbf{a}||\mathbf{b})$  to denote the concatenated vector of  $\mathbf{a}$  and  $\mathbf{b}$ . For an  $m \times n$  matrix  $\mathbf{A}$ , let  $\text{height}(\mathbf{A}) = m$  and  $\text{width}(\mathbf{A}) = n$ .

**Tensor Products.** The tensor product  $\mathbf{A} \otimes \mathbf{B}$  of an  $m_1 \times n_1$  matrix  $\mathbf{A}$  and an  $m_2 \times n_2$  matrix  $\mathbf{B}$ , both over a commutative ring  $\mathcal{R}$ , is the  $m_1 m_2 \times n_1 n_2$  matrix

consisting of  $m_2 \times n_2$  blocks. The  $(i, j)$ -th block of the matrix  $\mathbf{A} \otimes \mathbf{B}$  is  $a_{i,j} \cdot \mathbf{B}$ , where  $a_{i,j}$  denotes the  $(i, j)$ -th element of  $\mathbf{A}$ . For any scalar  $r \in \mathcal{R}$ , we have

$$r(\mathbf{A} \otimes \mathbf{B}) = (r\mathbf{A}) \otimes \mathbf{B} = \mathbf{A} \otimes (r\mathbf{B}).$$

We use the mixed product property, which says that

$$(\mathbf{A} \otimes \mathbf{B}) \cdot (\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD})$$

for any matrices  $\mathbf{A}, \mathbf{B}, \mathbf{C}$  and  $\mathbf{D}$  with compatible dimensions. In particular, it holds that

$$\begin{aligned} (\mathbf{A} \otimes \mathbf{B}) &= (\mathbf{A} \otimes \mathbf{I}_{\text{height}(\mathbf{B})}) \cdot (\mathbf{I}_{\text{width}(\mathbf{A})} \otimes \mathbf{B}) \\ &= (\mathbf{I}_{\text{height}(\mathbf{A})} \otimes \mathbf{B}) \cdot (\mathbf{A} \otimes \mathbf{I}_{\text{width}(\mathbf{B})}). \end{aligned}$$

**Approximations.** In this paper, we consider the noisy equations. We use the notation  $\approx$  to denote the equation including some additive error, and we always show a bound on the magnitude of this error. For example,

$$x \approx y \quad (\text{error } E)$$

means that  $x = y + e$  for some error  $e \in [-E, E]$ .

**Learning with Errors.** The Learning With Errors (LWE) assumption was first introduced by Regev [Reg05]. The decision version of the LWE problem is called the DLWE problem.

**Definition 1 (DLWE).** For a security parameter  $\lambda$ , let  $n = n(\lambda)$  be an integer dimension,  $q = q(\lambda) \geq 2$  be a modulus and  $\chi = \chi(\lambda)$  be an error distribution over  $\mathbb{Z}$ . The DLWE $_{n,q,\chi}$  problem is the problem to distinguish the following two distributions. For any  $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ , the LWE distribution  $A_{\mathbf{s},\chi}$  is sampled by choosing a uniformly random  $\mathbf{a} \leftarrow \mathbb{Z}_q^n$  and an error term  $e \leftarrow \chi$  and outputting  $(\mathbf{a}, b = \langle \mathbf{s}, \mathbf{a} \rangle + e) \in \mathbb{Z}_q^{n+1}$ . The uniform distribution  $U_n$  is sampled by choosing a uniformly random  $(\mathbf{a}, b) \leftarrow \mathbb{Z}_q^{n+1}$ . The DLWE $_{n,q,\chi}$  assumption states that it is hard for any PPT adversary to solve the DLWE $_{n,q,\chi}$  problem.

**Leftover Hash Lemma.** We state a version of the leftover hash lemma [ILL89] relating to matrix-vector multiplication, as it was stated in, for example, [BV11a].

**Lemma 1 (Leftover Hash Lemma).** Let  $\lambda$ ,  $n$  and  $q$  be integers and  $m \geq n \log q + 2\lambda$ . Let  $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$  be a uniformly random matrix, let  $\mathbf{r} \leftarrow \{0, 1\}^m$  and  $\mathbf{y} \leftarrow \mathbb{Z}_q^n$ . Then, it holds

$$\Delta((\mathbf{A}, \mathbf{Ar}), (\mathbf{A}, \mathbf{y})) \leq 2^{-\lambda},$$

where  $\Delta(\mathcal{A}, \mathcal{B})$  denotes the statistical distance between the distributions  $\mathcal{A}$  and  $\mathcal{B}$ .

**Gadget Matrix and Bit Decomposition.** We define a gadget vector. For simplicity, throughout this paper we use the powers of two gadget vector

$$\mathbf{g} = (1, 2, 4, \dots, 2^{\ell-1}) \in \mathbb{Z}_q^\ell,$$

where  $\ell = \lceil \log q \rceil$ . We also define the operation  $\mathbf{g}^{-1} : \mathbb{Z}_q \rightarrow \{0, 1\}^\ell$  that takes an input  $a \in \mathbb{Z}_q$  and outputs a binary column vector consisting of the binary representation of the input  $a$ . As such, it satisfies the identity  $\mathbf{g} \cdot \mathbf{g}^{-1}[a] = a$ .

More generally, we define the operation  $(\mathbf{I}_n \otimes \mathbf{g}^{-1})[\cdot]$ , which applies  $\mathbf{g}^{-1}$  entrywise to a height- $n$  vector and outputs a height- $n\ell$  binary vector that satisfies the identity

$$(\mathbf{I}_n \otimes \mathbf{g}) \cdot (\mathbf{I}_n \otimes \mathbf{g}^{-1})[\mathbf{a}] = \mathbf{a}.$$

It is clear that these operations can be applied to matrices.

### 3 Multi-key Homomorphic Proxy Re-Encryption

In this section, we introduce a notion of multi-key homomorphic proxy re-encryption (MH-PRE) and a security definition for MH-PRE schemes.

#### 3.1 Syntax of MH-PRE

We define the syntax of MH-PRE by applying the concept of PRE to MHE. Considering the notion of single-hop PRE scheme, we append the re-encryption key generation algorithm and the re-encryption algorithm to the tuple of the algorithms of MHE. We state the formal definition of MH-PRE below.

**Definition 2 (Multi-key Homomorphic Proxy Re-Encryption).** *A multi-key homomorphic proxy re-encryption (MH-PRE) scheme is a tuple of PPT algorithms MH-PRE = (Setup, KGen, Enc, Dec, Eval, RKGen, ReEnc) having the following properties:*

- $\mathbf{pp} \leftarrow \text{Setup}(1^\lambda, 1^k, 1^d)$  : given a security parameter  $\lambda$ , a bound  $k$  on the number of keys and a bound  $d$  on the circuit depth, outputs a public parameter  $\mathbf{pp}$ . (All the following algorithms implicitly take  $\mathbf{pp}$  as an input.)
- $(\mathbf{pk}, \mathbf{sk}) \leftarrow \text{KGen}(\mathbf{pp})$  : given a public parameter  $\mathbf{pp}$ , outputs a public key  $\mathbf{pk}$  and a secret key  $\mathbf{sk}$ .
- $c \leftarrow \text{Enc}(\mathbf{pk}, m)$  : given a public key  $\mathbf{pk}$  and a message  $m \in \mathcal{M}$ , outputs a ciphertext  $c$ .
- $\tilde{m} \leftarrow \text{Dec}(\mathbf{sk}_1, \dots, \mathbf{sk}_s, c)$  : given a tuple of secret keys  $\mathbf{sk}_1, \dots, \mathbf{sk}_s$  for  $s \leq k$ , outputs a message  $\tilde{m} \in \mathcal{M}$ .
- $c^* \leftarrow \text{Eval}(C, (\{\mathbf{pk}_\ell\}_{\ell \in I_1}, c_1) \dots, (\{\mathbf{pk}_\ell\}_{\ell \in I_s}, c_s))$  : given a circuit  $C : \mathcal{M}^s \rightarrow \mathcal{M}$  and tuples of a ciphertext and public keys  $(\{\mathbf{pk}_\ell\}_{\ell \in I_1}, c_1) \dots, (\{\mathbf{pk}_\ell\}_{\ell \in I_s}, c_s)$  for indices of the keys  $I_i \subset \{1, \dots, k\}$ , outputs a ciphertext  $c^*$ .
- $\mathbf{rk}_{i \rightarrow j} \leftarrow \text{RKGen}(\mathbf{sk}_i, \mathbf{pk}_j)$  : given a secret key  $\mathbf{sk}_i$  and a public key  $\mathbf{pk}_j$ , outputs a re-encryption key  $\mathbf{rk}_{i \rightarrow j}$ .

- $c^* \leftarrow \text{ReEnc}(\{\text{rk}_{\ell \rightarrow j}\}_{\ell \in \bigcup_{i=1}^s I_i}, c)$  : given a tuple of re-encryption keys  $\{\text{rk}_{\ell \rightarrow j}\}_{\ell \in \bigcup_{i=1}^s I_i}$  for indices of the keys  $I_i \subset \{1, \dots, k\}$  and a ciphertext  $c$ , outputs a ciphertext  $c^*$ .

An MH-PRE scheme should satisfy the following two properties.

**Correctness.** For all positive integers  $\lambda, k$  and  $d$ , for every circuit  $C: \mathcal{M}^s \rightarrow \mathcal{M}$  of depth at most  $d$ , for every  $j \in \{1, \dots, k\}$ , for every  $i \in \{1, \dots, s\}$  and for every  $m_i \in \mathcal{M}$ , the following holds. Let  $\text{pp} \leftarrow \text{Setup}(1^\lambda, 1^k, 1^d)$ ,  $(\text{pk}_j, \text{sk}_j) \leftarrow \text{KGen}(\text{pp})$ , ciphertexts  $c_i$  such that  $\text{Dec}(\{\text{sk}_\ell\}_{\ell \in I_i}, c_i) = m_i$  for indices of the keys  $I_i \subset \{1, \dots, k\}$ ,  $c_{\text{Eval}} \leftarrow \text{Eval}(C, (\{\text{pk}_\ell\}_{\ell \in I_1}, c_1), \dots, (\{\text{pk}_\ell\}_{\ell \in I_s}, c_s))$ ,  $\text{rk}_{\ell \rightarrow j} \leftarrow \text{RKGen}(\text{sk}_\ell, \text{pk}_j)$  for all  $\ell \in \bigcup_{i=1}^s I_i$ , and  $c_{\text{ReEnc}} \leftarrow \text{ReEnc}(\{\text{rk}_{\ell \rightarrow j}\}_{\ell \in \bigcup_{i=1}^s I_i}, c_{\text{Eval}})$ . Then, it holds that

$$\begin{aligned} \Pr[\text{Dec}(\{\text{sk}_\ell\}_{\ell \in \bigcup_{i=1}^s I_i}, c_{\text{Eval}}) \neq C(m_1, \dots, m_s)] &= \text{negl}(\lambda), \\ \Pr[\text{Dec}(\text{sk}_j, c_{\text{ReEnc}}) \neq C(m_1, \dots, m_s)] &= \text{negl}(\lambda). \end{aligned}$$

**Compactness.** An MH-PRE scheme is compact if there exists a polynomial  $p = \text{poly}(\cdot, \cdot, \cdot)$  such that  $|c_{\text{Eval}}| \leq p(\lambda, k, d)$  and  $|c_{\text{ReEnc}}| \leq p(\lambda, k, d)$ . In other words, the length of  $c_{\text{Eval}}$  and  $c_{\text{ReEnc}}$  are independent of the circuit  $C$  and the number of inputs, but can depend on  $\lambda, k$  and  $d$ .

### 3.2 Security Definition

We define the IND-CPA security for MH-PRE schemes. Our security definition follows the IND-CPA security of Polyakov et al. [PRSV17] and one for single-hop PRE schemes [LV08]. In the ordinary IND-CPA security for PRE schemes, an adversary is allowed to query a re-encryption key generation between arbitrary honest users. In the definition of [PRSV17], an adversary can only make re-encryption key generation queries according to a re-encryption graph that is a directed acyclic graph whose vertices correspond to honest users and edges represent possible directions of re-encryptions.

**Definition 3 (IND-CPA Security).** Let MH-PRE = (Setup, KGen, Enc, Dec, Eval, RKGen, ReEnc) be an MH-PRE scheme. We define the IND-CPA game between an adversary  $\mathcal{A}$  and a challenger as follows.

#### Phase 1:

- *Setup:* The challenger computes  $\text{pp} \leftarrow \text{Setup}(1^\lambda, 1^k, 1^d)$  and sends  $\text{pp}$  to  $\mathcal{A}$ . Let  $E = \emptyset$  be the set of edges of a re-encryption graph.
- *Uncorrupted key generation:*  $\mathcal{A}$  sends the number of uncorrupted keys  $N_U$  to the challenger. For  $i \in \{1, \dots, N_U\}$ , the challenger computes  $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KGen}(\text{pp})$  and sends  $\text{pk}_i$  to  $\mathcal{A}$ . Let  $\Gamma_H$  be the set of the honest public keys.

- *Corrupted key generation:*  $\mathcal{A}$  sends the number of corrupted keys  $N_C$  to the challenger. For  $i \in \{1, \dots, N_C\}$ , the challenger computes  $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KGen}(\text{pp})$  and sends  $(\text{pk}_i, \text{sk}_i)$  to  $\mathcal{A}$ . Let  $\Gamma_C$  be the set of the corrupted public keys.

**Phase 2:**  $\mathcal{A}$  can issue a polynomial number of these queries in arbitrary order.

- *Re-encryption key generation:*  $\mathcal{A}$  sends  $(i, j)$  to the challenger. If  $i, j \in \Gamma_H$  and the graph  $G = (\Gamma_H, E \cup (i, j))$  is a directed acyclic graph, the challenger adds  $(i, j)$  to  $E$  and returns  $\text{rk}_{i \rightarrow j} \leftarrow \text{RKGen}(\text{sk}_i, \text{pk}_j)$  to  $\mathcal{A}$ . Otherwise the challenger sends  $\perp$  to  $\mathcal{A}$ .
- *Re-encryption:*  $\mathcal{A}$  sends  $(i_1, \dots, i_s, j, c)$  to the challenger. If  $j \in \Gamma_C$  and  $c = c^*$ , then the challenger returns  $\perp$ . Otherwise, the challenger computes  $c_j \leftarrow \text{ReEnc}(\text{RKGen}(\text{sk}_{i_1}, \text{pk}_j), \dots, \text{RKGen}(\text{sk}_{i_s}, \text{pk}_j), c)$  and sends  $c_j$  to  $\mathcal{A}$ .
- *Challenge:*  $\mathcal{A}$  sends  $(i^*, m_0, m_1)$  where  $m_0, m_1 \in \mathcal{M}$  and  $i^* \in \Gamma_H$  to the challenger. The challenger chooses a random bit  $b \in \{0, 1\}$  and returns  $c^* \leftarrow \text{Enc}(\text{pk}_{i^*}, m_b)$  to  $\mathcal{A}$ .  $\mathcal{A}$  can make the challenge query only once.

**Phase 3:**

- $\mathcal{A}$  outputs a bit  $b' \in \{0, 1\}$ .

In this game, we define the advantage of the adversary  $\mathcal{A}$  as

$$\text{Adv}_{\text{MH-PRE}, \mathcal{A}}^{\text{ind-cpa}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

We say that MH-PRE is IND-CPA secure if for any PPT adversary  $\mathcal{A}$ ,

$$\text{Adv}_{\text{MH-PRE}, \mathcal{A}}^{\text{ind-cpa}}(\lambda) = \text{negl}(\lambda)$$

holds.

## 4 Construction

In this section, we show an instantiation of MH-PRE by extending the MHE scheme proposed by Peikert and Shiehian [PS16]. The extension is done by applying the “key switching” [BGV12] technique, which allows us to change a key required for decrypting a ciphertext.

### 4.1 Construction of [PS16]

First, we recall the construction of [PS16] scheme. Their scheme is described in a symmetric-key setting for simplicity in [PS16]. We describe their scheme in a public-key setting using a standard transformation since our MH-PRE scheme requires a public key to generate a re-encryption key.

- **Setup**( $1^\lambda, 1^k, 1^d$ ) : let  $n, q$  and  $\chi$  be the DLWE parameters. Let  $\chi$  be the standard discrete Gaussian error distribution with parameter  $2\sqrt{n}$ . The samples produced from  $\chi$  have magnitudes bounded by some  $E = \Theta(n)$  except with exponentially small probability. Let  $m = \lceil 2n \log q \rceil$ . Output a uniformly random  $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$  as a public parameter **pp**.
- **KGen**(**pp**) : choose  $\bar{\mathbf{t}} \leftarrow \chi^{n-1}$  and define  $\mathbf{t} = (-\bar{\mathbf{t}} \| 1) \in \mathbb{Z}^n$ . Choose  $\mathbf{e} \leftarrow \chi^m$  and define

$$\begin{aligned} \mathbf{b} &= \mathbf{t}\mathbf{A} + \mathbf{e} \\ &\approx \mathbf{t}\mathbf{A} \in \mathbb{Z}_q^m \quad (\text{error } E). \end{aligned} \quad (3)$$

Output a secret key  $\text{sk} = \mathbf{t}$  and a public key  $\text{pk} = \mathbf{b}$ .

- **Enc**( $\text{pk}, \mu \in \{0, 1\}$ ): define

$$\mathbf{B} = \mathbf{A} - \mathbf{e}_n^T \otimes \mathbf{b},$$

where  $\mathbf{e}_n^T \in \mathbb{Z}^n$  is the  $n$ -th standard basis vector of dimension  $n$ . Note that

$$\mathbf{t}\mathbf{B} = \mathbf{t}\mathbf{A} - (\mathbf{t} \otimes 1) \cdot (\mathbf{e}_n^T \otimes \mathbf{b}) \approx \mathbf{0} \quad (\text{error } E). \quad (4)$$

Do the following and output a ciphertext  $c = (\mathbf{C}, \mathbf{F}, \mathbf{D})$ .

1. Choose  $\mathbf{X}_C \leftarrow \{0, 1\}^{m \times n\ell}$  and define  $\bar{\mathbf{C}} = \mathbf{B}\mathbf{X}_C \in \mathbb{Z}_q^{n \times n\ell}$  and

$$\mathbf{C} = \bar{\mathbf{C}} + \mu(\mathbf{I}_n \otimes \mathbf{g}) \in \mathbb{Z}_q^{n \times n\ell}.$$

Notice that  $\mathbf{C}$  is simply a GSW ciphertext of [GSW13] encrypting the message  $\mu$  under the secret key  $\mathbf{t}$ , so it satisfies for  $\mathbf{t}$

$$\mathbf{t}\mathbf{C} = \mathbf{t}\mathbf{B}\mathbf{X}_C + \mu(\mathbf{t} \otimes 1) \cdot (\mathbf{I}_n \otimes \mathbf{g}) \approx \mu(\mathbf{t} \otimes \mathbf{g}) \quad (\text{error } n\ell \cdot E = E_C). \quad (5)$$

2. Choose a uniformly random  $\mathbf{R} \leftarrow \{0, 1\}^{m \times n\ell}$  and define

$$\mathbf{F} = \mathbf{A}\mathbf{R} + \mu(\mathbf{I}_n \otimes \mathbf{g}) \in \mathbb{Z}_q^{n \times n\ell}. \quad (6)$$

3. Choose  $\mathbf{X}_D \leftarrow \{0, 1\}^{nm\ell \times n\ell}$  and define  $\bar{\mathbf{D}} = (\mathbf{1}_{m\ell \times n\ell} \otimes \mathbf{B}) \cdot \mathbf{X}_D \in \mathbb{Z}_q^{nm\ell \times n\ell}$ . We note that

$$(\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot \bar{\mathbf{D}} = (\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot (\mathbf{1}_{m\ell \times n\ell} \otimes \mathbf{B}) \cdot \mathbf{X}_D \approx \mathbf{0} \quad (\text{error } n^3 m^2 \ell^3 \cdot E = E_D).$$

Define

$$\mathbf{D} = \bar{\mathbf{D}} + (\mathbf{R} \otimes \mathbf{g}^T \otimes \mathbf{e}_n^T) \in \mathbb{Z}_q^{nm\ell \times n\ell}.$$

Therefore, we have

$$\begin{aligned} (\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot \mathbf{D} &= (\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot \bar{\mathbf{D}} + (\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot (\mathbf{R} \otimes \mathbf{g}^T \otimes \mathbf{e}_n^T) \\ &\approx \mathbf{R} \otimes \mathbf{g}^T \quad (\text{error } E_D). \end{aligned} \quad (7)$$

- **Dec**( $\text{sk}_1, \dots, \text{sk}_s, c$ ): parse  $c = (\mathbf{C}, \mathbf{F}, \mathbf{D})$ . Define  $\mathbf{t} = (\mathbf{t}_1 \| \dots \| \mathbf{t}_s)$ . Let  $\mathbf{c}$  be the penultimate column of  $\mathbf{C}$  and output  $\tilde{\mu} = \lfloor \mathbf{t} \cdot \mathbf{c} / 2^{\ell-2} \rfloor$ .
- **Eval**( $\mathbf{C}, (\text{pk}_1, (\mathbf{C}_1, \mathbf{F}_1, \mathbf{D}_1)), (\text{pk}_2, (\mathbf{C}_2, \mathbf{F}_2, \mathbf{D}_2))$ ): output the evaluated ciphertext  $\mathbf{c}^*$ . This algorithm works exactly the same as the scheme described in Sect. 3 of [PS16]. We describe this in Appendix A for completeness.



### 4.2 Re-Encryption Key Generation and Re-Encryption

In this section, we describe the re-encryption key generation algorithm RKGen and the re-encryption algorithm ReEnc.

- RKGen( $sk_i, pk_j$ ) : choose  $\mathbf{X}_i \leftarrow \{0, 1\}^{m \times n\ell}$  and define  $\mathbf{B}_j = \mathbf{A} - \mathbf{e}_n^T \otimes \mathbf{b}_j$ .  
Output

$$rk_{i \rightarrow j} = \mathbf{B}_j \mathbf{X}_i + \begin{pmatrix} \mathbf{0}_{(n-1) \times n\ell} \\ \mathbf{t}_i \cdot (\mathbf{I}_n \otimes \mathbf{g}) \end{pmatrix}.$$

- ReEnc( $rk_{1 \rightarrow j}, \dots, rk_{s \rightarrow j}, (\mathbf{C}, \mathbf{F}, \mathbf{D})$ ) : suppose  $\mathbf{C} \in \mathbb{Z}_q^{n' \times n' \ell}$ , where  $n' = ns$ .  
Define

$$\mathbf{C}^* = (rk_{1 \rightarrow j} \parallel \dots \parallel rk_{s \rightarrow j}) \cdot (\mathbf{I}_{n'} \otimes \mathbf{g}^{-1})[\mathbf{C}]$$

and output the penultimate column of  $\mathbf{C}^*$  as a re-encrypted ciphertext  $\mathbf{c}^*$ .

Below, we show the correctness of our re-encryption algorithm. In order to perform homomorphic evaluations on ciphertexts under different keys, the MHE scheme [PS16] extends the ciphertexts while satisfying Eq. (5). Specifically, the extended ciphertext consists of

$$\mathbf{C} \in \mathbb{Z}_q^{n' \times n' \ell}, \mathbf{F} \in \mathbb{Z}_q^{n \times n\ell}, \mathbf{D} \in \mathbb{Z}_q^{n' m\ell \times n\ell},$$

where  $n' = ns$  for any positive integer  $s \leq k$  and satisfies Eq. (5) for the concatenation of the secret keys  $\mathbf{t} = (\mathbf{t}_1 \parallel \dots \parallel \mathbf{t}_s) \in \mathbb{Z}^{n'}$ .

In order to decrypt the ciphertext  $\mathbf{C}$ , we require re-encryption keys  $rk_{i \rightarrow j} \leftarrow \text{RKGen}(sk_i = \mathbf{t}_i, pk_j = \mathbf{B}_j)$  for  $i \in \{1, \dots, s\}$ . Each re-encryption key  $rk_{i \rightarrow j}$  satisfies the following equation for the secret key  $\mathbf{t}_j$ :

$$\begin{aligned} \mathbf{t}_j \cdot rk_{i \rightarrow j} &= \mathbf{t}_j \mathbf{B}_j \mathbf{X}_i + (\overline{\mathbf{t}_j} \parallel 1) \begin{pmatrix} \mathbf{0}_{(n-1) \times n\ell} \\ \mathbf{t}_i \cdot (\mathbf{I}_n \otimes \mathbf{g}) \end{pmatrix} \\ &\approx \mathbf{t}_i \cdot (\mathbf{I}_n \otimes \mathbf{g}) \quad (\text{from Equation (4)}). \end{aligned}$$

Hence, the concatenation of the re-encryption keys satisfies the following equation for the concatenated the secret key  $\mathbf{t} = (\mathbf{t}_1 \parallel \dots \parallel \mathbf{t}_s)$ :

$$\begin{aligned} \mathbf{t}_k \cdot (rk_{1 \rightarrow j} \parallel \dots \parallel rk_{s \rightarrow j}) &= (\mathbf{t}_k \cdot rk_{1 \rightarrow j} \parallel \dots \parallel \mathbf{t}_k \cdot rk_{s \rightarrow j}) \\ &\approx (\mathbf{t}_1 \cdot (\mathbf{I}_n \otimes \mathbf{g}) \parallel \dots \parallel \mathbf{t}_s \cdot (\mathbf{I}_n \otimes \mathbf{g})) \quad (\text{from the above equation}) \\ &= (\mathbf{t}_1 \parallel \dots \parallel \mathbf{t}_s) \cdot (\mathbf{I}_{n'} \otimes \mathbf{g}). \end{aligned}$$

Then, for the re-encrypted ciphertext  $\mathbf{C}^*$  and secret key  $\mathbf{t}_j$ , it holds that

$$\begin{aligned} \mathbf{t}_j \mathbf{C}^* &= (\mathbf{t}_1 \parallel \dots \parallel \mathbf{t}_s) \cdot (\mathbf{I}_{n'} \otimes \mathbf{g}) \cdot (\mathbf{I}_{n'} \otimes \mathbf{g}^{-1})[\mathbf{C}^*] \\ &= (\mathbf{t}_1 \parallel \dots \parallel \mathbf{t}_s) \cdot \mathbf{C}^* \\ &\approx \mu \cdot ((\mathbf{t}_1 \parallel \dots \parallel \mathbf{t}_s) \otimes \mathbf{g}) \quad (\text{from Equation (5) holds for } \mathbf{t} \text{ and } \mathbf{C}). \end{aligned}$$

So, as well as [PS16], we can correctly decrypt the message from the  $(\ell - 1)$ -th element of  $\mathbf{t}_j \mathbf{C}^*$ .

This equation shows that the re-encrypted ciphertext  $\mathbf{C}^*$  leaks the information on  $\mathbf{t}_1, \dots, \mathbf{t}_s$  to the owner of  $\mathbf{t}_j$ . To prevent the leakage of the secret keys, ReEnc outputs the penultimate column of  $\mathbf{C}^*$  as a re-encrypted ciphertext  $\mathbf{c}^*$  instead of the matrix  $\mathbf{C}^*$  as it is. It is obvious that we can decrypt the re-encrypted ciphertext  $\mathbf{c}^*$  since Dec uses only a penultimate column of the ciphertext matrix  $\mathbf{C}^*$ . By outputting only  $\mathbf{c}^*$  as a re-encrypted ciphertext, it is not possible to re-encrypt any more. Therefore, our scheme can be seen as a single-hop PRE scheme. Also for the same reason it is not possible to perform homomorphic evaluation on re-encrypted ciphertexts while it can be performed any times before re-encryption.

## 5 IND-CPA Security of MH-PRE

We show that our MH-PRE scheme  $\text{MH-PRE} = (\text{Setup}, \text{KGen}, \text{Enc}, \text{Dec}, \text{Eval}, \text{RKGen}, \text{ReEnc})$  described in Sect. 4 is IND-CPA secure in the sense of Definition 3. Formally, we prove the following theorem.

**Theorem 1.** *MH-PRE is IND-CPA secure assuming the hardness of the  $DLWE_{n-1, q, \chi}$  problem.*

*Proof.* Let  $\mathcal{A}$  be an adversary for the IND-CPA security of MH-PRE. Then, we consider the following sequence of games.

**Game 0:** This is the original IND-CPA game of MH-PRE. Without loss of generality, we can assume that  $\Gamma_H = \{1, \dots, N\}$  and  $\Gamma_C = \{N + 1, \dots, M\}$ . Furthermore without loss of generality, let  $1, \dots, N$  be the topological order dictated by the re-encryption graph, namely there are no edges from  $i$  to  $j$  if  $i < j$ . In other words,  $\mathcal{A}$  can only make re-encryption key generation queries to generate  $\text{rk}_{i \rightarrow j}$  where  $i > j$ .

We consider Game  $k$  ( $k = 1, \dots, N$ ) divided into Game  $k.1$  and Game  $k.2$ .

**Game  $k.1$ :** Same as Game  $k - 1.2$  expect the following. Here, let Game 0.2 denote Game 0. When  $\mathcal{A}$  makes an uncorrupted key generation query, the challenger generates a public key  $\text{pk}_i$  by choosing a random vector  $\mathbf{b}_i \leftarrow \mathbb{Z}_q^m$  for all  $i \leq k$  and computes  $\text{pk}_i \leftarrow \text{KGen}(\text{pp})$  for all  $k < i \leq N$ .

**Game  $k.2$ :** Same as Game  $k.1$ , expect the following. When  $\mathcal{A}$  makes a re-encryption key generation query, the challenger generates a re-encryption key  $\text{rk}_{i \rightarrow j}$  where  $i > j$  by choosing a random matrix from  $\mathbb{Z}_q^{n \times n\ell}$  for all  $i, j \leq k$  and computes  $\text{rk}_{i \rightarrow j} \leftarrow \text{RKGen}(\text{sk}_j, \text{pk}_i)$  for all  $k < i, j \leq N$ .

**Game final:** Same as Game  $N.2$  expect the following. When  $\mathcal{A}$  makes a challenge query, the challenger generates a ciphertext  $(\mathbf{C}, \mathbf{F}, \mathbf{D})$  by choosing random matrices  $\mathbf{C} \leftarrow \mathbb{Z}_q^{n \times n\ell}$ ,  $\mathbf{F} \leftarrow \mathbb{Z}_q^{n \times n\ell}$  and  $\mathbf{D} \leftarrow \mathbb{Z}_q^{nm\ell \times n\ell}$ .

Here, let  $Adv_{\mathcal{A}}^{\text{Game } i}(\lambda)$  be the advantage of  $\mathcal{A}$  in Game  $i$ . Below, for the above sequence of the games, we estimate the differences of the advantages of  $\mathcal{A}$  between each game. First of all, since Game 0 is the original IND-CPA game of MH-PRE, we have

$$Adv_{\text{MH-PRE}, \mathcal{A}}^{\text{ind-cpa}}(\lambda) = Adv_{\mathcal{A}}^{\text{Game } 0}(\lambda).$$

Next, for Game  $k.1$  and Game  $k.2$ , we show the following lemma.

**Lemma 2.**  $|Adv_{\mathcal{A}}^{\text{Game } k.1}(\lambda) - Adv_{\mathcal{A}}^{\text{Game } k.2}(\lambda)| = \text{negl}(\lambda)$ .

*Proof.* In Game  $k.1$ , the re-encryption keys  $\text{rk}_{k \rightarrow i}$  where  $i < k$  generated by the challenger satisfy the following properties. In this game,  $\text{rk}_{k \rightarrow i}$  is computed by  $\text{rk}_{k \rightarrow i} = \mathbf{B}_i \mathbf{X}_k + \begin{pmatrix} \mathbf{0}^{(n-1) \times n\ell} \\ \mathbf{t}_k \cdot (\mathbf{I}_n \otimes \mathbf{g}) \end{pmatrix}$ , where  $\mathbf{X}_k \leftarrow \{0, 1\}^{m \times n\ell}$ . Since  $\mathbf{b}_i$  and  $\mathbf{A}$  are uniformly random for all  $i < k$ ,  $\mathbf{B}_i = \mathbf{A} - \mathbf{e}_n^T \otimes \mathbf{b}_i$  is also uniformly random. Moreover, since  $\mathbf{B}_i$  and  $\mathbf{X}_k$  are uniformly random over  $\mathbb{Z}_q^{n \times m}$  and  $\{0, 1\}^{m \times n\ell}$ , respectively,  $\mathbf{B}_i \mathbf{X}_k$  is statistically indistinguishable from a uniformly random matrix by the leftover hash lemma. As a result,  $\text{rk}_{k \rightarrow i}$  is statistically indistinguishable from a uniformly random matrix and Game  $k.1$  and Game  $k.2$  are statistically indistinguishable.  $\square$

For Game  $k - 1.2$  and Game  $k.1$ , the following lemma holds.

**Lemma 3.**  $|Adv_{\mathcal{A}}^{\text{Game } k-1.2}(\lambda) - Adv_{\mathcal{A}}^{\text{Game } k.1}(\lambda)| = \text{negl}(\lambda)$ .

*Proof.* To prove the lemma, we construct the following PPT algorithm  $\mathcal{B}$  by using the adversary  $\mathcal{A}$ .  $\mathcal{B}$  distinguishes the LWE distribution  $A_{\bar{\mathbf{t}}, \chi}$  for some  $\bar{\mathbf{t}} \leftarrow \chi^{n-1}$  from the uniform distribution  $U_n$ .  $\mathcal{B}$  is given an arbitrary number of vectors  $\mathbf{x}$  sampled from the LWE distribution or the uniform distribution.

**Phase 1:**

- Setup:  $\mathcal{B}$  computes  $\bar{\mathbf{A}} = (\mathbf{x}_1^T \| \dots \| \mathbf{x}_m^T)$  and chooses a random vector  $\mathbf{b}_k \leftarrow \mathbb{Z}_q^m$ . Then,  $\mathcal{B}$  sets a public parameter  $\mathbf{A} = \bar{\mathbf{A}} + \mathbf{e}_n^T \otimes \mathbf{b}_k$  and sends it to  $\mathcal{A}$ .
- Uncorrupted key generation: When  $\mathcal{A}$  makes an uncorrupted key generation query,  $\mathcal{B}$  responds as follows.
  - For  $i < k$ ,  $\mathcal{B}$  chooses a random vector  $\mathbf{b}_i \leftarrow \mathbb{Z}_q^m$  and sets  $\text{pk}_i = \mathbf{b}_i$ .
  - For  $i = k$ ,  $\mathcal{B}$  sets  $\text{pk}_i = \mathbf{b}_k$  where  $\mathbf{b}_k$  is the vector chosen in the setup phase.
  - For  $i > k$ ,  $\mathcal{B}$  computes  $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KGen}(\mathbf{A})$ .

Finally  $\mathcal{B}$  sends  $\text{pk}_i$  ( $i \in \{1, \dots, k\}$ ) to  $\mathcal{A}$ .

- Corrupted key generation: When  $\mathcal{A}$  makes a corrupted key generation query,  $\mathcal{B}$  computes  $(\text{pk}_i, \text{sk}_i) \leftarrow \text{KGen}(\mathbf{A})$  and sends  $(\text{pk}_i, \text{sk}_i)$  to  $\mathcal{A}$ .

**Phase 2:**

- Re-encryption key generation: When  $\mathcal{A}$  makes a re-encryption key generation query  $(i, j)$ ,  $\mathcal{B}$  returns  $\text{rk}_{i \rightarrow j} \in \mathbb{Z}_q^{n \times n\ell}$  chosen uniformly at random if  $i < j < k$ , and  $\text{rk}_{i \rightarrow j} \leftarrow \text{RKGen}(\text{sk}_i, \text{pk}_j)$  if  $k < i < j$ .

- Re-encryption: When  $\mathcal{A}$  makes a re-encryption query  $(i_1, \dots, i_s, j, c)$ ,  $\mathcal{B}$  returns  $c_j \leftarrow \text{ReEnc}(\text{rk}_{i_1 \leftarrow j}, \dots, \text{rk}_{i_s \leftarrow j}, c)$ .
- Challenge: When  $\mathcal{A}$  makes a challenge query  $(i^*, m_0, m_1)$ ,  $\mathcal{B}$  chooses a random bit  $b \in \{0, 1\}$  and returns  $c^* \leftarrow \text{Enc}(\text{pk}_{i^*}, m_b)$ .

**Phase 3:**

- When  $\mathcal{A}$  terminates with output  $b' \in \{0, 1\}$ ,  $\mathcal{B}$  outputs 1 if  $b = b'$ . Otherwise  $\mathcal{B}$  outputs 0.

We see that if the input distribution to  $\mathcal{B}$  is the LWE distribution  $A_{\bar{\mathbf{t}}, \chi}$  for some  $\bar{\mathbf{t}} \leftarrow \chi^{n-1}$ ,  $\mathcal{B}$  simulates Game  $k - 1.2$  for  $\mathcal{A}$ . The first  $n - 1$  rows of  $\bar{\mathbf{A}}$  and  $\mathbf{b}_k$  are uniformly random, hence  $\mathbf{A}$  is uniformly random by the construction. Moreover,  $\mathbf{b}_k \approx (-\bar{\mathbf{t}}\|1) \cdot \mathbf{A}$  has the same distribution as in the real game. Finally, since  $\text{rk}_{i \rightarrow j}$  where  $i < j < k$  is replaced with a uniformly random matrix in the previous games,  $\mathcal{B}$  perfectly simulates Game  $k - 1.2$  for  $\mathcal{A}$ . By contrast, if the input distribution is the uniform distribution  $U_n$ , it is clear that  $\mathcal{B}$  simulates Game  $k.1$  for  $\mathcal{A}$  by the construction of  $\mathcal{B}$ .

From the above arguments and the DLWE assumption, we have

$$|\text{Adv}_{\mathcal{A}}^{\text{Game } k-1.2}(\lambda) - \text{Adv}_{\mathcal{A}}^{\text{Game } k.1}(\lambda)| = \text{negl}(\lambda). \quad \square$$

For Game  $N.2$  and Game final, the following lemma holds.

**Lemma 4.**  $|\text{Adv}_{\mathcal{A}}^{\text{Game } N.2}(\lambda) - \text{Adv}_{\mathcal{A}}^{\text{Game } \text{final}}(\lambda)| = \text{negl}(\lambda)$ .

*Proof.* In Game final, the ciphertext matrices are chosen uniformly at random. Since the public keys  $\mathbf{b}_i$  for all  $i \in \{1, \dots, N\}$  has been replaced with the uniformly random vectors in the previous games,  $\mathbf{B}_k$  computed in  $\text{Enc}$  is also uniformly random. Therefore since  $\mathbf{A}$  and  $\mathbf{B}$  are uniformly random,  $\bar{\mathbf{C}}$ ,  $\mathbf{AR}$ ,  $\bar{\mathbf{D}}$  is statistically indistinguishable from uniformly random by the leftover hash lemma. As a result, we see that Game  $N.2$  and Game final are statistically indistinguishable.  $\square$

Finally, for the advantage of  $\mathcal{B}$  in Game final, we have

$$\text{Adv}_{\mathcal{A}}^{\text{Game } \text{final}}(\lambda) = \text{negl}(\lambda).$$

From Lemmas 2, 3 and 4, the following inequality holds.

$$\begin{aligned} \text{Adv}_{\text{MH-PRE}, \mathcal{A}}^{\text{ind-cpa}}(\lambda) &\leq \sum_{k=1}^N |\text{Adv}_{\mathcal{A}}^{\text{Game } k-1.2}(\lambda) - \text{Adv}_{\mathcal{A}}^{\text{Game } k.1}(\lambda)| \\ &\quad + |\text{Adv}_{\mathcal{A}}^{\text{Game } N.2}(\lambda) - \text{Adv}_{\mathcal{A}}^{\text{Game } \text{final}}(\lambda)| + \text{Adv}_{\mathcal{A}}^{\text{Game } \text{final}}(\lambda) \\ &= \text{negl}(\lambda) \end{aligned}$$

Since the choice of  $\mathcal{A}$  is arbitrary, MH-PRE is IND-CPA secure.  $\square$

## 6 Conclusion

In this paper, we proposed the model of *multi-key homomorphic proxy re-encryption* (MH-PRE) by applying the notion of proxy re-encryption to multi-key homomorphic encryption (MHE). While all secret keys corresponding to an evaluated ciphertext are necessary for decryption in ordinary MHE, an evaluated ciphertext of our scheme can be decrypted with a specific secret key. Moreover, we showed a concrete instantiation of MH-PRE by applying the *key switching technique* of [BGV12] to the MHE scheme of [PS16] and proved the IND-CPA security under the DLWE assumption. We will consider other instantiations based on other MHE schemes [BP16, CZW17] as a future work.

**Acknowledgment.** We would like to thank the anonymous reviewers of ISC 2018 for their careful reading and comments.

## Appendix A The Description of the Eval Algorithm

For completeness, in this section, we describe the algorithm Eval of the MHE scheme of [PS16]. This algorithm works exactly same as the scheme described in Sect. 3 of [PS16].

**Ciphertext Extending.** First, in order to perform homomorphic operations correctly in the Eval algorithm, all the input ciphertexts must correspond to the same secret key. Therefore, when the input ciphertexts correspond to the different secret keys, each ciphertext is expanded so that the same secret key properly corresponds.

Consider the ciphertext  $c = (\mathbf{C}, \mathbf{F}, \mathbf{D})$  and the associated secret key  $\mathbf{t} \in \mathbb{Z}^{n'}$ , where  $n' = ns$  for some positive integer  $s$  and  $\mathbf{t}$  is the concatenation of  $s$  individual secret keys. Therefore the ciphertext  $c$  consists of component matrices

$$\mathbf{C} \in \mathbb{Z}_q^{n' \times n' \ell}, \mathbf{F} \in \mathbb{Z}_q^{n \times n \ell}, \mathbf{D} \in \mathbb{Z}_q^{n' n \ell \times n \ell}$$

that satisfy Eqs. (5), (6) and (7) for some randomness  $\mathbf{R} \in \mathbb{Z}^{m \times n \ell}$ . Our goal is to extend  $c = (\mathbf{C}, \mathbf{F}, \mathbf{D})$  to a new ciphertext  $c' = (\mathbf{C}', \mathbf{F}', \mathbf{D}')$  that satisfies Eqs. (5), (6) and (7) for the concatenated secret key  $\mathbf{t}' = (\mathbf{t} \parallel \mathbf{t}^*) \in \mathbb{Z}^{n'+n}$  where  $\mathbf{t}^*$  is an additional secret key and some randomness  $\mathbf{R}'$  without changing the encrypted message. For ciphertext extending, the public key  $\mathbf{b}^* \approx \mathbf{t}^* \mathbf{A}$  corresponding to the additional secret key  $\mathbf{t}^*$  is used. We do so as follows.

- $\mathbf{F}$  and the randomness is unchanged: Define  $\mathbf{F}' = \mathbf{F}$  and  $\mathbf{R} = \mathbf{R}'$ .
- Define

$$\mathbf{D}' = (\mathbf{I}_{m \ell} \otimes \begin{pmatrix} \mathbf{I}_{n'} \\ \mathbf{0}_{n \times n'} \end{pmatrix}) \cdot \mathbf{D}.$$

Then, Eq. (7) is preserved:  $(\mathbf{I}_{m \ell} \otimes \mathbf{t}') \cdot \mathbf{D}' = (\mathbf{I}_{m \ell} \otimes \mathbf{t}) \cdot \mathbf{D} \approx \mathbf{R} \otimes \mathbf{g}^T$ .

– Define

$$\mathbf{C}' = \begin{pmatrix} \mathbf{C} & \mathbf{X} \\ & \mathbf{F} \end{pmatrix},$$

where  $\mathbf{X}$  is defined as follows:

$$\begin{aligned} \mathbf{s} &= [-\mathbf{b}^*](\mathbf{I}_m \otimes \mathbf{g}^{-T}) \in \{0, 1\}^{m\ell} \\ \mathbf{X} &= (\mathbf{s} \otimes \mathbf{I}_{n'}) \cdot \mathbf{D} \in \mathbb{Z}_q^{n' \times n\ell}. \end{aligned}$$

Note that by the construction,

$$\begin{aligned} \mathbf{tX} &= (\mathbf{1} \otimes \mathbf{t}) \cdot (\mathbf{s} \otimes \mathbf{I}_{n'}) \cdot \mathbf{D} \\ &= (\mathbf{s} \otimes \mathbf{t}) \cdot \mathbf{D} \\ &= (\mathbf{s} \otimes \mathbf{1}) \cdot (\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot \mathbf{D} \\ &\approx \mathbf{s} \cdot (\mathbf{R} \otimes \mathbf{g}^t) && (\text{error } m\ell \cdot E_D) \\ &= [\mathbf{b}^*](\mathbf{I}_m \otimes \mathbf{g}^{-t})(\mathbf{R} \otimes \mathbf{g}) \\ &= -\mathbf{b}^*\mathbf{R}. \end{aligned}$$

Finally, we see that Eq. (5) is preserved:

$$\begin{aligned} \mathbf{t}'\mathbf{C}' &= \mathbf{t}' \cdot \begin{pmatrix} \mathbf{C} & \mathbf{X} \\ & \mathbf{F} \end{pmatrix} \\ &= (\mathbf{tC} \quad \mathbf{tX} + \mathbf{t}^*\mathbf{F}) \\ &\approx (\mu(\mathbf{t} \otimes \mathbf{g}) \quad \mathbf{tX} + \mathbf{t}^*\mathbf{AR} + \mu(\mathbf{t}^* \otimes \mathbf{g})) \quad (\text{error } E_C) \\ &\approx (\mu(\mathbf{t} \otimes \mathbf{g}) \quad \mathbf{tX} + \mathbf{b}^*\mathbf{R} + \mu(\mathbf{t}^* \otimes \mathbf{g})) \quad (\text{error } m\|\mathbf{R}\|_\infty \cdot E) \\ &\approx \mu(\mathbf{t}' \otimes \mathbf{g}) \quad (\text{error } m\ell \cdot E_D). \end{aligned}$$

**Homomorphic Operation.** Next, we describe homomorphic addition and multiplication. Suppose two ciphertexts  $c_1 = (\mathbf{C}_1, \mathbf{F}_1, \mathbf{D}_1)$  and  $c_2 = (\mathbf{C}_2, \mathbf{F}_2, \mathbf{D}_2)$  that respectively encrypt  $\mu_1$  and  $\mu_2$ , with the randomness  $\mathbf{R}_1$  and  $\mathbf{R}_2$ , under a common secret key  $\mathbf{t} \in \mathbb{Z}^{n'}$ .

– **Homomorphic Additions:** Add the corresponding matrices,

$$(\mathbf{C}_{\text{add}}, \mathbf{F}_{\text{add}}, \mathbf{D}_{\text{add}}) = (\mathbf{C}_1 + \mathbf{C}_2, \mathbf{F}_1 + \mathbf{F}_2, \mathbf{D}_1 + \mathbf{D}_2).$$

We verify that Eqs. (5), (6) and (7) hold for the new ciphertext with the message  $\mu_1 + \mu_2$  and the randomness  $\mathbf{R}_{\text{add}} = \mathbf{R}_1 + \mathbf{R}_2$ .

$$\begin{aligned} \mathbf{t} \cdot (\mathbf{C}_1 + \mathbf{C}_2) &= \mathbf{t} \cdot (\overline{\mathbf{C}}_1 + \mu_1(\mathbf{I}_n \otimes \mathbf{g}) + \overline{\mathbf{C}}_2 + \mu_2(\mathbf{I}_n \otimes \mathbf{g})) \\ &\approx (\mu_1 + \mu_2)(\mathbf{t} \otimes \mathbf{g}) \quad (\text{error } E_{C_1} + E_{C_2}) \end{aligned}$$

$$\begin{aligned} \mathbf{F}_1 + \mathbf{F}_2 &= \mathbf{AR}_1 + \mu_1(\mathbf{I}_n \otimes \mathbf{g}) + \mathbf{AR}_2 + \mu_2(\mathbf{I}_n \otimes \mathbf{g}) \\ &= \mathbf{AR}_{\text{add}} + (\mu_1 + \mu_2)(\mathbf{I}_n \otimes \mathbf{g}) \end{aligned}$$

$$\begin{aligned}
(\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot (\mathbf{D}_1 + \mathbf{D}_2) &= (\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot \mathbf{D}_1 + (\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot \mathbf{D}_2 \\
&\approx (\mathbf{R}_1 \otimes \mathbf{g}) + (\mathbf{R}_2 \otimes \mathbf{g}) \\
&= \mathbf{R}_{\text{add}} \otimes \mathbf{g} \quad (\text{error } E_{D_1} + E_{D_2})
\end{aligned}$$

– **Homomorphic Multiplications:** Define the following matrices:

$$\begin{aligned}
\mathbf{S}_c &= (\mathbf{I}_{n'} \otimes \mathbf{g}^{-1})[\mathbf{C}_2] \\
\mathbf{S}_f &= (\mathbf{I}_n \otimes \mathbf{g}^{-1})[\mathbf{F}_2] \\
\mathbf{S}_d &= (\mathbf{I}_{n'm\ell} \otimes \mathbf{g}^{-1})[\mathbf{D}_2]
\end{aligned}$$

and output the ciphertext

$$\begin{aligned}
\mathbf{C}_{\text{mul}} &= \mathbf{C}_1 \cdot \mathbf{S}_c \\
\mathbf{F}_{\text{mul}} &= \mathbf{F}_1 \cdot \mathbf{S}_f \\
\mathbf{D}_{\text{mul}} &= \mathbf{D}_1 \cdot \mathbf{S}_f + (\mathbf{I}_{m\ell} \otimes \mathbf{C}_1) \cdot \mathbf{S}_d.
\end{aligned}$$

The associated randomness is defined as

$$\mathbf{R}_{\text{mul}} = \mathbf{R}_1 \mathbf{S}_f + \mu_1 \mathbf{R}_2.$$

We show that the output ciphertext of the homomorphic multiplications satisfies Eqs. (5), (6) and (7) for the secret key  $\mathbf{t}$ , the message  $\mu_1 \mu_2$  and the randomness  $\mathbf{R}_{\text{mul}}$ . First, we can see  $\mathbf{C}_{\text{mul}}$  satisfies Eq. (5):

$$\begin{aligned}
\mathbf{t} \mathbf{C}_{\text{mul}} &= \mathbf{t} \mathbf{C}_1 \cdot \mathbf{S}_c = \mathbf{t} \overline{\mathbf{C}}_1 \cdot \mathbf{S}_c + \mu_1 (\mathbf{t} \otimes \mathbf{g}) \cdot \mathbf{S}_c \\
&\approx \mu_1 (\mathbf{t} \otimes \mathbf{g}) \cdot \mathbf{S}_c \quad (\text{error } n'\ell \cdot E_{C_1}) \\
&= \mu_1 (\mathbf{t} \otimes \mathbf{g}) \cdot (\mathbf{I}_{n'} \otimes \mathbf{g}^{-1})[\mathbf{C}_2] \\
&= \mu_1 \mathbf{t} \mathbf{C}_2 \\
&\approx \mu_1 \mu_2 (\mathbf{t} \otimes \mathbf{g}) \quad (\text{error } \mu_1 E_{C_2}).
\end{aligned}$$

Similarly, Eq. (6) is preserved by construction of  $\mathbf{F}_{\text{mul}}$ :

$$\begin{aligned}
\mathbf{F}_{\text{mul}} &= \mathbf{F}_1 \cdot \mathbf{S}_f = (\mathbf{A} \mathbf{R}_1 + \mu_1 (\mathbf{I}_n \otimes \mathbf{g})) \cdot \mathbf{S}_f \\
&= \mathbf{A} \mathbf{R}_1 \cdot \mathbf{S}_f + \mu_1 (\mathbf{I}_n \otimes \mathbf{g}) \cdot (\mathbf{I}_n \otimes \mathbf{g}^{-1})[\mathbf{F}_2] \\
&= \mathbf{A} \mathbf{R}_1 \cdot \mathbf{S}_f + \mu_1 \mathbf{F}_2 \\
&= \mathbf{A} \mathbf{R}_1 \cdot \mathbf{S}_f + \mu_1 \mathbf{A} \mathbf{R}_2 + \mu_1 \mu_2 (\mathbf{I}_n \otimes \mathbf{g}) \\
&= \mathbf{A} \mathbf{R}_{\text{mul}} + \mu_1 \mu_2 (\mathbf{I}_n \otimes \mathbf{g}).
\end{aligned}$$

Finally, to see that Eq. (7) holds for  $\mathbf{D}_{\text{mul}}$ , first note that

$$\begin{aligned}
(\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot \mathbf{D}_1 \cdot \mathbf{S}_f &= ((\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot \overline{\mathbf{D}}_1 + (\mathbf{R}_1 \otimes \mathbf{g}^T)) \cdot \mathbf{S}_f \\
&\approx (\mathbf{R}_1 \otimes \mathbf{g}^T) \cdot (\mathbf{S}_f \otimes \mathbf{1}) \quad (\text{error } n\ell \cdot E_{D_1}) \\
&= (\mathbf{R}_1 \cdot \mathbf{S}_f) \otimes \mathbf{g}^T.
\end{aligned}$$

In addition,

$$\begin{aligned}
& (\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot (\mathbf{I}_{m\ell} \otimes \mathbf{C}_1) \cdot \mathbf{S}_d \\
&= (\mathbf{I}_{m\ell} \otimes \mathbf{tC}_1) \cdot \mathbf{S}_d \\
&= ((\mathbf{I}_{m\ell} \otimes \mathbf{t}\bar{\mathbf{C}}_1) + \mu_1(\mathbf{I}_{m\ell} \otimes \mathbf{t} \otimes \mathbf{g})) \cdot \mathbf{S}_d \\
&\approx \mu_1(\mathbf{I}_{m\ell} \otimes \mathbf{t} \otimes \mathbf{g}) \cdot \mathbf{S}_d \quad (\text{error } n'\ell \cdot E_{C_1}) \\
&= \mu_1(\mathbf{I}_{m\ell} \otimes \mathbf{t} \otimes \mathbf{g}) \cdot (\mathbf{I}_{n'm\ell} \otimes \mathbf{g}^{-1})[\mathbf{D}_2] \\
&= \mu_1(\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot \mathbf{D}_2 \\
&\approx \mu_1(\mathbf{R}_2 \otimes \mathbf{g}^T) \quad (\text{error } \mu_1 \cdot E_{D_2}).
\end{aligned}$$

Thus, Eq. (7) holds for  $\mathbf{D}_{\text{mul}}$ :

$$\begin{aligned}
& (\mathbf{I}_{m\ell} \otimes \mathbf{t}) \cdot \mathbf{D}_{\text{mul}} \\
&\approx (\mathbf{R}_1 \cdot \mathbf{S}_f) \otimes \mathbf{g}^T + \mu_1(\mathbf{R}_2 \otimes \mathbf{g}^T) \quad (\text{error } n\ell \cdot E_{D_1} + n'\ell \cdot E_{C_1} + \mu_1 \cdot E_{D_2}) \\
&= \mathbf{R}_{\text{mul}} \otimes \mathbf{g}^T.
\end{aligned}$$

## References

- [AFGH06] Ateniese, G., Kevin, F., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.* **9**(1), 1–30 (2006)
- [BGV12] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: *ITCS*, pp. 309–325. ACM (2012)
- [BHP17] Brakerski, Z., Halevi, S., Polychroniadou, A.: Four round secure computation without setup. In: Kalai, Y., Reyzin, L. (eds.) *TCC 2017, Part II*. LNCS, vol. 10677, pp. 645–677. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70500-2\\_22](https://doi.org/10.1007/978-3-319-70500-2_22)
- [BP16] Brakerski, Z., Perlman, R.: Lattice-based fully dynamic multi-key FHE with short ciphertexts. In: Robshaw, M., Katz, J. (eds.) *CRYPTO 2016*. LNCS, vol. 9814, pp. 190–213. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53018-4\\_8](https://doi.org/10.1007/978-3-662-53018-4_8)
- [BV11a] Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) LWE. In: *FOCS*, pp. 97–106. IEEE Computer Society (2011)
- [BV11b] 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). [https://doi.org/10.1007/978-3-642-22792-9\\_29](https://doi.org/10.1007/978-3-642-22792-9_29)
- [CZW17] Chen, L., Zhang, Z., Wang, X.: Batched multi-hop multi-key FHE from Ring-LWE with compact ciphertext extension. In: Kalai, Y., Reyzin, L. (eds.) *TCC 2017*. LNCS, vol. 10678, pp. 597–627. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70503-3\\_20](https://doi.org/10.1007/978-3-319-70503-3_20)
- [CM15] Clear, M., McGoldrick, C.: Multi-identity and multi-key leveled FHE from learning with errors. In: Gennaro, R., Robshaw, M. (eds.) *CRYPTO 2015*. LNCS, vol. 9216, pp. 630–656. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-48000-7\\_31](https://doi.org/10.1007/978-3-662-48000-7_31)



- [DGHV10] van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully Homomorphic Encryption over the Integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-13190-5\\_2](https://doi.org/10.1007/978-3-642-13190-5_2)
- [DRS17] Derler, D., Ramacher, S., Slamanig, D.: Homomorphic proxy re-authenticators and applications to verifiable multi-user data aggregation. In: Kiayias, A. (ed.) FC 2017. LNCS, vol. 10322, pp. 124–142. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70972-7\\_7](https://doi.org/10.1007/978-3-319-70972-7_7)
- [Gen09] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178. ACM (2009)
- [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). [https://doi.org/10.1007/978-3-642-40041-4\\_5](https://doi.org/10.1007/978-3-642-40041-4_5)
- [ID03] Ivan, A.-A., Dodis, Y.: Proxy cryptography revisited. In: NDSS. The Internet Society (2003)
- [ILL89] Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-random generation from one-way functions (extended abstracts). In STOC, pp. 12–24. ACM (1989)
- [LTV12] López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: STOC, pp. 1219–1234. ACM (2012)
- [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). [https://doi.org/10.1007/978-3-540-78440-1\\_21](https://doi.org/10.1007/978-3-540-78440-1_21)
- [MLO16] Ma, C., Li, J., Ouyang, W.: A homomorphic proxy re-encryption from lattices. In: Chen, L., Han, J. (eds.) ProvSec 2016. LNCS, vol. 10005, pp. 353–372. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-47422-9\\_21](https://doi.org/10.1007/978-3-319-47422-9_21)
- [MW16] Mukherjee, P., Wichs, D.: Two round multiparty computation via multi-key FHE. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 735–763. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-49896-5\\_26](https://doi.org/10.1007/978-3-662-49896-5_26)
- [PS16] Peikert, C., Shiehian, S.: Multi-key FHE from LWE, revisited. In: Hirt, M., Smith, A. (eds.) TCC 2016-B, Part II. LNCS, vol. 9986, pp. 217–238. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53644-5\\_9](https://doi.org/10.1007/978-3-662-53644-5_9)
- [PRSV17] Polyakov, Y., Rohloff, K., Sahu, G., Vaikuntanathan, V.: Fast proxy re-encryption for publish/subscribe systems. *ACM Trans. Priv. Secur.* **20**(4), 14:1–14:31 (2017)
- [Reg05] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC, pp. 84–93. ACM (2005)



# Verifiable Decryption for Fully Homomorphic Encryption

Fucai Luo<sup>1,2,3(✉)</sup> and Kunpeng Wang<sup>1,2,3</sup>

<sup>1</sup> School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

<sup>2</sup> State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China  
luofucai@iie.ac.cn, kpwang@sina.cn

<sup>3</sup> Data Assurance and Communication Security Research Center, Chinese Academy of Sciences, Beijing, China

**Abstract.** Verifiable decryption allows one to prove the correct decryption of encrypted data. When the encrypted data is derived from homomorphic evaluations in the context of fully homomorphic encryption (FHE), verifiable decryption will be very useful in cloud computing or cryptographic protocols, e.g., secure medical computation, cryptographically verifiable election, etc. In this paper, we consider the problem of proving the correct decryption of an FHE ciphertext. Namely, we are interested in zero-knowledge proofs of knowledge of triples  $(m, s, c)$  such that the message  $m$  is the correct decryption of a ciphertext  $c$  for a secret key  $s$ . While analogous statements admit efficient zero-knowledge proof protocols in the discrete logarithm setting, they have never been addressed in FHE so far. We provide such verifiable decryption for Brakerski-Gentry-Vaikuntanathan (BGV) scheme, since this scheme was recognized as one of the most efficient leveled FHE schemes. Our solution is nearly “one shot”, in the sense that a single instance of the proof already has negligible soundness error, yielding compact proofs even for individual ciphertexts. Furthermore, to illustrate the applicability of verifiable decryption, we also give two example instantiations.

**Keywords:** Verifiable decryption · FHE · Zero-knowledge proofs  
Secure medical computation · Cryptographically verifiable election

## 1 Introduction

Fully Homomorphic Encryption (FHE) [19, 38], which is a very attractive cryptographic primitive that enables computations of any computable functions on encrypted data without knowing the secret key, is a powerful tool for handling many problems in cloud computing, such as private information retrieval, SQL query, outsourcing storage and computation, and secure multi-party computation (MPC). Since Gentry constructed the first FHE scheme and put forward a remarkable “bootstrapping” theorem to convert any non-fully homomorphic

encryption scheme into a full fledged one (that is, FHE) in his breakthrough work [18], a series of papers have been presented (see [1, 4, 6, 9, 14, 15, 20, 27, 39, 41]), with much progress mainly in efficiencies and well understood security assumptions.

Among the above FHE schemes, the optimized version [22] of the Ring Learning with Errors (RLWE)-based FHE scheme proposed by Brakerski, Gentry and Vaikuntanathan [4] (BGV) was universally accepted as one of the most efficient leveled FHE (capable of evaluating arbitrary polynomial-depth circuits) schemes, mainly due to the plaintexts of BGV that are polynomial ring elements. But one obvious drawback is that BGV needs user's "evaluation key". In Crypto 2013, Gentry, Sahai and Waters [20] (GSW) used two novel techniques of so-called *approximate eigenvector* and *flatten* to construct a matrix-style leveled FHE scheme with simpler and more directly homomorphic operations. Subsequently, Alperin-Sheriff and Peikert [1] (GSW variant) improved the GSW by using a "gadget matrix"  $\mathbf{G}$  developed by Micciancio and Peikert [35]. The GSW and GSW variant do not need user's "evaluation key" and have an interesting property of asymmetric noise growth, but they are only suitable for encrypting bits. This is due to the fact that the RLWE-based version of GSW (and GSW variant) needs to implement binary decomposition on polynomial ciphertexts when performing homomorphic multiplication, and this breaks polynomial character of the ciphertexts. Consequently, the *Fast Fourier Transform* (FFT) technique that can be used to improve polynomial operations is unavailable. Moreover, the noise may grow exponentially because of its asymmetric growth pattern.

In some cases, it is necessary for the secret key holder to provide zero-knowledge proofs of decryption on FHE ciphertexts. For example, as described in [8], a number of users with secret inputs want to compute some function  $f$  on those combined inputs, then they can use MPC if they are a small group and are online regularly; whereas in large group or in the asynchronous setting this will not work. But they can resort to FHE. More precisely, in the context of FHE, they can choose some semi-trusted party who has secret key and has published the corresponding public key. Then, based on FHE systems, all users use the public key to encrypt their secret inputs to get the corresponding ciphertexts and then post the ciphertexts. After that, the semi-trusted party evaluates the function  $f$  on these ciphertexts to derive a ciphertext (we call it derived ciphertext). Finally, the semi-trusted party decrypts the derived ciphertext and distributes the resulting value to the users. Meanwhile, without revealing any additional information beyond the resulting value, the semi-trusted party proves to the users that the resulting value is the plaintext of the derived ciphertext (i.e., the decryption was indeed done correctly). Note that the users can also get the derived ciphertext but cannot decrypt it. An obvious realisation of this scenario is cryptographic election, where the voters vote anonymously and the final voting results can be verified by anyone through a zero-knowledge proof protocol. We will give the definition of zero-knowledge proofs in Sect. 2.

From the perspective of zero-knowledge proofs, the above issue can be regarded as a problem of verifiable decryption. In a verifiable decryption scheme, a prover who has a secret key  $sk$  decrypts a ciphertext  $c = \text{Enc}(pk, x)$  and then

obtains a resulting value  $x$ . Then, he/she produces a zero-knowledge proof of knowledge  $\pi$  showing that the resulting value  $x$  is the plaintext of the ciphertext  $c$ , that is, the prover provides a zero-knowledge proof of decryption on the ciphertext  $c$ . Proving the correct decryption of the ciphertext  $c$  can be equivalent to proving the knowledge of the secret key  $sk$  along with proving that this resulting value  $x$  and the secret key  $sk$  satisfy the relation  $x = \text{Dec}(sk, c)$ . In this work, we consider an instance of BGV scheme, and construct a zero-knowledge proof for decryption of BGV ciphertexts.

### 1.1 Zero-Knowledge Proofs for Decryption of FHE Ciphertexts – Prior Work

Aiming at solving the problem of verifiable decryption, recently Carr et al. showed in [8] a zero-knowledge proof of correct decryption on Gentry’s original FHE ciphertexts. Carr et al. intended to use some Schnorr-like zero-knowledge proof protocol to solve this problem. But in this setting, applying the Schnorr-like zero-knowledge proof protocols to LWE-based FHE schemes will be very difficult. More specific reasons about this can be found in [8]. Based on this, they turned their attention to Gentry’s original FHE that is not based on LWE problem. In more detail, they observed that BGV is one of the most efficient leveled FHE schemes but they couldn’t construct a zero-knowledge proof of correct decryption on BGV ciphertexts; while Gentry’s original FHE is inefficient but it seems possible to build a zero-knowledge proof of correct decryption on its ciphertexts. Hence, they firstly presented a bootstrapping-like protocol to switch from BGV to Gentry’s original FHE, and then they constructed a simple “one shot” zero-knowledge proof protocol on Gentry’s original FHE ciphertexts. However, their zero-knowledge proof protocol is only able to prove the correct decryption of ciphertexts that are designed to encrypt message  $m = 0$ . This limitation will seriously affect the practicality of verifiable decryption. By contrast, in this work, we will provide a zero-knowledge proof protocol that can be used to prove the correct decryption of BGV ciphertexts that are designed to encrypt messages  $m \in R_2$ , where  $R$  is a polynomial ring.

### 1.2 Our Contributions

**Our Results.** We consider the problem of proving the correct decryption of BGV ciphertexts. We firstly review the BGV scheme, and then put forward an interactive zero-knowledge proof protocol to vouch for the correctness decryption of BGV ciphertexts. Finally, we show two example instantiations to illustrate the applicability of verifiable decryption. The problem considered in this paper can be described as follows: given a sequence of BGV ciphertexts  $\mathbf{c}_1, \dots, \mathbf{c}_N \in R_q^2$  and a function  $f$  (assume that the polynomial-depth of  $f$  is within maximum allowable level of the given BGV scheme), anyone (e.g. a verifier) can homomorphically perform the function  $f$  on these ciphertexts and then obtains a derived ciphertext  $\mathbf{c} = f(\mathbf{c}_1, \dots, \mathbf{c}_N) \in R_q^2$ . The secret key holder or we say the prover obtains a value  $m \in R_2$  (we call it “targeted message”) by decrypting

the derived ciphertext  $\mathbf{c}$ . The prover then sends the value  $m \in R_2$  to the verifier, and proves to the verifier that this value is the plaintext of the derived ciphertext  $\mathbf{c}$ , without revealing any additional information beyond the value. As previously mentioned, this is a verifiable decryption problem. In this work, we will transform this verifiable decryption problem into proving a linear relation of the form  $\mathbf{A}\mathbf{s} = \mathbf{0}$ .

We noted that an abstraction of Stern’s protocol [40] was suggested by Libert et al. [25] to address a similar setting when one has to prove a number of linear relations with a unique modulus, and Libert et al. [26] subsequently generalized the framework in [25], so as to handle correlated witnesses across relations modulo distinct integers. But the Stern’s protocol is unfortunately very impractical since each round of the protocol has soundness error  $2/3$ , and therefore it needs to be repeated 192 times to achieve 128-bit security. Alternatively, we found that a “one shot” verifiable encryption scheme proposed by Lyubashevsky and Neven [32] considered an analogous problem – verifiable encryption (we will introduce it in Related Work and Sect. 3). Their approach is “one shot” – that is, without repeating a protocol to amplify soundness, since a single instance of the proof already has negligible soundness error. Hence, this is a very efficient zero-knowledge proof protocol. Technically, the authors in [32] constructed the “one shot” verifiable encryption scheme by combining the framework of “Fiat-Shamir with Aborts” zero-knowledge proofs of linear relations (which we will present in Sect. 2) with a RLWE-based encryption scheme. Analogously, we will also combine the framework of “Fiat-Shamir with Aborts” zero-knowledge proofs of linear relations with BGV scheme, and then construct an interactive zero-knowledge proof protocol to solve the verifiable decryption problem effectively. Our proposed protocol only needs one extra communication between the prover and the verifier, and hence if we complete this communication off-line, then it is also “one shot”.

**Our Techniques.** The crucial ingredient of our solution is to transform the verifiable decryption problem into proving a linear relation. Concretely, given  $N$  BGV ciphertexts  $\mathbf{c}_1, \dots, \mathbf{c}_N \in R_q^2$  and a function  $f$ , both the prover and the verifier can homomorphically evaluate the function  $f$  on these ciphertexts and obtain a derived BGV ciphertext  $\mathbf{c} = f(\mathbf{c}_1, \dots, \mathbf{c}_N) \in R_q^2$ . We assume the derived ciphertext  $\mathbf{c}$  (decrypting to message  $m \in R_2$ ) is valid in the sense that, given the secret key  $\mathbf{s} = (\mathbf{s}', 1)$ , we have  $\langle \mathbf{s}, \mathbf{c} \rangle = m \bmod q \bmod 2$  (according to the correctness of the BGV scheme, skip to Sect. 3). After the prover decrypts the derived ciphertext  $\mathbf{c}$  and gets the “targeted message”  $m \in R_2$ , he/she sends the “targeted message” to the verifier and appends a valid proof that the decryption was indeed done correctly – that is, he/she has the secret key  $\mathbf{s}$  satisfying the following modular equation

$$\langle \mathbf{s}, \mathbf{c} - (0, m)^t \rangle = 0 \bmod q \bmod 2.$$

But even if the prover successfully proves this modular equation, the verifier is still not fully convinced that this vector  $\mathbf{s}$  is the secret key. This is mainly due to the fact that this derived ciphertext  $\mathbf{c}$  may not be computationally

indistinguishable from uniform in  $R_q^2$ , and hence we cannot prevent a dishonest prover from forging a valid proof. More details about this will be addressed in Sect. 3.

Based on this, we let the verifier provide a ciphertext  $\bar{\mathbf{c}}$  in advance that is designed to encrypt message  $\bar{m} = 0 \in R_2$  and is computationally indistinguishable from uniform in  $R_q^2$ . Then, the prover proves to the verifier that he/she has the secret key  $\mathbf{s}$  such that

$$\begin{bmatrix} \mathbf{c}'^t \\ \bar{\mathbf{c}}^t \end{bmatrix} \mathbf{s} = \mathbf{0} \pmod{q \pmod{2}},$$

where  $\mathbf{c}' \triangleq \mathbf{c} - (0, m)^t$ , without revealing any additional information beyond the “targeted message”  $m \in R_2$ . From the above, we successfully convert this verifiable decryption problem into proving a linear relation of the form  $\mathbf{A}\mathbf{s} = \mathbf{0} \pmod{q \pmod{2}}$ .

### 1.3 Related Work

**Verifiable Encryption and Proofs of Plaintext Knowledge.** As described in [32], verifiable encryption allows one to prove properties about encrypted data. Specifically, in a verifiable encryption scheme, there is a relation  $R_L$  and a language  $L = \{x : \exists w \text{ s.t. } R_L(x, w) = 1\}$ , where the value  $w$  is a witness to the fact that  $x$  is in the language  $L$ . The witness  $w$  possessed by the prover is private, while the relation  $R_L$  and the element  $x$  are public. The prover then produces an encryption  $t = \text{Enc}(w)$  as well as a zero-knowledge proof of knowledge  $\pi$  of the value  $w = \text{Dec}(t)$  and that  $w$  satisfies  $R_L(x, w) = 1$ . The proofs of plaintext knowledge can be seen as a verifiable encryption scheme without a relation, or where the relation is trivially satisfied. Roughly, in a proof of plaintext knowledge, a prover who has a message  $m$  generates a ciphertext  $t = \text{Enc}(m)$  and provides a zero-knowledge proof of knowledge  $\pi$  proving that he/she knows the value of  $\text{Dec}(t)$ .

On a high level, the verifiable decryption problem to be solved in this paper is very similar to the verifiable encryption problem described above. Thus, for better understanding the verifiable decryption and searching for solutions, it is necessary to study some work related to the verifiable encryption. Actually, in Crypto 2017, Baum et al. [2] proposed a new zero-knowledge proof protocol applicable to additively homomorphic functions that map integer vectors to an Abelian group. This protocol demonstrates knowledge of a short preimage and achieves amortised efficiency comparable to the approach of Cramer and Damgård [11], but gives a much tighter bound on what we can extract from a dishonest prover. Moreover, this protocol yields improved proofs of plaintext knowledge for (Ring-) LWE-based cryptosystems. This technique was subsequently refined in [12, 13].

Notably, in Eurocrypt 2017, Lyubashevsky and Neven [32] presented a construction of a verifiable encryption scheme, based on the hardness of the RLWE problem in the random-oracle model, for short solutions to linear equations over

polynomial rings. In [32], the authors combined the framework of “Fiat-Shamir with Aborts” zero-knowledge proofs of linear relations with the RLWE based encryption scheme in [34], and thus obtained a “one shot” verifiable encryption scheme. This verifiable encryption scheme is “one shot”, since a single instance of the proof already has negligible soundness error. However, it is well-known that verifiable encryption usually guarantees that the decryption can recover a witness for the original language, but the decryption of the “one shot” verifiable encryption scheme can only recover a witness of a related but extended language.

### 1.4 Organization

The rest of the paper is organized as follows. In Sect. 2, we summarize some notations throughout this paper, and we also introduce hardness assumptions including RLWE and RSIS problems, the definition of interactive zero-knowledge proofs and “Fiat-Shamir with Aborts” zero-knowledge proofs of linear relations. Section 3 simply recalls the BGV scheme and presents an interactive zero-knowledge proof protocol for decryption of BGV ciphertexts. In Sect. 4, we give two example instantiations for verifiable decryption. This paper ends with conclusion in Sect. 5.

## 2 Preliminaries

**Notations.** We say that a function is negligible, written  $\text{negl}(\lambda)$ , if  $\text{negl}(\lambda)$  is smaller than all polynomial fractions for sufficiently large  $\lambda$ . Let PPT denote probabilistic polynomial-time. In this work, we will use a ring of integer  $R = \mathbb{Z}[x]/(\Phi_m(x))$ , where  $\Phi_m(x)$  is a  $m$ -th cyclotomic polynomial which has degree  $n = \varphi(m)$ , the Euler’s totient of  $m$ . For ease of use, we let  $[n] \triangleq \{1, \dots, n\}$ . Moreover, for the positive integer  $q$ , we define the quotient ring  $R_q = R/qR = \mathbb{Z}_q[x]/(\Phi_m(x))$ , where all coefficients of polynomials in  $R_q$  are in  $(-q/2, q/2]$ . All definitions that follow apply both to  $R$  and  $R_q$ . We denote the elements of  $R_q$  by lowercase letters (e.g.  $a \in R_q$ ), elements of vectors in  $R_q^k$  by bold lowercase letters (e.g.  $\mathbf{a} \in R_q^k$ ), and elements of matrices in  $R_q^{k \times \ell}$  by bold uppercase letters (e.g.  $\mathbf{A} \in R_q^{k \times \ell}$ ). We denote by  $\mathbf{x}^t$  (resp.  $\mathbf{A}^T$ ) the transpose of the vector  $\mathbf{x}$  (resp. the matrix  $\mathbf{A}$ ), and  $\begin{bmatrix} \mathbf{a}^t \\ \mathbf{b}^t \end{bmatrix} \in R_q^{2 \times k}$  the row concatenation of column vectors  $\mathbf{a}, \mathbf{b} \in R_q^k$ .

For an element  $a = \sum_{i=0}^{n-1} \alpha_i x^i \in R$ , we define the  $\ell_1, \ell_2$  and  $\ell_\infty$  norms as follows:

$$\|a\|_1 = \sum_{i=0}^{n-1} |\alpha_i|, \|a\| = \sqrt{\sum_{i=0}^{n-1} \alpha_i^2} \text{ and } \|a\|_\infty = \max_i(|\alpha_i|)$$

respectively. Then for  $k$ -dimensional vectors  $\mathbf{a} = (a_1, \dots, a_k) \in R^k$ , we have

$$\|\mathbf{a}\|_1 = \sum_{i=1}^k \|a_i\|_1, \|\mathbf{a}\| = \sqrt{\sum_{i=1}^k \|a_i\|^2} \text{ and } \|\mathbf{a}\|_\infty = \max_i(\|a_i\|_\infty).$$

**Distribution.** We let  $x \stackrel{\$}{\leftarrow} \mathcal{D}$  denote that  $x$  is sampled uniformly at random from a distribution  $\mathcal{D}$ , and  $x \stackrel{\$}{\leftarrow} \mathcal{S}$  (e.g.  $R_q$ ) denote that  $x$  is uniform over a set  $\mathcal{S}$ . Moreover, we let  $\Delta(X, Y)$  denote the statistical distance between two distributions  $X, Y$ .

**Discrete Gaussians.** Let  $\mathcal{L}$  be a subset of  $\mathbb{Z}^m$ , for any  $\sigma > 0$ , define the Gaussian function on  $\mathbb{R}^m$  centered at  $\mathbf{c} \in \mathbb{R}^m$  with parameter  $\sigma$ :

$$\forall \mathbf{x} \in \mathbb{R}^m, \rho_{\sigma, \mathbf{c}}(\mathbf{x}) = \exp(-\pi \|\mathbf{x} - \mathbf{c}\|^2 / \sigma^2).$$

Then the discrete Gaussian distribution over the set  $\mathcal{L}$  with center  $\mathbf{c} \in \mathbb{R}^m$  and parameter  $\sigma$  is defined as:

$$\forall \mathbf{x} \in \mathcal{L}, \mathcal{D}_{\sigma, \mathbf{c}, \mathcal{L}}(\mathbf{x}) = \frac{\rho_{\sigma, \mathbf{c}}(\mathbf{x})}{\rho_{\sigma, \mathbf{c}}(\mathcal{L})}$$

where  $\rho_{\sigma, \mathbf{c}}(\mathcal{L}) = \sum_{\mathbf{x} \in \mathcal{L}} \rho_{\sigma, \mathbf{c}}(\mathbf{x})$ . As noted in [32], for the special case of  $\mathcal{L} = \mathbb{Z}^m$ ,

it holds that  $\Pr[\|\mathbf{x}\|_{\infty} > t\sigma : \mathbf{x} \stackrel{\$}{\leftarrow} \mathcal{D}_{\sigma, \mathbf{0}, \mathbb{Z}^m}] < 2me^{-t^2/2}$ . This implies that for  $t = 6$ , the probability that any coefficient of  $\mathbf{x}$  is greater than  $6\sigma$  is less than  $2^{-25}m$ .

## 2.1 Hardness Assumptions

**Ring Learning with Errors (RLWE) Problem.** The Ring Learning with Errors (RLWE) problem was firstly introduced by Lyubashevsky, Peikert and Regev [33], who also gave a reduction proving that the RLWE problem is not easier than the well-established worst-case GapSVP problem on ideal lattices. Here we consider a simplified version of RLWE problem that is easier to work with [4, 6].

**Definition 1** ([4]). *For security parameter  $\lambda$ , let  $q = q(\lambda) \geq 2$  be an integer. Let  $R = \mathbb{Z}[x]/(\Phi_m(x))$ ,  $R_q = R/qR$  be polynomial rings and  $\chi = \chi(\lambda)$  be a distribution over  $R$ , where  $\Phi_m(x)$  is a  $m$ -th cyclotomic polynomial which has degree  $n = \varphi(m)$ , the Euler's totient of  $m$ . The  $RLWE_{n, q, \chi}$  problem is to distinguish the following two distributions: In the first distribution, one first draws  $s \stackrel{\$}{\leftarrow} R_q$  uniformly and then samples  $(a_i, b_i) \in R_q^2$  by sampling  $a_i \stackrel{\$}{\leftarrow} R_q$ ,  $e_i \stackrel{\$}{\leftarrow} \chi$ , and setting  $b_i = a_i \cdot s + e_i$ . In the second distribution, one samples  $(a_i, b_i)$  uniformly from  $R_q^2$ . The  $RLWE_{n, q, \chi}$  assumption is that the  $RLWE_{n, q, \chi}$  problem is infeasible.*

Typically, in a RLWE cryptosystem, one chooses the noise distribution  $\chi$  according to a Gaussian distribution. As remarked in [33], the Gaussian distribution may need to be “ellipsoidal” for certain reductions to go through. Moreover, it has been shown for RLWE problem that one can equivalently assume that the secret ring element  $s$  is alternatively sampled from the noise distribution  $\chi$  [33].

**The Ring-SIS Problem.** Let  $q, k$  be positive integers and  $\beta$  be real, and let polynomial ring  $R_q = \mathbb{Z}_q[x]/(\Phi_m(x))$  of degree  $n = \varphi(m)$ . Then the  $RSIS_{q, k, \beta}$  problem is defined as follows.



**Definition 2** ([30]). *Given  $k$  ring elements  $a_1, \dots, a_k$  chosen uniformly at random from  $R_q$ , find  $k$  non-zero ring elements  $z_1, \dots, z_k \in R_q$  such that  $\|z_i\| \leq \beta$  and*

$$\sum_{i=1}^k a_i z_i = 0 \pmod{q}.$$

The  $\text{RSIS}_{q,k,\beta}$  problem is known to be as hard as certain worst-case problems (e.g. GapSVP) on ideal lattices [30,31]. In this work, our solution is based on a variant of Ring-SIS problem that we called  $\text{RSIS}_{q,2,k,\beta}$  problem, since it involves two moduli, that is,

$$\sum_{i=1}^k a_i z_i = 0 \pmod{q \pmod{2}},$$

where the modulus  $q$  is an odd number.

## 2.2 Interactive Proof Systems

In this paper, we focus on the interactive proof system with zero knowledge. Informally, in an interactive zero-knowledge proof, by interacting with a sceptical verifier, an honest prover can convince the sceptical verifier that some claim is true (and in some cases that he/she knows a proof) while revealing no other knowledge than the fact that the claim is true; whereas it’s nearly impossible for a cheating prover (e.g. an adversary) to convince the sceptical verifier that some claim is true in any case.

In an interactive proof system, let  $P$  and  $V$  denote a computational unbounded prover and a polynomial time verifier respectively, and  $P^*$  denote a computational bounded cheating prover. Then, for a promise problem  $\Pi = (\Pi^Y, \Pi^N)$ , where  $\Pi^Y$  and  $\Pi^N$  are disjoint sets, representing YES instance and NO instance respectively, we can define an interactive proof system as follows.

**Definition 3** ([36]). *( $P; V$ ) is an Interactive Proof System (IPS) for a promise problem  $\Pi = (\Pi^Y, \Pi^N)$ , and security parameter  $\lambda$  if*

- **Completeness.** *For every  $x \in \Pi^Y$ ,  $\Pr[(P; V)(x, r) \text{ accepts}] \geq 1 - c(\lambda)$ .*
- **Soundness.** *For every  $x \in \Pi^N$ ,  $\Pr[(P^*; V)(x, r) \text{ accepts}] \leq s(\lambda)$ .*

*The probabilities are taken over the choice of the random input  $r$  and the random choices of  $P$ . The random input  $r$  is generally chosen uniformly at random from  $\{0, 1\}^{p(\lambda)}$  for some fixed polynomial  $p(\lambda)$ . The function  $c(\lambda)$  denotes the completeness error, and the function  $s(\lambda)$  denotes the soundness error. For non-triviality, we require  $c(\lambda) + s(\lambda) \leq 1 - 1/\text{poly}(\lambda)$ .*

By standard techniques, completeness and soundness errors can be reduced via parallel repetition. We remark that our definition of soundness is non-adaptive, that is, the NO instance is fixed in advance of the random input  $r$ . Some applications may require adaptive soundness, in which there do not exist any instance  $x \in \Pi^N$  and valid proof  $\pi$ , except with negligible probability over the choice of  $r$ . For proof systems, a simple argument shows that

non-adaptive soundness implies adaptive soundness error  $2^{-p(\lambda)}$  for any desired  $p(\lambda) = \text{poly}(\lambda)$ : let  $B(\lambda) = \text{poly}(\lambda)$  be a bound on the length of any instance in  $\Pi^N$ , and compose the proof system in parallel some  $\text{poly}(\lambda)$  times to achieve (non-adaptive) soundness  $2^{-p(\lambda)-B(\lambda)}$ . Then by a union bound over all  $x \in \Pi^N$ , the resulting proof system has adaptive soundness  $2^{-p(\lambda)}$ . We feel that the subtleties involved in the various definitions of soundness will only be of interest to the zero-knowledge experts and we do not go further on this topic.

An interactive proof system also needs to satisfy an additional property of zero-knowledge. Informally, an interactive proof system  $(P; V)$  is zero-knowledge if there exists a PPT algorithm  $\text{Sim}$ , called the simulator  $S$ , such that for any  $x \in \Pi^Y$ , the output of  $(P; V)(x, r)$  and the output of  $\text{Sim}(x)$  are “indistinguishable”. In this paper, we restrict our attention to statistical proof systems, where the corresponding notion is that of statistical indistinguishability.

**Definition 4 (Simulatability).** *An interactive proof system  $(P; V)$  for a promise problem  $\Pi = (\Pi^Y, \Pi^N)$  is statistical zero knowledge (SZK) if there exists a PPT algorithm  $\text{Sim}$  (called a simulator) such that for all  $x \in \Pi^Y$ , the statistical distance between  $\text{Sim}(x)$  and  $(P; V)(x, r)$  is negligible in  $\lambda$ :*

$$\Delta(\text{Sim}(x), (P; V)(x, r)) \leq \text{negl}(\lambda).$$

*That is,  $\text{Sim}$  only gets the public input  $x$  and has no interaction with  $P$ , but still manages to output something indistinguishable from whatever  $V$  learned in the interaction.*

### 2.3 “Fiat-Shamir with Aborts” Zero-Knowledge Proofs of Linear Relations

Lyubashevsky [28, 29] introduced a technique for constructing practical digital signatures based on the hardness of lattice problems. In [28, 29], the author mainly considered a zero-knowledge proof of knowledge that, given a secret vector  $\mathbf{s} \in R^k$  satisfying the relation  $\mathbf{A}\mathbf{s} = \mathbf{t} \bmod q$ , proves the knowledge of low-norm  $\bar{\mathbf{s}}$  and  $\bar{\mathbf{c}}$  that satisfy  $\mathbf{A}\bar{\mathbf{s}} = \bar{\mathbf{c}}\mathbf{t} \bmod q$ . The main idea in [28, 29] is to construct a  $\Sigma$ -protocol with the main twist being that the prover does not always output the result. In particular, the protocols use rejection sampling to tailor the distribution so that it does not depend on the secret vector  $\mathbf{s}$ . Moreover, this rejection sampling was achieved by making the resulting distribution uniform in a box by [28], or was achieved by a more efficient approach that making the resulting distribution a discrete Gaussian by [29]. The interactive protocol is then converted into a non-interactive one in the random-oracle model [3] using the Fiat-Shamir technique [16]. This combined technique is sometimes referred to as “Fiat-Shamir with Abort”. The focus of the work [28, 29] was on signature schemes, while Lyubashevsky and Neven [32] formalized the approach as a non-interactive zero-knowledge (NIZK) proof system for “relaxed” relations, i.e., where extraction yields a witness from a different language than is used to produce the proof. Analogously, in this paper we formalize the approach as a non-interactive zero-knowledge (NIZK) proof system for linear relations, and

then use this NIZK proof system to construct an interactive zero-knowledge proof protocol for decryption of FHE ciphertexts.

---

**Algorithm 1.** “Fiat-Shamir with Aborts” zero-knowledge proof of knowledge

---

**Input:** A matrix  $\mathbf{A} \in R_q^{k \times \ell}$ , a vector  $\mathbf{s} \in \mathcal{S} \subset R_q^\ell$  and a vector  $\mathbf{t} \in R_q^k$  such that  $\mathbf{As} = \mathbf{t} \pmod{q}$  mod 2. Challenge domain  $\mathcal{C} \subset R_q$ . Cryptographic hash function  $H : \{0, 1\}^* \rightarrow \mathcal{C}$  and a standard deviation  $\sigma \geq 11 \cdot \max_{c \in \mathcal{C}, \mathbf{s} \in \mathcal{S}} \|\mathbf{cs}\|$ .

**Output:**  $\pi = (\mathbf{z}, c)$ , such that  $\mathbf{z} \in \mathcal{D}_{\sigma, 0, R_q^\ell}$ ,  $c \in \mathcal{C}$  and  $c = H(\mathbf{A}, \mathbf{t}, \mathbf{Az} - \mathbf{ct} \pmod{q} \pmod{2})$ .

1.  $\mathbf{y} \stackrel{\$}{\leftarrow} \mathcal{D}_{\sigma, 0, R_q^\ell}$
  2.  $c \leftarrow H(\mathbf{A}, \mathbf{t}, \mathbf{Ay} \pmod{q} \pmod{2})$
  3.  $\mathbf{z} \leftarrow c \cdot \mathbf{s} + \mathbf{y} \pmod{q}$
  4. with probability  $\frac{\mathcal{D}_{\sigma, 0, R_q^\ell}(\mathbf{z})}{3 \cdot \mathcal{D}_{\sigma, \mathbf{cs}, R_q^\ell}(\mathbf{z})}$ , continue, else goto 1
  5. if  $\|\mathbf{z}\|_\infty > 6\sigma$ , goto 1
  6. return  $\pi = (c, \mathbf{z})$
- 

---

**Algorithm 2.** “Fiat-Shamir with Aborts” verification algorithm

---

**Input:** A matrix  $\mathbf{A} \in R_q^{k \times \ell}$ , a vector  $\mathbf{t} \in R_q^k$  and a tuple  $\pi = (c, \mathbf{z}) \in \mathcal{C} \times R_q^\ell$ . Cryptographic hash function  $H : \{0, 1\}^* \rightarrow \mathcal{C}$  and a positive real  $\sigma$ .

**Output:** Bit 0 or 1 corresponding to Reject/Accept.

1. if  $\|\mathbf{z}\|_\infty > 6\sigma$ , return 0
  2. if  $c \neq H(\mathbf{A}, \mathbf{t}, \mathbf{Az} - \mathbf{ct} \pmod{q} \pmod{2})$ , return 0
  3. return 1
- 

Algorithm 1 (we borrowed it from [32]) is a variation of the signing protocol from [29]. It shows that the output  $\mathbf{z}$  is distributed according to  $\mathcal{D}_{\sigma, 0, R_q^\ell}$ . In particular, the rejection sampling stem on line 4 has the effect that the distribution of  $\mathbf{z}$  is independent of the secret vector  $\mathbf{s}$  and the challenge  $c$ . This algorithm is therefore honest-verifier zero knowledge since a simulator can simply output  $\mathbf{z} \stackrel{\$}{\leftarrow} \mathcal{D}_{\sigma, 0, R_q^\ell}, c \stackrel{\$}{\leftarrow} \mathcal{C}$  and program  $c = H(\mathbf{A}, \mathbf{t}, \mathbf{Az} - \mathbf{ct} \pmod{q} \pmod{2})$ . We remark that the simulation soundness of the algorithm is essentially the same as that of [32], so we refer interested readers to [32] for more details about it. Note that in [29, 32], one modular equation only contains one modulus, but the modular equations in our Algorithms 1 and 2 involves two moduli. As remarked in [32], one does not need to use the same modulus  $q$  for every row of the modular equation, since the modulus does not affect the “Fiat-Shamir with Aborts” zero-knowledge proofs of knowledge, so do our moduli in the above algorithms.

### 3 The Verifiable Decryption for BGV Scheme

In this section, for ease of exposition, we firstly recall the variant of BGV scheme. Then, based on the framework of non-interactive “Fiat-Shamir with Aborts” zero-knowledge proofs of linear relations presented in Sect. 2, we construct an interactive zero-knowledge proof protocol for decryption of BGV ciphertexts. Compared with non-interactive procedure, our interactive protocol only needs one extra communication between the prover and the verifier. Actually, if we complete this communication off-line, then our protocol can also be regarded as non-interactive.

#### 3.1 The BGV Scheme

We simply review the variant of BGV scheme, which was obtained by combing the original BGV scheme [4] with a variant of RLWE-based cryptosystem. This RLWE-based cryptosystem variant was first described in the full version of [33]. Since in this work we mainly focus on the zero-knowledge proofs of decryption, the *key switching* and *modulus switching* techniques involved in BGV’s homomorphic operations will not be presented.

- **BGV.Setup**( $1^\lambda$ ). Choose an odd modulus  $q = q(\lambda)$  and let  $R_q = \mathbb{Z}_q[x]/(\Phi_m(x))$  be polynomial ring, where  $\Phi_m(x)$  is a  $m$ -th cyclotomic polynomial of degree  $n = \varphi(m)$ . Let Gaussian noise distribution  $\chi = \chi(\lambda)$  be defined over  $R_q$  with a small norm bounded by a bound  $B$ . Set  $params = (q, n, \chi)$ .
- **BGV.KeyGen**( $params$ ). Choose uniformly at random three ring elements  $a \xleftarrow{\$} R_q$ , and  $e, s' \xleftarrow{\$} \chi$ . Generate  $b \leftarrow -(as' + 2e) \bmod q$ . Set the secret key  $sk = \mathbf{s} = (s', 1) \in R_q^2$  and the public key  $pk = (a, b) \in R_q^2$ .
- **BGV.Enc**( $pk, m$ ). To encrypt a plaintext polynomial  $m \in R_2$ , sample  $u, r_0, r_1 \xleftarrow{\$} \chi$  and then compute the ciphertext

$$\mathbf{c} = (c_0, c_1) = (au + 2r_0, bu + 2r_1 + m) \in R_q^2.$$

- **BGV.Dec**( $sk, \mathbf{c}$ ). For the ciphertext  $\mathbf{c} \in R_q^2$ , recover  $m$  as follows:

$$m = \langle \mathbf{s}, \mathbf{c} \rangle \bmod q \bmod 2.$$

**Correctness and Security.** The ciphertext is valid as long as the noise  $e' \triangleq 2s'r_0 - 2eu + 2r_1$  does not wrap around modulo  $q$ . We prove this as follows:

$$\begin{aligned} \langle \mathbf{s}, \mathbf{c} \rangle &= s'c_0 + c_1 = as'u + 2s'r_0 + bu + 2r_1 + m \\ &= as'u + 2s'r_0 + -(as'u + 2eu) + 2r_1 + m \\ &= 2s'r_0 - 2eu + 2r_1 + m = m \bmod q \bmod 2. \end{aligned}$$

As for the security, under the  $RLWE_{n,q,\chi}$  assumption, the BGV scheme is IND-CPA secure. This can be proved using a standard hybrid argument similar to [5]. Roughly, under the  $RLWE_{n,q,\chi}$  assumption, it is easy to prove that the public key  $pk = (a, b) \in R_q^2$  is computationally indistinguishable from uniform in  $R_q^2$ , and then by a standard hybrid argument we can draw the conclusion that the ciphertext  $\mathbf{c} = (c_0, c_1) \in R_q^2$  is also computationally indistinguishable from uniform in  $R_q^2$ .

### 3.2 Zero-Knowledge Proofs for Decryption of BGV Ciphertexts

As described in the introduction, Lyubashevsky and Neven [32] defined verifiable encryption and presented a construction of verifiable encryption scheme based on RLWE problem in the random-oracle model. Their definition of verifiable encryption is borrowed from Camenisch and Shoup [7], but the soundness defined in [7] requires that the decryption of a valid ciphertext always recovers a valid witness; whereas the construction in [32] can only achieve a relaxed property that recovers a witness for a related “extended” language that includes the original language. This is mainly due to the fact that given the linear relation  $\mathbf{B}\mathbf{m} = \mathbf{u} \bmod p$ , their construction only yields a witness  $(\bar{\mathbf{m}}, \bar{c})$  with small coefficients satisfying  $\mathbf{B}\bar{\mathbf{m}} = \bar{c}\mathbf{u} \bmod p$ . Nevertheless, in order to prove that this weaker property suffices for many practical applications of verifiable encryption, Lyubashevsky and Neven gave two concrete examples – key escrow for RLWE encryption and verifiably encrypted signatures. The core idea in [32] is to combine the framework of “Fiat-Shamir with Aborts” zero-knowledge proofs of linear relations with a RLWE based encryption scheme, and the challenge is to convert the modular equation involved in the encryption of the RLWE based encryption scheme into the linear relations that are compatible with “Fiat-Shamir with Aborts” zero-knowledge proofs of linear relations. This challenge also exists in verifiable decryption problem considered in this paper.

As for the verifiable decryption for BGV scheme, at first glance, it can be transformed into the problem that: given a sequence of ciphertexts  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_N \in R_q^2$  and a function  $f$ , the verifier and the prover both perform the function  $f$  on these ciphertexts and then obtain a derived ciphertext  $\mathbf{c} = f(\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_N) \in R_q^2$ . Then, the prover sends a “targeted message”  $m \in R_2$  to the verifier. Meanwhile, for the “targeted message”  $m \in R_2$  and the publicly computable derived ciphertext  $\mathbf{c}$ , the prover argues in zero-knowledge the possession of the secret key  $\mathbf{s} \in R_q^2$  such that  $\langle \mathbf{s}, \mathbf{c} - (0, m)^t \rangle = 0 \bmod q \bmod 2$  (by the correctness of the BGV scheme). However, we cannot use  $\text{RSIS}_{q,2,2,\beta}$  hard problem to prevent a dishonest prover from deceiving the verifier. More precisely, the derived ciphertext  $\mathbf{c} \in R_q^2$  may not be computationally indistinguishable from uniform in  $R_q^2$ , though any vector  $\mathbf{c}_i \in R_q^2$  for  $i \in [N]$  is computationally indistinguishable from uniform in  $R_q^2$  (due to the IND-CPA security of BGV scheme). This makes the modular equation  $\langle \mathbf{s}, \mathbf{c} - (0, m)^t \rangle = 0 \bmod q \bmod 2$  unable to connect to the  $\text{RSIS}_{q,2,2,\beta}$  problem (go back to The Ring-SIS Problem in Sect. 2). In other words, the implicit difficulty in the modular equation  $\langle \mathbf{s}, \mathbf{c} - (0, m)^t \rangle = 0 \bmod q \bmod 2$  cannot be guaranteed by the  $\text{RSIS}_{q,2,2,\beta}$  problem. In short, assume  $m$  is the plaintext of the ciphertext  $\mathbf{c}$ , for the message  $\neg m \in R_2$  claimed by the prover but  $m \neq \neg m$ , we cannot reduce the probability of the event that, the prover successfully proves to the verifier that he/she has a vector  $\neg \mathbf{s} \in R_q^2$  such that  $\langle \neg \mathbf{s}, \mathbf{c} - (0, \neg m)^t \rangle = 0 \bmod q \bmod 2$ , to be negligible. Note that for  $m \neq \neg m$ , it does not hold that  $\langle \mathbf{s}, \mathbf{c} - (0, \neg m)^t \rangle = 0 \bmod q \bmod 2$  for the secret key  $\mathbf{s} \in R_q^2$ . Therefore, whether the prover has the secret key or not, he/she may still deceive the verifier.

To overcome the challenges mentioned above, we insert the  $\text{RSIS}_{q,2,2,\beta}$  problem into the verifiable decryption by providing a ciphertext  $\bar{\mathbf{c}} \in R_q^2$  and requiring the prover to be able to prove that  $\langle \bar{\mathbf{s}}, \bar{\mathbf{c}} \rangle = 0 \pmod{q \pmod{2}}$ , where the ciphertext  $\bar{\mathbf{c}} \in R_q^2$  decrypting to message  $0 \in R_2$  is computationally indistinguishable from uniform in  $R_q^2$ . The prover cannot find a vector  $\bar{\mathbf{s}} \in R_q^2$  such that  $\langle \bar{\mathbf{s}}, \bar{\mathbf{c}} \rangle = 0 \pmod{q \pmod{2}}$ , if he/she does not have the secret key of the given BGV scheme. In other words, the vector  $\bar{\mathbf{s}} \in R_q^2$  satisfying  $\langle \bar{\mathbf{s}}, \bar{\mathbf{c}} \rangle = 0 \pmod{q \pmod{2}}$  cannot be anything but the secret key. Otherwise, the prover can solve the  $\text{RSIS}_{q,2,2,\beta}$  problem. Next, we will construct an interactive zero-knowledge proof protocol for decryption of BGV ciphertexts. We reiterate that the  $N$  ciphertexts  $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_N \in R_q^2$  and the function  $f$  are publicly known, and then both the prover and the verifier can compute the derived ciphertext  $\mathbf{c} = f(\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_N) \in R_q^2$ .

The interactive zero-knowledge proof protocol, between a prover  $\mathcal{P}$  and a verifier  $\mathcal{V}$ , is constructed as follows.

1. After receiving the “targeted message”  $m \in R_2$ , the verifier  $\mathcal{V}$  encrypts a message  $\bar{m} = 0 \in R_2$  using the given BGV scheme and obtains the corresponding ciphertext  $\bar{\mathbf{c}} \in R_q^2$ . After that,  $\mathcal{V}$  sends the ciphertext  $\bar{\mathbf{c}}$  to the prover  $\mathcal{P}$ . (This is an extra communication compared with non-interactive procedure.)
2. The prover  $\mathcal{P}$  runs Algorithm 3 and gets a triple  $\pi = (m, c, \mathbf{z})$ . Then, the prover  $\mathcal{P}$  sends the triple  $\pi$  to the verifier  $\mathcal{V}$ .
3. The verifier  $\mathcal{V}$  runs Algorithm 4. Output 1 if the Algorithm 4 outputs 1. Otherwise, output 0.

---

**Algorithm 3.** One-shot verifiable decryption  $\mathbf{Gen}(sk, \mathbf{c}, \bar{\mathbf{c}})$

---

**Input:** Two vectors  $\mathbf{c}, \bar{\mathbf{c}} \in R_q^2$  and a witness  $sk = \mathbf{s} = (s', 1) \in R_q^2$ , such

that  $\begin{bmatrix} \mathbf{c}'^t \\ \bar{\mathbf{c}}^t \end{bmatrix} \mathbf{s} = \mathbf{0} \pmod{q \pmod{2}}$ , where  $\mathbf{c}' \triangleq \mathbf{c} - (0, m)^t$  and

$m = \langle \mathbf{s}, \mathbf{c} \rangle \pmod{q \pmod{2}}$ . Challenge domain

$\mathcal{C} = \{c \in R : \|c\|_\infty = 1, \|c\|_1 \leq 36\}$ . Cryptographic hash function

$H : \{0, 1\}^* \rightarrow \mathcal{C}$  and a standard deviation

$\sigma = 11 \cdot \max_{c \in \mathcal{C}} \|c\|_1 \cdot \sqrt{2n(3 + \gamma)}$ . Values of  $\gamma$  less than 1.005 are

believed to achieve at least 128-bits security [17].

**Output:**  $\pi = (m, c, \mathbf{z})$ , where  $\mathbf{z} \in R_q^2$ .

1.  $m \leftarrow \langle \mathbf{s}, \mathbf{c} \rangle \pmod{q \pmod{2}}$  and  $\mathbf{c}' \leftarrow \mathbf{c} - (0, m)^t$

2.  $\mathbf{y} \xleftarrow{\$} \mathcal{D}_{\sigma, 0, R_q^2}$

3.  $c \leftarrow H\left(\begin{bmatrix} \mathbf{c}'^t \\ \bar{\mathbf{c}}^t \end{bmatrix}, \mathbf{0}, \begin{bmatrix} \mathbf{c}'^t \\ \bar{\mathbf{c}}^t \end{bmatrix} \mathbf{y} \pmod{q \pmod{2}}\right)$

4.  $\mathbf{z} \leftarrow c \cdot \mathbf{s} + \mathbf{y} \pmod{q}$

5. with probability  $\frac{\mathcal{D}_{\sigma, 0, R_q^2}(\mathbf{z})}{3 \cdot \mathcal{D}_{\sigma, cs, R_q^2}(\mathbf{z})}$ , continue, else goto 2

6. if  $\|\mathbf{z}\|_\infty > 6\sigma$ , goto 2

7. return  $\pi = (m, c, \mathbf{z})$

---

---

**Algorithm 4.** One-shot verification  $\mathbf{V}(\mathbf{c}, \bar{\mathbf{c}}, \pi)$

---

**Input:** Two vectors  $\mathbf{c}, \bar{\mathbf{c}} \in R_q^2$ , a triple  $\pi = (m, c, \mathbf{z})$ , a cryptographic hash function  $H$  and a positive real  $\sigma$  as in Algorithm 3.

**Output:** Bit 0 or 1 corresponding to Reject/Accept.

1.  $\mathbf{c}' \leftarrow \mathbf{c} - (0, m)^t$
  2. if  $\|\mathbf{z}\|_\infty > 6\sigma$ , return 0
  3. if  $c \neq H\left(\begin{bmatrix} \mathbf{c}'^t \\ \bar{\mathbf{c}}^t \end{bmatrix}, \mathbf{0}, \begin{bmatrix} \mathbf{c}'^t \\ \bar{\mathbf{c}}^t \end{bmatrix} \mathbf{z} \bmod q \bmod 2\right)$ , return 0
  4. return 1
- 

### 3.3 Correctness and Security

**Completeness.** Completeness follows from the completeness of the proof system of Sect. 2.3.

**Soundness.** Given the derived ciphertext  $\mathbf{c}$  decrypting to message  $m \in R_2$  (assume that the derived ciphertext  $\mathbf{c}$  is valid under the given BGV scheme) and  $\bar{\mathbf{c}}$  that is computationally indistinguishable from uniform in  $R_q^2$ , if the prover claims that the message  $\neg m \in R_2$  is the plaintext of the derived ciphertext  $\mathbf{c}$ , but  $m \neq \neg m$ , and argues in zero-knowledge the possession of a vector  $\mathbf{s} \in R_q^2$  such that

$$\begin{bmatrix} \neg \mathbf{c}^t \\ \bar{\mathbf{c}}^t \end{bmatrix} \mathbf{s} = \mathbf{0} \bmod q \bmod 2,$$

where  $\neg \mathbf{c} \triangleq \mathbf{c} - (0, \neg m)^t$ , then we have  $\langle \mathbf{s}, \bar{\mathbf{c}} \rangle = 0 \bmod q \bmod 2$ , and hence the vector  $\mathbf{s}$  must be the secret key. Otherwise, the prover solves the  $\text{RSIS}_{q,2,2,\beta}$  problem related to  $\bar{\mathbf{c}}$ . Since the vector  $\mathbf{s}$  is the secret key, and the derived ciphertext  $\mathbf{c}$  is a valid ciphertext, it follows that  $m = \langle \mathbf{s}, \mathbf{c} \rangle \bmod q \bmod 2$ . Rearranging, we obtain

$$\langle \mathbf{s}, \mathbf{c} - (0, m)^t \rangle = 0 \bmod q \bmod 2.$$

So that we have

$$\langle \mathbf{s}, \mathbf{c} - (0, \neg m)^t \rangle \neq 0 \bmod q \bmod 2,$$

which proves the soundness of the interactive zero-knowledge proof protocol.

**Simulatability.** The simulator  $\text{Sim}$  creates a ciphertext  $\bar{\mathbf{c}}$  of message  $\tilde{m} = 0$  using the given BGV scheme and runs the zero-knowledge simulator for the proof system of Sect. 2.3 to create a valid-looking proof  $\tilde{\pi} = (\tilde{m}, \bar{\mathbf{c}}, \tilde{\mathbf{z}})$  for  $(\mathbf{c}, \bar{\mathbf{c}})$ . This proof is indistinguishable from the real proof, due to the zero-knowledge property of the proof system and the IND-CPA security of the given BGV scheme.

The above properties are summarized in the following theorem.

**Theorem 1.** *Assuming the hardness of  $\text{RSIS}_{q,2,2,\beta}$  problem, the interactive zero-knowledge proof protocol is a zero-knowledge proof of knowledge for the given statement, with perfect completeness, negligible soundness error, and communication cost about  $2n \log q$ . In particular, there is an efficient simulator that, on*

*input  $(\mathbf{c}, \tilde{\mathbf{c}})$ , outputs a valid-looking proof  $\tilde{\pi} = (\tilde{m}, \tilde{c}, \tilde{\mathbf{z}})$  statistically close to that generated by the real prover.*

## 4 Applications

### 4.1 Secure Medical Computation

Sharing the medical records of individuals among health care providers and researchers around the world can accelerate advances in medical research. While the idea seems increasingly practical due to cloud data services, maintaining patient privacy is of crucial importance. We can use some standard encryption schemes to encrypt sensitive data so as to protect them from outside attackers, but these encryption algorithms cannot be used to compute on this sensitive data while being encrypted. Whether this private-preserving sensitive data can withstand arbitrary computations is very important, since the computational results are very useful to help researchers and clinicians to conduct research or diagnose. Fortunately, FHE can solve this issue since it presents a very useful tool that can compute on encrypted data without the need to decrypt it. As described in [24], in a medical data/key-distribution system, a central medical authority will be responsible for generating the secret and public keys. The public keys will then be distributed, using the wide-area network, among medical laboratories and downloaded to portable and wearable medical devices. The medical data generated by the medical personnel in laboratories and by patients using their devices will then be encrypted and uploaded to the cloud. All patient medical data can be stored on the cloud servers safely as the FHE scheme is IND-CPA secure. Administrators, researchers, analysts, or clinicians can run experiments on the encrypted medical data without having any secret keys as the FHE scheme supports arbitrary computations on encrypted data. In order to finally decrypt the encrypted experiment results, they will need to gain access to the secret keys from the key authority system using a secure channel.

However, the above procedures that clinicians (or analysts, administrators and researchers) have to access to the secret keys from the key authority system to decrypt the encrypted experiment results will incur many problems. For example, if the clinicians who have gained access to the secret keys before are corrupted, then the key authority has to remove the public keys and the secret keys and re-generate them. Moreover, the encrypted medical data that already existed in the system must be re-encrypted. Fortunately, we can use verifiable decryption for FHE to avoid these problems. In details, we can let the clinicians send the encrypted experiment results (derived ciphertexts) to the key authority. Then, the key authority performs the decryption procedure on the derived ciphertexts to get the corresponding decryption results. Finally, the key authority sends the decryption results to the clinicians. Meanwhile, without revealing any additional information beyond the decryption results, the key authority proves to the clinicians that the decryption results are decrypted correctly from the derived ciphertexts.



## 4.2 Cryptographically Verifiable Election

Traditionally, the cryptosystems used in voting were additively or multiplicatively homomorphic [23,37] which places significant restrictions on the kind of counting functions that can be computed. While FHE greatly expands the applicability of homomorphic tallying, such as quantum-safe voting scheme with better voter verifiability shown by Gjøsteen-Strand [21], who applied BGV scheme to internet voting in Norway. However, Gjøsteen-Strand [21] mainly considered the voter verifiability which is equivalent to verifiable encryption rather than verifiable decryption considered in this paper. Specifically, in their construction, the voter encrypts he/her ballot with a symmetric scheme, and attaches the key encrypted by BGV scheme. (The attached key encrypted by BGV scheme is used to check the validity of the ballot.) After that, the ballot box counts the ballots, and then sends a receipt to the voter. Finally, the decryption service decrypts the result submitted by the ballot box. In the process described above, however, the decryption service does not prove the correctness of decryption. Therefore, if the decryption service is corrupted, the voting system is not secure.

Alternatively, our proposed verifiable decryption for BGV scheme can be used to build the following cryptographically verifiable election: voters encrypt their ballots by the given BGV scheme and then post the encrypted ballots on some bulletin boards. A central authority collects the encrypted ballots, and homomorphically evaluates the election counting circuit on these encrypted ballots and then obtains an encrypted result (a derived ciphertext). Note that the encrypted votes are published on some bulletin boards by voters so that voters can also perform the election counting circuit evaluation themselves. The central authority proceeds to decrypt the derived ciphertext and obtains the corresponding plaintext. Then the central authority posts the plaintext and appends a valid proof that the decryption was indeed done correctly. This process of homomorphic tallying [10] is applicable when the counting function used in an election can be efficiently computed on encrypted ballots. From the above, this approach greatly simplifies public verifiability of a voting system, as correctness follows from the homomorphic properties of BGV scheme and the correctness of our proposed verifiable decryption for BGV scheme.

## 5 Conclusion

This paper presented an interactive zero-knowledge proof protocol for decryption of BGV ciphertexts, by combining “Fiat-Shamir with Aborts” zero-knowledge proofs of linear relations with BGV scheme. The interactive zero-knowledge proof protocol only needs one extra communication between the prover and the verifier. Hence, if the communication is completed off-line, then it is also “one shot”. Furthermore, in order to illustrate the point that verifiable decryption is an important tool in a number of applications, we showed two example instantiations – secure medical computation and cryptographically verifiable election. The efficiency of our proposed interactive zero-knowledge proof protocol is mostly affected by the size of the modulus  $q$  and the witness  $s$  in the

relation. The larger these values, the larger the proof and ciphertext will be. The witness  $s$  is the secret key of the given BGV scheme, so it will be a vector of small norm. While the modulus  $q$  depends on the security level and the homomorphic capacity of the given BGV scheme.

**Acknowledgments.** This research is supported in part by the National Basic Research Program of China (973 project, No. 2014CB340603) and the National Nature Science Foundation of China (Nos. 61672030 and 61272040). The authors would like to thank the anonymous reviewers for their detailed reviews and helpful comments.

## References

1. Alperin-Sheriff, J., Peikert, C.: Faster bootstrapping with polynomial error. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 297–314. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-662-44371-2\\_17](https://doi.org/10.1007/978-3-662-44371-2_17)
2. Baum, C., Damgård, I., Larsen, K.G., Nielsen, M.: How to prove knowledge of small secrets. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 478–498. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53015-3\\_17](https://doi.org/10.1007/978-3-662-53015-3_17)
3. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: CCS 1993, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, 3–5 November 1993, pp. 62–73 (1993)
4. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, 8–10 January 2012, pp. 309–325 (2012)
5. 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)
6. 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). [https://doi.org/10.1007/978-3-642-22792-9\\_29](https://doi.org/10.1007/978-3-642-22792-9_29)
7. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003). [https://doi.org/10.1007/978-3-540-45146-4\\_8](https://doi.org/10.1007/978-3-540-45146-4_8)
8. Carr, C., Costache, A., Davies, G.T., Gjøsteen, K., Strand, M.: Zero-knowledge proof of decryption for the ciphertexts. Technical report, Cryptology ePrint Archive, Report 2018/026 (2018). <https://eprint.iacr.org/2018/026>
9. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Faster fully homomorphic encryption: bootstrapping in less than 0.1 seconds. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 3–33. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53887-6\\_1](https://doi.org/10.1007/978-3-662-53887-6_1)
10. Cohen, J.D., Fischer, M.J.: A robust and verifiable cryptographically secure election scheme (extended abstract). In: 26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21–23 October 1985, pp. 372–382 (1985)
11. 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). [https://doi.org/10.1007/978-3-642-03356-8\\_11](https://doi.org/10.1007/978-3-642-03356-8_11)

12. Cramer, R., Damgård, I., Xing, C., Yuan, C.: Amortized complexity of zero-knowledge proofs revisited: achieving linear soundness slack. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 479–500. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56620-7\\_17](https://doi.org/10.1007/978-3-319-56620-7_17)
13. del Pino, R., Lyubashevsky, V.: Amortization with fewer equations for proving knowledge of small secrets. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10403, pp. 365–394. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-63697-9\\_13](https://doi.org/10.1007/978-3-319-63697-9_13)
14. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46800-5\\_24](https://doi.org/10.1007/978-3-662-46800-5_24)
15. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive, 2012:144 (2012)
16. 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). [https://doi.org/10.1007/3-540-47721-7\\_12](https://doi.org/10.1007/3-540-47721-7_12)
17. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-78967-3\\_3](https://doi.org/10.1007/978-3-540-78967-3_3)
18. Gentry, C.: A Fully Homomorphic Encryption Scheme. Stanford University (2009)
19. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31–June 2 2009, pp. 169–178 (2009)
20. 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). [https://doi.org/10.1007/978-3-642-40041-4\\_5](https://doi.org/10.1007/978-3-642-40041-4_5)
21. Gjøsteen, K., Strand, M.: A roadmap to fully homomorphic elections: stronger security, better verifiability. In: Brenner, M., et al. (eds.) FC 2017. LNCS, vol. 10323, pp. 404–418. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70278-0\\_25](https://doi.org/10.1007/978-3-319-70278-0_25)
22. Halevi, S., Shoup, V.: Bootstrapping for  $\text{HElib}$ . In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 641–670. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-46800-5\\_25](https://doi.org/10.1007/978-3-662-46800-5_25)
23. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-45539-6\\_38](https://doi.org/10.1007/3-540-45539-6_38)
24. Khedr, A., Gulak, P.G.: Securemed: secure medical computation using gpu-accelerated homomorphic encryption scheme. IEEE J. Biomed. Health Inf. **22**(2), 597–606 (2018)
25. 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). [https://doi.org/10.1007/978-3-662-53890-6\\_13](https://doi.org/10.1007/978-3-662-53890-6_13)
26. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based PRFs and applications to E-cash. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10626, pp. 304–335. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70700-6\\_11](https://doi.org/10.1007/978-3-319-70700-6_11)

27. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, 19–22 May 2012, pp. 1219–1234 (2012)
28. 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). [https://doi.org/10.1007/978-3-642-10366-7\\_35](https://doi.org/10.1007/978-3-642-10366-7_35)
29. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-29011-4\\_43](https://doi.org/10.1007/978-3-642-29011-4_43)
30. Lyubashevsky, V.: Digital signatures based on the hardness of ideal lattice problems in all rings. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 196–214. Springer, Heidelberg (2016). [https://doi.org/10.1007/978-3-662-53890-6\\_7](https://doi.org/10.1007/978-3-662-53890-6_7)
31. 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). [https://doi.org/10.1007/11787006\\_13](https://doi.org/10.1007/11787006_13)
32. Lyubashevsky, V., Neven, G.: One-shot verifiable encryption from lattices. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 293–323. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-56620-7\\_11](https://doi.org/10.1007/978-3-319-56620-7_11)
33. 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). [https://doi.org/10.1007/978-3-642-13190-5\\_1](https://doi.org/10.1007/978-3-642-13190-5_1)
34. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. *J. ACM (JACM)* **60**(6), 43 (2013)
35. 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). [https://doi.org/10.1007/978-3-642-29011-4\\_41](https://doi.org/10.1007/978-3-642-29011-4_41)
36. Peikert, C., Vaikuntanathan, V.: Noninteractive statistical zero-knowledge proofs for lattice problems. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 536–553. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-85174-5\\_30](https://doi.org/10.1007/978-3-540-85174-5_30)
37. Peng, K., Aditya, R., Boyd, C., Dawson, E., Lee, B.: Multiplicative homomorphic E-voting. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 61–72. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-30556-9\\_6](https://doi.org/10.1007/978-3-540-30556-9_6)
38. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. *Found. Secur. Comput.* **4**(11), 169–180 (1978)
39. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-13013-7\\_25](https://doi.org/10.1007/978-3-642-13013-7_25)
40. Stern, J.: A new paradigm for public key identification. *IEEE Trans. Inf. Theory* **42**(6), 1757–1768 (1996)
41. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-13190-5\\_2](https://doi.org/10.1007/978-3-642-13190-5_2)

# **Privacy-Preserving Applications**



# Platform-Independent Secure Blockchain-Based Voting System

Bin Yu<sup>1</sup>, Joseph K. Liu<sup>1</sup>(✉), Amin Sakzad<sup>1</sup>, Surya Nepal<sup>2</sup>, Ron Steinfeld<sup>1</sup>, Paul Rimba<sup>2</sup>, and Man Ho Au<sup>3</sup>

<sup>1</sup> Monash University, Melbourne, Australia  
joseph.liu@monash.edu

<sup>2</sup> CSIRO, Sydney, Australia

<sup>3</sup> The Hong Kong Polytechnic University, Kowloon, Hong Kong

**Abstract.** Cryptographic techniques are employed to ensure the security of voting systems in order to increase its wide adoption. However, in such electronic voting systems, the public bulletin board that is hosted by the third party for publishing and auditing the voting results should be trusted by all participants. Recently a number of blockchain-based solutions have been proposed to address this issue. However, these systems are impractical to use due to the limitations on the voter and candidate numbers supported, and their security framework, which highly depends on the underlying blockchain protocol and suffers from potential attacks (e.g., force-abstention attacks). To deal with two aforementioned issues, we propose a practical platform-independent secure and verifiable voting system that can be deployed on any blockchain that supports an execution of a smart contract. Verifiability is inherently provided by the underlying blockchain platform, whereas cryptographic techniques like Paillier encryption, proof-of-knowledge, and linkable ring signature are employed to provide a framework for system security and user-privacy that are independent from the security and privacy features of the blockchain platform. We analyse the correctness and coercion-resistance of our proposed voting system. We employ Hyperledger Fabric to deploy our voting system and analyse the performance of our deployed scheme numerically.

**Keywords:** Evoting · Blockchain · Ring signature  
Homomorphic encryption

## 1 Introduction

Voting plays a significant role in a democratic society. Almost every local authority allots a significant amount of budget on providing a more robust and trustworthy voting system. Cryptographic techniques like homomorphic encryption and Mix-net [7] are usually applied in contemporary electronic voting systems to achieve the voting result verifiability while preserving voters' secrecy. However, incidents like a security flaw that has erased 197 votes from the computer

database in the 2008 United States elections [34] and the compromise of 66,000 electronic votes in the 2015 New South Wales (NSW) state election in Australia [30] raise the public concerns on the security of electronic voting systems. For voting systems based on bulletin (e.g., [1,9]), one of the major concerns is whether the voting result that is published on the bulletin can be trusted. Blockchain with the growing popularity and remarkable success in cryptocurrency provides a new paradigm to achieve the public verifiability in such electronic voting systems.

Recently a number of blockchain-based electronic voting systems have been developed by exploiting its inherent features. These systems can be classified into three broad categories. (1) Cryptocurrency based voting systems (e.g., [21,33,43]). The ballots to a candidate are based on the payment he/she receives from the voters; the problem with such systems are malicious voters may refuse to “pay” the candidates to retain the money. Furthermore, a centralised trusted party, who coordinates the payment between the candidates and voters must exist. (2) Smart contract based voting system [28], which only supports two candidates and the voting is restricted to limited number of participants. Furthermore, it requires all voters to cast their ballots before reaching an agreement on the voting result. (3) Using blockchain as a ballot box to maintain the integrity of the ballots [11,35].

In summary, the security of these blockchain-based systems highly depends on the specific cryptocurrency protocol they employed. Additionally, these voting systems can only work with a specific blockchain platform, and support a limited number of candidates and voters. Based on our observations and studies, we believe that any blockchain-based voting systems should have the following three features: (1) Platform-independence — this means the changes in the underlying blockchain protocols should not affect the voting schemes. (2) Security framework — the voting system should be implemented with comprehensive security features (the detail of security features are discussed in Sect. 5). The nature of the blockchain allows everyone to obtain the data on it; thus, the comprehensive security features have critical importance to ensure that the ballots are fully secured on the blockchain. (3) Practical — it should be scalable, which means the large amount of ballots casting and tallying can be finished in a reasonable time.

**Our Contributions:** In this paper, we propose an electronic voting (evoting) system that supports the above identified three features as follows.

1. *Our voting system does not depend on a centralised trusted party for ballots tallying and result publishing.* Compared with traditional voting systems, which highly depend on a centralised trusted party to tally the ballots and publish the result, our voting system takes the advantage of a blockchain protocol to eliminate the need for a centralised trusted party. The details of the blockchain trustworthiness and voting system trust assumptions are discussed in Sects. 4 and 5, respectively.
2. *Our voting system is platform-independent and provides comprehensive security assurances.* Existing blockchain-based voting systems highly depend on the underlying cryptocurrency protocols. Receipt-freeness [31] and correctness

of the voting result are hard to achieve (we analyse the blockchain-based voting system explicitly in Sect. 2). The security of our voting system is achieved by cryptographic techniques provided by our voting protocol itself, thus, our voting system can be deployed on any blockchain that supports smart contract. To achieve the goal of providing a comprehensive security, we employ Paillier system to enable ballots to be counted without leaking candidature information in the ballots. Proof-of-knowledge is employed to convince the voting system that the ballot casted by a voter is valid without revealing the content of the ballot. Linkable ring signature is employed to ensure that the ballot is from one of the valid voters, while no one can trace the owner of the ballot. The detail of security features that we achieved are discussed in Sect. 5.

3. *Our voting system is scalable and applicable.* In order to support voting scalability, we propose two optimised short linkable ring signature key accumulation algorithms which are described in our full paper [41] to achieve a reasonable latency in large scale voting. We evaluate our system performance with 1 million voters to show the feasibility of running a large scale voting with the comprehensive security requirements.

The rest of the paper is organised as follows: we discuss the cryptographic techniques applied in voting systems and analyse some typical voting systems in Sect. 2. Cryptographic primitives and our voting protocol are presented in Sects. 3 and 4, respectively. We analyse the correctness and security of our voting system in Sect. 5. In Sect. 6, we deploy our voting system and analyse its performance.

## 2 Related Work

Secure evoting is considered as one of the most difficult problems in security literature as it involves many security requirements. To satisfy these security requirements, cryptographic techniques are mostly applied in constructing a secure evoting system. In this section, we discuss the existing voting systems based on traditional public bulletin and blockchain technology.

### 2.1 Public Bulletin Based Voting Systems:

In the following, we outline the key cryptographic techniques used in public bulletin based voting systems and how such techniques address the corresponding security requirements.

- **Homomorphic encryption:** Homomorphism feature allows one to operate on ciphertexts without decrypting them [13]. For a voting system, this property allows the encrypted ballots to be counted by any third party without leaking any information in the ballot [10, 14, 18]. Typical cryptosystems applied in a voting system are Paillier encryption [32, 40] and ElGamal encryption [19, 21].



- **Mix-net:** Mix-net was proposed in 1981 by Chaum [7]. The main idea of mix-net is to perform a re-encryption over a set of ciphertexts and shuffle the order of those ciphertexts. Mix node only knows the node that it immediately received the message from and the immediate destination to send the shuffled messages to. The voting systems proposed in [1, 17, 20] apply mix-net to shuffle the ballots from different voters, thus the authority cannot relate a ballot to a voter. For the mix-net based voting systems, they need enough amount of mix nodes and ballots to be mixed.
- **Zero-knowledge proof:** Zero-knowledge proof is often employed in a voting system [9, 29, 39] to let the prover to prove that the statement is indeed what it claimed without revealing any additional knowledge of the statement itself. In a voting system, the voter should convince the authority that his ballot is valid by proving that the ballot includes only one legitimate candidate without revealing the candidate information.
- **Blind signature and linkable ring signature:** Voting systems like [12, 16, 22] employ blind signature [12] to convince the tallying centre that the ballot is from a valid voter while not revealing the owner of the ballot. Simultaneously, the authority who signs the ballot learns nothing about the voter’s selections. In blind signature, both voters and tallying centre must trust the signer. If the signer is compromised, the signature scheme may stop working. Unlike blind signature, linkable ring signature [3–5, 8, 23–27, 36, 42] is proposed to avoid the untrusted signer. Instead, it needs a certain number of voters to participate in the signing process. By comparing the linkability tag, the authority can easily tell whether this voter has already voted. When the voter signs on the ballot, he/she needs to include other voters’ public keys to make his/her signature indistinguishable from other voters’ signatures.

## 2.2 Blockchain-Based Voting Systems

The blockchain-based voting systems can be discussed under three broad categories as follows.

- **Voting systems using cryptocurrency:** In [43], authors propose a voting system based on Bitcoin. In their voting system, the ballot does not need to be encrypted/decrypted as they employ the protocol for lottery. Random numbers are used to hide the ballot that are distributed via zero-knowledge proof. Making deposit before voting may keep the voters to comply with the voting protocol while the malicious voters can still forfeit the voting by refusing to vote. However, only supporting “yes/no” voting may restrict the adoption of this voting system.

In [33], authors proposed a voting system on the Zcash payment protocol [15] without altering the inner working of Zcash protocol. The voter’s anonymity is ensured by the Zcash address schemes. The correctness of the voting is guaranteed by the trusted third-party and the candidates. In this system, the authority, who manages the Zcash and voter status database should be

trusted. If the authority is compromised, double-voting or tracing the source of the ballot is possible.

- **Voting systems using smart contract:** In [28], the authors claim that their open voting network is the first implementation of a decentralised and self-tallying Internet voting protocol with maximum voter privacy using Blockchain. They employ smart contract as a public bulletin to achieve self-tallying.<sup>1</sup> However, their voting system can only work with two candidates voting (yes/no voting) and the limitation of 50 voters makes it impractical for a real large scale voting system.
- **Voting systems using blockchain as a ballot box:** Tivi and Followmyvote [11,35] are commercial voting systems which employ the blockchain as a ballot box<sup>2</sup>. They claim to achieve verifiability and accessibility anytime anywhere, while the voters' privacy protection in these systems is hard to evaluate.

To summarize, most traditional voting systems need a centralised trusted party to coordinate the whole voting process. In these systems, the centralised trusted party plays a critical role in storing the ballots, counting the ballots, and publishing the voting result. Although the existing blockchain-based voting systems take advantage of blockchain public verifiability, the system security and user privacy of these systems depend on the features provided by the underlying blockchain platform. Our proposed approach not only takes the benefits of a decentralised trust offered by the blockchain technology to remove the need of a centralised trusted party to do the ballots tallying, voting result decoding and publishing, but also considers key security primitives proposed in the literature including traditional voting systems to build a practical platform independent secure voting protocol that can be deployed to any blockchain platforms that support smart contract.

### 3 Cryptographic Primitives

In this section, we describe the cryptography primitives borrowed from traditional voting systems and apply in our voting system. Note that the syntaxes, correctness conditions, and security models of a linkable ring signature and a public key encryption are given in Appendix A and Appendix B, respectively in our full paper [41].

#### 3.1 Message Encode and Decode

Before the voting starts, we must encode the candidate ID to make it suitable for vote tallying. The encode/decode functions are defined as follows:

---

<sup>1</sup> Self-tallying means that after the casting phase, voters can count the ballots themselves.

<sup>2</sup> The authors call the storage of the ballot as the ballot box.

- **Candidate encoding:** We define  $\zeta := \text{Encode}(m) \in \mathbb{Z}$  as the candidate encoding function. For  $\rho$  candidates, each labeled with an ID from  $\mathcal{P} = \{1, 2, \dots, \rho\}$ ,  $\beta = 2^{\rho+1}$  be the basis of encoding operation. We encode the  $m^{\text{th}}$  candidate as  $\zeta = \beta^m$  where  $m \in \mathcal{P}$ . We choose 2 as the basis of  $\beta$  as the division operation can be replaced by the CPU register right shift instruction to achieve a better performance.
- **Candidate decoding:** Let  $k = k_t\beta^t + \dots + k_n\beta^n + k_{n-1}\beta^{n-1} + \dots + k_1\beta + k_0$  be the representation of  $k$  base  $\beta$ ,  $k \in \mathbb{Z}$ , then we define the right shift  $k$  with  $n$  positions as  $\text{rsh}(k, n) = k_t\beta^{t-n} + k_{t-1}\beta^{t-n-1} + \dots + k_{n+1}\beta + k_n$ . Let  $\text{sum} = s_\rho\beta^{\rho-1} + s_{\rho-1}\beta^{\rho-2} + \dots + s_2\beta + s_1$  be the addition of all the ballots where  $s_j$  is the total number of ballots that the candidate  $j^{\text{th}}$  acquires. We define  $s_j := \text{Decode}(\text{sum}, j)$  for  $1 \leq j \leq \rho$  and is computed as  $s_j = \text{rsh}(\text{sum}, \beta^{j-1}) \bmod \beta$ .

### 3.2 Paillier Encryption System [37]

Paillier Encryption system is employed to enable our voting system to tally the encrypted ballots. In our voting system, we implement the following functions in Paillier system and the detail of these functions are described in Appendix B.1 in the full paper [41].

- **Key Generation:**  $(\text{sk}_{\text{Paillier}}, \text{pk}_{\text{Paillier}}) := \text{Gen}_{\text{Paillier}}(K_{\text{len}})$  is the function to generate the secret key  $\text{sk}_{\text{Paillier}}$  and the corresponding public key  $\text{pk}_{\text{Paillier}}$  with the given key length  $K_{\text{len}}$ . The voting administrator invokes this function to generate the key pair and uploads the public key  $\text{pk}_{\text{Paillier}}$  to the blockchain.
- **Encryption:**  $C_{\text{Ballot}} \leftarrow \text{Enc}_{\text{Paillier}}(\zeta_{\text{Ballot}}, \text{pk}_{\text{Paillier}})$  where  $\zeta_{\text{Ballot}} \in \mathbb{Z}_n$  is the plaintext ballot to be encrypted. Voters download the  $\text{pk}_{\text{Paillier}}$  from the blockchain and call this function to encrypt their ballots. This function generates the encrypted ballot  $C_{\text{Ballot}}$ .
- **Decryption:**  $\zeta_{\text{Res}} := \text{Dec}_{\text{Paillier}}(C_{\text{Res}}, \text{sk}_{\text{Paillier}})$  where  $C_{\text{Res}} \in \mathbb{Z}_n^*$  is the encrypted voting result; the voting administrator invokes this function to decrypt the voting result.
- **Message Membership Proof of Knowledge [6]:**  $\{v_j, e_j, u_j\}_{j \in \mathcal{P}} := \text{PoK}_{\text{mem}}(C_{\text{Ballot}}, \mathcal{Y})$  where  $C_{\text{Ballot}}$  is the encrypted ballot,  $\mathcal{Y}$  is the set of the encoded candidates. When a voter publishes his/her ballot, he/she invokes this function to generate the proof  $\{v_j, e_j, u_j\}_{j \in \mathcal{P}}$  that demonstrates his/her ballot encrypts only one of the encoded candidates in  $\mathcal{Y}$ .
- **Decryption Correctness Proof of Knowledge:**  $(\zeta_{\text{Res}}, r) := \text{PoK}(C_{\text{Res}}, \text{sk}_{\text{Paillier}})$ , where  $\zeta_{\text{Res}}$  is the plaintext and  $r$  is the random factor that generate the encrypted voting result  $C_{\text{Res}}$ . After publishing the voting result, the voting administrator invokes this function to generate a unique value pair  $(\zeta_{\text{Res}}, r)$  that constructs the  $C_{\text{Res}}$  to prove that he/she decrypts the voting result  $C_{\text{Res}}$  correctly.

### 3.3 Linkable Ring Signatures

Linkable ring signature (LRS) can be applied in our system to protect the privacy of the voters. In practice, we apply the short linkable ring signature (SLRS) [2]

which extends all the SLRS features to make the signature size constant with the growth of voter numbers, it has the following features: (1) every ballot that is accepted by the system is from one of the valid users, (2) the voter can check whether his ballot is counted by the blockchain, (3) the size of the signature is constant to support scalability and (4) double-voting is prevented. In our voting system, we implement the function tuple (Setup, KeyGen, Sign, Verify, Link). The details of these functions are explained in Appendix A.2 in the full paper [41].

- **Setup:**  $\text{param} \leftarrow \text{Setup}(\lambda)$  is a function that takes  $\lambda$  as the security parameter and generates system-wide public parameters  $\text{param}$  such as the group of quadratic residues modulo a safe prime product  $N$  (explained in Appendix A.2 in the full paper [41]) denoted as  $\text{QR}(N)$ , the length of the key, and a random generator  $\tilde{g} \in \text{QR}(N)$ .
- **Key Generation:**  $(\text{sk}_i, \text{pk}_i) \leftarrow \text{KeyGen}(\text{param})$  is a function to generate a key pair for each voter  $i$ .
- **Signature:**  $\sigma \leftarrow \text{Sign}(\mathcal{Y}, \text{sk}, \text{msg})$  is a function to generate the signature  $\sigma$  using all voters' public keys  $\mathcal{Y} = \{y_1, y_2, \dots, y_b\}$ , the message  $\text{msg}$  to be signed, and the voter's secret key  $\text{sk}$ . Voters should invoke this function to sign on his/her encrypted ballot.
- **Verification:**  $\text{accept/reject} \leftarrow \text{Verify}(\sigma, \mathcal{Y}, \text{msg})$ ; our voting system invokes this function to test the validity of every ballot. Based on the input of the encrypted ballot itself, the voter's signature and all the voters' public keys, the blockchain accepts or rejects this ballot to be put on the chain.
- **Linkability:**  $\text{Link}(\sigma_1, \sigma_2) \rightarrow \text{linked/unlinked}$ . When a voter casts his/her vote, our voting system invokes this function to check whether this voter has already casted his vote. If this function returns linked, our voting system rejects this ballot; otherwise, the ballot is recorded on the chain.

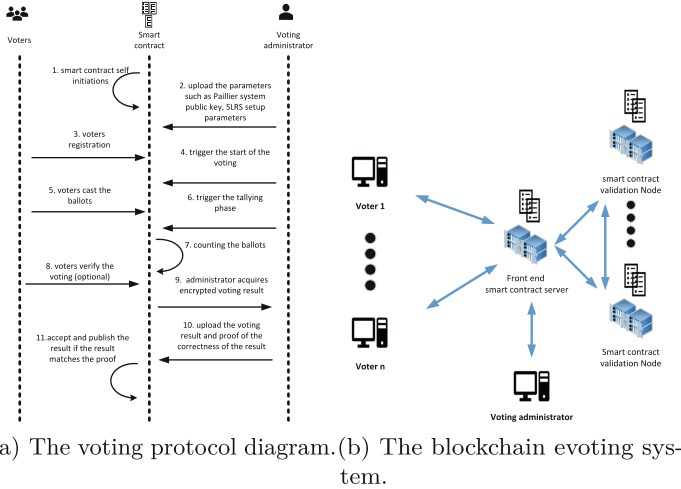
## 4 The Voting Protocol

In this section, we first provide an overview of the whole voting protocol and then discuss each step in details. The whole voting process can be divided into 11 steps as shown in Fig. 1(a). Except smart contract administrator, three entities are involved: voters, smart contract, and voting administrator.

The voting system consists of one front-end smart contract and several validation nodes as shown in Fig. 1(b). The role of a validation node is to replicate the execution of the smart contract codes to ensure its correct execution. For a practical voting, the validation nodes could be held by different entities/stakeholders, thus all ballots on the blockchain have been verified by different entities/stakeholders. As all the entities/stakeholders have the agreement on the data stored on the blockchain, the blockchain built on the servers owned by different entities can be regarded as trustworthy.

### 4.1 Entities in the Voting Process

Four entities should be involved in our voting system shown in Fig. 1(a), and details are explained as follows:



**Fig. 1.** The general voting protocol and how entities are connected.

- **smart contract administrator:** he/she has the ability to access the smart contract platform to deploy/terminate smart contract. In Hyperledger fabric, this account is authorised by the membership service and a permission is granted to deploy/terminate the Smart contract. In our voting system, we need at least one smart contract administrator to deploy the voting smart contract.
- **voting administrator:** The role of voting administrator is to organize the vote by setting up the voting parameters and triggering the tallying and result publishing phase. Although there are underlying mechanisms in hyperledger to authenticate users, we use SLRS to prevent administrator from linking the ballots with the users.
- **smart contract:** The role of smart contract include (1) store the encrypted ballots. (2) verify the validity of the ballots. (3) count the encrypted ballot. (4) verify the correctness of the voting result. (5) publish the voting result and provide the platform for the voters to verify the voting process.
- **voters:** Voters are the people who have the rights to cast their vote. They need to register into the voting system before they cast their vote.

### 4.2 Voting System Set Up

During the system set up, the voting administrator uploads the following three parameters to the blockchain:

- The public key of the Paillier system  $pk_{Paillier}$ .
- A set of encryptions of zero denoted as  $T$ , generated by the administrator’s Paillier public key  $pk_{Paillier}$ . For voting with 1 million voters, we suggest the set

should contain enough elements to make the randomised pool large enough and the detail of  $T$  is discussed in receipt-freeness analysis<sup>3</sup>.

- The SLRS scheme parameters,  $\text{param}$ .

### 4.3 Voter Registration

Bob must register this voting system with his identity. The registration information could be: (1) email address with a desired password, or (2) the identity number with a desired password, or (3) an invitation URL sent by the administrator with a desired password. After Bob passes the identity check conducted by the smart contract, he can login with the desired password to download the SLRS  $\text{param}$  and the administrator's Paillier public key  $\text{pk}_{\text{Paillier}}$ , then generates the SLRS key pair  $(\text{sk}_i, \text{pk}_i)$  by calling  $\text{KeyGen}(\text{param})$ ; Bob then uploads the public key  $\text{pk}_i$  to the smart contract (Bob's secret key is kept off-chain by himself). If the smart contract accepts his SLRS public key, the smart contract puts his public key  $\text{pk}_i$  on the blockchain to complete his registration phase.

### 4.4 Vote Casting Phase

During this phase, Bob casts his vote as follows: (1) Bob chooses one of the candidates  $m \in \mathcal{P}$  and encodes it as  $\zeta := \text{Encode}(m)$ . (2) Bob invokes the Paillier encryption function  $C \leftarrow \text{Enc}_{\text{Paillier}}(\zeta, \text{pk}_{\text{Paillier}})$ . (3) Bob needs to prove that  $C$  is an encryption of an element in  $\{\zeta_1, \dots, \zeta_\rho\}$  (set of all encoded candidates) by calling  $\{v_j, e_j, u_j\}_{j \in \mathcal{P}} := \text{PoK}_{\text{mem}}(C, \mathcal{Y})$ ; hence he sends  $\pi = \{C, \{v_j, e_j, u_j\}_{j \in \mathcal{P}}\}$  to the smart contract.<sup>4</sup> (4) Upon receiving  $\{v_j, e_j, u_j\}_{j \in \mathcal{P}}$ , the smart contract verifies the validation of the encrypted ballot  $C$ . We denote  $\phi$  as a mapping of this transaction's session id to  $T$  domain. If  $C$  is valid, the smart contract takes an encryption of zero at  $\phi^{\text{th}}$  position. Let  $\epsilon$  be the addition of  $T[\phi]$  and  $C$ . The smart contract signs on  $\epsilon$  denoted as  $s$  and sends  $(\epsilon, s)$  back to Bob. (5) If Bob accepts  $s$ , he invokes  $(v, \tilde{y}, \sigma') := \text{Sign}(\epsilon, (\text{pk}, \text{sk}), \mathcal{Y})$  to generate the  $\text{Sign}_{\text{bob}}$  on  $\epsilon$  and sends  $(\epsilon, \text{Sign}_{\text{bob}})$  to the smart contract. (6) If the smart contract detects that  $\text{Sign}_{\text{bob}}$  has already been recorded on the blockchain or cached in the memory, it rejects Bob's vote; otherwise,  $(\epsilon, \text{Sign}_{\text{bob}})$  is put on the blockchain.

### 4.5 Ballots Tallying and Result Publishing

Due to the Paillier system's homomorphic feature, counting the vote is as simple as fetching the encrypted ballots from the blockchain and adding them together. The result publishing mechanism is described in the following steps: (1) Let  $E_{\text{sum}}$

<sup>3</sup> To avoid requiring the administrator to upload the encryption of zero pool, coin flipping protocol can be applied to generate the consistent encryption of zero on all the validation nodes and this is our future work.

<sup>4</sup> Bob can choose none of the actual candidates by casting his ballot to a dummy candidate. When the smart contract publishes the voting result, it ignores all the ballots that the dummy candidate gets.

be the sum of all the encrypted votes and  $\text{Sign}_s$  be the signature signed by the smart contract on  $E_{\text{sum}}$ . The smart contract sends  $(E_{\text{sum}}, \text{Sign}_s)$  to the administrator. (2) The administrator invokes  $\text{sum} := \text{Dec}_{\text{Paillier}}(E_{\text{sum}}, \text{sk}_{\text{Paillier}})$  to compute plaintext  $\text{sum}$ , which encodes the ballots of each candidate. The administrator also invokes  $(\text{sum}, r) := \text{PoK}(E_{\text{sum}}, \text{sk}_{\text{Paillier}})$  to calculate the random  $r$  that constructs this  $E_{\text{sum}}$ , and sends  $(\text{sum}, r)$  to the smart contract. (3) The smart contract verifies the correctness of  $(\text{sum}, r)$  by checking if  $E_{\text{sum}} \stackrel{?}{=} g^{\text{sum}r^n}$  ( $g$  is one of the elements of  $\text{pk}_{\text{Paillier}}$ ). (4) If the smart contract accepts the  $\text{sum}$ , it iteratively invokes  $m := \text{Decode}(\text{sum}, i)$  to compute the ballots for each candidate  $i$ . Let  $\Pi$  be the dictionary holding the voting result of all candidates. The smart contract finally puts  $\Pi$  on the blockchain.

#### 4.6 Ballot Verifying

After tallying ballots and before the voting administrator decrypts the tallying result, the public can verify ballots on the blockchain to make sure the validity of the voting process. We define two roles for people who can verify the voting. The first one is those who have the right to access the data on the blockchain while they do not have the right to vote. The second one is those who have both rights to vote and access the data on the blockchain. The public verifiability to those who have first role is as follows (1) Checking the number of ballots that were counted and the number of people registered for this voting. (2) Checking the correctness of the tallying result by downloading and adding all the encrypted ballots to match with the tallying result published on the blockchain. Compared to those who have the first role, people assigned to the second role can also verify that his/her ballot is recorded on the blockchain by checking whether there exists one ballot that is signed with his/her signature; This ensure his/her vote is recorded and counted.

In practice, we suggest enhancing the trustworthiness of the blockchain by allowing different political parties/stakeholders host their own validation nodes. The data on the blockchain is verified by different entities/stakeholders, and it is unlikely that these entities/stakeholders collude with each other to publish a false voting result.

## 5 Correctness and Security Analysis

### 5.1 Correctness Analysis

The correctness of our voting system is guaranteed by the public verifiability provided by the smart contract and the proof of knowledge provided by the cryptographic schemes. More than that, the smart contract ensures the consistency of a transaction execution. Any inconsistencies generate an error which results in the rejection of the transaction. This means the voting participants can be assured that every transaction on the blockchain is verified and accepted by all participating nodes. This prevents compromised nodes from putting an invalid

data on the blockchain unless the adversary can take control of a proportion of the nodes in the whole blockchain network<sup>5</sup>.

## 5.2 Security Features of Our Voting System

- **Privacy:** The ballots on the public ledger are encrypted and only the voting administrator who initiates the voting can decrypt the ballots. This ensures that the tallying center can count the ballots without knowing the content of the ballots.
- **Anonymity:** The voters, candidates, or smart contract cannot tell the public key of the signer with a probability larger than  $1/b$ , where  $b$  is the number of the voters. This can be guaranteed by the anonymity property of the linkable ring signature (LRS) scheme. Details are explained in Appendix A.2 in the full paper [41].
- **Double-voting-avoided:** In our voting system, we take the advantage of linkability provided by the short ring signature scheme. This means it is infeasible for a voter to generate two signatures such that they are determined to be unlinked. Our system can detect whether the signatures are from the same voter. Hence, a voter can only sign on one ballot and cast his/her ballot only once. This can be guaranteed by the linkable property of the LRS scheme. Details are explained in Appendix A.2 in the full paper [41].
- **Slanderability-avoided:** A voter cannot generate a signature which is determined to be linked with another signature not generated by him/her. In other words, an adversary cannot fake other voters' signature. This can be guaranteed by the non-slanderability property of the LRS scheme. Details are explained in Appendix A.2 in the full paper [41].
- **Receipt-freeness:** Even if an adversary obtains a voter's secret key, the adversary cannot prove that this voter voted for a specific candidate. This is guaranteed by the addition of encryption of zero which provides additional randomness to the ciphertext which is unknown to the voter. Thus, even if the voter's secret key is disclosed, no one can prove his casted ballot. For our voting system, the security level of the receipt-freeness is affected by the size of the encryption of zero pool, as the voters can collude with each other to reconstruct the encryption of zero pool. One solution is increase the size of zero pool thus more voters is required to reconstruct the pool. Another solution is applying coin flipping protocol on all validation node to work out a consistent randomness encryption of zero for each valid ballot. We have taken the first approach in this paper with 4096 encryptions of zeros.
- **Public verifiability:** Anyone who has the relevant rights to access the blockchain can verify that all the ballots are counted correctly; moreover, voters can also verify whether their ballots have been recorded.
- **Correctness:** Proof-of-knowledge ensures the correctness of the voting process. Voting participants need to prove the correctness of the interactions

---

<sup>5</sup> The number of nodes to be compromised depends on the underlying consensus protocol.



with the blockchain. Even if some blockchain nodes are compromised, others can still verify whether the proof is correct.

- **Vote-and-Go:** Compared with the voting system proposed in [31], our voting system does not need the voter to trigger the tallying phase. Moreover, in our system, voters can also cast their vote and quit before the voting ends, unlike [28] which needs all voters to finish voting before tallying the ballots.

### 5.3 Security and Coercion-Resistance Analysis

To address the security and coercion-resistance, we make the following assumptions:

- The trustworthiness of the blockchain platform is achieved by allowing different stakeholders/entities to host the blockchain validation nodes.
- Voters cast their ballots in a secure terminal, which means it is assumed that no one stand behind a voter or uses digital devices to record the voting process. We do not take the physical voting environment security into our consideration.
- The possibility of an attacker to create a blockchain and apply a social engineering technique to launch a phishing attack is beyond our research scope.
- The administrator should not reveal the Paillier system secret key and the encryption of zeros to anyone.
- Voters should cast their ballots by themselves. No one else can cast the ballot with a voter's identification except the voter himself.

We demonstrate the robustness of our system under two typical attacks below.

*Man-in-the-Middle Attacks:* Our voting system has strong resistance to this attack. First, as the voters and the smart contract both sign their messages and the voting data is encrypted, it is impossible for an adversary to forge the signature or alter the data on any parties involved in the transactions. Second, the public keys used for signature verification are all published on the blockchain, preventing the adversary from cheating any parties by replacing the original public key with the adversary's public key. The encryption of the ballot also eliminates the possibility of the ballot leakage.

*Denial-of-Service (DoS) Attacks:* DoS attack is feasible to launch since the network service is provided in a relatively centralised way. In addition, the servers have relatively limited processing ability for a large number of requests. Distributing the service on different nodes is one of the solutions to DoS attack as it is almost impossible for the adversary to compromise all the servers.

**Coercion-Resistance Analysis:** Coercion-Resistance means it is infeasible for the adversary to determine whether a coerced voter complies with the demand. Our voting system security features discussed in Sect. 5.2 make it impossible to launch the Ballots-buyer attack and Double voting attack. Additionally, our voting system is free from randomization attack.

For the Ballots-buyer attack, an attacker coerces a voter by requiring that he submits a randomly composed ballot. Under such circumstances, both the attacker and the voter have no idea about which candidate this voter casts the ballot for. The purpose of this attack is to nullify the ballots. For our system, it is impossible to launch this attack as the voter should prove that the ciphertext is one out of  $\rho$  encrypted candidates by calling  $\{v_j, e_j, u_j\}_{j \in \mathcal{P}} := \text{PoK}_{\text{mem}}(\text{Enc}_{\text{Paillier}}(\zeta, \text{pk}), \mathcal{Y})$ . Since  $\mathcal{Y}$  is held by the smart contract, any ballot that is not the encryption of any element in  $\mathcal{Y}$  is rejected and the voter is notified that this transaction is failed.

## 6 System Deployment and Experiments

### 6.1 System Deployment

Our voting system can be deployed in any blockchain platforms with smart contract capability and achieve the same level of security. There might be some other reasons to choose a particular platform such as voting latency and flexibility requirements. Different consensus protocols have significant impact on the blockchain network latency and node scalability [38]. If the ballots' confirmation latency is not a major issue for the voting system, the PoW-based blockchain system could be a good option to achieve maximum node scalability. Otherwise, a BFT-based blockchain platform is a better solution. In our scenario, we employ the BFT-based blockchain platform Hyperledger Fabric and deploy our voting system in a practical scenario.

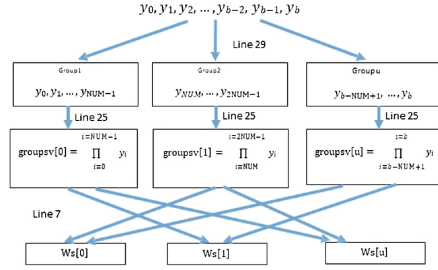
### 6.2 Experiments and Performance Evaluation

We deploy our system in docker containers running on a desktop with 4 cores i5-6500 CPU and 8 GB DDR3 memory. We conduct 1 million voters voting process on the blockchain that consists of 4 validation nodes and 1 PBFT leader node. Each of the validation nodes runs in one dedicated container; thus, we run five docker containers to build our testing blockchain system. We set a voter's public key as 1024 and 2048 bits respectively and the Paillier key pairs as 1024 bits. The deployment pattern is shown in Fig. 1(b). We summarize the time spent on our employed cryptographic processes for 1 million voters' voting in Table 1.

**Voting Parameters Setting Up Time (Administrator Side):** To initialise the voting, the administrator is responsible for uploading the voting parameters as discussed in Sect. 3. Let  $t_{\text{cal}}$  be the time taken to generate  $T$ , and  $t_{\text{upload}}$  be the time spent on uploading  $T$  to the blockchain. With 1024 bits key length, the pool size is 1 MB. According to our test,  $t_{\text{upload}}$  is  $< 1$  s and  $T_{\text{cal}}$  is about 14 s. In conclusion, under 100 MB bandwidth network, on the smart contract side, the majority of the time is spent on bottom half key accumulation, and on the administrator side, the most time-consuming phase is generating and uploading  $T$  (the pool of encryption of zeros).

**Table 1.** Time consumed on each step.

Step	Time
Generate $T$	13,560 ms
Bottom half key accumulation	<34 s
Top half key accumulation	<23 ms
Download parameters	4 ms
Upload ballots	≈776 ms
Tally	3,850 ms
Decode and publish	<2,000 ms

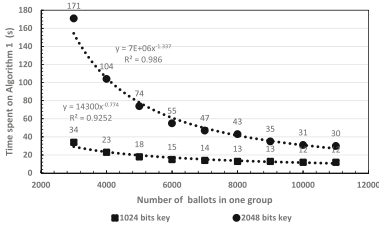


**Fig. 2.** The diagram of Algorithm 1.

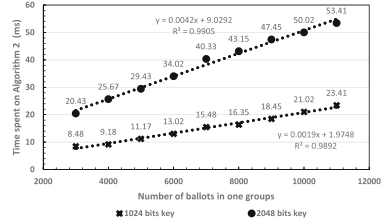
**SLRS Parameter Setting Up Time:** Compared with LRS, SLRS enables the size of the signature constant no matter how many signers are involved in this signature. This feature is critical important for a large scale voting (i.e. the number of voters  $> 100,000$  1024-bits keys) as the signature should be constant to be suitable for storing on the blockchain. Compared with LRS, SLRS needs an extra step that accumulates all the signers’ public keys. Let  $y_i$  be the public key of  $i^{\text{th}}$  voter and  $\psi$  be the SLRS public parameter for all voters. We define the key accumulation operation for all the voters’ SLRS public keys for the  $i^{\text{th}}$  voter as  $w_i = \psi^{y_1 y_2 \dots y_{i-1} y_{i+1} y_{i+2} \dots y_b}$ . In order to make the time spent on key accumulation acceptable, we divide the key accumulation into two halves. The bottom half is run on smart contract and the top half is run by each voter.

**Bottom Half Time Consumption (Smart Contract Side):** For the bottom half (the bottom half algorithm is shown in our full paper [41]), on the smart contract, we divide the voter SLRS public keys into  $m$  groups and pre-calculate the accumulation of all the public keys except the keys in the given group  $i$  and denote this key accumulation as  $ws_i$ . A diagram that shows how bottom half algorithm works is also given in Fig. 2. We only discuss a case in which the number of the voters is larger than 500; otherwise, the voters can generate the key accumulation themselves within a reasonable computation time. We denote  $G$  as the group that contains the voters’ SLRS public key  $pk$  and  $f$  the public key accumulation function. We invoke an array operation function *append* to add an element into the array. We distribute the voters’ SLRS public keys into  $u$  groups and each group has *Num* of keys (except the last group). We denote the array  $WS$  to store all  $ws$ , and  $gkeys$  to store the voters’ SLRS public keys groups. The most time-consuming part is the multiplication of public keys for each group. In our implementation, we calculate the  $WS$  using four threads to save time.

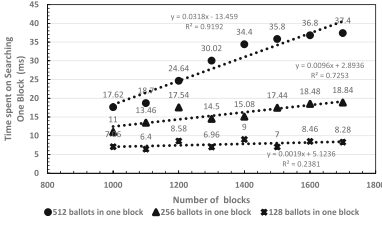
We evaluate the performance for 1 million voters’ voting and the result is shown in Fig. 3(a). We find that the time spent on calculating  $WS$  decreases with the growth of the voter numbers in one group. For example, for 1024 bits length and 2048 bit length key accumulation, it decreases from 34s and 171 s for the group that contains 3000 voters to 13 s and 35 s for the group that contains 8,000



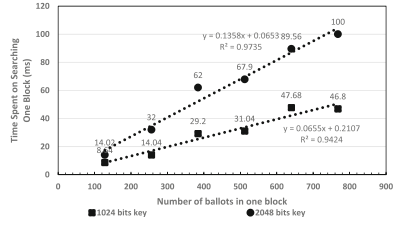
(a) Time spent in Algorithm 1.



(b) Time spent in Algorithm 2.



(c) Time versus growth of ballots in a block. (d) Time versus growth of voter numbers. block.



**Fig. 3.** SLRS public key accumulation and searching a block on a given blockchain in 1 million voters' voting

voters, respectively. This is due to (1) the time spent on running the exponential computation loop in bottom half algorithm that dominates the time spent for the whole algorithm (2). For the 1 million voters' public key accumulation, we decrease the number of groups by increasing the number of the voters in a group to decrease the time spent on the loop in bottom half execution.

**Top Half Time Consumption (Voter Side Key Accumulation):** For the top half execution (The top half algorithm is shown in our full paper [41]), the voter downloads the array  $WS$  and the key group that his key belongs to in  $keys$  from the blockchain. The time spent on downloading these parameters is acceptable because of two reasons (1) the key size and the element size in  $WS$  are constant. (2) The number of groups is relatively small. For instance, if we have 1 million voters and we group 5000 voters in one group, and set the public key size as 1024 bits and  $N$  as 1024 bits, then the size of all the public keys in this group, denoted as  $\mathcal{Y}'$ , is approximately 624 KB. The size of an element in  $WS$  is restricted by SLRS parameter  $N$ ; thus with the parameters above,  $WS$  is 256 KB. Therefore, the total size of  $\mathcal{Y}'$  and  $WS$  is about 880 KB. The voter only needs to do one exponential operation, regardless of the voting scale. From Fig. 3(b), it can be seen that with the key size of 1024 bits, it increase from 8.48ms for the group that contains 3000 voters to 16.35ms for the group that contains 8000 voters. The increase of time spent for the key accumulation on the voter's side can be explained as the increase of time spent in top half execution, as it dominates the total time spent for the voter side key accumulation.

For 1 million voter's voting, we could set the number of voters in one group smaller to reduce the time spent on the voter side for key accumulation. For the key length of 1024 bits, we recommend to set each group contains about 7000 voters so that it takes 14s and 15.48ms on the smart contract side and the voter side key accumulation, respectively.

Based on the above experiments, it is clear that both the increases of the block size and chain length increase the time spent on searching one given block in the blockchain. For 1 million voters' voting system, the blockchain that consists of smaller blocks has better search performance. However, the drawback of the smaller block size is that it increases the number of searching operations (e.g., if we put all ballots in one block, users only searches once to get them; whereas if we allocate all of them in 10 blocks, users have to search 10 times to get them). Based on the experiment results shown in Fig. 3(c) and (d), in practice, we recommend to set each block contains 640 ballots to achieve both a reasonable search time latency and the average number of search operations.

## 7 Conclusion

To solve the problems that the current blockchain voting system cannot provide the comprehensive security features, and most of them are platform dependent, we have proposed a blockchain-based voting system that the voters' privacy and voting correctness are guaranteed by homomorphic encryption, linkable ring signature, and PoKs between the voter and blockchain. We analyse the correctness and the security of our voting system. The experimental results show that our voting system achieves a reasonable performance even in a large scale voting.

## References

1. Adida, B.: Helios: web-based open-audit voting. In: USENIX Security Symposium, vol. 17, pp. 335–348 (2008)
2. Au, M.H., Chow, S.S.M., Susilo, W., Tsang, P.P.: Short linkable ring signatures revisited. In: Atzeni, A.S., Lioy, A. (eds.) EuroPKI 2006. LNCS, vol. 4043, pp. 101–115. Springer, Heidelberg (2006). [https://doi.org/10.1007/11774716\\_9](https://doi.org/10.1007/11774716_9)
3. 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). [https://doi.org/10.1007/11941378\\_26](https://doi.org/10.1007/11941378_26)
4. Au, M.H., Liu, J.K., Susilo, W., Yuen, T.H.: Certificate based (linkable) ring signature. In: Dawson, E., Wong, D.S. (eds.) ISPEC 2007. LNCS, vol. 4464, pp. 79–92. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-72163-5\\_8](https://doi.org/10.1007/978-3-540-72163-5_8)
5. Au, M.H., Liu, J.K., Susilo, W., Yuen, T.H.: Secure ID-based linkable and revocable-iff-linked ring signature with constant-size construction. *Theor. Comput. Sci.* **469**, 1–14 (2013)
6. Baudron, O., Fouque, P.A., Pointcheval, D., Stern, J., Poupard, G.: Practical multi-candidate election system. In: Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing, pp. 274–283. ACM (2001)

7. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* **24**(2), 84–90 (1981)
8. Chow, S.S.M., Susilo, W., Yuen, T.H.: Escrowed linkability of ring signatures and its applications. In: Nguyen, P.Q. (ed.) *VIETCRYPT 2006*. LNCS, vol. 4341, pp. 175–192. Springer, Heidelberg (2006). [https://doi.org/10.1007/11958239\\_12](https://doi.org/10.1007/11958239_12)
9. Chow, S.S., Liu, J.K., Wong, D.S.: Robust receipt-free election system with ballot secrecy and verifiability. In: *NDSS*, vol. 8, pp. 81–94 (2008)
10. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. *Trans. Emerg. Telecommun. Technol.* **8**(5), 481–490 (1997)
11. follow my vote. <https://followmyvote.com/>. Accessed 24 June 2017
12. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Seberry, J., Zheng, Y. (eds.) *AUSCRYPT 1992*. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993). [https://doi.org/10.1007/3-540-57220-1\\_66](https://doi.org/10.1007/3-540-57220-1_66)
13. Gentry, C.: A Fully Homomorphic Encryption Scheme. Stanford University (2009)
14. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) *EUROCRYPT 2000*. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-45539-6\\_38](https://doi.org/10.1007/3-540-45539-6_38)
15. Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification. Technical report, 2016–1.10. Zerocoin Electric Coin Company (2016)
16. Joaquim, R., Zúquete, A., Ferreira, P.: Revs—a robust electronic voting system. *IADIS Int. J. WWW/Internet* **1**(2), 47–63 (2003)
17. Juels, A., Catalano, D., Jakobsson, M.: Coercion-resistant electronic elections. In: *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, pp. 61–70. ACM (2005)
18. Katz, J., Myers, S., Ostrovsky, R.: Cryptographic counters and applications to electronic voting. In: Pfitzmann, B. (ed.) *EUROCRYPT 2001*. LNCS, vol. 2045, pp. 78–92. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44987-6\\_6](https://doi.org/10.1007/3-540-44987-6_6)
19. Kiayias, A., Yung, M.: Self-tallying elections and perfect ballot secrecy. In: Naccache, D., Paillier, P. (eds.) *PKC 2002*. LNCS, vol. 2274, pp. 141–158. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-45664-3\\_10](https://doi.org/10.1007/3-540-45664-3_10)
20. Laskowski, S.J., Autry, M., Cugini, J., Killam, W., Yen, J.: Improving the usability and accessibility of voting systems and products. NIST Special Publication, 256,500 (2004)
21. Lee, B., Kim, K.: Receipt-free electronic voting scheme with a tamper-resistant randomizer. In: Lee, P.J., Lim, C.H. (eds.) *ICISC 2002*. LNCS, vol. 2587, pp. 389–406. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-36552-4\\_27](https://doi.org/10.1007/3-540-36552-4_27)
22. Li, C.T., Hwang, M.S., Lai, Y.C.: A verifiable electronic voting scheme over the internet. In: *Sixth International Conference on Information Technology: New Generations, ITNG 2009*, pp. 449–454. IEEE (2009)
23. Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Linkable ring signature with unconditional anonymity. *IEEE Trans. Knowl. Data Eng.* **26**(1), 157–165 (2014)
24. Liu, J.K., Susilo, W., Wong, D.S.: Ring signature with designated linkability. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S. (eds.) *IWSEC 2006*. LNCS, vol. 4266, pp. 104–119. Springer, Heidelberg (2006). [https://doi.org/10.1007/11908739\\_8](https://doi.org/10.1007/11908739_8)
25. Liu, J.K., Wei, V.K., Wong, D.S.: Linkable spontaneous anonymous group signature for Ad Hoc groups. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) *ACISP 2004*. LNCS, vol. 3108, pp. 325–335. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-27800-9\\_28](https://doi.org/10.1007/978-3-540-27800-9_28)

26. Liu, J.K., Wong, D.S.: Linkable ring signatures: security models and new schemes. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganà, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3481, pp. 614–623. Springer, Heidelberg (2005). [https://doi.org/10.1007/11424826\\_65](https://doi.org/10.1007/11424826_65)
27. Liu, J.K., Wong, D.S.: Enhanced security models and a generic construction approach for linkable ring signature. *Int. J. Found. Comput. Sci.* **17**(6), 1403–1422 (2006)
28. McCorry, P., Shahandashti, S.F., Hao, F.: A smart contract for boardroom voting with maximum voter privacy. *IACR Cryptology ePrint Archive* 2017, 110 (2017)
29. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pp. 116–125. ACM (2001)
30. Nsw election result could be challenged over ivote security flaw (2015). <https://www.theguardian.com/australia-news/2015/mar/23/nsw-election-result-could-be-challenged-over-ivote-security-flaw>
31. Okamoto, T.: Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) *Security Protocols* 1997. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0028157>
32. Ryan, P.Y.: Prêt à voter with paillier encryption. *Math. Comput. Model.* **48**(9), 1646–1662 (2008)
33. Tarasov, P., Tewari, H.: Internet voting using zcash. *Cryptology ePrint Archive, Report* 2017/585 (2017). <http://eprint.iacr.org/2017/585>
34. Theguardian: Why machines are bad at counting votes (2009). <https://www.theguardian.com/technology/2009/apr/30/e-voting-electronic-polling-systems>
35. Tivi voting. <https://tivi.io/>. Accessed 24 June 2017
36. Tsang, P.P., Au, M.H., Liu, J.K., Susilo, W., Wong, D.S.: A suite of non-pairing ID-based threshold ring signature schemes with different levels of anonymity (extended abstract). In: Heng, S.-H., Kurosawa, K. (eds.) *ProvSec* 2010. LNCS, vol. 6402, pp. 166–183. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-16280-0\\_11](https://doi.org/10.1007/978-3-642-16280-0_11)
37. Volkhausen, T.: Paillier cryptosystem: a mathematical introduction. In: *Seminar Public-Key Kryptographie (WS 05/06) bei Prof. Dr. J. Blömer* (2006)
38. Vukolić, M.: The quest for scalable blockchain fabric: proof-of-work vs. BFT replication. In: Camenisch, J., Kesdoğan, D. (eds.) *iNetSec* 2015. LNCS, vol. 9591, pp. 112–125. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-39028-4\\_9](https://doi.org/10.1007/978-3-319-39028-4_9)
39. Weber, S.: A coercion-resistant cryptographic voting protocol-evaluation and prototype implementation. Darmstadt University of Technology (2006). <http://www.cdc.informatik.tudarmstadt.de/reports/reports/StefanWeber.diplom.pdf>
40. Xia, Z., Schneider, S.A., Heather, J., Traoré, J.: Analysis, improvement, and simplification of prêt à voter with paillier encryption. In: *EVT 2008 Proceedings of the Conference on Electronic Voting Technology* (2008)
41. Yu, B., et al.: Platform-independent secure blockchain-based voting system. *Cryptology ePrint Archive, change me* (2018). <http://eprint.iacr.org/2018/>
42. Yuen, T.H., Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Efficient linkable and/or threshold ring signature without random oracles. *Comput. J.* **56**(4), 407–421 (2013)
43. Zhao, Z., Chan, T.-H.H.: How to vote privately using Bitcoin. In: Qing, S., Okamoto, E., Kim, K., Liu, D. (eds.) *ICICS* 2015. LNCS, vol. 9543, pp. 82–96. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-29814-6\\_8](https://doi.org/10.1007/978-3-319-29814-6_8)



# Privacy in Crowdsourcing: A Systematic Review

Abdulwhab Alkharashi<sup>1</sup>(✉) and Karen Renaud<sup>1,2</sup>(✉)

<sup>1</sup> University of Glasgow, Glasgow, Scotland  
a.alkharashi.1@research.gla.ac.uk

<sup>2</sup> Abertay University, Dundee, Scotland  
k.renaud@abertay.ac.uk

**Abstract.** The advent of crowdsourcing has brought with it multiple privacy challenges. For example, essential monitoring activities, while necessary and unavoidable, also potentially compromise contributor privacy. We conducted an extensive literature review of the research related to the privacy aspects of crowdsourcing. Our investigation revealed interesting gender differences and also differences in terms of individual perceptions. We conclude by suggesting a number of future research directions.

**Keywords:** Privacy principles · Privacy aspects · Crowdsourcing

## 1 Introduction

Crowdsourcing concatenates the words ‘crowd’ and ‘outsourcing’ to reflect platforms that facilitate the recruiting of “crowds” to undertake tasks. The crowdsourcing approach has the potential to provide organizations with access to new ideas and solutions, to engender sustained consumer engagement and opportunities. It constitutes a step change in the way many people work, hire, and market labour [13, 42].

Crowdsourced labour is not always remunerated. In particular, Wikipedia is a widely known and used crowdsourcing platform where members donate their time to contribute to a publicly-available online encyclopedia. The outcome is the most inclusive encyclopedia in the world [14] that ranks as the fifth [68] most-viewed website worldwide.

The principle of crowdsourcing is that many heads are better than one. By recruiting a large crowd, it is possible to gather ideas, benefit from a wide variety of skills, and encourage participation. The quality of content and idea generation will be superior to anything produced by a solo person or small team [74].

Crowdsourcing, in addition to its positive aspects, also renders its users vulnerable to significant privacy risks. In this paper, we use previously-proposed privacy dimensions to evaluate the effectiveness of high-level guidance for enhancing privacy [27]. These include privacy categories [57, 59], privacy principles [32, 36, 76], privacy concerns [16] and privacy enhancements [76].



The contribution of this paper is to provide an overview of existing research into crowdsourcing-related privacy concerns. Our review revealed a gender difference in terms of crowdsourcing labourers and allows us to suggest possible future research directions.

## 2 Privacy

Solove [69, p. 1] defines privacy as “*a concept in disarray. Nobody can articulate what it means. Currently, privacy is a sweeping concept, encompassing (among other things) freedom of thought, control over one’s body, solitude in one’s home, control over personal information, freedom from surveillance, protection of one’s reputation, and protection from searches and interrogations*”.

This definition informs our discussion of privacy challenges related to crowdsourcing. Computational systems have often not managed the enormous amount of data gathered by all these systems in a secure or confidential fashion. This could result in personal data being leaked and/or compromised [1]. Most of all, personal privacy could be sacrificed, and privacy, once lost, can never be regained.

In this section we outline the dimensions that informed our investigation. We will consider privacy from four distinct perspectives, and report on the interactions of these in published literature. The orthogonal dimensions are:

1. three basic **layers** of privacy derived from Patil and Kobsa [59]: *social, technical and legal*.
2. five privacy **principles** which are typically reflected in privacy legislation and regulations [32, 36, 76].
3. five privacy **concerns** experienced by people who give their personal data to others [16],
4. five privacy **enhancement** techniques that are typically applied by those who collect personal data in order to address specific individual concerns of the data owners [76].

These dimensions (depicted in Fig. 1) provide the structure we used to inform our investigation.

### (1) Privacy Layers

An extended view of a *layered framework* [57] was adapted from Patil and Kobsa [59] to allow us to analyze privacy risks from both user and service-provider perspectives.

**Normative/Legal:** this layer emphasizes laws and policies that protect the individual from the privacy-invasive practices engaged in by corporations, governments, and other individuals.

**Technical:** this layer describes measures put in place to protect personal data and to allow information owners to control access to their own information.

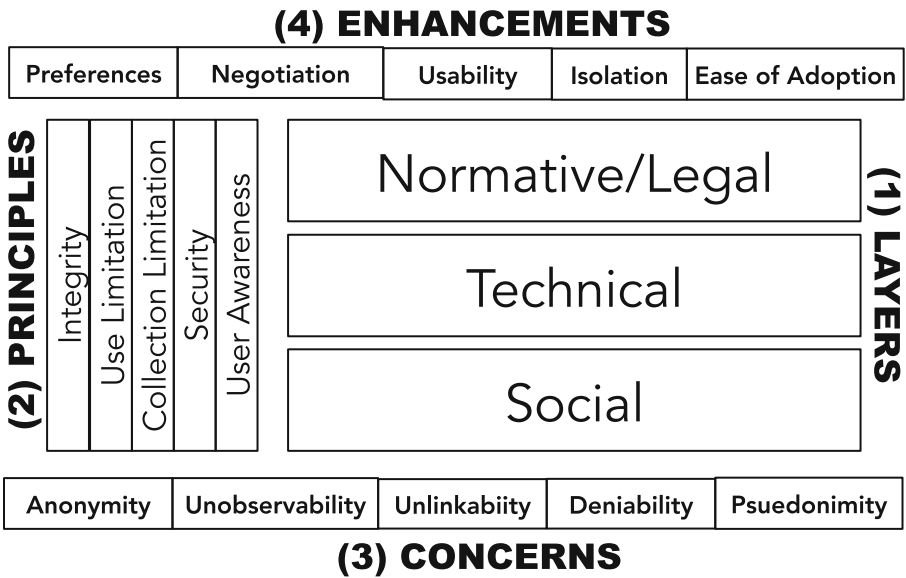


Fig. 1. Privacy dimensions

**Social:** this layer concerns the management of the boundary between people’s private and public lives. Any information people divulge happens with an understanding of the context within which it is shared, and privacy is lost when the information is shared outwith that context.

We used these layers to identify the research gaps between privacy layers from legal, social and technical perspectives to identify the factors that shape privacy behaviours among online communities.

**(2) Privacy Principles**

Privacy legislation and regulations typically instantiate fundamental privacy *principles*. We performed our analysis using a core set of privacy principles that are frequently addressed in privacy laws and regulations. The principles are briefly described below [76]:

**User Awareness.** This indicates the level of clarity and knowledge of privacy when collecting or providing data [36].

**Security.** This concerns the reasonable security safeguards used to protect personal information and defend it against risks such as loss or unauthorized access, destruction, use, modification or disclosure of data [32].

**Collection Limitation.** This concerns the limitations imposed onto the collection of personal data and the fact that any such data should be obtained by lawful and fair means [32].

**Use Limitation.** This addresses the fact that personal data should not be disclosed, made available or otherwise used for purposes other than those specified during collection [32].

**Integrity.** This addresses the need for collected personal data to be sufficiently accurate and up-to-date to support the intended purposes. A data controller should ensure that all corrections are propagated in a timely manner to all parties that have received or supplied any inaccurate data that is identified [36].

### (3) Privacy Concerns

Privacy *concerns* apply to an individual's particular views of justice within the context of privacy. People mostly have idiosyncratic views and interpretations of what data it is fair to collect, and how they rank their personal information from least to most sensitive. Campbell [16] suggests the following list of concepts that encapsulate people's concerns.

**Anonymity.** Ability to hide identity completely.

**Pseudonymity.** Appearances suggest identity hiding, but in reality the person can be identified.

**Unobservability.** Ability to use a system or website without all such accesses being logged.

**Unlinkability.** Ability for separate accesses not to be connected to each other by a data controller.

**Deniability.** Ability for users to deny some of their characteristics or actions, with the understanding that the system will not provide proof to refute such claims.

### (4) Privacy Enhancement

A number of techniques are recommended for privacy *enhancement* [76].

**User Preference.** A data controller should specify a service's privacy practices in line with each individual user's preferences.

**Negotiation.** Systems will facilitate a negotiation between a user and a website in terms of privacy standards.

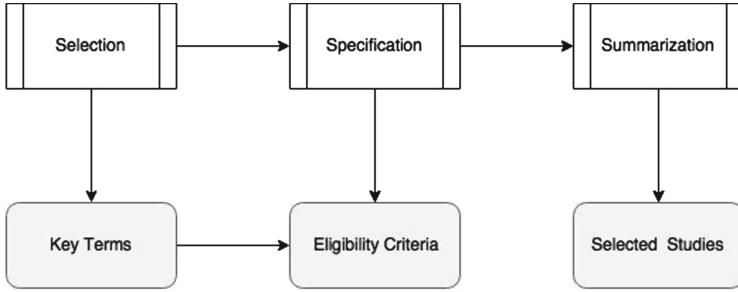
**Ease of Adoption.** This principle relates to the readiness of organizations to adopt a particular privacy protection, irrespective of the need for multiple infrastructures or technologies.

**Usability.** This principle relates to the ease with which users can convey their privacy decisions to the system.

**Isolation.** This principle relates to users being able to deny some of their characteristics or actions, and the understanding that others will not verify the veracity of their claims.

### 3 Systematic Review

In this section, we introduce a reproducible model of the systematic literature review process we conducted [37, 85]. The process, as shown in Fig. 2, describes main stages of the review process: (1) selection, (2) specification and (3) summarizing.



**Fig. 2.** Systematic review process

**Selection.** In this process, we consider two important factors during selection. Firstly, we choose a particular key terms related to the research scope including: “crowdsourcing privacy”, “crowd sourcing privacy”; or “crowdsourcing privacy” added with “social behavior”, “user awareness”, “security attacks”, “concerns”, “data protection”, “trust”, “anonymity”, “integrity”, “collection”.

Secondly, we use multiple well-known digital library databases to collect all resources from: Web of Science, Directory of Open Access Journals, Microsoft Academic, Google Scholar, ProQuest, Research Gate, science Direct, IEEE Xplore Digital Library, arXiv (Cornell library) and Wiley.

**Specification.** To manage our search results from a database source, we apply two simple rules of validation: date of publication and relevance of study. We only use papers that we can access online. We restricted our search to papers published from 2013 to 2017. Papers also should have enough information and must not be out of the research domain.

**Summarizing:** After we had filtered the papers, we recorded each paper’s reference in our summary tables, finalized our full review of findings and discussed potential research directions.

### 4 Findings

Our search results on the online database delivered a total of 635 original research papers. We retained roughly 30% (212 papers): those that specifically discussed privacy in crowdsourcing.

### Approaches Proposed by Researchers

We selected publications within four major approaches of research that correspond to crowdsourcing and privacy domains. These approaches are framework, algorithm, model and survey. Figure 3 shows that the number of published papers which presented model of privacy in crowdsourcing is research work (55%), algorithm (6%), survey (5%) and framework (34%). This also indicates that there is a research activity mostly in modeling and framework of privacy in crowdsourcing.

### Privacy Principles

The papers were examined in terms of the privacy layers, principles, concerns and enhancements, as detailed in Sect. 2 as shown in Tables 1, 2 and 3.

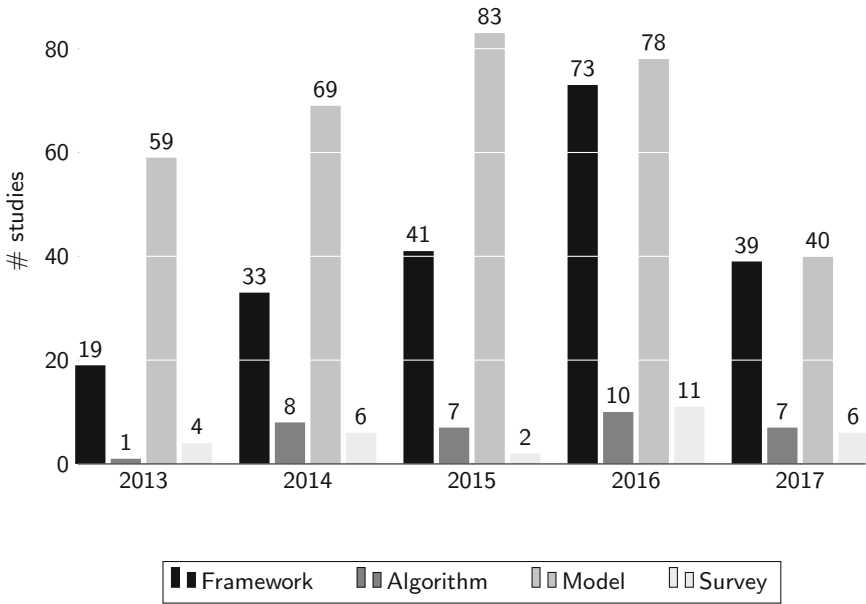


Fig. 3. The number of crowdsourcing privacy publications by research approach.

Table 1. Summary of references dealing with privacy principles in crowdsourcing.

	User awareness	Security	Collection limitation	Use limitation	Integrity
Privacy attitudes	[53, 58]	[55, 70]	[17]	[15, 17, 26]	•
Trust & evaluation	[81]	[62]	•	•	[28]
Intelligent applications	[7]	[2, 25]	•	•	[84]
Protection measures	[17, 72]	[35]	[22]	•	•
Authentication methods	[8]	[63]	•	•	[49]

**Table 2.** Summary of references related to privacy concerns in crowdsourcing.

	Anonymity	Pseudonymity	Unobservability	Unlinkability	Deniability
Privacy attitudes	[47, 50, 75]	[34]	[34]	[34]	[34]
Trust & evaluation	[43, 62]	[71]	•	[77]	•
Intelligent applications	[23, 38]	•	[40]	•	•
Protection measures	[37]	•	•	•	•
Authentication methods	[10, 60]	•	[64]	•	•

**Table 3.** Summary of references relating to privacy enhancements in crowdsourcing.

	User preference	Negotiation	Ease of adoption	Usability	Isolation
Privacy attitudes	[73]	[31]	•	[44]	[54]
Trust & evaluation	•	[56]	•	[56]	•
Intelligent applications	•	[41]	•	[20, 65]	[23]
Protection measures	•	•	•	[30]	•
Authentication methods	[4]	[78]	[12, 24]	[12]	[29]

## 5 Discussion and Limitations

Two poorly researched areas were identified during the review: (1) Gender and Privacy, and (2) Individual Privacy Perceptions. We were not specifically looking for the first but it emerged during the analysis and we considered it worth reporting.

### Gender and Privacy

Many studies report a gender gap in online knowledge sharing e.g. Wikipedia [6, 33, 51]. Researchers have shown that females are more concerned about online privacy than males [67] and it is just possible that privacy concerns are discouraging females from contributing. It would be interesting to test social psychology theory models in order gain a deeper understand of why this gap really exists and to gain insights into gender-specific privacy behaviours in this context. The main areas of gender gap revealed by reviewed literature are as follows:

*Contribution.* One study [51] shows that females contributed less to crowd-sourced platforms during 2009. Only 16.1% of the 38,497 editors who started editing on Wikipedia were female. The study examined multiple social behaviour-related hypotheses by conducting statistical experiments when extracting Wiki page data. Another study reported that both males and females made the same number of revisions, and the most active female Wikipedians make more revisions than most active male Wikipedians.

*Vandalism and Trolling.* Both acts have similar ultimate goals in the context of online discussion communities. However, these terms are used interchangeably in the research literature. Research around vandalism or trolling behaviour has tended to be essentially qualitative, commonly involving deep case-study analyses of a small number of manually-detected activities. These analyses include the

different types of trolling that have been carried out [39], the motivations behind doing so [66] and the different approaches in terms of responding to trolls [9, 19]. Another study reports on the evolution of users' anti-social behaviours from initial joining to final banishment [18].

*Measurement.* The most common approach used by researchers when trying to understand behaviour is to use a measurement tool. One study [5] examines how contributor motivations affect the type of contributions made to Wikipedia by presenting a retention rate to measure the reliability of an article written by both registered and anonymous users. Another study has presented a machine learning approach to detect vandalism edits on Wikipedia by using a logistic regression model [61].

This particular finding suggests that the gender gap is an area that would benefit from further investigation, with a particular emphasis on gender-specific privacy-preserving behaviour in crowdsourcing.

### Individual Privacy Perceptions

Understanding privacy-based perceptions can be difficult. Most studies [45, 46, 48] suggest that crowd workers have similar amounts of personal information online. Yet different cultures have differing perspectives with respect to online anonymity and privacy [11]. The impact of culture and gender on privacy in crowdsourcing environments is a rich avenue for future investigation.

### Research Limitations

Although there is a huge intersect between the Internet of Things and crowdsourcing, we restricted our review analysis to papers that applied privacy principles to the crowdsourcing context. We also included papers dealing with ubiquitous computing since these were also relevant. We restricted the date range to those published after 2014 to focus only on the most recent research. In a quickly-changing and developing research area, such as this one, research ages very quickly and old research is often no longer relevant in reflecting extant *status quo* research.

## 6 Related Research

Extensive research has been carried out related to privacy protection in ubiquitous computing [3, 52, 79]. One study [3] presents a mechanism to detect when users access private data. The idea is to provide a crowdsourced privacy recommendation engine on mobile applications to allow users to evaluate their privacy dimensions. There is an undeniable link between security and privacy and a number of research projects were conducted to reveal crowdsourcing-related security threats [80]. These systems are mostly useful for tracking and analysing the usage of sensitive data.

In public safety, crowdsourcing was used to study the information security factors when data is being collected from citizens that participate in crowdsourcing smart city project. [22]. In particular, it allows citizens to report unusual

public-safety events by using mobile phone sensing applications to detect the location of crowdsourcing participants [21].

Several survey papers were presented in the context of crowdsourcing systems in general to describe the categories and characteristics of crowdsourcing applications [83], and to judge a crowdsourcing system to introduce solutions to address the challenges of crowdsourcing systems [82].

This systematic review revealed some interesting areas for future research in crowdsourcing privacy. Both privacy principles, concerns and enhancements have been addressed, yet the idea of combining these to study the gaps in crowdsourcing privacy research is a new one.

## 7 Conclusion

Although crowdsourcing platforms seem to grow so quickly in terms of both users and data, it is evident that privacy gaps still exist and are poorly covered in the research literature. Having reviewed the latest research on privacy in crowdsourcing, we plan to proceed to study editing behaviour in crowdsourcing next.

## References

1. Ackerman, M., Darrell, T., Weitzner, D.J.: Privacy in context. *Hum. Comput. Interact.* **16**(2–4), 167–176 (2001)
2. Adamkó, A., Kollár, L.: A system model and applications for intelligent campuses. In: 18th International Conference on Intelligent Engineering Systems (INES), pp. 193–198. IEEE (2014)
3. Agarwal, Y., Hall, M.: ProtectMyPrivacy: detecting and mitigating privacy leaks on iOS devices using crowdsourcing. In: Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services, pp. 97–110. ACM (2013)
4. Alt, F., Memarovic, N., Greis, M., Henze, N.: UniDisplay – a research prototype to investigate expectations towards public display applications. In: IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 519–524. IEEE (2014)
5. Anthony, D., Smith, S.W., Williamson, T.: Reputation and reliability in collective goods the case of the online encyclopedia wikipedia. *Ration. Soc.* **21**(3), 283–306 (2009)
6. Antin, J., Yee, R., Cheshire, C., Nov, O.: Gender differences in Wikipedia editing. In: Proceedings of the 7th International Symposium on Wikis and Open Collaboration, pp. 11–14. ACM (2011)
7. Ashraf, M.S., Saha, S., Shatabda, S.: A participatory sensing framework for environment pollution monitoring and management (2017). arXiv preprint: [arXiv:1701.06429](https://arxiv.org/abs/1701.06429)
8. Assal, H., Hurtado, S., Imran, A., Chiasson, S.: What’s the deal with privacy apps? A comprehensive exploration of user perception and usability. In: Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia, pp. 25–36. ACM (2015)



9. Baker, P.: Moral panic and alternative identity construction in Usenet. *J. Comput. Mediat. Commun.* **7**(1) (2001)
10. Balicki, J., Brudlo, P., Szpryngier, P.: Crowdsourcing and volunteer computing as distributed approach for problem solving. In: *Proceedings of the 13th International Conference on Software Engineering, Parallel and Distributed Systems, SEPADS*, vol. 14, pp. 115–121 (2014)
11. Bellman, S., Johnson, E.J., Kobrin, S.J., Lohse, G.L.: International differences in information privacy concerns: a global survey of consumers. *Inf. Soc.* **20**(5), 313–324 (2004)
12. Bhagavatula, C., Ur, B., Iacovino, K., Kywe, S.M., Cranor, L.F., Savvides, M.: Biometric authentication on iPhone and Android: usability, perceptions, and influences on adoption. In: *Proceedings of USEC*, pp. 1–2 (2015)
13. Brabham, D.C.: Crowdsourcing as a model for problem solving: an introduction and cases. *Convergence* **14**(1), 75–90 (2008)
14. Bratvold, D.: What is crowdsourcing? *Daily Crowdsourc* (2017). <https://dailycrowdsourc.com/>. Accessed 18 June 2017
15. Breaux, T.D., Smullen, D., Hibshi, H.: Detecting repurposing and over-collection in multi-party privacy requirements specifications. In: *IEEE 23rd International Requirements Engineering Conference (RE)*, pp. 166–175 (2015)
16. Campbell, A.J.: Relationship marketing in consumer markets: a comparison of managerial and consumer attitudes about information privacy. *J. Interact. Mark.* **11**(3), 44–57 (1997)
17. Casanovas, P.: Open source intelligence, open social intelligence and privacy by design. In: *ECSI*, pp. 174–185 (2014)
18. Cheng, J., Danescu-Niculescu-Mizil, C., Leskovec, J.: Antisocial behavior in online discussion communities (2015). arXiv preprint: [arXiv:1504.00680](https://arxiv.org/abs/1504.00680)
19. Chesney, T., Coyne, I., Logan, B., Madden, N.: Griefing in virtual worlds: causes, casualties and coping strategies. *Inf. Syst. J.* **19**(6), 525–548 (2009)
20. Cheung, T.C.-H., Cheung, H., Mark, K.-P.: A study of the impact of a crowd wisdom online learning community platform on student learning. In: *PACIS*, p. 266 (2014)
21. Christin, D., Reinhardt, A., Kanhere, S.S., Hollick, M.: A survey on privacy in mobile participatory sensing applications. *J. Syst. Softw.* **84**(11), 1928–1946 (2011)
22. Cilliers, L., Flowerday, S.: Information security in a public safety, participatory crowdsourcing smart city project. In: *World Congress on Internet Security (World-CIS)*, pp. 36–41 (2014)
23. Clark, B.Y., Zingale, N., Logan, J.: Intelligence and information gathering through deliberative crowdsourcing. *J. Public Nonprofit Aff.* **3**(1), 55–78 (2016)
24. Clark, G.D., Lindqvist, J.: Engineering gesture-based authentication systems. *IEEE Pervasive Comput.* **14**(1), 18–25 (2015)
25. De Cunha, J.F., Galvão, T.: State of the art and future perspectives for smart support services for public transport. In: Borangiu, T., Trentesaux, D., Thomas, A. (eds.) *Service Orientation in Holonic and Multi-Agent Manufacturing and Robotics*. SCI, vol. 544, pp. 225–234. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-04735-5\\_15](https://doi.org/10.1007/978-3-319-04735-5_15)
26. Demchak, C.C., Fenstermacher, K.D.: Institutionalizing behavior-based privacy. *Adm. Soc.* **41**(7), 783–814 (2009)
27. Dinev, T., Hart, P.: Internet privacy concerns and their antecedents-measurement validity and a regression model. *Behav. Inf. Technol.* **23**(6), 413–422 (2004)

28. Dragos, V., Rein, K.: What's in a message? Exploring dimensions of trust in reported information. In: 19th International Conference on Information Fusion (FUSION), pp. 2125–2132. IEEE (2016)
29. Esnaashari, E.: Users' decisions about the security of mobile applications. Master's thesis, Aachen University (2014)
30. Fenton, N.: Effective Bayesian modelling with knowledge before data (2014). Accessed 29 Apr 2015
31. Fogues, R.L., Murukannaiah, P.K., Such, J.M., Singh, M.P.: Sharing policies in multiuser privacy scenarios: incorporating context, preferences, and arguments in decision making. *ACM Trans. Comput. Hum. Interact. (TOCHI)* **24**(1), 5 (2017)
32. Organisation for Economic Co-operation and Development. Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. OECD (1981)
33. Forte, A., Andalibi, N., Greenstadt, R.: Privacy, anonymity, and perceived risk in open collaboration: a study of tor users and wikipedians. In: CSCW, pp. 1800–1811 (2017)
34. Friedman, A., Knijnenburg, B.P., Vanhecke, K., Martens, L., Berkovsky, S.: Privacy aspects of recommender systems. In: Ricci, F., Rokach, L., Shapira, B. (eds.) *Recommender Systems Handbook*, pp. 649–688. Springer, Boston, MA (2015). [https://doi.org/10.1007/978-1-4899-7637-6\\_19](https://doi.org/10.1007/978-1-4899-7637-6_19)
35. Fu, C., Shaobin, Z., Guangjun, S., Mengyuan, G.: Crowdsourcing leakage of personally identifiable information via sina microblog. In: Hsu, R.C.-H., Wang, S. (eds.) *IOV 2014. LNCS*, vol. 8662, pp. 262–271. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-11167-4\\_26](https://doi.org/10.1007/978-3-319-11167-4_26)
36. Galinsky, E.: The study of children in family child care and relative care-key findings and policy recommendations. *Young Child.* **50**(1), 58–61 (1994)
37. Gander, M., Sauerwein, C., Breu, R.: Assessing real-time malware threats. In: 2015 IEEE International Conference on Software Quality, Reliability and Security-Companion (QRS-C), pp. 6–13 (2015)
38. Guzdial, M., Harrison, B., Li, B., Riedl, M.: Crowdsourcing open interactive narrative. In: *Foundations of Digital Games Conference*, Pacific Grove, CA, 22–25 June 2015
39. Hardaker, C.: Trolling in asynchronous computer-mediated communication: from user discussions to academic definitions. *J. Politeness Res.* **6**(2), 215–242 (2010)
40. Holmgård, C., Liapis, A., Togelius, J., Yannakakis, G.N.: Generative agents for player decision modeling in games. In: *Foundations of Digital Games Conference*, Ft. Lauderdale, FL (2014)
41. Holtgrewe, U.: New new technologies: the future and the present of work in information and communication technology. *New Technol. Work Employ.* **29**(1), 9–24 (2014)
42. Howe, J.: The rise of crowdsourcing. *Wired Mag.* **14**(6), 1–4 (2006)
43. Iren, D., Bilgen, S.: Cost of quality in crowdsourcing. *Hum. Comput.* **1**(2), 283–314 (2014)
44. Ismail, Q., Ahmed, T., Kapadia, A., Reiter, M.K.: Crowdsourced exploration of security configurations. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 467–476. ACM (2015)
45. Jensen, C., Potts, C.: Privacy policies as decision-making tools: an evaluation of online privacy notices. In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 471–478. ACM (2004)
46. Jensen, C., Potts, C., Jensen, C.: Privacy practices of internet users: self-reports versus observed behavior. *Int. J. Hum. Comput. Stud.* **63**(1), 203–227 (2005)

47. Kang, R., Brown, S., Dabbish, L., Kiesler, S.: Privacy attitudes of mechanical Turk workers and the US public. In: Symposium on Usable Privacy and Security (SOUPS) (2014)
48. Kang, R., Brown, S., Kiesler, S.: Why do people seek anonymity on the internet?: Informing policy and design. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2657–2666. ACM (2013)
49. Kishi, G.T., McLean, J.G., Pickover, C.A., Winarski, D.J.: Virtual avatar authentication. US Patent 9,641,507, 2 May 2017
50. Kobsa, A., Knijnenburg, B.P., Livshits, B.: Let's do it at my place instead?: Attitudinal and behavioral study of privacy in client-side personalization. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 81–90. ACM (2014)
51. Lam, S.T.K., Uduwage, A., Dong, Z., Sen, S., Musicant, D.R., Terveen, L., Riedl, J.: Wp: clubhouse?: An exploration of wikipedia's gender imbalance. In: Proceedings of the 7th International Symposium on Wikis and Open Collaboration, pp. 1–10. ACM (2011)
52. Lin, J., Amini, S., Hong, J.I., Sadeh, N., Lindqvist, J., Zhang, J.: Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, pp. 501–510. ACM (2012)
53. Malandrino, D., Petta, A., Scarano, V., Serra, L., Spinelli, R., Krishnamurthy, B.: Privacy awareness about information leakage: who knows what about me? In: Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society, pp. 279–284. ACM (2013)
54. Minkus, T., Memon, N.: Leveraging personalization to facilitate privacy (2014). <https://ssrn.com/abstract=2448026> or <http://dx.doi.org/10.2139/ssrn.2448026>. Accessed 27 June 2018
55. Mitropoulos, D., Spinellis, D.: Securing e-voting against MITM attacks. In: 13th Panhellenic Conference on Informatics, Corfu Island (2009)
56. Monticolo, D., Chang, T.K., Lahoud, I.: An agent based approach to annotate ideas during creativity challenges in an engineering school of innovation. In: Proceedings from the First International Workshop on Educational Knowledge Management (EKM 2014), vol. 104, pp. 11–18. Linköping University Electronic Press, Linköping, 24 November 2014
57. Netter, M., Herbst, S., Pernul, G.: Interdisciplinary impact analysis of privacy in social networks. In: Altshuler, Y., Elovici, Y., Cremers, A., Aharony, N., Pentland, A. (eds.) *Security and Privacy in Social Networks*, pp. 7–26. Springer, New York (2013). [https://doi.org/10.1007/978-1-4614-4139-7\\_2](https://doi.org/10.1007/978-1-4614-4139-7_2)
58. Pandita, R., Xiao, X., Yang, W., Enck, W., Xie, T.: WHYPER: towards automating risk assessment of mobile applications. *USENIX Secur.* **13**(20) (2013)
59. Patil, S., Kobsa, A.: Privacy considerations in awareness systems: designing with privacy in mind. In: Markopoulos, P., De Ruyter, B., Mackay, W. (eds.) *Awareness Systems*. Human-Computer Interaction Series, pp. 187–206. Springer, London (2009). [https://doi.org/10.1007/978-1-84882-477-5\\_8](https://doi.org/10.1007/978-1-84882-477-5_8)
60. Petsas, T., Tsiirantonakis, G., Athanasopoulos, E., Ioannidis, S.: Two-factor authentication: is the world ready?: Quantifying 2FA adoption. In: Proceedings of the Eighth European Workshop on System Security, p. 4. ACM (2015)
61. Potthast, M., Stein, B., Gerling, R.: Automatic vandalism detection in Wikipedia. In: Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., White, R.W. (eds.) *ECIR 2008*. LNCS, vol. 4956, pp. 663–668. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-78646-7\\_75](https://doi.org/10.1007/978-3-540-78646-7_75)

62. Ren, J., Zhang, Y., Zhang, K., Shen, X.: Exploiting mobile crowdsourcing for pervasive cloud services: challenges and solutions. *IEEE Commun. Mag.* **53**(3), 98–105 (2015)
63. Rodrigues, J.G.P., Aguiar, A., Barros, J.: Sensemycity: crowdsourcing an urban sensor (2014). arXiv preprint: [arXiv:1412.2070](https://arxiv.org/abs/1412.2070)
64. Rosoff, H., Cui, J., John, R.S.: Behavioral experiments exploring victims' response to cyber-based financial fraud and identity theft scenario simulations. In: *SOUPS*, pp. 175–186 (2014)
65. Sárkány, A., et al.: Maintain and improve mental health by smart virtual reality serious games. In: Serino, S., Matic, A., Giakoumis, D., Lopez, G., Cipresso, P. (eds.) *MindCare 2015*. CCIS, vol. 604, pp. 220–229. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-32270-4\\_22](https://doi.org/10.1007/978-3-319-32270-4_22)
66. Shachaf, P., Hara, N.: Beyond vandalism: Wikipedia trolls. *J. Inf. Sci.* **36**(3), 357–370 (2010)
67. Sheehan, K.B.: An investigation of gender differences in on-line privacy concerns and resultant behaviors. *J. Interact. Mark.* **13**(4), 24–38 (1999)
68. SimilarWeb. Top websites ranking (2018). <https://www.similarweb.com/top-websites>. Accessed 28 June 2018
69. Solove, D.J.: *Understanding Privacy*. Harvard University Press, Cambridge (2008)
70. Sommerard, R., Rouvoy, R.: Towards privacy-preserving data dissemination in crowd-sensing middleware platform. In: 11èmes journées francophones Mobilité et Ubiquité (UbiMob 2016), p. 6 (2016)
71. Hussain, R., Nawaz, W., Lee, J.Y., Son, J., Seo, J.T.: A hybrid trust management framework for vehicular social networks. In: Nguyen, H.T.T., Snasel, V. (eds.) *CSoNet 2016*. LNCS, vol. 9795, pp. 214–225. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-42345-6\\_19](https://doi.org/10.1007/978-3-319-42345-6_19)
72. Spinelli, R.: The value of privacy: concerns, attitudes, behaviors online, and information protection measures (2015). <http://elea.unisa.it/handle/10556/1920>
73. Stevenson, D.M., Pasek, J.: Privacy concern, trust, and desire for content personalization. In: *TPRC 43: The 43rd Research Conference on Communication, Information and Internet Policy Paper* (2015)
74. Surowiecki, J.: *The Wisdom of Crowds: Why the Many Are Smarter Than the Few*. Abacus (2005)
75. Volkamer, M., Renaud, K., Kulyk, O., Emeröz, S.: A socio-technical investigation into smartphone security. In: Foresti, S. (ed.) *STM 2015*. LNCS, vol. 9331, pp. 265–273. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-24858-5\\_17](https://doi.org/10.1007/978-3-319-24858-5_17)
76. Wang, Y.: Privacy-enhancing technologies. In: *Handbook of Research on Social and Organizational Liabilities in Information Security*, pp. 203–227. IGI Global (2009)
77. Wang, Y., Jia, X., Jin, Q., Ma, J.: Mobile crowdsourcing: framework, challenges, and solutions. *Concurr. Comput. Pract. Exp.* **29**(3), e3789 (2017)
78. Wolf, F., Kuber, R., Aviv, A.J.: Preliminary findings from an exploratory qualitative study of security-conscious users of mobile authentication. In: *Twelfth Symposium on Usable Privacy and Security (SOUPS)*. USENIX Association (2016)
79. Yang, D., Xue, G., Fang, X., Tang, J.: Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing. In: *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, pp. 173–184. ACM (2012)
80. Yang, K., Zhang, K., Ren, J., Shen, X.: Security and privacy in mobile crowdsourcing networks: challenges and opportunities. *IEEE Commun. Mag.* **53**(8), 75–81 (2015)

81. Ye, B., Wang, Y., Liu, L.: Crowd trust: a context-aware trust model for worker selection in crowdsourcing environments. In: 2015 IEEE International Conference on Web Services (ICWS), pp. 121–128. IEEE (2015)
82. Yin, X., Liu, W., Wang, Y., Yang, C., Lu, L.: What? How? Where? A survey of crowdsourcing. In: Li, S., Jin, Q., Jiang, X., Park, J.J.J.H. (eds.) *Frontier and Future Development of Information Technology in Medicine and Education*. LNEE, vol. 269, pp. 221–232. Springer, Dordrecht (2014). [https://doi.org/10.1007/978-94-007-7618-0\\_22](https://doi.org/10.1007/978-94-007-7618-0_22)
83. Yuen, M.-C., King, I., Leung, K.-S.: A survey of crowdsourcing systems. In: IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third International Conference on Social Computing (SocialCom), pp. 766–773 (2011)
84. Zogaj, S., Bretschneider, U., Leimeister, J.M.: Managing crowdsourced software testing: a case study based insight on the challenges of a crowdsourcing intermediary. *J. Bus. Econ.* **84**(3), 375–405 (2014)
85. Zurynski, Y.: Writing a systematic literature review: resources for students and trainees. Australian Paediatric Surveillance Unit (2014). <http://www.apsu.org.au>. Accessed 18 June 2017

# **Advanced Signatures**



# Anonymous yet Traceable Strong Designated Verifier Signature

Veronika Kuchta<sup>1</sup>, Rajeev Anand Sahu<sup>2</sup>(✉), Vishal Saraswat<sup>3</sup>,  
Gaurav Sharma<sup>2</sup>, Neetu Sharma<sup>4</sup>, and Olivier Markowitch<sup>2</sup>

<sup>1</sup> Monash University, Clayton, Australia

<sup>2</sup> Université Libre de Bruxelles, Brussels, Belgium

rajeev.sahu@ulb.ac.be

<sup>3</sup> Indian Institute of Technology Jammu, Jammu, India

<sup>4</sup> PRS University, Raipur, Raipur, India

**Abstract.** In many privacy-preserving protocols, protection of the user's identity, called anonymity, is a desirable feature. Another issue is that, if a signed document is leaked then anyone can be convinced of the authenticated data, which is strictly not allowed for sensitive data, instead the authentication only by a designated receiver is recommended. There are many scenarios in real life, for example e-auction, where both the functionalities— anonymity and designated verification are required simultaneously. For such an objective, in this paper we introduce a compact scheme of identity-based strong designated verifier group signature (ID-SDVGS) by combining the good features of strong designated verifier signature and group signature in ID-based setting. This scheme provides anonymity to the signer of a designated verifier signature with the feature of the revocation of signer's identity in case of misuse or dispute. Moreover, our scheme fulfils all the security properties of the individual components. We have obtained an ID-based instantiation of the generic group signature given by Bellare et al. in Eurocrypt 2003, and have proposed our scheme on that framework. To the best of our knowledge, this is the first construction of ID-SDVGS.

## 1 Introduction

There are numerous cryptographic protocols made up by combination of different primitives to achieve desired features for required applications. But it is important that a scheme designed for an optimal solution should not render additional security breaches. This work attempts to construct a cryptographic solution for a situation where authentication is desired only by the authorized receiver while protecting identity of the sender. In cryptography, anonymous signatures offer the anonymity of signer, while the designated verifier signatures (DVS) offer authorised verification with the property of *non-transferability* of the verification. Achieving signer's anonymity in a DVS is crucial in many application. Unfortunately, this issue has not been widely addressed and this functionality (anonymity) has not been achieved yet for such a signature (DVS) on identity

(ID)-based setting with achieving all the security properties. In this paper, we come up with a secure cryptographic construction of ID-based strong designated verifier signature which provides anonymity to the signer.

Designated verifier signatures [15] belong to special class of digital signature, which enables the signer to sign a document for an authorised recipient. In such signature, the verifier can validate the authenticity of the signature without being able to transfer the conviction to a third party. Group signature [7] is a candidate tool for construction of an anonymous signature with the property of revocation of signer's identity when required. In a group signature scheme any member of the group can sign on behalf of the group and the signature can be verified by a common public key of the group. There is a group manager who holds an opening key by which he can *open* the identity of the signer, for a given signature. However, there are situations where both the properties i.e. authorised verification and signer's anonymity are required together. In this paper we present an ID-based strong designated verifier group signature (ID-SDVGS) scheme, which fulfils both the above properties in a single compact construction.

## 1.1 Related Work

**Strong Designated Verifier Signature.** The idea of designated verifier signature (DVS) was first introduced in 1996 by Jakobsson et al. [15]. In the DVS schemes the property of strongness was first achieved by Saeednia et al. [23]. The first ID-based strong designated verifier signatures (ID-SDVS) scheme is due to Susilo et al. [24] which can be viewed as an ID-based variant of [23]. Lipmaa et al. [19] discussed an issue of *delegability* towards the security of SDVS. To address the issue, Zhang et al. [25] proposed a non-delegatable ID-SDVS scheme. But Kang et al. [16] observed a security flaw in the strongness property of [25] and proposed another scheme with security guarantees, however, it was observed in [18] that the signature in [16] is universally forgeable. Another variant of ID-SDVS [17] was presented to address the issue in the short ID-SDVS of [14], but they did not address various security properties of an SDVS, moreover their scheme has been examined to be universally forgeable in [10].

**Group Signature.** The idea of group signature is due to Chaum and van Heyst [7]. In 1997, Park et al. [22] presented the first ID-based group signature scheme. In 2000, Ateniese et al. [1] proposed first practical and provably secure coalition-resistant construction for group signature scheme based on knowledge proof signature. In 2003, Bellare et al. [2] analysed security properties of group signature and proposed a generic construction of group signature. They captured important properties including anonymity and traceability in their BMW model of security in this paper. Bellare et al. [3] later addressed the securities also for the dynamic group structure. Motivated by the BMW model [2] and its variant [5], Boyen and Waters [6] proposed compact group signature in the standard model in bilinear groups by combining provably secure hierarchical signature and the



Non-Interactive Zero Knowledge (NIZK) proof [13]. In the existing literature, the schemes [8, 9, 11] are similar in many contexts to the presented work. However, in [8] the anonymity has been realised by the ring signature which has no provision of identity revocation of the actual signer, also we have addressed more security properties like strongness with compare to [8]. In contrast to the schemes [9, 11], ours is on the ID-based setting which meets all the requirements for an effective enterprise key management system. Moreover our scheme qualifies more security properties like unverifiability, non-transferability, strongness etc. Though the property ‘non-transferability’ is defined in [11], we concretely show that our scheme actually achieves it.

## 1.2 Contribution

Though there are signature schemes in isolation like ID-based signature, group signature, strong designated verifier signature etc., yet for many applications, there is need of a compact single signature which can address properties of all these signatures in one algorithm. We provide a rigorous construction of *ID-based strong designated verifier group signature* to achieve it. Security of our construction relies on the standard assumptions, the decisional bilinear Diffie-Hellman (DBDH) assumption and the decisional linear (DLIN) assumption. The proposed scheme is suitable for the cloud-based electronic health record (EHR) where patients require access to all their medical records in a fine-grained secure way, as discussed in [9]. Our scheme also enjoys application in biometric authentication and identity-management as mentioned in [11].

## 1.3 Outline of the Paper

In Sect. 2, we introduce some related mathematical definitions, problems and assumptions. In Sect. 3, we present the formal definition of an ID-based strong designated verifier group signature scheme and a formal security model for it. Our proposed scheme is presented in Sect. 4. Lastly in Sect. 6 we briefly conclude the outcomes of the paper presented in Sect. 4. In Sect. 5 we analyze the security of the proposed scheme.

## 2 Preliminaries

In this section, we introduce the notations used in the paper, some relevant definitions, computational problems and hardness assumptions.

A probabilistic polynomial time (PPT) algorithm is a probabilistic/randomized algorithm that runs in time polynomial in the length of input. We denote by  $y \leftarrow A(x)$  the operation of running a randomized or deterministic algorithm  $A$  with input  $x$  and storing the output to the variable  $y$ . If  $X$  is a set, then  $v \xleftarrow{\$} X$  denotes the operation of choosing an element  $v$  of  $X$  according to the uniform random distribution on  $X$ . We say that a given function  $f : N \rightarrow [0, 1]$  is *negligible in  $n$*  if  $f(n) < 1/p(n)$  for any polynomial  $p$  for sufficiently large  $n$  [20]. For a group  $G$  and  $g \in G$ , we write  $G = \langle g \rangle$  if  $g$  is a generator of  $G$ .

**Definition 1 (Bilinear Map).** Let  $G_1$  be an additive cyclic group with generator  $P$  and  $G_2$  be a multiplicative cyclic group. Let both the groups are of the same prime order  $q$ . Then a map  $e : G_1 \times G_1 \rightarrow G_2$  is called a *cryptographic bilinear map* if it satisfies the following properties.

**Bilinearity:** For all  $a, b \in \mathbb{Z}_q^*$ ,  $e(aP, bP) = e(P, P)^{ab}$ , or equivalently, for all  $Q, R, S \in G_1$ ,  $e(Q+R, S) = e(Q, S)e(R, S)$  and  $e(Q, R+S) = e(Q, R)e(Q, S)$ .

**Non-Degeneracy:** There exists  $Q, R \in G_1$  such that  $e(Q, R) \neq 1$ . Note that since  $G_1$  and  $G_2$  are groups of prime order, this condition is equivalent to the condition  $g := e(P, P) \neq 1$ , which again is equivalent to the condition that  $g := e(P, P)$  is a generator of  $G_2$ .

**Computability:** There exists an efficient algorithm (viz. Miller’s algorithm [21]) to compute  $e(Q, R) \in G_2$  for all  $Q, R \in G_1$ .

**Definition 2 (Bilinear Diffie-Hellman Problem).** Given a security parameter  $\lambda$ , let  $\langle q, e : G_1 \times G_1 \rightarrow G_2, P, g \rangle \leftarrow \mathfrak{B}(\lambda)$ . Let  $BDH : G_1 \times G_1 \times G_1 \rightarrow G_2$  be a map defined by  $BDH(X, Y, Z) = \omega$  where  $X = xP$ ,  $Y = yP$ ,  $Z = zP$  and  $\omega = e(P, P)^{xyz}$ .

The bilinear Diffie-Hellman problem (BDHP) is to evaluate  $BDH(X, Y, Z)$  given  $X, Y, Z \stackrel{\$}{\leftarrow} G_1$ . (Without the knowledge of  $x, y, z \in \mathbb{Z}_q$  – obtaining  $x \in \mathbb{Z}_q$ , given  $P, X \in G_1$  is solving the discrete logarithm problem (DLP)).

**Definition 3 (Decisional Bilinear Diffie-Hellman Problem).** Given a security parameter  $\lambda$ , let  $\langle q, e : G_1 \times G_1 \rightarrow G_2, P, g, X, Y, Z \rangle \leftarrow \mathfrak{C}(\lambda)$ . Let  $\omega \stackrel{\$}{\leftarrow} G_2$ . The *decisional bilinear Diffie-Hellman problem* (DBDHP) is to decide if

$$\omega = BDH(X, Y, Z).$$

That is, if  $X = xP, Y = yP, Z = zP$ , for some  $x, y, z \in \mathbb{Z}_q$ , then the DBDHP is to decide if  $\omega = e(P, P)^{xyz}$ .

**Definition 4 (Decisional Bilinear Diffie-Hellman Assumption).** Given the parameters mentioned in the above Definition 3 of DBDHP, the *decisional bilinear Diffie-Hellman assumption* (DBDHA) states that, for any PPT algorithm  $\mathcal{A}$  which attempts to solve DBDHP, its *advantage*  $\text{Adv}_{\mathfrak{D}}(\mathcal{A})$ , defined as

$$|\Pr[\mathcal{A}(q, e : G_1 \times G_1 \rightarrow G_2, P, g, X, Y, Z, BDH(X, Y, Z)) = 1] - \Pr[\mathcal{A}(q, e : G_1 \times G_1 \rightarrow G_2, P, g, X, Y, Z, \omega) = 1]|$$

is negligible in  $\lambda$ .

**Definition 5 (Decisional Linear Problem).** Given a security parameter  $\lambda$ , let the instance  $\rho = (q, G_1, G_2, P, aP, bP, arP, bsP, Y_\beta) \leftarrow \mathfrak{D}_\beta^{\text{DLIN}}(\lambda)$ . Where  $a, b, r, s \in \mathbb{F}_q$ . The *decisional linear problem* (DLINP) is to decide whether  $\beta = 0$  or 1, where  $Y_0 := (r + s)P$ , and  $Y_1 \leftarrow G_1$ . Thus the DLINP is to decide if

$$Y_\beta = Y_0 \text{ or } Y_\beta = Y_1.$$

**Definition 6 (Decisional Linear Assumption).** Given the parameters mentioned in the above Definition 5 of DLINP, the *decisional linear assumption* (DLINA) states that, for any PPT algorithm  $\mathcal{A}$  which attempts to solve DLINP, its *advantage*  $\text{Adv}_{\mathcal{A}}^{\text{DLIN}}(\lambda)$ , defined as

$$|\Pr[\mathcal{A}(Y_0, \rho) = 1 | \rho \leftarrow \mathfrak{D}_0^{\text{DLIN}}(\lambda)] - \Pr[\mathcal{A}(Y_1, \rho) = 1 | \rho \leftarrow \mathfrak{D}_1^{\text{DLIN}}(\lambda)]|$$

is negligible in  $\lambda$ .

### 2.1 Non-Interactive Zero-Knowledge Proof [12]

A Non-Interactive Zero-Knowledge (NIZK) proof [12] is a well studied system in public key cryptography. Due to page constraint we omit the details here. In the full version of this paper, a brief discussion on it has been mentioned.

## 3 Identity-Based Strong Designated Verifier Group Signature (ID-SDVGS) Scheme

We present here the formal definition of an ID-Based Strong Designated Verifier Group Signature (ID-SDVGS) scheme and formalise a security model for it. We rely on the strong one-time signature (SOTS) scheme [12]. Our scheme achieves the CMA-unforgeability i.e. secure against one-time chosen message attack, following the security notion of [12].

### 3.1 ID-SDVGS Scheme

In our ID-SDVGS scheme there is a group of  $n + 2$  members, where  $i = 1, \dots, n$  are the users with identity  $\text{ID}_i$  who can generate signatures for a fixed designated verifier  $\mathcal{V}$ , there is a certificate issuing authority (CIA) who issues certificates for the  $n$  users and assist them in joining the group. Additionally, there is a group manager (GM) who holds a secret key and can revoke the identity of the signer in case of dispute, without learning private keys of the users. Lastly, in our ID-based setting there is a private key generator (PKG) who issues keys for all the  $n + 2$  members of the group and for the designated verifier. The structure of our ID-SDVGS scheme is as follows:

1.  $params \leftarrow \text{DVGSetup}(\lambda)$ : On input security parameter  $\lambda$ , this algorithm generates the system's public parameters  $params$  and a common reference string (CRS)  $\Sigma$ . In all the algorithms from here onward,  $params$  will be considered as an implicit input.
2.  $(Q_{\text{ID}}, S_{\text{ID}}) \leftarrow \text{DVGGen}(\text{ID})$ : This is the *Key Extraction* algorithm run by the PKG. On input user's identity  $\text{ID}$ , the algorithm generates user's (public key, private key) pair  $(Q_{\text{ID}}, S_{\text{ID}})$ .

3.  $(Cert_i) \leftarrow \text{DVGJoin}((i, ID_i), S_{ID_K})$ : This algorithm is an interactive protocol between the user, and the CIA. On input credentials  $(i, ID_i)$  of the user  $i$  and private key  $S_{ID_K}$  of the CIA and  $S_{ID_K}$  by  $S_{ID_C}$  and also everywhere in the paper, this algorithm generates user's membership certificate  $Cert_i$  with respect to its credentials.
4.  $\tilde{\sigma} \leftarrow \text{DVGSig}(\text{SOTS}, S_{ID_V}, ID_V, (i, ID_i), Cert_i, ID_{GM}, \Sigma, M)$ : This is the *Signature* algorithm run by the signer. On input the SOTS [12], signer's secret key  $S_{ID_V}$ , designated verifier's identity  $ID_V$ , signer's credentials  $(i, ID_i)$ , signer's membership certificate  $Cert_i$ , group manager's identity  $ID_{GM}$ , CRS  $\Sigma$  and the message  $M$ , this probabilistic algorithm finally generates an ID-SDVGS  $\tilde{\sigma}$  on message  $M$ .
5.  $b \leftarrow \text{DVGVer}(\text{SOTS}, S_{ID_V}, Q_{ID_V}, M, \tilde{\sigma})$ : This is the *Verification* algorithm run by the designated verifier. On input the SOTS [12], verifier's secret key  $S_{ID_V}$ , the signer's public key  $Q_{ID_V}$ , the message  $M$  and the ID-SDVGS  $\tilde{\sigma}$ , this deterministic algorithm confirms whether the signature  $\tilde{\sigma}$  is valid or invalid.
6.  $\sigma' \leftarrow \text{DVGTran}(\text{SOTS}, S_{ID_V}, Q_{ID_V}, M)$ : This is the *Transcript Simulation* algorithm run by the designated verifier. On input the SOTS [12], verifier's secret key  $S_{ID_V}$ , signer's public keys  $Q_{ID_V}$  and the message  $M$  this deterministic algorithm outputs a valid ID-SDVGS  $\sigma'$ .
7.  $(i, ID_i) \leftarrow \text{DVGOpen}(\text{SOTS}, S_{ID_{GM}}, \tilde{\sigma})$ : This is the *Open* algorithm run by the group manager GM. On input SOTS, group manager's secret key  $gmsk = S_{ID_{GM}}$ , and signature  $\tilde{\sigma}$  this deterministic algorithm outputs the actual signer's credential  $(i, ID_i)$ .

### 3.2 Security Model for ID-SDVGS Scheme

An ID-SDVGS scheme must satisfy the following security properties.

1. **Correctness:** The verification algorithm takes place properly for the correctly generated signature, i.e. if a signature on a message  $M$  is correctly computed by a signer, then the designated verifier must be able to verify the correctness of the signature, on the given message.
2. **Unforgeability:** It is computationally infeasible to construct a valid ID-SDVGS signature without the knowledge of the private key of either the signer or the designated verifier.
3. **Unverifiability:** It is computationally infeasible to verify the validity of an ID-SDVGS signature without the knowledge of the private key of either the signer or the designated verifier. We define below *existential designated unverifiability against an adaptive chosen message and adaptive chosen identities attack*.

**Definition 7 (Unverifiability).** An ID-SDVGS scheme is said to be *existential designated unverifiable against adaptive chosen message and adaptive chosen identities attack* if for any security parameter  $\lambda$ , no PPT adversary

$\mathcal{A}(q_{H_i}, q_J, q_E, q_i, q_{\mathcal{V}}, \varepsilon_{\mathcal{A}}(\lambda), t)$  which runs in time  $t$  has a non-negligible advantage

$$\begin{aligned} \text{Adv}_{\text{ID-SDVGS}, \mathcal{A}}^{\text{EDV-CID2-CMA2}}(\lambda) := & \\ & |\Pr[\mathcal{A}(Q_{\text{ID}_i^*}, Q_{\text{ID}_{\mathcal{V}}^*}, m^*, \text{DVGSig}(\text{S}_{\text{ID}_i^*}, \text{D}_{\text{ID}_i^*}, \text{Q}_{\text{ID}_{\mathcal{V}}^*}, m^*)) = 1] \\ & - \Pr[\mathcal{A}(Q_{\text{ID}_i^*}, Q_{\text{ID}_{\mathcal{V}}^*}, m^*, \sigma^*) = 1]| \quad (1) \end{aligned}$$

against the challenger  $\mathcal{B}$  in the below security experiment:

1. *Setup*:  $\mathcal{B}$  generates *params* for security parameter  $\lambda$ .
  2. *Oracle Queries*:  $\mathcal{A}$  may request: up to (a)  $q_{H_i}, i \in \mathbb{N}$  hash queries on its adaptively chosen identities and messages (b)  $q_E$  key extraction queries on its adaptively chosen identities (c)  $q_J$  join queries on its adaptively chosen identities (d)  $q_i$  signature queries on its adaptively chosen messages and adaptively chosen identities (e)  $q_{\mathcal{V}}$  verification queries on signatures on its adaptively chosen messages  $m$  and adaptively chosen identities; and obtain responses for each of its query from  $\mathcal{B}$  who acts as a random oracle.
  3. *Challenge*: At some point,  $\mathcal{A}$  outputs a message  $m^*$  and identities  $\text{ID}_i^*$  of the signer and  $\text{ID}_{\mathcal{V}}^*$  of the designated verifier on which it wishes to be challenged such that  $\mathcal{A}$  has never submitted  $\text{ID}_i^*$  or  $\text{ID}_{\mathcal{V}}^*$  during the key extraction queries. The challenger  $\mathcal{B}$  responds with a “signature”  $\sigma^*$  and challenges  $\mathcal{A}$  to verify if it is valid or not.
  4. *Query Phase 2*:  $\mathcal{A}$  continues its queries as in Query Phase 1 with an additional restriction that now it cannot submit a verification query on  $\sigma^*$ .
  5. *Output*: Finally,  $\mathcal{A}$  outputs a bit  $b^*$  which is 1 if the signature is valid and 0 if invalid.
4. **Non-transferability**: Given a signature  $\sigma$  on message  $m$ , it is infeasible for any PPT adversary  $\mathcal{A}$  to decide whether  $\sigma$  was produced by the signer or by the designated verifier, even if  $\mathcal{A}$  is also given the private keys of the signer and the designated verifier. In other words, it is impossible for the designated verifier to prove (i.e. to convince) to an outsider that the signature is actually generated by the signer.

**Definition 8 (Non-transferability)**. An ID-SDVGS scheme is said to achieve non-transferability if the signature generated by the signer is computationally indistinguishable from that generated by the designated verifier, that is,

$$\begin{aligned} \tilde{\sigma} \leftarrow \text{DVGSig}(\text{SOTS}, \text{S}_{\text{ID}_i}, \text{ID}_{\mathcal{V}}, (i, \text{ID}_i), \text{Cert}_i, \text{ID}_{\text{GM}}, \Sigma, M) \\ \approx \sigma' \leftarrow \text{DVGTran}(\text{SOTS}, \text{S}_{\text{ID}_{\mathcal{V}}}, \text{Q}_{\text{ID}_i}, M). \end{aligned}$$

5. **Strongness**: Let  $\tilde{\sigma} \leftarrow \text{DVGSig}(\text{SOTS}, \text{S}_{\text{ID}_i}, \text{ID}_{\mathcal{V}}, (i, \text{ID}_i), \text{Cert}_i, \text{ID}_{\text{GM}}, \Sigma, M)$  be a signature on a message  $M$  from a signer  $i$  to a designated verifier  $\mathcal{V}$ . *Strongness* requires that  $\tilde{\sigma}$  could have been produced by any other third party  $i^*$  for some designated verifier  $\mathcal{V}^*$  other than  $\mathcal{V}$ .

**Definition 9 (Strongness).** An ID-SDVGS scheme is said to be *strong designated* if given  $\tilde{\sigma} \leftarrow \text{DVGSig}(\text{SOTS}, \text{S}_{\text{ID}_i}, \text{ID}_{\mathcal{V}}, (i, \text{ID}_i), \text{Cert}_i, \text{ID}_{\text{GM}}, \Sigma, \text{M})$ , anyone, say  $\mathcal{V}^*$ , other than the designated verifier  $\mathcal{V}$  can produce identically distributed transcripts that are indistinguishable from those of  $\tilde{\sigma}$  from someone, say  $i^*$ , except the signer  $i$ . That is,

$$\begin{aligned} \tilde{\sigma} &\leftarrow \text{DVGSig}(\text{SOTS}, \text{S}_{\text{ID}_i}, \text{ID}_{\mathcal{V}}, (i, \text{ID}_i), \text{Cert}_i, \text{ID}_{\text{GM}}, \Sigma, \text{M}) \\ &\approx \tilde{\sigma} \leftarrow \text{DVGSig}(\text{SOTS}, \text{S}_{\text{ID}_i^*}, \text{ID}_{\mathcal{V}}^*, (i^*, \text{ID}_i^*), \text{Cert}_i^*, \text{ID}_{\text{GM}}, \Sigma, \text{M}). \end{aligned}$$

6. **Anonymity:** By *anonymity* we mean that no one except the group manager should be able to determine the identity of the original signer from the dynamic group. The formal definition is provided as follows:

**Definition 10 (Anonymity).** Let  $\mathcal{A}_{ano}$  be an adversary against the anonymity of our ID-based strong designated verifier group signature scheme (ID-SDVGS). An ID-SDVGS scheme is said to be *anonymous* if for any security parameter  $\lambda$ , no probabilistic polynomial time adversary  $\mathcal{A}_{ano}(q_{H_i}, q_J, q_E, q_O, \varepsilon_{\mathcal{A}}(\lambda), t)$  which runs in time  $t$  has a non-negligible advantage

$$\begin{aligned} \text{Adv}_{\text{ID-SDVGS}, \mathcal{A}_{ano}}^{\text{ANO}}(\lambda) &= |\Pr[\text{Expt}_{\text{ID-SDVGS}, \mathcal{A}_{ano}}^{\text{ANO}-1}(\lambda) = 1] \\ &\quad - \Pr[\text{Expt}_{\text{ID-SDVGS}, \mathcal{A}_{ano}}^{\text{ANO}-0}(\lambda) = 1]| \quad (2) \end{aligned}$$

in the security below security experiment:

1. *Setup:* On input security parameter  $\lambda$ , the challenger  $\mathcal{B}$  generates the group public key  $gpk$ , the issuing key  $ik$  and the opening key  $ok$ .
2. *Oracle Queries:* The adversary  $\mathcal{A}$  may request up to (a)  $q_{H_i}, i \in \mathbb{N}$  hash queries  $q_E$  key extraction queries  $q_J$  join queries, as described in the security experiment of the unverifiability property. Here,  $\mathcal{A}$  is also allowed to pose up to  $q_{Ch_b}$  queries to the challenge oracle on input  $\text{ID}_{i_0}, \text{ID}_{i_1}$  of identities and a message  $M$  to obtain a signature of the message under the signing key of  $\text{ID}_{i_b}$  for  $b \in \{0, 1\}$ , and up to  $q_O$  open queries to the opening oracle on input a message  $M$  and a signature  $\tilde{\sigma}$  in order to obtain the output of the open algorithm.
3. *Output:* At some point,  $\mathcal{A}$  outputs a credential  $(i^*, \text{ID}_i^*)$  of the signer.
4. *Solution to DLINP:* Challenger outputs a solution of DLINP.
7. **Traceability:** *Traceability* is an underlying property of group signature schemes. The property requires that in case of malicious signature, signer's identity should be recoverable by the group manager. In other words, it means that no subgroup of members, even the whole group should be able to generate a valid signature which cannot be opened by the manager in case of misuse and cannot be traced back to the malicious signer or to a member of the coalition.
8. **Non-frameability:** By *Non-frameability* we denote the property which implies that an honest user cannot form a valid signature which can be opened by the group manager and can be traced back to another user which has not generated the signature.

## 4 Proposed Scheme

In this section, we present our proposed ID-SDVGS. As described in Sect. 3, the proposed scheme consists of the following seven algorithms:  $DVGSetup$ ,  $DVGGen$ ,  $DVGJoin$ ,  $DVGSign$ ,  $DVGVer$ ,  $DVGTran$  and  $DVGOpen$ .

**DVGSetup:** On input security parameter  $\lambda$ , this algorithm generates the system's public parameters  $params = (\lambda, G_1, P, G_2, q, e, H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8)$  where  $G_1$  is an additive cyclic group of prime order  $q$  with generator  $P$ ,  $G_2$  is a multiplicative cyclic group of prime order  $q$ ,  $e : G_1 \times G_1 \rightarrow G_2$  is a bilinear map defined in Sect. 2 and  $H_1 : \{0, 1\}^* \rightarrow G_1$ ,  $H_2 : \{0, 1\}^* \times G_1 \rightarrow \mathbb{Z}_q^*$ ,  $H_3 : G_1 \times G_1 \times G_1 \rightarrow \mathbb{Z}_q^*$ ,  $H_4 : \{0, 1\}^* \times G_1 \times G_1 \times G_2 \rightarrow \mathbb{Z}_q^*$ ,  $H_5 : G_1 \times G_1 \rightarrow \mathbb{Z}_q^*$ ,  $H_6 : \{0, 1\}^* \times G_1 \times G_1 \times G_2 \rightarrow \{0, 1\}^\lambda$ ,  $H_7 : G_2 \rightarrow \{0, 1\}^\lambda$  and  $H_8 : \mathbb{Z}_q^* \rightarrow \{0, 1\}^\lambda$  are secure cryptographic hash functions. This algorithm also generates a CRS following the NIZK construction [12] as:

- choose  $x_e, y_e \xleftarrow{\$} \mathbb{Z}_q^*$ ;
- compute  $f_e = x_e P, h_e = y_e P$ ;
- choose  $r_c, s_c \xleftarrow{\$} \mathbb{Z}_q$ ;
- compute  $c_1 = E_{f_e, h_e}(P; r_c, s_c) = (r_c f_e, s_c h_e, (r_c + s_c)P)$ ;
- run the key generation algorithm of [12] and output commitment key  $ck \leftarrow K(q, G_1, G_2, e, P)$ .

The CRS is  $\Sigma = (pk, f_e, h_e, c_1, ck)$ . The simulated ciphertext is computed by the simulator as  $c_1 = E_{f_e, h_e}(1; r_c, s_c)$ . The simulator outputs  $(\Sigma, \tau, \xi)$ , where  $\tau = (sk, r_c, s_c)$  is a trapdoor key and  $\xi = (x_e, y_e)$  is extraction key.

**DVGGen:** Everyone – the group manager (GM), the CIA, the group members  $\{i = 1, 2, \dots, n\}$  and the designated verifier  $\mathcal{V}$  submit their respective identities  $ID_{GM}$ ,  $ID_C$ ,  $ID_i$  and  $ID_{\mathcal{V}}$  to the PKG. The PKG chooses a random  $s \xleftarrow{\$} \mathbb{Z}_q$ , sets  $P_{pub} = sP$  as system's public value and keeps the master secret  $s$  confidential. Further for a user with identity  $ID \in \{0, 1\}^*$ , the PKG computes

- public key as  $Q_{ID} = H_1(ID) \in G_1$  and
- corresponding private key as  $S_{ID} = sQ_{ID} \in G_1$ .

Finally, following the above extraction PKG shares the private keys to the corresponding users via a secure channel.

**DVGJoin:** To join the group, a user provide its credential  $(i, ID_i)$  to the KIA. The KIA outputs a certificate  $Cert_i$ , for the user  $i$ , by signing its credential  $(i, ID_i)$  as following:

- chooses  $t \xleftarrow{\$} \mathbb{Z}_q^*$ ; and computes
- $T_1 = tP \in G_1$ ;  $h_2 = H_2(i || ID_i, T_1)$ ;  $T_2 = tP_{pub} + h_2 S_{ID_K}$ .

Finally, sends the  $Cert_i = (T_1, T_2)$  to the user  $i$  as its membership certificate of the group. On receiving the certificate, the member can check its authenticity by checking the equality  $e(T_2, P) = e(T_1 + h_2 Q_{ID_K}, P_{pub})$ .

**DVGSign:** Prior to sign any message for the designated verifier, the group member runs the key generation algorithm of SOTS [12] to generate the verification and private key pair  $(vk_{\text{sots}}, sk_{\text{sots}})$ , namely  $((f_s, h_s), (x_s, y_s)) \leftarrow \text{KeyGen}_{\text{sots}}(q, G_1, G_2, e, P)$  where  $vk_{\text{sots}} = (f_s, h_s) = (x_s P, y_s P)$  and  $sk_{\text{sots}} = (x_s, y_s)$ , for  $x_s, y_s \xleftarrow{\$} \mathbb{Z}_q^*$ .

Further, using the secret key  $S_{\text{ID}_i}$ , the user with identity  $\text{ID}_i$  generates a signature on verification key  $vk_{\text{sots}}$  as follows:

- chooses  $x \xleftarrow{\$} \mathbb{Z}_q^*$  and computes
- $V_1 = xP \in G_1$ ;
- $h_{(3, vk_{\text{sots}})} = H_3(vk_{\text{sots}}, V_1) = H_3(f_s, h_s, V_1)$ ;
- $V_2 = xP_{\text{pub}} + h_{3, vk_{\text{sots}}} S_{\text{ID}_i} \in G_1$ ;
- $V = e(V_2, Q_{\text{ID}_\nu}) \in G_2$ .

The signature on  $vk_{\text{sots}}$  is  $\sigma = (V_1, V_2, V)$ . The member encrypts the above signed verification key  $\sigma$  along with its credential  $(i, \text{ID}_i)$  and membership certificate  $cert_i = (T_1, T_2)$  motivated by the Boneh-Franklin CCA-secure encryption scheme [4], using group manager's public key  $Q_{\text{ID}_{\text{GM}}}$  as follows:

- chooses  $\gamma \xleftarrow{\$} \{0, 1\}^\lambda$ ,
- computes  $r_1 = H_4(\gamma || i || \text{ID}_i, \sigma)$  and  $r_2 = H_5(T_1, T_2)$ ,  $h_\sigma = H_6(i || \text{ID}_i, \sigma)$ .
- Computes the ciphertext  $Ct$  as the following tuple:

$$\begin{aligned} Ct &= \langle A, B, C, D, E \rangle \\ &= \langle r_1 P, r_2 P, h_\sigma \oplus H_7(g_{\text{ID}_{\text{GM}}}^{r_1}), H_8(r_2) \oplus \gamma, \gamma \oplus H_7(g_{\text{ID}_{\text{GM}}}^{r_2}) \rangle \end{aligned}$$

where  $g_{\text{ID}_{\text{GM}}} = e(Q_{\text{ID}_{\text{GM}}}, P_{\text{pub}}) \in G_2$ . Furthermore, the user provides a proof of satisfiability of pairing product equations and the statement that  $(i || \text{ID}_i, cert_i, \sigma)$  is a plaintext of  $Ct$  corresponding to the technique in [12], such that

$$\pi \leftarrow P(\Sigma, S_{gs}(gpk, vk_{\text{sots}}, Ct), W_{gs}(gpk, vk_{\text{sots}}, pk_i, cert_i, \sigma, R))$$

where  $S_{gs}$  is a set of pairing equations which are used to verify the group signature (as used in the DVGVer below),  $W_{gs}$  is the witness of the NIZK proof and  $R = (r_1, r_2)$  is randomness used in the encryption.

Finally the member forms a strong one-time signature (SOTS)  $\sigma_{\text{sots}}$  on message  $M$ , ciphertext  $Ct$ , proof  $\pi$  and the key  $vk_{\text{sots}}$  using SOTS' signing key  $sk_{\text{sots}}$ . According to the signing algorithm of SOTS from [12], the signature is formed as follows: Choose  $r \xleftarrow{\$} \mathbb{Z}_q^*$  and compute

$$\sigma_{\text{sots}} = (r, s) = (r, (x_s(r_s - r) + y_s s_s - H(M || vk_{\text{sots}} || Ct || \pi)) / y_s).$$

where 'H' is some suitable hash function. The final signature is  $\tilde{\sigma} = (Ct, \pi, \sigma_{\text{sots}}, vk_{\text{sots}})$ .



DVGVer: The verification proceeds in two steps.

1. Firstly, receiver of the signature  $\tilde{\sigma}$  runs the verification algorithm of SOTS-scheme from [12] such that

$$c_s = H(M || vk_{\text{sots}} || Ct || \pi) P + r f_s + s h_s$$

2. Secondly, he runs the verification part of the NIZK proof

$$V(\Sigma, S_{g_s}(gpk, vk_{\text{sots}}, Ct)).$$

That is, he checks the satisfiability of the following bilinear equations:

$$\begin{aligned} e(T_2, P) &= e(T_1 + h_2 Q_{\text{ID}_K}, P_{\text{pub}}), \\ V &= e(V_1 + h_{(3, vk_{\text{sots}})} Q_{\text{ID}_i}, S_{\text{ID}_V}) \end{aligned}$$

where the first equation satisfies the witness represented by the certificate and the second equation validates the signature  $\sigma = (V_1, V_2, V)$  of the  $i$ -th member of the group on the verification key  $vk_{\text{sots}}$ , for designated verifier  $\mathcal{V}$ . The correctness of the equations are described in Sect. 5. Finally, satisfying all the above proofs and equalities, the designated verifier validates truthfulness of the signature  $\tilde{\sigma}$ .

DVGTran: It can be evidenced that upon receiving a signature from the group member  $i$  possessing identity  $\text{ID}_i$ , the designated verifier  $\mathcal{V}$  can simulate the signature using its secret key by following the signature algorithm of the scheme. It is sufficient here to show that a designated verifier can generate an identical signature on  $vk_{\text{sots}}$  as follows:

- chooses  $x' \xleftarrow{\$} \mathbb{Z}_q^*$  and computes;
- $V'_1 = x' P \in G_1$ ;
- $h'_{(3, vk_{\text{sots}})} = H_3(vk_{\text{sots}}, V'_1) = H_3(f_s, h_s, V'_1)$ ;
- $V'_2 = x' P_{\text{pub}} + h'_{3, vk_{\text{sots}}} S_{\text{ID}_V} \in G_1$ ;
- $V' = e(V'_2, Q_{\text{ID}_i}) \in G_2$ ;
- the signature tuple is  $(V'_1, V'_2, V')$ .

It follows from the correctness of the scheme that the above simulation is identical to the signature generated by the user  $i$  for the verifier  $\mathcal{V}$

$$V' = e(V'_2, Q_{\text{ID}_i}) = e(V'_1 + h'_{3, vk_{\text{sots}}} Q_{\text{ID}_V}, S_{\text{ID}_i})$$

DVGOpen: On input group manager's secret key  $gmsk$ , group public key  $gpk$ , message  $M$ , signature  $\tilde{\sigma} = (Ct, \pi, \sigma_{\text{sots}}, vk_{\text{sots}})$ , the group manager verifies the signature and returns 0 if  $V(\Sigma, S_{g_s}(gpk, vk_{\text{sots}}, Ct)) = 0$ . Otherwise it decrypts  $Ct$  using his  $gmsk = S_{\text{ID}_{\text{GM}}}$  as follows:

- set

$$\begin{aligned} Ct &= \langle A, B, C, D, E \rangle \\ &= \langle r_1 P, r_2 P, h_\sigma \oplus H_7(g_{\text{ID}_{\text{GM}}}^{r_1}), H_8(r_2) \oplus \gamma, \gamma \oplus H_7(g_{\text{ID}_{\text{GM}}}^{r_2}) \rangle \end{aligned}$$

- compute  $C \oplus H_7(e(S_{\text{ID}_{\text{GM}}}, A)) = h_\sigma = H_6(i||\text{ID}_i||, \sigma)$ ,
- compute  $E \oplus H_7(e(S_{\text{ID}_{\text{GM}}}, B)) = \gamma$ ,
- compute  $D \oplus \gamma = H_8(H_5(T_1, T_2))$ ,
- set  $r_1 = H_4(\gamma||i||\text{ID}_i||, \sigma)$ ,  $r_2 = H_5(T_1^*, T_2^*)$ ,
- check  $A = r_1P$  and  $B = r_2P$ . If not, reject the ciphertext.

Note that, GM can compute  $r_1$  for those identities from the list of users, which satisfy  $h_\sigma = H_6(i||\text{ID}_i||, \sigma)$ , and select  $(T_1^*, T_2^*)$  from the list of certificates which satisfies  $D \oplus \gamma = H_8(H_5(T_1^*, T_2^*))$ . Further, the GM perform the decryption and output  $(i||\text{ID}_i||, \sigma, \text{cert}_i, \sigma_{\text{sots}})$ . Finally, he runs the verification algorithm of SOTS-scheme and output  $i$  hence the  $\text{ID}_i$ .

## 5 Analysis of the Proposed Scheme

### 5.1 Correctness of the Proposed Scheme

The correctness of the scheme follows since: if  $(T_1, T_2)$  is a correctly generated certificate on user's public key,  $(V_1, V_2, V)$  is a valid signature on the verification key  $vk_{\text{sots}}$  of the underlying strong one-time signature scheme and  $\sigma_{\text{sots}}$  is a valid signature on a message  $M$  from a signer with identity  $\text{ID}_i$  for a designated verifier with identity  $\text{ID}_v$ , it follows from the following equalities and proof:

$$\begin{aligned} e(T_2, P) &= e(tP_{\text{pub}} + h_2S_{\text{ID}_K}, P) = e(T_1 + h_2Q_{\text{ID}_K}, P_{\text{pub}}); \\ V &= e(V_2, Q_{\text{ID}_v}) = e(xP_{\text{pub}} + h_{(3, vk_{\text{sots}})}S_{\text{ID}_i}, Q_{\text{ID}_v}) \\ &= e(xP + h_{(3, vk_{\text{sots}})}Q_{\text{ID}_i}, S_{\text{ID}_v}) = e(V_1 + h_{(3, vk_{\text{sots}})}Q_{\text{ID}_i}, S_{\text{ID}_v}); \end{aligned}$$

as by the definition of  $g_{\text{ID}}$  in the signature protocol.

Further, the correctness of the SOTS signature follows from [12]. Furthermore, to achieve perfect correctness we provide completeness of our NIZK proof below, which together with the correctness of the signature scheme completes the security property of perfect correctness of our ID-SDVGS scheme.

**Theorem 1.** *The non-interactive protocol of the underlying signature schemes is a perfectly complete non-interactive zero-knowledge proof of the statement that a member certificate is a signature on user's public key.*

*Proof.* According to the proof in [12], perfect completeness of our NIZK proof follows from the NIZK proof for commitment to zero. We remember the values defined in DVGSetup algorithm: for randomly chosen  $x_e, y_e \leftarrow \mathbb{Z}_q^*$ , set  $f_e = x_eP, h_e = y_eP$ . For random values  $r_c, s_c$  a relation describing commitments to 1 is given by  $R_1 := \{c_1, r_c, s_c | c_1 = \text{Com}(1, r_c, s_c) = (r_c f_e, s_c h_e, (r_c + s_c)P = (c_{11}, c_{12}, c_{13}))\}$ . The proof is given by  $\pi_1 = r_cP$ . The verification of the proof follows on input a commitment  $c_1$  if and only if  $e(P, r_c f_e) = e(\pi_1, f_e)$  and  $e(c_{12}, P) = e(h_e, c_{13} - \pi)$ . The latter equation follows, since the left side of the equation is equal to  $e(c_1 2, P) = e(s_c h_e, P) = e(s_c y_e P, P)$  and the right side of the equation is equal to

$$e(h_e, c_{13} - \pi) = e(y_e P, (r_c + s_c)P - r_c P) = e(y_e P, s_c P) = e(s_c y_e P, P).$$

### 5.2 Unforgeability

Unforgeability of our scheme relies on the one-time CMA security of the underlying SOTS scheme. Thus, in the following we provide security proof of the remained security properties, namely - unverifiability, non-transferability, strongness, anonymity, traceability and non-frameability.

### 5.3 Unverifiability

We now prove that the proposed ID-SDVGS is strongly designated. That is, any third party other than the signer and the designated verifier, cannot verify the validity of a signature from a signer for a designated verifier with non-negligible probability. We show that if there exists a PPT adaptive chosen message and adaptive chosen identity algorithm which can verify the proposed ID-SDVGS, then there exists another PPT algorithm which can use the earlier algorithm to solve the DBDHP. In particular, we prove the following theorem:

**Theorem 2.** *Given a security parameter  $\lambda$ , if there exists a PPT adversary  $\mathcal{A}(q_{H_1}, \dots, q_{H_8}, q_E, q_J, q_i, q_{\mathcal{V}}, \varepsilon_{\mathcal{A}}(\lambda), t)$  which breaks the designated unverifiability of the proposed ID-SDVGS scheme in time  $t$  with success probability  $\varepsilon_{\mathcal{A}}(\lambda)$ , then there exists a PPT adversary  $\mathcal{B}(t', \varepsilon_{\mathcal{B}}(\lambda))$  which solves DBDHP with success probability at least*

$$\varepsilon_{\mathcal{B}}(\lambda) \geq \left(1 - \frac{1}{q^2}\right)\left(1 - \frac{1}{q^4}\right)\left(1 - \frac{2}{q_{H_1}}\right)^{q_E + q_{\mathcal{V}} + q_i} \left(1 - \frac{2}{q_{H_2}q_{H_1}q_{H_3}q_{H_4}q_{H_5}}\right)^{q_i} \\ \left(1 - \frac{2}{q_{H_3}}\right)^{q_i + q_{\mathcal{V}}} \left(\frac{2}{q_{H_2}q_{H_3}(1 - q_{H_3}q_{H_2})}\right) \left(\frac{2}{q_{H_1}(q_{H_1} - 1)}\right) \varepsilon_{\mathcal{A}}(\lambda)$$

in time at most

$$t' \leq (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{H_6} + q_{H_7} + q_{H_8} + q_E + q_J + 3q_i + q_{\mathcal{V}})S_{G_1} \\ + (q_i + q_{\mathcal{V}})P_e + q_iO_{G_1} + S_{G_1} + S_{G_2} + P_e + t$$

where  $S_{G_1}, S_{G_2}, O_{G_1}, O_{G_2}$  is the time taken for one scalar multiplication in  $G_1$  (resp.  $G_2$ ) and  $O_{G_1}$  (resp.  $O_{G_2}$ ) is the time taken for one group operation in  $G_1$  (resp.  $G_2$ ), and  $P_e$  is the time taken for one pairing computation.

*Proof.* Let for a security parameter  $\lambda$ ,  $\mathcal{B}$  is challenged to solve the DBDHP for  $\langle q, e : G_1 \times G_1 \rightarrow G_2, P, aP, bP, cP, \omega \rangle$  where  $G_1$  is an additive cyclic group of prime order  $q$  with generator  $P$ ,  $G_2$  is a multiplicative cyclic group of prime order  $q$  with generator  $e(P, P)$ , and  $e : G_1 \times G_1 \rightarrow G_2$  is a cryptographic bilinear map as described in Sect. 2 and  $\omega \stackrel{\$}{\leftarrow} G_2$ .  $a, b, c \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$  are unknown to  $\mathcal{B}$ . The goal of  $\mathcal{B}$  is to solve DBDHP by verifying if  $e(P, P)^{abc} = \omega$  using  $\mathcal{A}$ , the adversary who claims to forge our proposed ID-SDVGS scheme. In order to simulate public parameters of our ID-SDVGS scheme,  $\mathcal{B}$  sets  $Q_{ID_j} = aP$  for  $j \in \{i, \mathcal{V}, GM\}$  denoting the identity either of the group member, verifier or group manager.  $\mathcal{B}$

simulates the security game for unverifiability with  $\mathcal{A}$  by running the *Setup* and by responding all hash queries  $H_i$  ( $i = 1, 2, \dots, 8$ ), join queries, key extraction queries, signature queries and verification queries appropriately.

*Output:* After  $\mathcal{A}$  has made its queries, it finally outputs a message  $M^*$ , an identity  $ID_S^*$  of a signer and an identity  $ID_V^*$  of a designated verifier on which it wishes to be challenged.

If  $\mathcal{A}$  did not make  $H_1$ -query for the identities  $ID_S^*$  and  $ID_V^*$ , then the probability that verification equality holds is less than  $1/q^2$ . Thus, with probability greater than  $1 - 1/q^2$ , both the public keys were computed using  $H_1$ -oracle and there exist indices  $i, j \in [1, q_{H_1}]$  such that  $ID_S^* = ID_i$  and  $ID_V^* = ID_j$ . If  $\{i, j\} \in RegList$  of registered entities, then abort.

*Solution to DBDHP:* Otherwise,  $\mathcal{B}$  chooses a random  $r \xleftarrow{\$} \mathbb{Z}_q^*$  and  $T \xleftarrow{\$} G_1$ ; sets  $V_1 = xP$ ; sets  $h_{(3, vk_{sots})} = H_3(vk_{sots}, V_1) = H_3(f_s, h_s, V_1)$ ;  $V_2 = xP_{pub} + h_{(3, vk_{sots})}S_{ID_i} \in G_1$ ; sets  $V = e(V_2, Q_{ID_V}) = e(bP, cP)^r \omega^h$ ; where he sets  $V_1 = cP$  and  $s = b$ , such that  $S_{ID_i} = abP$  and challenges  $\mathcal{A}$  to verify the validity of the signature  $(V_1, V_2, V)$ . Then, the verification holds if and only if each of the following equations holds

$$e(T_2, P) = e(T_1 + h_{(3,i)}Q_{ID_K}, P_{pub}), \quad V = e(V_2, Q_{ID_i}) = e(V_1 + h_{(3, vk_{sots})}Q_{ID_V}, S_{ID_i}),$$

$$g_{ID_{GM}}^{r_1} = e(r_1P, S_{ID_{GM}}), \quad g_{ID_{GM}}^{r_2} = e(r_2P, S_{ID_{GM}}).$$

where

$$\begin{aligned} \sigma &= e(V_1 + h_{(3, vk_{sots})}Q_{ID_V}, S_{ID_i}) = e(cP + h_{3, vk_{sots}} abP, bP_{pub}) \\ &= e(rP, bP_{pub})e(haP, bP_{pub}) = e(P, bP_{pub})^r e(aP, bP_{pub})^h \\ &= e(bP, P_{pub})^r e(aP, bP_{pub})^h = e(bP, cP)^r e(aP, bcP)^h \\ &= e(bP, cP)^r (e(P, P)^{abc})^h = e(bP, cP)^r \omega^h \\ &\Rightarrow \omega^h = (e(P, P)^{abc})^h \iff \omega = e(P, P)^{abc} \end{aligned}$$

Then, from the above equation,  $\mathcal{B}$  solves the DBDHP by simply returning the response of  $\mathcal{A}$  to the strongness challenge.

*Probability Calculation:* If  $\mathcal{B}$  does not abort during the simulation then  $\mathcal{A}$ 's view is identical to its view in the real attack. The responses to  $H_1$ -,  $H_2$ -,  $H_3$ - and  $H_4$ -queries are as in the real attack, since each response is uniformly and independently distributed in  $G_1$  or in  $G_2$  and  $\mathbb{Z}_q^*$  respectively. The key extraction, signature and verification queries are answered as in the real attack.

The probability that  $\mathcal{B}$  does not abort during the simulation is

$$\left(1 - \frac{2}{q_{H_1}}\right)^{q_E + q_V + q_i} \left(1 - \frac{2}{q_{H_3}}\right)^{q_i + q_V} \left(1 - \frac{2}{q_{H_1} q_{H_2} q_{H_3} q_{H_4} q_{H_5}}\right)^{q_i}.$$

The probability that  $\mathcal{A}$  did  $H_1$ -query for the identities  $ID_S^*$  and  $ID_V^*$  is  $\left(1 - \frac{1}{q^2}\right) \left(\frac{2}{q_{H_1}(q_{H_1}-1)}\right)$ .

The probability that  $\mathcal{A}$  issued  $H_2$ -query and  $H_3$ -query for the identities  $ID_i^*, ID_K^*$  is

$$\left(1 - \frac{1}{q^4}\right) \left(\frac{2}{q_{H_2} q_{H_3} (1 - q_{H_3} q_{H_2})}\right).$$

Clearly  $\mathcal{B}$ 's advantage  $\varepsilon_{\mathcal{B}}(\lambda)$  for solving the BDHP, that is, the total probability that  $\mathcal{B}$  succeeds to solve BDHP, is the product of  $\mathcal{A}$ 's advantage  $\varepsilon_{\mathcal{A}}(\lambda)$  of forging the proposed ID-SDVGS and the above three probabilities. Hence

$$\begin{aligned} \varepsilon_{\mathcal{B}}(\lambda) \geq & \left(1 - \frac{1}{q^2}\right) \left(1 - \frac{1}{q^4}\right) \left(1 - \frac{2}{q_{H_1}}\right)^{q_E + q_{\nu} + q_i} \left(1 - \frac{2}{q_{H_2} q_{H_1} q_{H_3} q_{H_4} q_{H_5}}\right)^{q_i} \\ & \left(1 - \frac{2}{q_{H_3}}\right)^{q_i + q_{\nu}} \left(\frac{2}{q_{H_2} q_{H_3} (1 - q_{H_3} q_{H_2})}\right) \left(\frac{2}{q_{H_1} (q_{H_1} - 1)}\right) \varepsilon_{\mathcal{A}}(\lambda) \end{aligned}$$

*Time Calculation:* It can be observed that running time of the algorithm  $\mathcal{B}$  is same as that of  $\mathcal{A}$  plus time taken to respond to the hash queries, key extraction queries, join queries, signature queries and verification queries,  $q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{H_6} + q_{H_7} + q_{H_8} + q_E + q_J + q_S + q_V$ . Hence the maximum running time required by  $\mathcal{B}$  to solve the BDHP is

$$\begin{aligned} t' \leq & (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{H_6} + q_{H_7} + q_{H_8} + q_E + q_J + 3q_i + q_{\nu}) S_{G_1} \\ & + (q_i + q_{\nu}) P_e + q_i O_{G_1} + O_{G_2} + S_{G_2} + t \end{aligned}$$

as  $\mathcal{B}$  requires to compute one scalar multiplication in  $G_1$  to respond to  $H_1, H_2, H_3$  and  $H_4$  hash query, one scalar multiplication in  $G_1$  to respond to key extraction query and join query, three scalar multiplications in  $G_1$  to respond to signature query, one scalar multiplication in  $G_1$  to respond to verification query; one pairing computation to respond to signature query, one pairing computation to respond to verification query, one group operation in  $G_1$  to respond to signature query, and, one scalar multiplication in  $G_1$ , one scalar multiplication in  $G_2$  and one pairing computation to output a solution of DBDHP.

#### 5.4 Non-transferability

As described in Sect. 3, the property of non-transferability implies that the signatures simulated by the designated verifier are indistinguishable from those that he receives from the signer. In DVGTran of Sect. 4 (proposed scheme) it has been already shown that we achieve this property in our scheme.

#### 5.5 Strongness

To prove this property, we will show that if the  $i$ -th signer, with identity  $ID_i$ , of the group outputs  $V = \text{Sig}(S_{ID_i}, Q_{ID_{\nu}}, vk_{\text{sots}})$  using his private key  $S_{ID_i}$  and the public key  $Q_{ID_{\nu}}$  of the designated verifier  $\mathcal{V}$  with identity  $ID_{\nu}$  as its signature on the verification key  $vk_{\text{sots}}$  during the  $\text{DVGSign}(\text{SOTS}, Q_{ID_{\nu}}, S_{ID_i}, m)$ , then the

same value  $V$  can be generated using the private key  $S_{\text{ID}_i^*}$  of a signer with identity  $\text{ID}_i^*$  (other than the  $i$ -th signer) and the public key  $Q_{\text{ID}_\mathcal{V}^*}$  of a designated verifier with identity  $\text{ID}_\mathcal{V}^*$  (other than the verifier  $\mathcal{V}$ ). That is, we show that  $V = \text{Sig}(S_{\text{ID}_i^*}, Q_{\text{ID}_\mathcal{V}^*}, vk_{\text{sots}})$  (where  $Q_{\text{ID}_\mathcal{V}^*}$  and  $S_{\text{ID}_i^*}$  are defined as in the following) since

$$\begin{aligned}
V &= e(V_2, Q_{\text{ID}_\mathcal{V}^*}) \\
&= e(xP_{\text{pub}} + h_{(3, vk_{\text{sots}})} S_{\text{ID}_i}, tQ_{\text{ID}_\mathcal{V}^*}) \quad \text{where } Q_{\text{ID}_\mathcal{V}^*} = tQ_{\text{ID}_\mathcal{V}^*} \\
&= e(xtP_{\text{pub}} + h_{(3, vk_{\text{sots}})} tS_{\text{ID}_i}, Q_{\text{ID}_\mathcal{V}^*}) \\
&= e(xP_{\text{pub}} + x(t-1)P_{\text{pub}} + h_{(3, vk_{\text{sots}})} tS_{\text{ID}_i}, Q_{\text{ID}_\mathcal{V}^*}) \\
&= e(xP_{\text{pub}} + x(t-1)h_{(3, vk_{\text{sots}})} P'_{\text{pub}} + h_{(3, vk_{\text{sots}})} tS_{\text{ID}_i}, Q_{\text{ID}_\mathcal{V}^*}) \\
&\quad (\text{where } P'_{\text{pub}} = h_{(3, vk_{\text{sots}})}^{-1} P_{\text{pub}}) \\
&= e(xP_{\text{pub}} + h_{(3, vk_{\text{sots}})} (x(t-1)P'_{\text{pub}} + tS_{\text{ID}_i}), Q_{\text{ID}_\mathcal{V}^*}) \\
&= e(xP_{\text{pub}} + h_{(3, vk_{\text{sots}})} S_{\text{ID}_i^*}, Q_{\text{ID}_\mathcal{V}^*}) \\
&= e(xP + h_{(3, vk_{\text{sots}})} Q_{\text{ID}_i^*}, S_{\text{ID}_\mathcal{V}^*}) \\
&= e(V_1 + h_{(3, vk_{\text{sots}})} Q_{\text{ID}_i^*}, S_{\text{ID}_\mathcal{V}^*}).
\end{aligned}$$

where  $S_{\text{ID}_i^*} = x(t-1)P'_{\text{pub}} + tS_{\text{ID}_i}$

## 5.6 Anonymity

**Theorem 3 (Anonymity).** *Our ID-SDVGS described above is anonymous if DLIN assumption holds for  $G_1$ .*

*Proof.* According to the anonymity experiment in Definition 10 an adversary against anonymity of our ID-SDVGS scheme has access to the following oracles: key extraction oracle, join oracle, challenge oracle and opening oracle. Furthermore,  $\mathcal{A}_{\text{ano}}$  can issue hash queries to the four hash oracles. According to [12] we are applying a hybrid argument to our proof technique. The consequence of that is that it is sufficient to consider only the challenge oracle queries in order to prove anonymity of the scheme.

Let  $b \in \{0, 1\}$  be a bit and  $\text{ID}_{i_0}, \text{ID}_{i_1}, M$  is the input for the challenge oracle query which produces a challenge signature as  $\sigma^* \leftarrow \text{DVGSig}(\text{gpk}, \text{gsk}[\text{ib}], M)$ . Furthermore, we assume that  $\mathcal{A}_{\text{ano}}$  has access to all the opening outputs except the one on input  $(M, \sigma^*)$ . It also knows all the secret keys of the group members as well as the issuing key for the certificate generation.

We simulate first the challenge oracle on input  $\text{ID}_{i_0}$ . Following the proof technique from [12], we amend the oracle that we receive the simulation-extractor which we run in the setup algorithm in order to receive the common reference string  $\Sigma$ . The output by the simulation-extractor is  $\Sigma = (pk, f_e, h_e, c_1, \sigma)$  where  $c_1 = E_{f_e, h_e}(1; r_c, s_c)$  is the ciphertext which encrypts 1. In the challenge phase, we have to simulate the NIZK proof  $\pi$  for the CCA-ciphertext  $Ct$  which encrypts  $(i_0 || \text{ID}_{i_0}, \text{cert}_{i_0}, \sigma_{i_0})$ .

Furthermore we note that it is impossible to reuse the verification key  $vk_{\text{sots}}$  of the strong one-time signature scheme in any of the issued opening queries, since the signature can be used only once. This leads to the conclusion that the tuple  $(gpk, vk_{\text{sots}}, \tilde{\sigma})$ , where  $\tilde{\sigma} = (Ct, \pi, \sigma_{\text{sots}})$  is different from the challenge tuple during the challenge signature formation. Therefore for the extraction of the plaintext  $(pk_i, cert_i, \sigma)$  we use the knowledge extractor of the NIZK proof instead of the group master secret key  $gmsk$ . The extraction works as follows: given a proof  $\pi$ , the extraction algorithm uses extraction key  $\xi = (x_e, y_e)$  to decrypt ciphertext  $Ct$ . From the perfect soundness of the used NIZK proof we know that  $Ct$  encrypts either a satisfying assignment  $(pk_i, cert_i, \sigma)$  or it encrypts 1. Since  $Ct$  does not encrypt 1 it follows that it encrypts  $(pk_i, cert_i, \sigma)$  which implies that  $S_{g_s}$  is a satisfiable set and we get opening possible with satisfying probability.

Since we are using a CCA-secure encryption procedure motivated by the scheme in [4], we note that adversary's advantage in winning the anonymity experiment in Definition 10 is the same as if the ciphertext  $Ct$  would be encrypting 1 instead of  $(pk_i, cert_i, \sigma)$ .

The same argumentation as above we can apply to the scenario where  $\mathcal{A}_{\text{ano}}$  has access to the challenge oracle on input  $ID_{i_1}$ , where the challenge consists of the ciphertext encrypting 1 and the corresponding simulated proof  $\pi$ .

## 5.7 Traceability

Let  $\tilde{\sigma} = (Ct, \pi, \sigma_{\text{sots}}, vk_{\text{sots}})$  be a valid group signature on a message  $M$ . From the perfect-soundness of the NIZK proof we know that  $Ct$  encrypts either a satisfying assignment  $(pk^*, cert^*, \sigma_{\text{sots}}^*)$ , where the designated verification algorithm on input  $(cert^*, \sigma_{\text{sots}}^*)$  outputs 1, or it encrypts 1. After running the opening algorithm the output  $i$  corresponds to the registered challenge verification key  $vk^*$ , or it aborts if no registration of the key took place. Consequently, the non-registered verification key means that no honest signature took place on that key  $vk^*$  and the existing signature  $\sigma^*$  must be a forgery. However this is not possible due to the CMA-security of the underlying signature scheme.

## 5.8 Non-frameability

Let  $\mathcal{A}_{\text{n-fra}}$  be an adversary against non-frameability property of our ID-SDVGS scheme. The adversary outputs a valid signature  $\tilde{\sigma} = (C, \pi, \sigma_{\text{sots}}, vk_{\text{sots}})$  on a message  $M$ . Furthermore he succeeds in opening procedure and outputs  $(i || ID_i || \sigma, cert^*, \sigma_{\text{sots}}^*)$  which associates with the user  $i$ . Let  $\tilde{\sigma}'$  be the signatures generated by the group member with this revealed identity. It means that the user formed a signature  $\sigma'_{\text{sots}}$  on the verification key  $vk'_{\text{sots}}$  using  $sk'_i$ . Assuming the strong unforgeability of the underlying strong one-time signature scheme,  $\mathcal{A}_{\text{n-fra}}$  cannot reuse the key  $vk'_{\text{sots}}$ . That means  $\mathcal{A}_{\text{n-fra}}$  needed a new  $vk_{\text{sots}}$  which was never signed by the member with identity  $ID_i$ . Consequently,  $\sigma_{\text{sots}}^*$  is a forged signature on  $vk_{\text{sots}}$ , which contradicts to the CMA-unforgeability of the underlying one-time signature scheme [12].

## 6 Conclusion

To realise a compact secure cryptographic construction for the situations where signer's anonymity is desired with the designated verification, in this paper we have proposed an ID-based strong designated verifier group signature (ID-SDVGS) scheme by combining the good features of ID-based strong designated verifier signature and the group signature. The scheme is proved secure under standard security notions. More particularly, we have considered all the security properties of the ingredient signatures of the proposed compact signature. More particularly, the *unverifiability* and the *strongness* are essential security properties of a SDVS, however they have not been addressed properly in the literature. We have provided proofs for both the properties of our scheme along with other security proofs. We have realized the proposed construction by obtaining an ID-based instantiation of the generic group signature frame, given by Bellare et al. in Eurocrypt 2003. To the best of our knowledge this is the first construction of ID-based *strong* designated verifier *group* signature.

## References

1. 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). [https://doi.org/10.1007/3-540-44598-6\\_16](https://doi.org/10.1007/3-540-44598-6_16)
2. 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). [https://doi.org/10.1007/3-540-39200-9\\_38](https://doi.org/10.1007/3-540-39200-9_38)
3. 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). [https://doi.org/10.1007/978-3-540-30574-3\\_11](https://doi.org/10.1007/978-3-540-30574-3_11)
4. 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). [https://doi.org/10.1007/3-540-44647-8\\_13](https://doi.org/10.1007/3-540-44647-8_13)
5. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-28628-8\\_3](https://doi.org/10.1007/978-3-540-28628-8_3)
6. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006). [https://doi.org/10.1007/11761679\\_26](https://doi.org/10.1007/11761679_26)
7. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991). [https://doi.org/10.1007/3-540-46416-6\\_22](https://doi.org/10.1007/3-540-46416-6_22)
8. Chen, Y., Susilo, W., Mu, Y.: Identity-based anonymous designated ring signatures. In: International Conference on Wireless Communications and Mobile Computing, pp. 189–194. ACM (2006)
9. Derler, D., Krenn, S., Slamanig, D.: Signer-anonymous designated-verifier redactable signatures for cloud-based data sharing. In: Foresti, S., Persiano, G. (eds.) CANS 2016. LNCS, vol. 10052, pp. 211–227. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-48965-0\\_13](https://doi.org/10.1007/978-3-319-48965-0_13)



10. Du, H., Wen, Q.: Attack on Kang et al.'s identity-based strong designated verifier signature scheme. IACR Cryptology ePrint Archive, 2008:297 (2008)
11. Emura, K., Miyaji, A., Omote, K.: An anonymous designated verifier signature scheme with revocation: how to protect a company's reputation. In: Heng, S.-H., Kurosawa, K. (eds.) ProvSec 2010. LNCS, vol. 6402, pp. 184–198. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-16280-0\\_12](https://doi.org/10.1007/978-3-642-16280-0_12)
12. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006). [https://doi.org/10.1007/11935230\\_29](https://doi.org/10.1007/11935230_29)
13. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (2006). [https://doi.org/10.1007/11761679\\_21](https://doi.org/10.1007/11761679_21)
14. Huang, X., Susilo, W., Yi, M., Zhang, F.: Short designated verifier signature scheme and its identity-based variant. *Int. J. Netw. Secur.* **6**(1), 82–93 (2008)
15. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996). [https://doi.org/10.1007/3-540-68339-9\\_13](https://doi.org/10.1007/3-540-68339-9_13)
16. Kang, B., Boyd, C., Dawson, E.: Identity-based strong designated verifier signature schemes: attacks and new construction. *Comput. Electr. Eng.* **35**(1), 49–53 (2009)
17. Kang, B., Boyd, C., Dawson, E.D.: A novel identity-based strong designated verifier signature scheme. *J. Syst. Softw.* **82**(2), 270–273 (2009)
18. Lee, J.-S., Chang, J.K., Lee, D.H.: Forgery attacks on Kang et al.'s identity-based strong designated verifier signature scheme and its improvement with security proof. *Comput. Electr. Eng.* **36**(5), 948–954 (2010)
19. Lipmaa, H., Wang, G., Bao, F.: Designated verifier signature schemes: attacks, new security notions and a new construction. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 459–471. Springer, Heidelberg (2005). [https://doi.org/10.1007/11523468\\_38](https://doi.org/10.1007/11523468_38)
20. Mao, W.: *Modern Cryptography: Theory and Practice*. Prentice Hall Professional Technical Reference (2003)
21. Miller, V.S.: Use of elliptic curves in cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986). [https://doi.org/10.1007/3-540-39799-X\\_31](https://doi.org/10.1007/3-540-39799-X_31)
22. Park, S., Kim, S., Won, D.: Id-based group signature. *Electron. Lett.* **33**(19), 1616–1617 (1997)
23. Saeednia, S., Kremer, S., Markowitch, O.: An efficient strong designated verifier signature scheme. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 40–54. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24691-6\\_4](https://doi.org/10.1007/978-3-540-24691-6_4)
24. Susilo, W., Zhang, F., Mu, Y.: Identity-based strong designated verifier signature schemes. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 313–324. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-27800-9\\_27](https://doi.org/10.1007/978-3-540-27800-9_27)
25. Zhang, J., Mao, J.: A novel id-based designated verifier signature scheme. *Inf. Sci.* **178**(3), 766–773 (2008)



# Strongly Unforgeable Signature Resilient to Polynomially Hard-to-Invert Leakage Under Standard Assumptions

Masahito Ishizaka<sup>(✉)</sup> and Kanta Matsuura

Institute of Industrial Science, The University of Tokyo, Tokyo, Japan  
{ishimasa,kanta}@iis.u-tokyo.ac.jp

**Abstract.** A signature scheme is said to be weakly unforgeable, if it is hard to forge a signature on a message not signed before. A signature scheme is said to be strongly unforgeable, if it is hard to forge a signature on any message. In some applications, the weak unforgeability is not enough and the strong unforgeability is required, e.g., the Canetti, Halevi and Katz transformation.

Leakage-resilience is a property which guarantees that even if secret information such as the secret-key is partially leaked, the security is maintained. Some security models with leakage-resilience have been proposed. The auxiliary (input) leakage model, or hard-to-invert leakage model, proposed by Dodis et al. in STOC'09 is especially meaningful one, since the leakage caused by a function which information-theoretically reveals the secret-key, e.g., one-way permutation, is considered.

In this work, we propose a generic construction of a signature scheme strongly unforgeable and resilient to polynomially hard-to-invert leakage which can be instantiated under standard assumptions such as the decisional linear assumption. We emphasize that our signature scheme is not only the first one resilient to polynomially hard-to-invert leakage under standard assumptions, but also the first one which is strongly unforgeable and has hard-to-invert leakage-resilience.

**Keywords:** Digital signature · Strong existential unforgeability  
Leakage-resilience · Hard-to-invert leakage · Auxiliary (input) leakage

## 1 Introduction

*Strongly Unforgeable Signature.* We say that a signature scheme is weakly (existentially) unforgeable if it is hard to forge a signature on a message not signed before. We say that a signature scheme is strongly (existentially) unforgeable if it is hard to forge a signature on any message which can be a message signed before. Since most of the signature schemes generate a signature randomly, there is a gap between the weak unforgeability and strong unforgeability. Moreover, in some applications, strongly unforgeable signature scheme is required, e.g., Canetti-Halevi-Katz transformation [11].

*Leakage-Resilient Cryptography.* Leakage-resilience is a property which guarantees that even if secret information such as the secret key is partially leaked, the security is maintained. Any scheme whose security has been proven only in a security model without leakage-resilience is not guaranteed to be secure when some information about the secret information such as the secret-key are leaked. There exist some side-channel attacks which are real threats to us, e.g., cold-boot attack [19], so leakage-resilient cryptographic schemes are practically desirable.

In the security model considering leakage-resilience, a side-channel attack caused by an adversary is modeled as a polynomial time computable function  $f : \{0, 1\}^{|Secret|} \rightarrow \{0, 1\}^*$ <sup>1</sup>. The adversary is allowed to choose an arbitrary leakage function  $f$ , query it to the leakage oracle, then learn  $f(Secret)$ . If we allow the adversary to choose the identity map as  $f$ , the adversary acquires the secret key entirely and is able to break the security model with no failures. Hence, we have to impose a restriction on  $f$ . Several security models in which different restrictions are imposed on  $f$  have been proposed.

In bounded leakage (BL) model [1], the output bit-length of  $f$  is restricted. More concretely, only  $f$  satisfying  $f : \{0, 1\}^{|Secret|} \rightarrow \{0, 1\}^{l(k)}$  such that  $l(k) < k$  can be chosen<sup>2</sup>. To make the output bit-length of  $f$  unbounded, noisy leakage (NL) model [25] was invented. In NL model, only  $f : \{0, 1\}^{|Secret|} \rightarrow \{0, 1\}^*$  such that, when we observe  $f(Secret)$ , the minimum entropy of the secret key  $sk$  drops by at most  $l(k) < k$  can be chosen. Any function which information-theoretically reveals the secret key  $sk$  is excluded in each one of the two models. Thus, for instance, one-way permutation cannot be chosen in each one of the models. To remove such a restriction, auxiliary (input) leakage (AL) model [14], or hard-to-invert leakage model, was invented. In AL model, the function  $f$  must be a hard-to-invert function. More concretely, only  $f$  such that, given  $f(Secret)$ , no PPT algorithm can compute  $sk$  with a probability larger than  $\mu(k)$  can be chosen, where  $\mu(\cdot)$  is a negligible function such that  $\mu(k) > 2^{-k}$ . The larger  $\mu(k)$  is, the larger the function class of  $f$  is. AL model is a generalization of BL and NL model, thus has a larger function class. Moreover, AL model is useful in the context of the composition. There may be the case when we want to use the same pair of public key and secret key of an auxiliary leakage-resilient cryptographic scheme for multiple schemes. Their composition remains secure as long as each one of the other schemes has been proven to be secure in the standard (non-leakage-resilient) security model [12, 29].

Large number of cryptographic schemes with leakage-resilience have been proposed. For instance, public-key encryption [1, 4, 12, 13], identity-based encryption [10, 21, 23, 29], attribute-based encryption [23, 31–33], identification [2, 13], and authenticated key agreement [2, 13], have been proposed.

*Related Works.* Katz and Vaikuntanathan [22] defined that fully leakage-resilient (FLR) signature is a signature resilient to not only the direct leakage from the

<sup>1</sup>  $Secret$  denotes the secret information.  $|Secret|$  denotes the bit-length of  $Secret$ .

<sup>2</sup>  $k$  denotes the minimum entropy of the secret key  $sk$ . If the secret-key is generated uniformly at random,  $k$  is equivalent to the bit-length of the secret-key  $|sk|$ .

secret-key, but also the leakage from the randomness used to generate the secret-key and signatures in the signing oracle. FLR or non-FLR signature schemes secure in the bounded-leakage model have been proposed in [2, 8, 22, 24].

The concept of the auxiliary leakage-resilience was presented by Dodis et al. [14]. They proposed symmetric-key encryption schemes which is IND-CPA or IND-CCA secure and resilient to exponentially hard-to-invert leakage. In a subsequent work, Dodis et al. [12] started a research on public-key encryption with hard-to-invert leakage-resilience. They defined two leakage-function classes. The class  $\mathcal{H}_{\text{ow}}(\xi(\lambda))$  (resp.  $\mathcal{H}_{\text{pkow}}(\xi(\lambda))$ ) consists of every polynomial-time computable function  $f : \{0, 1\}^{|pk|+|sk|} \rightarrow \{0, 1\}^*$  such that any PPT algorithm  $\mathcal{A}$  which is given  $f(pk, sk)$  (resp.  $(pk, f(pk, sk))$ ) as input is able to guess  $sk$  correctly only with a probability smaller than  $\xi(\lambda)$ , where  $\xi(\lambda) > 2^{-k}$  is a negligible function and  $(pk, sk)$  is a randomly generated key-pair. They proved that the BHHO encryption scheme [5] and a slightly modified version of the GPV encryption scheme [16] are IND-CPA secure in the hard-to-invert leakage-resilience model w.r.t. the function class  $\mathcal{H}_{\text{ow}}(1/\mu_1(\lambda))$ , where  $\mu_1(\lambda)$  is a sub-exponential function. They also mentioned that a PKE scheme which is IND-CPA secure in the hard-to-invert leakage-resilience model w.r.t.  $\mathcal{H}_{\text{pkow}}(1/\mu_2(\lambda))$ , where  $\mu_2(\lambda)$  is a polynomial function, is given in its full paper [12].

Faust et al. firstly presented a research on digital signature with hard-to-invert leakage-resilience [15]. To construct a signature scheme secure in the AL model, there is an obstacle whom we have to overcome. It is how to prevent the adversary to choose the signing algorithm as the leakage-function, get a valid signature, then output the signature as a forged signature. Faust et al. proposed a signature scheme which is wEUF-CMA (weakly existentially unforgeability under adaptively chosen messages attack) secure in the hard-to-invert leakage-resilience model w.r.t. the function class  $\mathcal{H}_{\text{pkow}}(1/\mu_3(\lambda))$ , where  $\mu_3(\lambda)$  is an exponential function. Their solution to overcome the obstacle explained earlier is to add a ciphertext of the secret-key  $sk$  to a signature. Specifically, their signature scheme adopts the labeled public-key encryption (LPKE) as a building block, and adds a ciphertext of the secret-key to a signature. Moreover, for their signature scheme, the hardness parameter  $1/\mu_3(\lambda)$  in the leakage function class is set as  $1/\mu_3(\lambda) \ll 2^{-l_{dk}}$ , where  $l_{dk} \in \mathbb{N}$  denotes the bit-length of the decryption-key  $dk$  of the LPKE scheme. This solution effectively works. The reason is as follows. Since any PPT algorithm is able to guess the decryption-key  $dk$  correctly with probability  $2^{-l_{dk}}$ , any PPT inverter in the definition of the function class  $\mathcal{H}_{\text{pkow}}(1/\mu_3(\lambda))$  which is given a signature including a ciphertext  $C$  of the secret-key  $sk$  is able to guess  $sk$  correctly with probability  $2^{-l_{dk}} \gg 1/\mu_3(\lambda)$  by guessing the decryption-key  $dk$ , then decrypting the ciphertext  $C$  with the guessed  $dk$ . Hence, the signing algorithm is excluded from the class  $\mathcal{H}_{\text{pkow}}(1/\mu_3(\lambda))$ . By the way, they showed that their signature scheme can be instantiated under standard assumptions such as the DLIN assumption [3].

Independently of Faust et al. [15], Yuen et al. [30] also presented a research on signature secure in the AL model. To overcome the obstacle to construct a signature with auxiliary leakage resilience, Yuen et al. proposed an original

security model, which is named “selective auxiliary input model”. In the security model, the adversary is allowed to choose as the leakage-functions only functions which are independent of the public-key. They proposed a signature scheme secure in the security model. Their signature scheme is FLR and resilient to polynomially hard-to-invert leakage. Here, their definition of leakage function  $f$  being resilient to polynomially hard-to-invert leakage is as follows: any PPT algorithm which is given  $(pk, S, f(state))$  is able to guess  $sk$  correctly only with a negligible probability, where  $(pk, sk)$  is a randomly generated key-pair,  $S$  is a set of randomly generated signatures on the messages queried to the signing oracle, and  $state$  is a set of randomnesses used to generate  $sk$  and the signatures  $S$ . Their definition of leakage-function is undesirable since it depends on the signatures generated on the signing oracle.

Subsequently, Wang et al. [27] proposed a signature scheme secure in the selective auxiliary input model. Their signature scheme is FLR and resilient to polynomially hard-to-invert leakage. Their definition for a function to be resilient to polynomially hard-to-invert leakage is not the same as the one by Yuen et al. [30]. It is improved as follows: any PPT algorithm which is given  $f(sk)$  is able to identify  $sk$  only with a negligible probability. However, their scheme needs differing input obfuscator (diO), indistinguishable obfuscator (iO), and point-function obfuscator with auxiliary input (AIPO), each one of which has been constructed only under strong assumptions.

Note that each one of the signature schemes with auxiliary leakage resilience by Faust et al. Yuen et al. and Wang et al. is not strongly existentially unforgeable, but weakly existentially unforgeable.

Boneh et al. [7] proposed a method to transform a weakly unforgeable signature scheme into a strongly unforgeable one. However, their transformation can be applied to “partitioned” signatures only. In a subsequent work, Steinfeld et al. [26] proposed a method to transform “any” weakly unforgeable signature into a strongly unforgeable one. Note that each transformation by Boneh et al. and Steinfeld et al. has a common property such that each one of the public-key, secret-key and signature of the strongly unforgeable signature scheme becomes each one of the public-key, secret-key and signature of the weakly unforgeable signature whom some new elements are added to. Huang et al. [20] proposed a transformation which no new elements are added to the public-key, secret-key and signature.

Wang et al. [28] modified the transformation by Steinfeld et al. [26] to get a transformation from a signature weakly existentially unforgeable and FLR in the bounded leakage model to a strongly unforgeable one. The transformation by Steinfeld et al. utilizes two chameleon hash functions (with no leakage-resilience). In the transformation by Wang et al., one of the chameleon hash functions is assumed to satisfy a property such that any PPT algorithm cannot find a strong collision even if the algorithm is given a length-bounded information about the secret-key.

The transformation by Wang et al. needs to add some new elements to both the key-pair and signature. Huang et al. [18] modifies the transformation by

Huang et al. [20] to get a method to transform a signature weakly existentially unforgeable and FLR in the BL model into a strongly unforgeable one which no elements are added to the signature<sup>3</sup>.

*Our Results.* We propose a generic construction of signature scheme which is strongly unforgeable and resilient to polynomially hard-to-invert leakage under standard assumptions. Specifically, we give an example of instantiation under the decisional linear (DLIN) assumption [3]. Our security model is not the selective auxiliary leakage model [30], so the leakage-function can be chosen dependently on the public-key.

Our result is a desirable one because of the following two independent points. Firstly, our signature is the first one which is resilient to polynomially hard-to-invert leakage under standard assumptions. Secondly, our signature is the first one which is strongly unforgeable and has hard-to-invert leakage-resilience.

*Our Approach.* Our result is obtained by modifying the one by Faust et al. [15]. Before explaining how the modification is done, we explain the result by Faust et al. in detail.

Faust et al. proposed a generic construction of a signature scheme secure in the wEUF-CMA security model w.r.t. the function class  $\mathcal{H}_{\text{pkow}}(\xi(\lambda))$ . It consists of three building blocks. They are second pre-image resistant hash function (SPRHF), labeled PKE (LPKE) whose decryption-key  $dk$  has bit-size  $l_{dk} \in \mathbb{N}$ , and non-interactive zero-knowledge proof (NIZK) whose trapdoor  $td$  has bit-size  $l_{td} \in \mathbb{N}$ . The hardness parameter of the leakage function class is set as  $\xi(\lambda) = 2^{-(\lambda+l_{dk}+l_{td})}$ . A signature  $\sigma$  on a message  $m$  consists of a LPKE ciphertext  $c$  and a NIZK proof  $\pi$ . Concretely, the ciphertext  $c$  is a LPKE ciphertext encrypting the secret-key  $sk$  under the label  $m$ , and the proof  $\pi$  is a NIZK proof which proves that there exists a secret-key  $sk'$  such that the ciphertext  $c$  is a ciphertext of  $sk'$  on the label  $m$  and the hashed value of  $sk'$  is equivalent to the hashed value of the real secret-key  $sk$  which is included in the public-key  $pk$ .

Intuitively speaking, the security proof for the signature by Faust et al. is done as follows. By modifying the initial security game several times, we get the final game  $\text{Game}_{\text{final}}$ . In  $\text{Game}_{\text{final}}$ , for a signature  $\sigma = (c, \pi)$  on the signing oracle, the ciphertext  $c$  is generated by encrypting  $0^{|sk|}$  instead of  $sk$ , and the proof  $\pi$  is generated by using the trapdoor  $td$  instead of  $sk$ . In addition, the adversary is considered to win the game, if he successfully outputs a signature  $\sigma^* = (c^*, \pi^*)$  and a message  $m^*$  such that  $c^*$  is a valid ciphertext of  $sk^*$  on label  $m^*$ , and  $\pi^*$  is a valid proof. We prove that every PPT  $\mathcal{A}$  wins the game only with a negligible probability by a reduction to the hard-to-invert property of the leakage-function  $f \in \mathcal{H}_{\text{pkow}}(2^{-(\lambda+l_{dk}+l_{td})})$ . In the reduction, a simulator  $\mathcal{S}$  needs both  $td$  and  $dk$  to simulate  $\text{Game}_{\text{final}}$  and decrypt the ciphertext  $c^*$ . However, by the definition of the leakage function class  $\mathcal{H}_{\text{pkow}}(\cdot)$ ,  $\mathcal{S}$  is given neither  $td$  nor  $dk$ , so  $\mathcal{S}$  has to guess them, and the guess succeeds with probability  $2^{-(l_{dk}+l_{td})}$ .

<sup>3</sup> By the transformation in [18], some new elements are added to the public-key and secret-key.

By the above reason, the hardness parameter for Faust et al.’s signature scheme becomes  $2^{-(\lambda+l_{dk}+l_{td})}$ .

The above is the result by Faust et al. We modify the result with three steps.

In the first step, we generalize the second pre-image resistance (SPR) property of the SPRHF, which is one of the building blocks. Intuitively, the SPR property is a property such that no PPTA given a key-pair  $(pk, sk)$  is able to find a secret-key  $sk^*$  which is not  $sk$ , but has a hashed value equivalent to the hashed value of  $sk$  with a non-negligible probability. We generalize it to a property such that no PPTA given  $(pk, sk)$  is able to find a secret-key  $sk^*$  such that a relation holds between  $sk^*$  and  $sk$  and another relation also holds between  $sk^*$  and  $pk$  with a non-negligible probability.

The second step is to modify the definition of the leakage-function class  $\mathcal{H}_{\text{pkow}}(\cdot)$ . In the modified definition of the function class, the PPT algorithm (or inverter)  $\mathcal{A}$  is given not only the public-key of the key-pair  $(pk, sk)$ , but also some variables which are generated during generation of the key-pair and are not directly included in either  $pk$  or  $sk$ . Specifically, for our signature scheme, the variables are the decryption-key  $dk$  and the trapdoor  $td$ . If the definition of the leakage-function class is modified to such one, the simulator in the proof for  $\text{Game}_{\text{final}}$  is not forced to guess  $dk$  and  $td$  with probability  $2^{-(l_{dk}+l_{td})}$ , so the polynomially hard-to-invert leakage-resilience security is achieved. Instantiating the generic construction of the signature scheme, we can concretely generate the first signature scheme (weakly unforgeable and) resilient to polynomially hard-to-invert leakage under standard assumptions such as the DLIN assumption.

In the third step, we apply a methodology which is invented by modifying the one by Wang et al. [28] to the weakly unforgeable signature scheme in the second step, then get a strongly unforgeable one. Note that unlike Wang et al., we do not propose a generic transformation from a weakly unforgeable and resilient to hard-to-invert leakage to a strongly unforgeable one. In the transformation by Wang et al., a chameleon hash function with strong collision-resistance in the bounded leakage model (BLR-CHF) was used. We use a CHF with strong collision-resistance in the auxiliary leakage model (ALR-CHF). Moreover, the secret-key of the strongly unforgeable signature scheme obtained by the transformation by Wang et al. includes not only the “original” secret-key, i.e., the secret-key of the weakly unforgeable signature, but also the secret-key of the BLR-CHF. However, the secret-key of our strongly unforgeable signature includes the secret-key of the ALR-CHF only. By instantiating the signature scheme, we obtain a concrete construction of the first signature strongly unforgeable and resilient to polynomially hard-to-invert leakage under standard assumptions such as the DLIN assumption.

*Paper Organization.* This paper is organized as follows. In Sect. 2, we give basic notations and the syntax and the definition of security or property of labeled public-key encryption, non-interactive zero-knowledge proof, chameleon hash function, and digital signature. In Sect. 3, our generic construction of signature and its security proof are given. In Sect. 4, we show that the generic construction of signature in Sect. 3 can be instantiated under the DLIN assumption.



## 2 Preliminaries

*Notation.* For  $a, b \in \mathbb{N}$ ,  $[a, b]$  denotes  $\{x \in \mathbb{N} \mid a \leq x \leq b\}$ . For  $\lambda \in \mathbb{N}$ ,  $1^\lambda$  denotes a security parameter.  $\mathcal{G}$  is a function which takes  $1^\lambda$  as input, and randomly outputs  $(p, \mathbb{G}, g)$ , where  $p$  is a prime number whose bit-size is  $\lambda$ ,  $\mathbb{G}$  is a multiplicative cyclic group whose order is  $p$ , and  $g$  is a generator of  $\mathbb{G}$ . PPTA means probabilistic polynomial time algorithm.

### 2.1 Hardness Assumptions

*Discrete Logarithm (DL) Assumption.* For  $\lambda \in \mathbb{N}$ , let  $(p, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda)$ . DL assumption holds, if for every PPTA  $\mathcal{A}$ , the probability  $\Pr[x \leftarrow \mathcal{A}(p, \mathbb{G}, g, g^x) \mid x \xleftarrow{\text{U}} \mathbb{Z}_p]$  is negligible in  $\lambda$ .

*Decisional Linear (DLIN) Assumption [3].* For  $\lambda \in \mathbb{N}$ , let  $(p, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda)$ . DLIN assumption holds, if for every PPTA  $\mathcal{A}$ ,

$$\left| \Pr[1 \leftarrow \mathcal{A}(p, \mathbb{G}, g_1, g_2, g_3, g_1^{r_1}, g_2^{r_2}, g_3^{r_1+r_2}) \mid g_1, g_2, g_3 \xleftarrow{\text{U}} \mathbb{G}, r_1, r_2 \xleftarrow{\text{U}} \mathbb{Z}_p] - \Pr[1 \leftarrow \mathcal{A}(p, \mathbb{G}, g_1, g_2, g_3, g_1^{r_1}, g_2^{r_2}, g_3^u) \mid g_1, g_2, g_3 \xleftarrow{\text{U}} \mathbb{G}, r_1, r_2, u \xleftarrow{\text{U}} \mathbb{Z}_p] \right|$$

is negligible in  $\lambda$ .

### 2.2 Labeled Public Key Encryption

*Syntax.* Labeled public key encryption (LPKE) consists of three polynomial time algorithms  $\{\text{Gen}, \text{Enc}, \text{Dec}\}$ .  $\text{Gen}$  and  $\text{Enc}$  are probabilistic.  $\text{Dec}$  is deterministic.

$\text{Gen}(1^\lambda) \rightarrow (ek, dk)$ . The key generation algorithm takes  $1^\lambda$  as input, and outputs an encryption key  $ek$ , and a decryption key  $dk$ . Plaintext space  $\mathcal{M}$ , ciphertext space  $\mathcal{C}$ , and label space  $\mathcal{L}$  are uniquely determined by  $ek$ .

$\text{Enc}(ek, m, L) \rightarrow C$ . The encryption algorithm takes the encryption key  $ek$ , a plaintext  $M \in \mathcal{M}$ , and a label  $L \in \mathcal{L}$  as inputs, and outputs a ciphertext  $C$ .

$\text{Dec}(dk, C, L) \rightarrow M \mid \perp$ . The decryption algorithm<sup>4</sup> takes the decryption key  $dk$ , a ciphertext  $C \in \mathcal{C}$ , and a label  $L \in \mathcal{L}$  as inputs, and outputs a plaintext  $M$  or  $\perp$ .

A LPKE scheme must be correct. LPKE scheme  $\Sigma_{LPKE} = \{\text{Gen}, \text{Enc}, \text{Dec}\}$  is correct, if for every  $\lambda \in \mathbb{N}$ , every  $(ek, dk) \leftarrow \text{Gen}(1^\lambda)$ , every  $M \in \mathcal{M}$ , every  $L \in \mathcal{L}$ , and every  $C \leftarrow \text{Enc}(ek, M, L)$ , it holds that  $M \leftarrow \text{Dec}(dk, C, L)$ .

---

<sup>4</sup> Although  $\text{Dec}$  needs the encryption-key  $ek$  as an input since  $ek$  includes information such as the prime  $p$ , the group  $\mathbb{G}$ , and etc., we often omit  $ek$  as the input.



### Ciphertext Indistinguishability

To define weak ciphertext indistinguishability against adaptively chosen label and ciphertexts attacks (IND-wLCCA) for a LPKE scheme  $\Sigma_{LPKE} = \{\text{Gen}, \text{Enc}, \text{Dec}\}$ , we use the following game which is played between an adversary  $\mathcal{A}$  and challenger  $\mathcal{CH}$ .

**Key-Generation.**  $\mathcal{CH}$  runs  $(ek, dk) \leftarrow \text{Gen}(1^\lambda)$ , and sends  $ek$  to  $\mathcal{A}$ .

**Query.**  $\mathcal{A}$  is allowed to use the decryption oracle  $\text{Dec}$  adaptively.

**Dec( $C, L$ ):**  $\mathcal{A}$  queries a ciphertext  $C \in \mathcal{C}$  and a label  $L \in \mathcal{L}$ .  $\mathcal{CH}$  returns  $M / \perp \leftarrow \text{Dec}(dk, C, L)$ .

**Challenge( $M_0, M_1, L^*$ ).**  $\mathcal{A}$  sends two plaintexts  $M_0, M_1 \in \mathcal{M}$ , and a label  $L^* \in \mathcal{L}$ .  $\mathcal{CH}$  sets  $b \xleftarrow{\text{U}} \{0, 1\}$ , then returns  $C^* \leftarrow \text{Enc}(ek, M_b, L^*)$ .

**Query 2.**  $\mathcal{A}$  is allowed to use the decryption oracle  $\text{Dec}$  adaptively.

**Dec( $C, L$ ):**  $\mathcal{A}$  queries a ciphertext  $C \in \mathcal{C}$  and a label  $L \in \mathcal{L}$  such that  $L \neq L^*$ .  $\mathcal{CH}$  returns  $M / \perp \leftarrow \text{Dec}(dk, C, L)$ .

**Guess( $b'$ ).**  $\mathcal{A}$  sends  $b' \in \{0, 1\}$  to  $\mathcal{CH}$ .

**Definition 1.** LPKE scheme  $\Sigma_{LPKE}$  is IND-wLCCA secure if for any PPT adversary  $\mathcal{A}$ ,  $\text{Adv}_{\mathcal{A}, \Sigma_{LPKE}}^{\text{IND-wLCCA}}(\lambda) = |2 \cdot \Pr[b' = b] - 1|$  is negligible.

### 2.3 Non-Interactive Zero-Knowledge Proof

*Syntax.* Non-interactive zero-knowledge proof (NIZK)  $\Pi_{\text{NIZK}}$  for a language  $L$  consists of three polynomial time algorithms  $\{\text{Gen}, \text{Pro}, \text{Ver}\}$ . Each one of  $\text{Gen}$  and  $\text{Pro}$  is probabilistic.  $\text{Ver}$  is deterministic.  $R_L$  denotes the witness relation.

$\text{Gen}(1^\lambda) \rightarrow \text{crs}$ . The key-generation algorithm takes  $1^\lambda$  as an input, and outputs a common reference string (CRS)  $\text{crs}$ .

$\text{Pro}(\text{crs}, x, w) \rightarrow \pi$ . The proof-generation algorithm takes the CRS  $\text{crs}$ , a statement  $x$ , and a witness  $w$  as inputs, and outputs a proof  $\pi$ .

$\text{Ver}(\text{crs}, x, \pi) \rightarrow 1 / 0$ . The proof-verification algorithm takes the CRS  $\text{crs}$ , a statement  $x$ , and a proof  $\pi$  as inputs, and outputs 1 or 0.

A NIZK scheme must be correct. A NIZK scheme  $\Sigma_{\text{NIZK}} = \{\text{Gen}, \text{Pro}, \text{Ver}\}$  is correct if for every  $\lambda \in \mathbb{N}$ , every  $\text{crs} \leftarrow \text{Gen}(1^\lambda)$ , every  $(x, w)$  such that  $(x, w) \in R_L$ , and every  $\pi \leftarrow \text{Pro}(\text{crs}, x, w)$ , it holds that  $1 \leftarrow \text{Ver}(\text{crs}, x, \pi)$ .

We give the definitions of soundness and zero-knowledge for a NIZK scheme.

**Definition 2.** A NIZK scheme  $\Sigma_{\text{NIZK}} = \{\text{Gen}, \text{Pro}, \text{Ver}\}$  is sound if for every  $\lambda \in \mathbb{N}$ , every  $\text{crs} \leftarrow \text{Gen}(1^\lambda)$ , and every PPT  $\mathcal{A}$ ,

$$\Pr[\mathcal{A}(\text{crs}) \rightarrow (x, \pi) \text{ s.t. } [\text{Ver}(\text{crs}, x, \pi) \rightarrow 1] \wedge [x \notin L]]$$

is negligible.

**Definition 3.** A NIZK scheme  $\Sigma_{\text{NIZK}} = \{\text{Gen}, \text{Pro}, \text{Ver}\}$  is zero-knowledge if for every  $\lambda \in \mathbb{N}$  and every PPT  $\mathcal{A}$ , there exists a PPT  $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$  such that

$$\left| \Pr \left[ \mathcal{A}^{\mathcal{O}_0^{\text{crs}}(x,w)}(\text{crs}) \rightarrow 1 \mid \text{Gen}(1^\lambda) \rightarrow \text{crs} \right] - \Pr \left[ \mathcal{A}^{\mathcal{O}_1^{\text{crs},td}(x,w)}(\text{crs}) \rightarrow 1 \mid \mathcal{S}_1(1^\lambda) \rightarrow (\text{crs}, td) \right] \right|$$

is negligible, where  $\mathcal{O}_0^{\text{crs}}(x,w)$  returns  $\text{Pro}(\text{crs}, x, w)$  (resp.  $\perp$ ), if  $(x, w) \in R_L$  (resp.  $(x, w) \notin R_L$ ), and  $\mathcal{O}_1^{\text{crs},td}(x,w)$  returns  $\mathcal{S}_2(\text{crs}, x, td)$  (resp.  $\perp$ ), if  $(x, w) \in R_L$  (resp.  $(x, w) \notin R_L$ ).

### 2.4 Chameleon Hash Function

*Syntax.* A chameleon hash function (CHF) scheme consists of the polynomial time algorithms  $\{\text{Gen}, \text{Eval}, \text{TC}, \text{SKVer}, \text{SKVer2}\}$ .  $\text{Gen}$  and  $\text{TC}$  are probabilistic, and  $\text{Eval}$ ,  $\text{SKVer}$  and  $\text{SKVer2}$  are deterministic.

$\text{Gen}(1^\lambda) \rightarrow (pk, sk)$ . The key-generation algorithm takes a security parameter  $1^\lambda$ , where  $\lambda \in \mathbb{N}$ , as an input, and outputs a public-key  $pk$  and a secret-key (or trapdoor)  $sk$ . The message space  $\mathcal{M}$ , randomness space  $\mathcal{R}$  and hashed value space  $\mathcal{H}$  are uniquely determined by  $pk$ .

$\text{Eval}(pk, m; r) \rightarrow h$ . The evaluation algorithm takes the public-key  $pk$  and a message  $m \in \mathcal{M}$  as inputs, and outputs the hashed value  $h \in \mathcal{H}$  which was calculated under a randomness  $r \in \mathcal{R}$ .

$\text{TC}(pk, (m_1, r_1), m_2) \rightarrow r_2$ . The trapdoor collision finder algorithm takes the public-key  $pk$ , a pair of a message and randomness  $(m_1, r_1) \in \mathcal{M} \times \mathcal{R}$ , and a message  $m_2 \in \mathcal{M}$  as inputs, and outputs a randomness  $r_2 \in \mathcal{R}$ .

$\text{SKVer}(pk, sk') \rightarrow 1 / 0$ . The first secret-key-verification algorithm takes the public-key  $pk$  and a secret-key  $sk'$  as inputs, and outputs 1 or 0.

$\text{SKVer2}(pk, sk', sk^\dagger) \rightarrow 1 / 0$ . The second secret-key-verification algorithm takes the public-key  $pk$ , a secret-key  $sk'$ , and a secret-key  $sk^\dagger$  as inputs, and outputs 1 or 0. Even if the two secret-keys are inputted in the reversed order, the output is required to be equivalent. Thus, for any  $\lambda \in \mathbb{N}$ , any  $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$ , and any two valid secret-keys  $sk'$  and  $sk^\dagger$ , it holds that  $\text{SKVer2}(pk, sk', sk^\dagger) = \text{SKVer2}(pk, sk^\dagger, sk')$ .

A CHF scheme must be correct. A CHF scheme  $\Sigma_{\text{CHF}} = \{\text{Gen}, \text{Eval}, \text{TC}, \text{SKVer}, \text{SKVer2}\}$  is correct, if for every  $\lambda \in \mathbb{N}$ , every  $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$ , every  $m \in \mathcal{M}$ , every  $m' \in \mathcal{M}$ , every  $r \in \mathcal{R}$ , and every  $r' := \text{TC}(pk, (m, r), m')$ , it holds that  $[\text{Eval}(pk, m; r) = \text{Eval}(pk, m'; r')] \wedge [1 \leftarrow \text{SKVer}(pk, sk)] \wedge [1 \leftarrow \text{SKVer2}(pk, sk, sk)]$ .

We give the definitions of two standard properties for the CHF scheme. They are strong collision-resistance and random trapdoor collision.

**Definition 4.** A CHF scheme  $\Sigma_{\text{CHF}} = \{\text{Gen}, \text{Eval}, \text{TC}, \text{SKVer}, \text{SKVer2}\}$  is strongly collision-resistant, if for every  $\lambda \in \mathbb{N}$  and every PPT  $\mathcal{A}$ , it holds that

$$\Pr[\mathcal{A}(pk) \rightarrow ((m_1, r_1), (m_2, r_2)) \text{ s.t. } [(m_1, r_1) \neq (m_2, r_2)] \wedge [\text{Eval}(pk, m_1; r_1) = \text{Eval}(pk, m_2; r_2)]]$$

is negligible, where  $(pk, sk) \stackrel{R}{\leftarrow} \text{Gen}(1^\lambda)$ .

**Definition 5.** A CHF scheme  $\Sigma_{\text{CHF}} = \{\text{Gen}, \text{Eval}, \text{TC}, \text{SKVer}, \text{SKVer2}\}$  is said to have the property of random trapdoor collision, if for any  $\lambda \in \mathbb{N}$ , any  $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$  and any two messages  $m_1, m_2 \in \mathcal{M}$ , a randomness  $r_1$  chosen uniformly at random from  $\mathcal{R}$  distributes identically with  $r_2 := \text{TC}(pk, (m_1, r_1), m_2)$ .

We give the definition of an original property for a CHF scheme. The property is hard-to-compute-secret-key (HtC-SK).

**Definition 6.**  $\Sigma_{\text{CHF}} = \{\text{Gen}, \text{Eval}, \text{TC}, \text{SKVer}, \text{SKVer2}\}$  is said to have the property of HtC-SK, if for every  $\lambda \in \mathbb{N}$ , and every PPT  $\mathcal{A}$ , it holds that

$$\Pr[\mathcal{A}(pk, sk) \rightarrow sk^* \text{ s.t. } [1 \leftarrow \text{SKVer}(pk, sk^*)] \wedge [0 \leftarrow \text{SKVer2}(pk, sk^*, sk)]]$$

is negligible, where  $(pk, sk) \stackrel{R}{\leftarrow} \text{Gen}(1^\lambda)$ .

*Remark.* The property is related to the second pre-image resistance (SPR) [13, 15]. The algorithm SKVer given  $pk$  and  $sk$  as inputs is an algorithm which verifies whether or not a relation holds between  $pk$  and  $sk$ . The algorithm SKVer2 given two secret-keys  $sk$  and  $sk^\dagger$  can be defined as the algorithm outputting 1 iff the two keys are equivalent. Thus, the HtC-SK property can be the SPR property. We can say that the HtC-SK property is a generalization of the SPR property.

## 2.5 Digital Signature

*Syntax.* Digital signature consists of the polynomial time algorithms  $\{\text{Gen}, \text{Sig}, \text{Ver}, \text{SKVer}, \text{SKVer2}\}$ . Gen and Sig are probabilistic, and Ver, SKVer and SKVer2 are deterministic.

$\text{Gen}(1^\lambda) \rightarrow (pk, sk)$ . The key-generation algorithm takes  $1^\lambda$ , where  $\lambda \in \mathbb{N}$ , as an input, and outputs a public-key  $pk$  and a secret-key  $sk$ . The message space  $\mathcal{M}$  is uniquely determined by  $pk$ .

$\text{Sig}(pk, m, sk) \rightarrow \sigma$ . The signing algorithm takes the public-key  $pk$ , a message  $m \in \mathcal{M}$ , and the secret-key  $sk$  as inputs, and outputs a signature  $\sigma$ .

$\text{Ver}(pk, m, \sigma) \rightarrow 1 / 0$ . The signature-verification algorithm takes the public-key  $pk$ , a message  $m \in \mathcal{M}$ , and a signature  $\sigma$  as inputs, and outputs 1 or 0.

$\text{SKVer}(pk, sk') \rightarrow 1 / 0$ . This is the same as the algorithm SKVer of the CHF scheme.

$\text{SKVer2}(pk, sk', sk^\dagger) \rightarrow 1 / 0$ . This is the same as the algorithm SKVer2 of the CHF scheme.

A signature scheme must be correct. A signature scheme  $\Sigma_{\text{SIG}} = \{\text{Gen}, \text{Sig}, \text{Ver}, \text{SKVer}, \text{SKVer2}\}$  is correct if for every  $(pk, sk) \leftarrow \text{Gen}(1^\lambda)$ , every  $m \in \mathcal{M}$ , and every  $\sigma \leftarrow \text{Sig}(pk, m, sk)$ , it holds that  $[1 \leftarrow \text{Ver}(pk, m, \sigma)] \wedge [1 \leftarrow \text{SKVer}(pk, sk)] \wedge [1 \leftarrow \text{SKVer2}(pk, sk, sk)]$ .

**Strong Existential Unforgeability in the Auxiliary Leakage Model**

In this subsection, we define the strong existential unforgeability in the AL model for a signature scheme. Specifically, we define the strong existential unforgeability against adaptively chosen messages attacks in the auxiliary leakage model (AL-sEUF-CMA) for a signature scheme  $\Sigma_{\text{SIG}} = \{\text{Gen}, \text{Sig}, \text{Ver}, \text{SKVer}, \text{SKVer2}\}$ .

At first, we define a game which is played between an adversary  $\mathcal{A}$  and challenger  $\mathcal{CH}$  as follows. Note that a leakage function  $f : \{0, 1\}^{|pk|+|sk|} \rightarrow \{0, 1\}^*$  whose randomness space is denoted by  $\mathcal{R}$  is included in a class  $\mathcal{F}_{\Sigma_{\text{SIG}}}(\lambda)$ , i.e.,  $f \in \mathcal{F}_{\Sigma_{\text{SIG}}}(\lambda)$ <sup>5</sup>.

**Key-Generation.**  $\mathcal{CH}$  runs  $(pk, sk) \leftarrow \text{SIG.Gen}(1^\lambda)$ .  $\mathcal{CH}$  chooses  $r \xleftarrow{\mathcal{R}} \mathcal{R}$ , then computes  $f(pk, sk; r)$ .  $\mathcal{CH}$  sends  $(pk, f(pk, sk; r))$  to  $\mathcal{A}$ .  $\mathcal{CH}$  initializes the list  $\mathcal{L}_S$  as an empty set  $\emptyset$ .

**Query.**  $\mathcal{A}$  is allowed to use the signing oracle **Sign**, adaptively.

**Sign**( $m \in \mathcal{M}$ ):  $\mathcal{CH}$  generates  $\sigma \leftarrow \text{SIG.Sig}(pk, m, sk)$ , then sends  $\sigma$  to  $\mathcal{A}$ .

After that,  $\mathcal{CH}$  sets  $\mathcal{L}_S := \mathcal{L}_S \cup \{(m, \sigma)\}$ .

**Forgery**( $m^*, \sigma^*$ ).  $\mathcal{CH}$  receives  $(m^*, \sigma^*)$  sent by  $\mathcal{A}$ .

In the above game,  $\mathcal{A}$  is said to win the game if  $[1 \leftarrow \text{SIG.Ver}(pk, m^*, \sigma^*)] \wedge [(m^*, \sigma^*) \notin \mathcal{L}_S]$ . The advantage  $\text{Adv}_{\Sigma_{\text{SIG}}, \mathcal{A}}^{\mathcal{F}(\lambda)\text{-AL-sEUF-CMA}}(\lambda)$  is defined as the probability  $\text{Pr}[\mathcal{A} \text{ wins.}]$ .

**Definition 7.**  $\Sigma_{\text{SIG}}$  is AL-sEUF-CMA-secure with respect to the leakage-function class  $\mathcal{F}_{\Sigma_{\text{SIG}}}(\lambda)$ , if for every PPT  $\mathcal{A}$  and every function  $f \in \mathcal{F}_{\Sigma_{\text{SIG}}}(\lambda)$ ,  $\text{Adv}_{\Sigma_{\text{SIG}}, \mathcal{A}}^{\mathcal{F}(\lambda)\text{-AL-sEUF-CMA}}(\lambda)$  is negligible.

*Remark.* Weak existential unforgeability is defined in the same manner as the strong existential unforgeability except for the winning condition by the adversary  $\mathcal{A}$  in the security game. The adversary is said to win the game if the signature  $\sigma^*$  is a valid signature on the message  $m^*$ , i.e.,  $[1 \leftarrow \text{SIG.Ver}(pk, m^*, \sigma^*)]$ , and the message  $m^*$  has not been queried to the signing oracle **Sign**.

**3 Signature Strongly Existentially Unforgeable and Resilient to Polynomially Hard-to-Invert Leakage**

In Subsect. 3.1, the generic construction of our signature scheme is given. In Subsect. 3.2, the signature scheme is proven to be strongly existentially unforgeable and resilient to polynomially hard-to-invert leakage. In the next section, i.e., Sect. 4, we show that the signature scheme can be instantiated under the DLIN assumption.

---

<sup>5</sup> In this paper, the function class  $\mathcal{F}_{\Sigma_{\text{SIG}}}(\lambda)$  can be simply written as  $\mathcal{F}(\lambda)$ , if it is obvious that the function class is for the signature scheme  $\Sigma_{\text{SIG}}$ .

### 3.1 Construction

Our generic construction of signature scheme  $\Sigma_{\text{SIG}} = \{\text{SIG.Gen}, \text{SIG.Sig}, \text{SIG.Ver}, \text{SIG.SKVer}, \text{SIG.SKVer2}\}$  has the following 4 building blocks: A LPKE scheme  $\Sigma_{\text{LPKE}} = \{\text{LPKE.Gen}, \text{LPKE.Enc}, \text{LPKE.Dec}\}$ , a NIZK scheme  $\Sigma_{\text{NIZK}} = \{\text{NIZK.Gen}, \text{NIZK.Pro}, \text{NIZK.Ver}\}$ , a CHF scheme  $\Sigma_{\text{CHF}} = \{\text{CHF.Gen}, \text{CHF.Eval}, \text{CHF.TC}, \text{CHF.SKVer}, \text{CHF.SKVer2}\}$  and a CHF scheme  $\Sigma_{\text{CHF2}} = \{\text{CHF2.Gen}, \text{CHF2.Eval}, \text{CHF2.TC}\}$ .

The signature scheme  $\Sigma_{\text{SIG}}$  is generically constructed as follows.

**SIG.Gen( $1^\lambda$ ):** Run  $(ek, dk) \leftarrow \text{LPKE.Gen}(1^\lambda)$ ,  $(pk_1, sk_1) \leftarrow \text{CHF.Gen}(1^\lambda)$  and  $(pk_2, sk_2) \leftarrow \text{CHF2.Gen}(1^\lambda)$ . Run  $(crs, td) \leftarrow \mathcal{S}_1(1^\lambda)$ , where  $\mathcal{S}_1$  is the first simulator in the definition of zero-knowledge for  $\Sigma_{\text{NIZK}}$ .

$\mathcal{R}_E$  and  $\mathcal{R}_{E2}$  denote the randomness space of  $\text{CHF.Eval}$  and  $\text{CHF2.Eval}$ , respectively.  $\bar{\mathcal{M}}, \mathcal{M}, \mathcal{C}, \mathcal{P}$  and  $\mathcal{K}_1$  denote the message space of  $\Sigma_{\text{CHF}}$ , the label space of  $\Sigma_{\text{LPKE}}$  (or the hashed value space of  $\Sigma_{\text{CHF2}}$ ), the ciphertext space of  $\Sigma_{\text{LPKE}}$ , the proof space of  $\Sigma_{\text{NIZK}}$ , and the secret-key space of  $\Sigma_{\text{CHF}}$ , respectively.  $\mathcal{M}$  is a space satisfying  $\bar{\mathcal{M}} = \mathcal{M} \parallel \mathcal{C} \parallel \mathcal{P}$ .

Verification-key and signing-key are set as  $pk := (pk_1, pk_2, ek, crs)$  and  $sk := sk_1$ , respectively. **Return**  $(pk, sk)$ . We define language  $L$  as

$$L := \{(c, \bar{m}) \in \mathcal{C} \times \bar{\mathcal{M}} \mid \exists sk_1 \in \mathcal{K}_1 \text{ s.t. } [c \leftarrow \text{LPKE.Enc}(ek, sk_1, \bar{m})] \\ \wedge [1 \leftarrow \text{CHF.SKVer}(pk_1, sk_1)]\}.$$

**SIG.Sig( $pk, m \in \mathcal{M}, sk$ ):**  $pk$  is parsed as  $(pk_1, pk_2, ek, crs)$ .  $sk$  is written as  $sk_1$ . Do as follows in order.

- $r'_E \xleftarrow{\text{U}} \mathcal{R}_E, r_{E2} \xleftarrow{\text{U}} \mathcal{R}_{E2}, m' \xleftarrow{\text{U}} \mathcal{M}, c' \xleftarrow{\text{U}} \mathcal{C}, \pi' \xleftarrow{\text{U}} \mathcal{P}, \sigma' := (c', \pi')$ .
- $h := \text{CHF.Eval}(pk_1, m' \parallel \sigma'; r'_E), \bar{m} := \text{CHF2.Eval}(pk_2, h; r_{E2})$ .
- $c := \text{LPKE.Enc}(ek, sk_1, \bar{m}), x := (c, \bar{m}), w := sk_1, \pi := \text{NIZK.Pro}(crs, x, w)$ .
- $\sigma := (c, \pi), r_E := \text{CHF.TC}(pk_1, sk_1, (m' \parallel \sigma', r'_E), m \parallel \sigma)$ .

**Return**  $\sigma^\dagger := (\sigma, r_E, r_{E2}) = (c, \pi, r_E, r_{E2})$ .

**SIG.Ver( $pk, m \in \mathcal{M}, \sigma^\dagger$ ):**  $pk$  is parsed as  $(pk_1, pk_2, ek, crs)$ .  $\sigma^\dagger$  is parsed as  $(c, \pi, r_E, r_{E2})$ .  $h := \text{CHF.Eval}(pk_1, m \parallel \sigma; r_E)$ .  $\bar{m} := \text{CHF2.Eval}(pk_2, h; r_{E2})$ .  $x := (c, \bar{m})$ . **Return**  $\text{NIZK.Ver}(crs, x, \pi)$ .

**SIG.SKVer( $pk, sk$ ):**  $pk$  is parsed as  $(pk_1, pk_2, ek, crs)$ .  $sk$  is written as  $sk_1$ . **Return**  $\text{CHF.SKVer}(pk_1, sk_1)$ .

**SIG.SKVer2( $pk, sk, sk'$ ):**  $pk$  is parsed as  $(pk_1, pk_2, ek, crs)$ .  $sk$  and  $sk'$  are written as  $sk_1$  and  $sk'_1$ , respectively. **Return**  $\text{CHF.SKVer2}(pk_1, sk_1, sk'_1)$ .

### 3.2 Proof of Strong Unforgeability in Polynomially Hard-to-Invert Leakage Model

Before giving the theorem for the strong unforgeability in the hard-to-invert leakage model of the signature scheme  $\Sigma_{\text{SIG}}$ , we give the definitions of the leakage-function class  $\mathcal{F}_{\Sigma_{\text{SIG}}}^{\text{HtI}}(\lambda)$  for the signature scheme and the strong collision-resistance in the auxiliary leakage model w.r.t. the function class  $\mathcal{F}_{\Sigma_{\text{CHF}}}^{\text{HtI}}(\lambda)$  for the chameleon hash function  $\Sigma_{\text{CHF}}$ .

**Definition 8.** Function class  $\mathcal{F}_{\Sigma_{\text{SIG}}}^{\text{HtI}}(\lambda)$  consists of every polynomial-time computable probabilistic (or deterministic) function  $f : \{0, 1\}^{|pk_1|+|pk_2|+|ek|+|crs|+|sk_1|} \rightarrow \{0, 1\}^*$  which has a randomness space  $\mathcal{R}$  and satisfies the following condition: for every PPT  $\mathcal{B}$ ,

$$\begin{aligned} & \Pr[\mathcal{B}(pk_1, pk_2, ek, crs, sk_2, dk, td, f(pk_1, pk_2, ek, crs, sk_1; r)) \rightarrow sk_1^* \\ & \text{s.t. } [1 \leftarrow \text{CHF.SKVer}(pk_1, sk_1^*)] \wedge [1 \leftarrow \text{CHF.SKVer2}(pk_1, sk_1^*, sk_1)]] \end{aligned} \quad (1)$$

is negligible, where  $(pk_1, sk_1) \stackrel{\mathcal{R}}{\leftarrow} \text{CHF.Gen}(1^\lambda)$ ,  $(pk_2, sk_2) \stackrel{\mathcal{R}}{\leftarrow} \text{CHF2.Gen}(1^\lambda)$ ,  $(ek, dk) \stackrel{\mathcal{R}}{\leftarrow} \text{LPKE.Gen}(1^\lambda)$ ,  $(crs, td) \stackrel{\mathcal{R}}{\leftarrow} \mathcal{S}_1(1^\lambda)$  and  $r \stackrel{\mathcal{R}}{\leftarrow} \mathcal{R}$ .

*Remark.* If the chameleon hash function  $\Sigma_{\text{CHF}}$  is a CHF with the second pre-image resistance [13, 15], the algorithm CHF.SKVer2 is defined as the equality-checking algorithm, and the secret-key  $sk_1^*$  which satisfies  $[1 \leftarrow \text{CHF.SKVer}(pk_1, sk_1^*)] \wedge [1 \leftarrow \text{CHF.SKVer2}(pk_1, sk_1^*, sk_1)]$  is the original secret-key  $sk_1$  only. So, the probability (1) is simply written as  $\Pr[\mathcal{B}(pk_1, pk_2, ek, crs, sk_2, dk, td, f(pk_1, pk_2, ek, crs, sk_1; r)) \rightarrow sk_1]$ .

**Definition 9.** CHF scheme  $\Sigma_{\text{CHF}} = \{\text{CHF.Gen}, \text{CHF.Eval}, \text{CHF.TC}, \text{CHF.SKVer}, \text{CHF.SKVer2}\}$  is strongly collision-resistant in the auxiliary leakage model w.r.t. the function class  $\mathcal{F}_{\Sigma_{\text{SIG}}}^{\text{HtI}}(\lambda)$ , if for every PPT  $\mathcal{A}$  and every function  $f : \{0, 1\}^{|pk_1|+|pk_2|+|ek|+|crs|+|sk_1|} \rightarrow \{0, 1\}^*$  which satisfies  $f \in \mathcal{F}_{\Sigma_{\text{SIG}}}^{\text{HtI}}(\lambda)$  and has a randomness space  $\mathcal{R}$ ,

$$\begin{aligned} & \Pr[\mathcal{A}(pk_1, pk_2, ek, crs, sk_2, dk, td, f(pk_1, pk_2, ek, crs, sk_1; r)) \rightarrow ((m_1, r_1), (m_2, r_2)) \\ & \text{s.t. } [(m_1, r_1) \neq (m_2, r_2)] \wedge [\text{CHF.Eval}(pk_1, m_1; r_1) = \text{CHF.Eval}(pk_1, m_2; r_2)] \end{aligned}$$

is negligible, where  $(pk_1, sk_1) \stackrel{\mathcal{R}}{\leftarrow} \text{CHF.Gen}(1^\lambda)$ ,  $(pk_2, sk_2) \stackrel{\mathcal{R}}{\leftarrow} \text{CHF2.Gen}(1^\lambda)$ ,  $(ek, dk) \stackrel{\mathcal{R}}{\leftarrow} \text{LPKE.Gen}(1^\lambda)$ ,  $(crs, td) \stackrel{\mathcal{R}}{\leftarrow} \mathcal{S}_1(1^\lambda)$  and  $r \stackrel{\mathcal{R}}{\leftarrow} \mathcal{R}$ .

The strong unforgeability in the AL model of the signature scheme  $\Sigma_{\text{SIG}}$  is guaranteed by the following theorem.

**Theorem 1.**  $\Sigma_{\text{SIG}}$  is AL-sEUF-CMA w.r.t. the function class  $\mathcal{F}_{\Sigma_{\text{SIG}}}^{\text{HtI}}(\lambda)$ , if

- $\Sigma_{\text{LPKE}}$  is IND-wLCCA,
- $\Sigma_{\text{NIZK}}$  is sound and zero-knowledge,
- $\Sigma_{\text{CHF}}$  is strongly collision-resistant in the auxiliary leakage model w.r.t. the function class  $\mathcal{F}_{\Sigma_{\text{SIG}}}^{\text{HtI}}(\lambda)$ , random trapdoor collision, and HtC-SK, and
- $\Sigma_{\text{CHF2}}$  is strongly collision-resistant and random trapdoor collision.

*Proof of Theorem 1.* Hereafter,  $q_s \in \mathbb{N}$  denotes the number of times that PPT adversary  $\mathcal{A}$  uses the signing oracle **Sign**. To prove Theorem 1, we use multiple games **Game<sub>i</sub>**, where  $i \in \{0, 1, 2, 3, 4, 5, 6, 7, 7|1, \dots, 7|q_s\}$ .

The first game **Game<sub>0</sub>** is the normal AL-sEUF-CMA game w.r.t. the signature scheme  $\Sigma_{\text{SIG}}$  and the function class  $\mathcal{F}_{\Sigma_{\text{SIG}}}^{\text{HtI}}(\lambda)$ . Specifically, **Game<sub>0</sub>** is the following game.

**Key-Generation.**  $\mathcal{CH}$  runs  $(pk_1, sk_1) \leftarrow \text{CHF.Gen}(1^\lambda)$ ,  $(pk_2, sk_2) \leftarrow \text{CHF2.Gen}(1^\lambda)$ ,  $(ek, dk) \leftarrow \text{LPKE.Gen}(1^\lambda)$ , and  $(crs, td) \leftarrow \mathcal{S}_1(1^\lambda)$ .  $pk$  and  $sk$  are set as  $pk := (pk_1, pk_2, ek, crs)$  and  $sk := sk_1$ , respectively. For a function  $f \in \mathcal{F}_{\Sigma_{\text{SIG}}}^{\text{HtI}}(\lambda)$ ,  $\mathcal{CH}$  chooses  $r \xleftarrow{\text{R}} \mathcal{R}$ , then computes  $f(pk, sk; r)$ .  $\mathcal{CH}$  sends  $(pk, f(pk, sk; r))$  to  $\mathcal{A}$ .  $\mathcal{L}_S$  is set to  $\emptyset$ .

**Query.** When  $\mathcal{A}$  issues a message  $m \in \mathcal{M}$  as a query to the signing oracle **Sign**,  $\mathcal{CH}$  generates a signature  $(c, \pi, r_E, r_{E2})$  on the message as follows.

- $r'_E \xleftarrow{\text{U}} \mathcal{R}_E$ ,  $r_{E2} \xleftarrow{\text{U}} \mathcal{R}_{E2}$ ,  $m' \xleftarrow{\text{U}} \mathcal{M}$ ,  $c' \xleftarrow{\text{U}} \mathcal{C}$ ,  $\pi' \xleftarrow{\text{U}} \mathcal{P}$ ,  $\sigma' := (c', \pi')$ .
- $h := \text{CHF.Eval}(pk_1, m' || \sigma'; r'_E)$ ,  $\bar{m} := \text{CHF2.Eval}(pk_2, h; r_{E2})$ .
- $c := \text{LPKE.Enc}(ek, sk_1, \bar{m})$ ,  $x := (c, \bar{m})$ ,  $w := sk_1$ ,  $\pi := \text{NIZK.Pro}(crs, x, w)$ .
- $\sigma := (c, \pi)$ ,  $r_E := \text{CHF.TC}(pk_1, sk_1, (m' || \sigma', r'_E), m || \sigma)$ .

$\mathcal{CH}$  returns a signature  $(c, \pi, r_E, r_{E2})$  to  $\mathcal{A}$ .  $\mathcal{CH}$  sets  $\mathcal{L}_S := \mathcal{L}_S \cup \{(m, c, \pi, r_E, r_{E2})\}$ .

**Forgery** $(m^*, (c^*, \pi^*, r_E^*, r_{E2}^*))$ .  $\mathcal{CH}$  is given a message  $m^* \in \mathcal{M}$  and a signature  $(c^*, \pi^*, r_E^*, r_{E2}^*)$ .

$\mathcal{A}$  wins the game, if  $[1 \leftarrow \text{NIZK.Ver}(crs, x^*, \pi^*)] \wedge [(m^*, c^*, \pi^*, r_E^*, r_{E2}^*) \notin \mathcal{L}_S]$ , where  $h^* := \text{CHF.Eval}(pk_1, m^* || (c^*, \pi^*); r_E^*)$ ,  $\bar{m}^* := \text{CHF2.Eval}(pk_2, h^*; r_{E2}^*)$  and  $x^* := (c^*, \bar{m}^*)$ .

We define the games **Game<sub>i</sub>**, where  $i \in \{1, 2, 3, 4, 5, 6, 7, 7|1, \dots, 7|q_s\}$ , as follows.

**Game<sub>1</sub>.** **Game<sub>1</sub>** is the same as **Game<sub>0</sub>** except that  $\mathcal{CH}$  generates a common reference string  $crs$  by running  $crs \leftarrow \text{NIZK.Gen}(1^\lambda)$  in **Key-Generation**.

**Game<sub>2</sub>.** **Game<sub>2</sub>** is the same as **Game<sub>1</sub>** except that  $\mathcal{A}$ 's winning condition is changed to the following one, where  $sk_1^* := \text{LPKE.Dec}(dk, c^*, \bar{m}^*)$ :  $[1 \leftarrow \text{NIZK.Ver}(crs, x^*, \pi^*)] \wedge [(m^*, c^*, \pi^*, r_E^*, r_{E2}^*) \notin \mathcal{L}_S] \wedge [1 \leftarrow \text{CHF.SKVer}(pk_1, sk_1^*)]$ .

**Game<sub>3</sub>.** **Game<sub>3</sub>** is the same as **Game<sub>2</sub>** except that  $\mathcal{A}$ 's winning condition is changed to the following one:  $[1 \leftarrow \text{NIZK.Ver}(crs, x^*, \pi^*)] \wedge [(m^*, c^*, \pi^*, r_E^*, r_{E2}^*) \notin \mathcal{L}_S] \wedge [1 \leftarrow \text{CHF.SKVer}(pk_1, sk_1^*)] \wedge [1 \leftarrow \text{CHF.SKVer2}(pk_1, sk_1^*, sk_1)]$ .

**Game<sub>4</sub>.** **Game<sub>4</sub>** is the same as **Game<sub>3</sub>** except that  $\mathcal{A}$ 's winning condition is changed to the following one:  $[1 \leftarrow \text{NIZK.Ver}(crs, x^*, \pi^*)] \wedge [(m^*, c^*, \pi^*, r_E^*, r_{E2}^*) \notin \mathcal{L}_S] \wedge [1 \leftarrow \text{CHF.SKVer}(pk_1, sk_1^*)] \wedge [1 \leftarrow \text{CHF.SKVer2}(pk_1, sk_1^*, sk_1)] \wedge [[\bar{m}^* \notin \{\bar{m}_1, \dots, \bar{m}_{q_s}\}] \vee [\exists i \in [1, q_s] \text{ s.t. } [\bar{m}^* = \bar{m}_i] \wedge [(h^*, r_{E2}^*) = (h_i, r_{E2,i})] \wedge [(m^*, c^*, \pi^*, r_E^*) \neq (m_i, c_i, \pi_i, r_{E,i})]]]$ , where, for  $i \in [1, q_s]$ , each one of  $\bar{m}_i, h_i, r_{E2,i}, c_i, \pi_i$  and  $r_{E,i}$  is the element which was generated when computing the reply to the  $i$ -th signing oracle query.

**Game<sub>5</sub>.** **Game<sub>5</sub>** is the same as **Game<sub>4</sub>** except that when  $\mathcal{A}$  issues a message  $m \in \mathcal{M}$  as a query to the signing oracle **Sign**,  $\mathcal{CH}$  generates a signature  $(c, \pi, r_E, r_{E2})$  on the message as follows.

- $r_E, r'_E \xleftarrow{\text{U}} \mathcal{R}_E$ ,  $r'_{E2} \xleftarrow{\text{U}} \mathcal{R}_{E2}$ ,  $m' \xleftarrow{\text{U}} \mathcal{M}$ ,  $c' \xleftarrow{\text{U}} \mathcal{C}$ ,  $\pi' \xleftarrow{\text{U}} \mathcal{P}$ ,  $\sigma' := (c', \pi')$ .
- $h' := \text{CHF.Eval}(pk_1, m' || \sigma'; r'_E)$ ,  $\bar{m}' := \text{CHF2.Eval}(pk_2, h'; r'_{E2})$ .

- $c := \text{LPKE.Enc}(ek, sk_1, \bar{m}), x := (c, \bar{m}), w := sk_1, \pi := \text{NIZK.Pro}(crs, x, w)$ .
- $\sigma := (c, \pi), h := \text{CHF.Eval}(pk_1, m || \sigma; r_E)$ .
- $r_{E2} := \text{CHF2.TC}(pk_2, sk_2, (h', r'_{E2}), h)$ .

**Game<sub>6</sub>.** Game<sub>6</sub> is the same as Game<sub>5</sub> except that the following two points. Firstly,  $\mathcal{CH}$  generates a common reference string  $crs$  by running  $(crs, td) \leftarrow \mathcal{S}_1(1^\lambda)$  in **Key-Generation**. Secondly, when replying to a query to **Sign** in **Query**,  $\mathcal{CH}$  generates a proof  $\pi$  by using  $\mathcal{S}_2$ , instead of  $\text{NIZK.Pro}$ , where  $\mathcal{S}_2$  denotes the second simulator in the definition of zero-knowledge for  $\Sigma_{\text{NIZK}}$ .

**Game<sub>7</sub>(= Game<sub>7|0</sub>).** Game<sub>7</sub> is the same as Game<sub>6</sub> except that  $\mathcal{A}$ 's winning condition is changed to the following one:  $[1 \leftarrow \text{NIZK.Ver}(crs, x^*, \pi^*)] \wedge [(m^*, c^*, \pi^*, r_{E^*}, r_{E2^*}) \notin \mathcal{L}_S] \wedge [1 \leftarrow \text{CHF.SKVer}(pk_1, sk_1^*)] \wedge [1 \leftarrow \text{CHF.SKVer2}(pk_1, sk_1^*, sk_1)] \wedge [\bar{m}^* \notin \{\bar{m}_1, \dots, \bar{m}_{q_s}\}]$ .

**Game<sub>7|1, \dots, Game<sub>7|q\_s</sub>.</sub>** Game<sub>7|i</sub>, where  $i \in [1, q_s]$ , is the same as Game<sub>7|0</sub> except that when replying to the  $j$ -th signing oracle query, where  $j \leq i$ ,  $\mathcal{CH}$  generates the ciphertext  $c_j$  by running  $c_j \leftarrow \text{LPKE.Enc}(ek, 0^{|sk_1|}, \bar{m}_j)$ , where  $0^{|sk_1|}$  denotes the bitstring of  $|sk_1|$  number of 0.

Hereafter,  $W_i$ , where  $i \in \{0, 1, 2, 3, 4, 5, 6, 7, 7|1, \dots, 7|q_s\}$ , denotes the event that  $\mathcal{A}$  wins the game Game <sub>$i$</sub> . It holds obviously that

$$\begin{aligned} \text{Adv}_{\Sigma_{\text{SIG}}, \mathcal{A}}^{\mathcal{F}_{\Sigma_{\text{SIG}}}^{\text{HtI}}(\lambda) - \text{AL-sEUF-CMA}}(\lambda) &= \Pr[W_0] \\ &\leq \sum_{i=1}^7 |\Pr[W_{i-1} - W_i]| + \sum_{i=1}^{q_s} |\Pr[W_{7|i-1}] \\ &\quad - \Pr[W_{7|i}]| + \Pr[W_{7|q_s}]. \end{aligned}$$

Theorem 1 is proven by the above inequality and the following all lemmas.

**Lemma 1.**  $|\Pr[W_0] - \Pr[W_1]|$  is negligible if  $\Sigma_{\text{NIZK}}$  is zero-knowledge.

**Lemma 2.**  $|\Pr[W_1] - \Pr[W_2]|$  is negligible if  $\Sigma_{\text{NIZK}}$  is sound.

**Lemma 3.**  $|\Pr[W_2] - \Pr[W_3]|$  is negligible if  $\Sigma_{\text{CHF}}$  is HtC-SK.

**Lemma 4.**  $|\Pr[W_3] - \Pr[W_4]|$  is negligible if  $\Sigma_{\text{CHF2}}$  is strongly collision-resistant.

**Lemma 5.**  $|\Pr[W_4] - \Pr[W_5]|$  is negligible if each one of  $\Sigma_{\text{CHF}}$  and  $\Sigma_{\text{CHF2}}$  is random trapdoor collision.

**Lemma 6.**  $|\Pr[W_5] - \Pr[W_6]|$  is negligible if  $\Sigma_{\text{NIZK}}$  is zero-knowledge.

**Lemma 7.**  $|\Pr[W_6] - \Pr[W_{7|0}]|$  is negligible if  $\Sigma_{\text{CHF}}$  is strongly collision-resistant in the auxiliary leakage model w.r.t. the function class  $\mathcal{F}_{\Sigma_{\text{SIG}}}^{\text{HtI}}(\lambda)$ .

**Lemma 8.** For any  $i \in [1, q_s]$ ,  $|\Pr[W_{7|i-1}] - \Pr[W_{7|i}]|$  is negligible if  $\Sigma_{\text{LPKE}}$  is IND-wLCCA.

**Lemma 9.**  $\Pr[W_{7|q_s}]$  is negligible.

Proof of each lemma is given in the full version of this paper.  $\square$



## 4 Instantiation Under the DLIN Assumption

As a concrete construction for the chameleon hash function  $\Sigma_{\text{CHF}}$ , we adopt the chameleon hash function  $\Pi_{\text{CHF},n}$  given in Fig. 1 which is described in the last page of this paper. For  $\Pi_{\text{CHF},n}$ , we obtain Theorems 2, 3, and 4, whose proofs are given in the full version of this paper.

$\text{CHF.Gen}(1^\lambda, 1^n) :$ $(p, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda). x_1, \dots, x_n, a_1, \dots, a_n \xleftarrow{\text{U}} \mathbb{Z}_p. g_1 := g^{a_1}, \dots, g_n := g^{a_n}. y := \prod_{i=1}^n g_i^{x_i}.$ Return $pk := (p, \mathbb{G}, g_1, \dots, g_n, y)$ and $sk := (x_1, \dots, x_n).$
$\text{CHF.Eval}(pk, m; \mathbf{r}) :$ $\mathbf{r} \xleftarrow{\text{U}} \mathbb{Z}_p^n$ , where $\mathbf{r}$ is parsed as $(r_1, \dots, r_n)$ . Return $(y \cdot \prod_{i=1}^n g_i^{r_i})^{J(m)}$ .
$\text{CHF.TC}(pk, sk, (m, \mathbf{r}), m') :$ $sk \in \mathbb{Z}_p^n$ is parsed as $(x_1, \dots, x_n)$ . $\mathbf{r} \in \mathbb{Z}_p^n$ is parsed as $(r_1, \dots, r_n)$ . For $i \in [1, n]$ , $r'_i := J(m)(x_i - r_i) / J(m') - x_i$ . Return $\mathbf{r}' := (r'_1, \dots, r'_n)$ .
$\text{CHF.SKVer}(pk, sk^*) :$ $sk^* \in \mathbb{Z}_p^n$ is parsed as $(x_1^*, \dots, x_n^*)$ . Return 1, if $\left[ y = \prod_{i=1}^n g_i^{x_i^*} \right]$ . Return 0, otherwise;
$\text{CHF.SKVer2}(pk, sk^*, sk') :$ $sk^* \in \mathbb{Z}_p^n$ is parsed as $(x_1^*, \dots, x_n^*)$ . $sk' \in \mathbb{Z}_p^n$ is parsed as $(x'_1, \dots, x'_n)$ . Return 1, if $\left[ \bigwedge_{i=1}^n [x_i^* = x'_i] \right]$ . Return 0, otherwise;

**Fig. 1.** Construction of CHF Scheme  $\Pi_{\text{CHF},n}$ , where  $J : \{0, 1\}^* \rightarrow \mathbb{Z}_p \setminus \{0\}$  is a collision-resistant hash function.

**Theorem 2.** For any  $n \in \mathbb{N}$ ,  $\Pi_{\text{CHF},n}$  is HtC-SK under the DL assumption.

**Theorem 3.** For any  $n \in \mathbb{N}$ ,  $\Pi_{\text{CHF},n}$  is random trapdoor collision.

**Theorem 4.** For any chameleon hash function  $\Pi_{\text{CHF}2}$ , any LPKE scheme  $\Pi_{\text{LPKE}}$ , any NIZK scheme  $\Pi_{\text{NIZK}}$ , and any integer  $n \in \mathbb{N}$ ,  $\Pi_{\text{CHF},n}$  is strongly collision-resistant in the auxiliary leakage model w.r.t. the function class  $\mathcal{F}_{\text{SIG}}^{\text{HtI}}(\lambda)$  under the collision-resistance of the hash function  $J : \{0, 1\}^* \rightarrow \mathbb{Z}_p \setminus \{0\}$  and the DL assumption, where  $\Pi_{\text{SIG}}$  denotes the instantiation of the signature scheme  $\Sigma_{\text{SIG}}$  by  $\Pi_{\text{CHF}}$ ,  $\Pi_{\text{CHF}2}$ ,  $\Pi_{\text{LPKE}}$  and  $\Pi_{\text{NIZK}}$ .

As a concrete construction for the chameleon hash function  $\Sigma_{\text{CHF}2}$ , we adopt the chameleon hash function  $\Pi_{\text{CHF},1}$  which is  $\Pi_{\text{CHF},n}$  in Fig. 1 with  $n = 1$ . The following corollary is obtained by Theorem 4, obviously.

**Corollary 1.** For any  $n \in \mathbb{N}$ ,  $\Pi_{\text{CHF},n}$  is strongly collision-resistant under the collision-resistance of the hash function  $J : \{0, 1\}^* \rightarrow \mathbb{Z}_p \setminus \{0\}$  and the DL assumption.

<p>LPKE.Gen(<math>1^\lambda, 1^l</math>) :</p> <p><math>(p, \mathbb{G}, g) \leftarrow \mathcal{G}(1^\lambda)</math>. <math>a_1, \dots, a_l, b_1, b_2 \xleftarrow{\text{U}} \mathbb{Z}_p</math>. <math>\hat{g}_0 := g, \hat{g}_1 := g^{b_1}, \hat{g}_2 := g^{b_2}, g_1 := g^{a_1}, \dots, g_l := g^{a_l}</math>.</p> <p><math>u_1, u_2, u_3, v_1, v_2, v_3, w_1, w_2, w_3 \xleftarrow{\text{U}} \mathbb{Z}_p</math>. <math>d_1 := \hat{g}_0^{u_1} \cdot \hat{g}_1^{u_2}, d_2 := \hat{g}_0^{u_1} \cdot \hat{g}_2^{u_3}</math>,</p> <p><math>e_1 := \hat{g}_0^{v_1} \cdot \hat{g}_1^{v_2}, e_2 := \hat{g}_0^{v_1} \cdot \hat{g}_2^{v_3}, h_1 := \hat{g}_0^{w_1} \cdot \hat{g}_1^{w_2}, h_2 := \hat{g}_0^{w_1} \cdot \hat{g}_2^{w_3}</math>.</p> <p><math>ek := (p, \mathbb{G}, \hat{g}_0, \hat{g}_1, \hat{g}_2, g_1, \dots, g_l, d_1, d_2, e_1, e_2, h_1, h_2)</math>. <math>dk := (u_1, u_2, u_3, v_1, v_2, v_3, w_1, w_2, w_3)</math>.</p> <p>Return <math>(ek, dk)</math>.</p>
<p>LPKE.Enc(<math>ek, x \in \{0, 1\}^l, L \in \{0, 1\}^*</math>) :</p> <p>For <math>i \in [1, l]</math>, the <math>i</math>-th bit of <math>x \in \{0, 1\}^l</math> is denoted by <math>x_i \in \{0, 1\}</math>.</p> <p>For every <math>i \in [1, l]</math>, do:</p> <p><math>r_i, s_i \xleftarrow{\text{U}} \mathbb{Z}_p</math>. <math>\mathbf{y}_i := (y_{i,1}, y_{i,2}, y_{i,3}) := (\hat{g}_0^{r_i+s_i}, \hat{g}_1^{r_i}, \hat{g}_2^{s_i})</math>. <math>z_i := h_1^{r_i} \cdot h_2^{s_i} \cdot g_i^{x_i}</math>.</p> <p><math>c_i := (d_1 \cdot e_1^{r_i})^{x_i} \cdot (d_2 \cdot e_2^{s_i})^{1-x_i}</math>, where <math>t_i := H_{CL}(\mathbf{y}_i, z_i, L)</math>. <math>\mathbf{c}_i := (\mathbf{y}_i, z_i, c_i)</math>.</p> <p>Return <math>\mathbf{C} := \{\mathbf{c}_i\}_{i \in [1, l]}</math>.</p>
<p>LPKE.Dec(<math>ek, dk, \mathbf{C}, L</math>) :</p> <p>For every <math>i \in [1, l]</math>, do:</p> <p><math>\tilde{c}_i := y_{i,1}^{u_1+t_i v_1} \cdot y_{i,2}^{u_2+t_i v_2} \cdot y_{i,3}^{u_3+t_i v_3}</math>, where <math>t_i := H_{CL}(\mathbf{y}_i, z_i, L)</math>.</p> <p>If <math>\tilde{c}_i \neq c_i</math>, then return <math>\perp</math>.</p> <p>Else if <math>z_i / (d_1^{w_1} \cdot d_2^{w_2} \cdot y_{i,3}^{w_3}) = g_i</math>, then <math>x'_i := 1</math>. Else, then <math>x'_i := 0</math>.</p> <p>The <math>i</math>-th bit of <math>x'</math> is set as <math>x'_i</math>.</p> <p>Return <math>x' \in \{0, 1\}^l</math>.</p>

**Fig. 2.** Construction of LPKE Scheme  $\Pi_{LPKE,l}$ , where  $H_{CL} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$  is a collision-resistant hash function.

Thus, the random trapdoor collision and strong collision-resistance of the CHF scheme  $\Pi_{CHF,1}$  are guaranteed by Theorem 3 and Corollary 1, respectively.

As a concrete construction for the LPKE scheme  $\Sigma_{LPKE}$ , we adopt  $\Pi_{LPKE,l}$  given in Fig. 2 which is described in the last page of this paper. The LPKE scheme is a modification of the LPKE scheme by Camenisch et al. [9] which is IND-LCCA secure<sup>6</sup> under the DLIN assumption and the collision-resistance of hash function. Faust et al. [15] modifies the scheme by Camenisch et al. to get the LPKE scheme  $\Pi_{LPKE,l}$  which achieves a weaker security, i.e., IND-wLCCA, but encrypts a plaintext of arbitrary length. Thus,

**Theorem 5.** *For any  $l \in \mathbb{N}$ ,  $\Pi_{LPKE,l}$  is IND-wLCCA under the collision-resistance of the hash function  $H_{CL} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$  and the DLIN assumption.*

As a concrete construction for the non-interactive zero-knowledge proof  $\Sigma_{NIZK}$ , we adopt the Groth-Sahai proof  $\Pi_{NIZK}$  in [17] whose soundness and zero-knowledge are guaranteed under the DLIN assumption.

By the schemes  $\Pi_{CHF,n}, \Pi_{CHF2}, \Pi_{NIZK}$  and  $\Pi_{LPKE,n\lambda}$ , where  $\lambda$  denotes the integer in the security parameter  $1^\lambda$  of  $\Pi_{CHF,n}$ , our concrete signature scheme  $\Pi_{SIG}$  is constructed. Hereafter, for  $i \in [1, n]$  and  $j \in [1, \lambda]$ , the  $j$ -th bit of  $x_i \in \mathbb{Z}_p$  in  $\Pi_{CHF,n}$  is denoted by  $x_{ij} \in \{0, 1\}$ , and the prime and group in  $\Pi_{LPKE,n\lambda}$  are

<sup>6</sup> IND-LCCA is stronger security notion than IND-wLCCA. For the details, refer to [15].

written as  $\hat{p}$  and  $\hat{\mathbb{G}}$ , respectively. By the signing algorithm of  $\Pi_{\text{SIG}}$ , a LPKE ciphertext  $\mathbf{C}$  and a NIZK proof  $\pi$  are generated as follows.

The ciphertext  $\mathbf{C}$  is generated by running  $\mathbf{C} \leftarrow \text{LPKE.Enc}(ek, (x_1, \dots, x_n), \bar{m})$ , where  $\text{LPKE.Enc}$  is the encryption algorithm of  $\Pi_{\text{LPKE}, n, \lambda}$ . The ciphertext  $\mathbf{C}$  is parsed as  $\{\mathbf{c}_{ij}\}_{i \in [1, n], j \in [1, \lambda]}$ , and  $\mathbf{c}_{ij}$  is parsed as  $(\mathbf{y}_{ij}, z_{ij} \in \hat{\mathbb{G}}, c_{ij} \in \hat{\mathbb{G}})$ .  $\mathbf{y}_{ij}$  is parsed as  $(y_{ij,1}, y_{ij,2}, y_{ij,3}) \in \hat{\mathbb{G}}^3$ .

By using the proof-generation algorithm of the NIZK scheme  $\Pi_{\text{NIZK}}$ , we generate the proof  $\pi$ . Actually, The proof  $\pi$  is a proof which proves that

$$\begin{aligned} & \exists \{r_{ij} \in \mathbb{Z}_{\hat{p}}, s_{ij} \in \mathbb{Z}_{\hat{p}}, x_{ij} \in \{0, 1\}\}_{i \in [1, n], j \in [1, \lambda]} \text{ such that} \\ & \left[ \prod_{i=1}^n \prod_{j=1}^{\lambda} g_i^{2^{j-1} \cdot x_{ij}} = y \right] \bigwedge_{i \in [1, n], j \in [1, \lambda]} \left[ [\hat{g}_0^{r_{ij} + s_{ij}} = y_{ij,1}] \wedge [\hat{g}_1^{r_{ij}} = y_{ij,2}] \wedge [\hat{g}_2^{s_{ij}} = y_{ij,3}] \right] \\ & \wedge [h_1^{r_{ij}} \cdot h_2^{s_{ij}} \cdot g_{ij}^{x_{ij}} = z_{ij}] \wedge [(d_1 \cdot e_1^{t_{ij}})^{r_{ij}} \cdot (d_2 \cdot e_2^{t_{ij}})^{s_{ij}} = c_{ij}] \wedge [x_{ij}(1 - x_{ij}) = 0], \end{aligned}$$

where  $y = \prod_{i=1}^n g_i^{x_i} \in \mathbb{G}$ .

**Acknowledgements.** This work was supported by JSPS KAKENHI (Grant Number JP17KT0081). This work was also supported by JSPS and DST under the Japan-India Science Cooperative Program.

## References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-00457-5\\_28](https://doi.org/10.1007/978-3-642-00457-5_28)
2. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-03356-8\\_3](https://doi.org/10.1007/978-3-642-03356-8_3)
3. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-28628-8\\_3](https://doi.org/10.1007/978-3-540-28628-8_3)
4. 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). [https://doi.org/10.1007/978-3-642-14623-7\\_1](https://doi.org/10.1007/978-3-642-14623-7_1)
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). [https://doi.org/10.1007/978-3-540-85174-5\\_7](https://doi.org/10.1007/978-3-540-85174-5_7)
6. Brakerski, Z., Kalai, Y.T., Katz, J., Vaikuntanathan, V.: Overcoming the hole in the bucket: public-key cryptography resilient to continual memory leakage. In: FOCS 2010, pp. 501–510 (2010)

7. Boneh, D., Shen, E., Waters, B.: Strongly unforgeable signatures based on computational Diffie-Hellman. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 229–240. Springer, Heidelberg (2006). [https://doi.org/10.1007/11745853\\_15](https://doi.org/10.1007/11745853_15)
8. Boyle, E., Segev, G., Wichs, D.: Fully leakage-resilient signatures. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 89–108. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-20465-4\\_7](https://doi.org/10.1007/978-3-642-20465-4_7)
9. Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-01001-9\\_20](https://doi.org/10.1007/978-3-642-01001-9_20)
10. Chow, S.S.M., Dodis, Y., Rouselakis, Y., Waters, B.: Practical leakage-resilient identity-based encryption from simple assumptions. IN: ACMCCS 2010, pp. 152–161 (2010)
11. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-24676-3\\_13](https://doi.org/10.1007/978-3-540-24676-3_13)
12. Dodis, Y., Goldwasser, S., Tauman Kalai, Y., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 361–381. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-11799-2\\_22](https://doi.org/10.1007/978-3-642-11799-2_22)
13. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 613–631. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-17373-8\\_35](https://doi.org/10.1007/978-3-642-17373-8_35)
14. Dodis, Y., Kalai, Y.T., Lovett, S.: On Cryptography with auxiliary input. In: STOC 2009, pp. 621–630 (2009)
15. Faust, S., Hazay, C., Nielsen, J.B., Nordholt, P.S., Zottarel, A.: Signature schemes secure against hard-to-invert leakage. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 98–115. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-34961-4\\_8](https://doi.org/10.1007/978-3-642-34961-4_8)
16. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC 2008, pp. 197–206 (2008)
17. 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). [https://doi.org/10.1007/978-3-540-78967-3\\_24](https://doi.org/10.1007/978-3-540-78967-3_24)
18. Huang, J., Huang, Q., Pan, C.: A black-box construction of strongly unforgeable signature schemes in the bounded leakage model. In: Chen, L., Han, J. (eds.) ProvSec 2016. LNCS, vol. 10005, pp. 320–339. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-47422-9\\_19](https://doi.org/10.1007/978-3-319-47422-9_19)
19. Halderman, J.A., et al.: Lest we remember: cold boot attacks on encryption keys. In: USENIX Security Symposium, pp. 45–60 (2008)
20. Huang, Q., Wong, D.S., Zhao, Y.: Generic transformation to strongly unforgeable signatures. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 1–17. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-72738-5\\_1](https://doi.org/10.1007/978-3-540-72738-5_1)
21. Kurosawa, K., Trieu Phong, L.: Leakage resilient IBE and IPE under the DLIN assumption. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 487–501. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-38980-1\\_31](https://doi.org/10.1007/978-3-642-38980-1_31)

22. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-10366-7\\_41](https://doi.org/10.1007/978-3-642-10366-7_41)
23. Lewko, A., Rouselakis, Y., Waters, B.: Achieving leakage resilience through dual system encryption. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 70–88. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-19571-6\\_6](https://doi.org/10.1007/978-3-642-19571-6_6)
24. Malkin, T., Teranishi, I., Vahlis, Y., Yung, M.: Signatures resilient to continual leakage on memory and computation. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 89–106. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-19571-6\\_7](https://doi.org/10.1007/978-3-642-19571-6_7)
25. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-03356-8\\_2](https://doi.org/10.1007/978-3-642-03356-8_2)
26. 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). [https://doi.org/10.1007/11967668\\_23](https://doi.org/10.1007/11967668_23)
27. Wang, Y., Matsuda, T., Hanaoka, G., Tanaka, K.: Signatures resilient to uninvertible leakage. In: Zikas, V., De Prisco, R. (eds.) SCN 2016. LNCS, vol. 9841, pp. 372–390. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-44618-9\\_20](https://doi.org/10.1007/978-3-319-44618-9_20)
28. Wang, Y., Tanaka, K.: Generic transformation to strongly existentially unforgeable signature schemes with leakage resiliency. In: Chow, S.S.M., Liu, J.K., Hui, L.C.K., Yiu, S.M. (eds.) ProvSec 2014. LNCS, vol. 8782, pp. 117–129. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-12475-9\\_9](https://doi.org/10.1007/978-3-319-12475-9_9)
29. Yuen, T.H., Chow, S.S.M., Zhang, Y., Yiu, S.M.: Identity-based encryption resilient to continual auxiliary leakage. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 117–134. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-29011-4\\_9](https://doi.org/10.1007/978-3-642-29011-4_9)
30. Yuen, T.H., Yiu, S.M., Hui, L.C.K.: Fully leakage-resilient signatures with auxiliary inputs. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 294–307. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-31448-3\\_22](https://doi.org/10.1007/978-3-642-31448-3_22)
31. Zhang, M.: New model and construction of ABE: achieving key resilient-leakage and attribute direct-revocation. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. LNCS, vol. 8544, pp. 192–208. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-08344-5\\_13](https://doi.org/10.1007/978-3-319-08344-5_13)
32. Zhang, M., Shi, W., Wang, C., Chen, Z., Mu, Y.: Leakage-resilient attribute-based encryption with fast decryption: models, analysis and constructions. In: Deng, R.H., Feng, T. (eds.) ISPEC 2013. LNCS, vol. 7863, pp. 75–90. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-38033-4\\_6](https://doi.org/10.1007/978-3-642-38033-4_6)
33. Zhang, M., Wang, C., Takagi, T., Mu, Y.: Functional encryption resilient to hard-to-invert leakage. *Comput. J.* **58**, 735–749 (2013). <https://doi.org/10.1093/comjnl/bxt105>



# A Revocable Group Signature Scheme with Scalability from Simple Assumptions and Its Implementation

Keita Emura<sup>(✉)</sup> and Takuya Hayashi

National Institute of Information and Communications Technology (NICT),  
Tokyo, Japan  
{k-emura,t-hayashi}@nict.go.jp

**Abstract.** Group signatures are signatures providing signer anonymity where signers can produce signatures on behalf of the group that they belong to. Although such anonymity is quite attractive considering privacy issues, it is not trivial to check whether a signer has been revoked or not. Thus, how to revoke the rights of signers is one of the major topics in the research on group signatures. In particular, scalability, where the signing and verification costs and the signature size are constant in terms of the number of signers  $N$ , and other costs regarding signers are at most logarithmic in  $N$ , is quite important. In this paper, we propose a revocable group signature scheme which is currently more efficient compared to previous all scalable schemes. Moreover, our revocable group signature scheme is secure under simple assumptions (in the random oracle model), whereas all scalable schemes are secure under  $q$ -type assumptions. Finally, we implemented our scheme by employing the Barreto-Lynn-Scott curves over a 455-bit prime field (BLS455), and the Barreto-Naehrig curves over a 382-bit prime field (BN382), respectively, by using the RELIC library. We showed that the running times of our signing algorithm were approximately 21 ms (BLS455) and 17 ms (BN382), and those of our verification algorithm were approximately 31 ms (BLS455) and 24 ms (BN382), respectively.

## 1 Introduction

### 1.1 Revocable Group Signature

Group signatures [19] have been widely recognized as an extension of digital signatures. In conventional signature schemes, a signer-specific public key is used for verifying signatures whereas in a group signature scheme, a group public key is used. Thus signers are anonymous since a verifier just verifies that a signer belongs to the group. Although such anonymity is quite attractive considering privacy issues, on the other hand, it makes it difficult to provide revocation function. Furthermore, how to revoke the rights of signers in group signatures is not trivial. In the early stage, either the signing cost or the verification cost depend on the number of revoked signers  $R$ . Camenisch and Lysyanskaya [18] proposed a method to revoke users by using accumulators, where the signing algorithm

requires  $O(R)$  computations. Boneh and Shacham [15] proposed group signatures with verifier-local revocation (VLR-GS). In VLR-GS [15, 38, 43, 45, 46], no signer is involved to the revocation procedure. As a drawback, the verification cost is  $O(R)$ , and is not constant. In 2009, Nakanishi, Fujii, Hira, and Funabiki [44] broke this barrier by proposing a revocable group signature scheme with constant costs for signing and verifying. One drawback of their scheme is that the public key size is  $O(\sqrt{N})$  where  $N$  is the maximal number of signers. Later, Fan, Hsu, and Manulis [29] proposed a revocable group signature scheme with not only constant signing/verification costs but also constant size public key. However, the size of the revocation list is  $O(N)$ . Slamanig et al. [54] proposed linking-based revocation, and gave an instantiation based on the Delerablée-Pointcheval group signature scheme [24] by employing a generic compiler [53]. They introduced a dedicated authority, which they call revocation authority (RA), that can extract a revocation token from signatures by using a secret linking key. By using a simple look-up operation (and cuckoo hashing), the constant-time revocation check is realized. However, signatures are not publicly verifiable in the sense that the revocation check requires the secret linking key. Recently, some VLR-type schemes realized sub-linear/constant verification costs [26, 37, 51]. As a drawback, these schemes do not provide unlinkability, that is, they employed linkable parts contained in signatures for efficiently executing verification procedure.

A major breakthrough was implemented by Libert, Peters, and Yung (LPY) [41] in 2012. In the LPY scheme, a group manager periodically publishes a revocation list that contains ciphertexts of broadcast encryption which will be decrypted by non-revoked signers. A non-revoked signer proves the decryption ability of a ciphertext. The LPY scheme is scalable in the sense that it provides not only constant signing and verification costs, but also other costs regarding signers are at most logarithmic in  $N$ . They gave two schemes based on the complete subtree (CS) and the subset difference (SD) methods [48]. They further improved the efficiency of the LPY schemes by proposing a revocable group signature scheme with constant-size certification [40]. As followers of the LPY works, revocable group signatures with compact revocation list size were proposed [5, 6, 47, 52].

## 1.2 Actual Efficiency of Revocable Group Signatures with Scalability

Although the LPY scheme and other similar schemes are “asymptotically” very efficient, these schemes are not sufficiently efficient in practice. One reason for its inefficiency is that these schemes did not rely on random oracles, but rather employed the Groth-Sahai proofs [33]. Of course, avoiding random oracles and constructing schemes in the standard model are quite meaningful from a theoretical point of view. However, in general, random oracles yield efficient schemes, especially, in the group signature context. For example, the signature size of the LPY scheme is approximately 100 group elements, and even if we exclude revocation functionality, the signature size of the Groth scheme [32] and the (non-revocable) Libert-Peters-Yung scheme [42], which are

recognized as the most efficient group signature schemes in the standard model, are approximately 50 group elements. On the other hand, we can construct (non-revocable) group signature schemes whose signature size are less than 10 group elements when random oracles are introduced, e.g., Boneh-Boyen-Shacham [14], Furukawa-Imai [31], Delerablée-Pointcheval [24], Bichsel et al. [13], Pointcheval-Sanders [50], Derler-Slamani [25], and Libert-Mouhartem-Peters-Yung [39].

In terms of the running time of signing and verification, Begum et al. [11] gave an implementation of the Nakanishi-Funabiki scheme [47], which is a revocable group signature with scalability secure in the standard model, where the running time of the signing algorithm and the verification algorithm are approximately 500 ms and 900 ms, respectively. They employed the Barreto-Naehrig (BN) curves [10] over a 254-bit prime field and the embedding degree is 12, and utilized a library based on the “Cross-twisted  $\chi$ -based Ate (Xt-Xate) pairing” [3]. On the other hand, Emura, Hayashi, and Ishida [27] proposed a group signature scheme with time-bound keys secure in the random oracle model, where each signing key is associated with an expiry time, and they showed that the running time of their signing and verification algorithms were less than 4 ms and 12 ms, respectively. They also employed the BN curves over a 254-bit prime field, and utilized the RELIC library [4]. Of course we cannot directly compare these two implementation results due to differences in the functionalities and the selection of the underlying elliptic curves and parameters. However, these results somewhat indicate that group signature schemes in the random oracle model are significantly more efficient than those in the standard model.

In actual usage, Intel Software Guard Extensions (SGX) [2] employs the Intel Enhanced Privacy Identification (EPID) scheme [1,17], and the EPID scheme builds on top of the Boneh-Boyen-Shacham group signature scheme [14] and the Furukawa-Imai group signature scheme [31]. These group signature schemes are secure in the random oracle model. Thus, improving efficiency of revocable group signature schemes in the random oracle model seems meaningful for a practical usage. To the best of our knowledge, the Ohara et al. revocable group signature scheme [49] is the only scheme that provides scalability in the random oracle model. The costs of the Ohara et al. scheme are asymptotically the same as those of the CS-based LPY group signature scheme [41]. Moreover the Ohara et al. scheme is significantly more efficient than the LPY scheme due to the random oracle. For example, the signature size of the Ohara et al. scheme is 18 group elements whereas that of the LPY scheme is 98 group elements. One drawback of the Ohara et al. scheme is the underlying complexity assumptions, i.e., their scheme relies on a  $q$ -type assumption. Due to the Cheon attack [20], employing  $q$ -type assumption should be avoided as much as possible. Thus, proposing an efficient revocable group signature scheme with scalability from simple assumptions is still an open problem.

### 1.3 Our Contribution

In this paper, we propose a revocable group signature scheme with scalability from simple assumptions, and give its implementation results. Our scheme is



**Table 1.** Comparison of revocable group signature schemes with scalability

Scheme	Public key size	Signature size <sup>a</sup>	Certificate size	Revocation list size	Signing cost	Verification cost	Std/ROM <sup>c</sup>	Assumption
LPY1(CS) [41]	$O(1)$	$O(1)$ (96)	$O(\log N)$	$O(R \cdot \log(N/R))$	$O(1)$	$O(1)$	Std	$q$ -Type
LPY2(SD) [41]	$O(\log N)$	$O(1)$ (96)	$O(\log^3 N)$	$O(R)$	$O(\log N)$ <sup>b</sup>	$O(1)$	Std	$q$ -Type
LPY3 [40]	$O(\log N)$	$O(1)$ (144)	$O(1)$	$O(R)$	$O(1)$	$O(1)$	Std	$q$ -Type
AEHS [5, 6]	$O(1)$	$O(1)$ (98)	$O(R_{\max})$	$O(1)$	$O(R)$ <sup>b</sup>	$O(1)$	Std	$q$ -Type
NF [47]	$O(T \log N)$	$O(1)$ (143)	$O(T)$	$O(R/T)$	$O(T)$ <sup>b</sup>	$O(1)$	Std	$q$ -Type
SN [52]	$O(T + \log N)$	$O(1)$ (299)	$O(1)$	$O(R/T)$	$O(T)$ <sup>b</sup>	$O(1)$	Std	$q$ -Type
Ohara et al. [49]	$O(1)$	$O(1)$ (18)	$O(\log N)$	$O(R \cdot \log(N/R))$	$O(1)$	$O(1)$	ROM	$q$ -Type
Our Scheme	$O(1)$	$O(1)$ (16)	$O(\log N)$	$O(R \cdot \log(N/R))$	$O(1)$	$O(1)$	ROM	Simple

$N$ : The maximum number of group members.  
 $R$ : The number of revoked signers.  
 $R_{\max}$ : The maximum number of revoked signers.  
 $T$ : The parameter of the accumulated/vector commitment value in [47, 52].  
<sup>a</sup> We denote the number of group elements contained in a group signature on  $(\cdot)$ . This number contains both the number of  $G$  elements and  $Z_p$  elements.  
<sup>b</sup> This complexity is only required at the first signature generation of each revocation epoch.  
<sup>c</sup> Standard Model/Random Oracle Model

more efficient than previous all scalable schemes. We summarized the efficiency of scalable schemes in Table 1. We employed the methodology proposed by Ohara et al. [49], where the group manager publishes a revocation list containing signatures of non-revoked signers, and a signer proves that a signature corresponding to the signer is contained in the revocation list. In addition to this, we employed the signature scheme proposed by Libert-Mouhartem-Peters-Yung (LMPY) [39] which is secure under a simple assumption. The signature size of the LMPY scheme is constant regardless of the number of message blocks due to the Kiltz-Wee quasi-adaptive non-interactive zero-knowledge (QA-NIZK) arguments for linear subspaces [36]. Libert et al. proposed a group signature scheme based on the LMPY signature scheme. Since the scheme does not provide revocation functionality, our scheme can also be seen as a modification of the Libert et al. group signature scheme by adding revocation functionality without additional complexity assumptions.

Finally, we implemented our scheme by employing the Barreto-Lynn-Scott curves [9] over a 455-bit prime field (BLS455), and the Barreto-Naehrig curves [10] over a 382-bit prime field (BN382), respectively, by using the RELIC library. We showed that the running times of our signing algorithm were approximately 21 ms (BLS455) and 17 ms (BN382), and those of our verification algorithm were approximately 31 ms (BLS455) and 24 ms (BN382), respectively. These implementation results indicate that our scheme is feasible in practical settings.

## 2 Preliminaries

### 2.1 Cryptographic Assumptions

In this subsection, we give the definitions of the Decisional Diffie-Hellman (DDH) assumption, the Symmetric eXternal Diffie-Hellman (SXDH) assumption, and the Symmetric Discrete Logarithm (SDL) assumption. Let  $\lambda \in \mathbb{N}$  be a security parameter. Let  $G$ ,  $\widehat{G}$ , and  $G_T$  be groups with prime order  $p > 2^\lambda$ , and  $e :$

$\mathbb{G} \times \widehat{\mathbb{G}} \rightarrow \mathbb{G}_T$  be a bilinear map. For  $g \in \mathbb{G}$  and  $\widehat{h} \in \widehat{\mathbb{G}}$ ,  $e(g, \widehat{h}) \neq 1_{\mathbb{G}_T}$  holds unless  $g \neq 1_{\mathbb{G}}$  and  $\widehat{h} \neq 1_{\widehat{\mathbb{G}}}$ .

**Definition 1 (DDH Assumption).** Let  $a, b \xleftarrow{\$} \mathbb{Z}_p^*$  and  $Z \xleftarrow{\$} \mathbb{G} \setminus \{g^{ab}\}$ . We say that the DDH assumption holds in  $\mathbb{G}$  if for any probabilistic polynomial time (PPT) adversary  $\mathcal{A}$ , the advantage  $\text{Adv}_{\text{DDH}}(\lambda) := |\Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) \rightarrow \text{true}] - \Pr[\mathcal{A}(g, g^a, g^b, Z) \rightarrow \text{true}]|$  is negligible.

**Definition 2 (SXDH Assumption).** We say that the SXDH assumption holds if the DDH assumption holds in both  $\mathbb{G}$  and  $\widehat{\mathbb{G}}$ .

**Definition 3 (SDL Assumption [39]).** Let  $a \xleftarrow{\$} \mathbb{Z}_p^*$ . We say that the SDL assumption holds in  $(\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T)$  if for any PPT adversary  $\mathcal{A}$ , the advantage  $\text{Adv}_{\text{SDL}}(\lambda) := \Pr[\mathcal{A}(g, \widehat{g}, g^a, \widehat{g}^a) \rightarrow a]$  is negligible.

### 2.2 QA-NIZK Arguments for Linear Subspaces

In this subsection, we introduce the Kiltz-Wee QA-NIZK arguments for linear subspaces [36] that prove membership in the row space of a matrix  $\mathbf{M}$ . As in [39], we assume that all algorithms take as input the description of common public parameters  $\text{cp} = (\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T, p)$ . In QA-NIZK proofs, the common reference string (CRS) may depend on the language to be proved. In the Kiltz-Wee case, it depends a matrix  $\mathbf{M}$ . As in [39], for soundness,  $\mathbf{M}$  is required to be witness-samplable where the reduction has to know the discrete logarithms of the group elements of  $\mathbf{M}$ .

Bold capital letters, such as  $\mathbf{M}$ , denote matrices, and bold lowercase letters, such as  $\mathbf{v}$ , denote vectors. For  $\mathbf{M} \in \mathbb{G}^{t \times n}$ , we denote  $\mathbf{M} = (M_{i,j})_{i \in [1,t], j \in [1,n]} = (\mathbf{M}_1, \dots, \mathbf{M}_t)^T$  where  $M_{i,j} \in \mathbb{G}$  for  $i \in [1, t]$  and  $j \in [1, n]$  and  $\mathbf{M}_i = (M_{i,1}, M_{i,2}, \dots, M_{i,n})$  for  $i \in [1, t]$ .

**QA.KeyGen(cp, M):** This CRS and trapdoor generation algorithm takes as input  $\text{cp}$  and a matrix  $\mathbf{M} = (M_{i,j})_{i \in [1,t], j \in [1,n]} \in \mathbb{G}^{t \times n}$ . Choose  $\widehat{g}_z \xleftarrow{\$} \widehat{\mathbb{G}}$  and a trapdoor  $\text{tk} = (\chi_1, \dots, \chi_n) \xleftarrow{\$} \mathbb{Z}_p^n$ . Compute  $\widehat{g}_j = \widehat{g}_z^{\chi_j}$  for all  $j \in [1, n]$ .

Compute  $z_i = \prod_{j=1}^n M_{i,j}^{-\chi_j}$  for all  $i \in [1, t]$ . Output the common reference string  $\text{crs} = (\{z_i\}_{i=1}^t, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^n) \in \mathbb{G}^t \times \widehat{\mathbb{G}}^{n+1}$  and the trapdoor  $\text{tk} \in \mathbb{Z}_p^n$ .

**Prove(crs, v, {ω<sub>i</sub>}\_{i=1}^t, {ĝ<sub>j</sub>}\_{j=1}^n):** The proof generation algorithm takes as input  $\text{crs} = (\{z_i\}_{i=1}^t, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^n)$ , a vector  $\mathbf{v}$ , and witnesses  $\{\omega_i\}_{i=1}^t$  where  $\mathbf{v} = \mathbf{M}_1^{\omega_1} \cdot \mathbf{M}_2^{\omega_2} \dots \mathbf{M}_t^{\omega_t} = (\prod_{i=1}^t M_{i,1}^{\omega_i}, \dots, \prod_{i=1}^t M_{i,n}^{\omega_i})$  holds. Output a proof  $\pi = \prod_{i=1}^t z_i^{\omega_i}$  which proves that  $\mathbf{v}$  is a linear combination of the rows of  $\mathbf{M}$ .

**Sim(tk, v):** The simulation algorithm takes as input  $\text{tk} = (\chi_1, \dots, \chi_n)$  and  $\mathbf{v} = (v_1, \dots, v_n) \in \mathbb{G}^n$ , and output a simulated proof  $\pi = \prod_{j=1}^n v_j^{-\chi_j}$ .

**Verify(crs, v, π):** The verify algorithm takes as input  $\text{crs} = (\{z_i\}_{i=1}^t, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^n)$ ,  $\mathbf{v} = (v_1, \dots, v_n)$ , and  $\pi$ , and output 1 if  $(v_1, \dots, v_n) \neq (1_{\mathbb{G}}, \dots, 1_{\mathbb{G}})$  and  $1_{\mathbb{G}_T} = e(\pi, \widehat{g}_z) \prod_{j=1}^n e(v_j, \widehat{g}_j)$  hold, and 0 otherwise.

As we can see,  $\prod_{j=1}^n e(v_j, \widehat{g}_j) = \prod_{j=1}^n e(v_j, \widehat{g}_z^{X_j}) = e(\prod_{j=1}^n v_j^{X_j}, \widehat{g}_z)$  hold. Since  $\pi = \prod_{j=1}^n v_j^{-X_j}$ , the equation above holds.

### 2.3 The LMPY Signature Scheme

In this subsection, we introduce a signature scheme proposed by Libert-Mouhartem-Peters-Yung (LMPY) [39] which is unforgeable under the SXDH assumption. The signature scheme can efficiently sign block messages in  $\mathbb{Z}_p^\ell$ . By employing the Kiltz-Wee QA-NIZK arguments, the signature size is constant regardless of the number of blocks  $\ell$ . Moreover, the verification algorithm requires just 5 pairings.

**Sig.KeyGen**( $\lambda, \ell$ ): The key generation algorithm takes as input a security parameter  $\lambda$  and the block size  $\ell$ . Choose bilinear groups  $\text{cp} = (\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T, p)$  where  $p > 2^\lambda$  and  $g \xleftarrow{\$} \mathbb{G}$  and  $\widehat{g} \xleftarrow{\$} \widehat{\mathbb{G}}$ . Choose  $\omega, a \xleftarrow{\$} \mathbb{Z}_p$  and set  $h = g^\omega$  and  $\Omega = h^\omega$ . Choose  $\mathbf{v} = (v_1, \dots, v_\ell, W) \xleftarrow{\$} \mathbb{G}^{\ell+1}$ . For  $g \in \mathbb{G}$  and the identity matrix  $\mathbf{I}_{\ell+1}$ , we denote  $g^{\mathbf{I}_{\ell+1}}$  as the  $(\ell + 1) \times (\ell + 1)$  matrix whose diagonal components are  $g$  and other all elements are  $1_{\mathbb{G}}$ . Let  $\mathbf{1}_{\ell+1} = (1_{\mathbb{G}}, \dots, 1_{\mathbb{G}}) \in \mathbb{G}^{\ell+1}$ . Set a matrix  $\mathbf{M} \in \mathbb{G}^{(\ell+2) \times (2\ell+4)}$  as

$$\mathbf{M} = \begin{pmatrix} g & \mathbf{1}_{\ell+1} & \mathbf{1}_{\ell+1} & h \\ \mathbf{v}^T & g^{\mathbf{I}_{\ell+1}} & h^{\mathbf{I}_{\ell+1}} & \mathbf{1}_{\ell+1}^T \end{pmatrix}$$

Run **QA.KeyGen**( $\text{cp}, \mathbf{M}$ ), and get  $\text{crs} = (\{z_i\}_{i=1}^{\ell+2}, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^{2\ell+4})$  and  $\text{tk} \in \mathbb{Z}_p^{2\ell+4}$ . Output the signing key  $\text{sk} = \omega$ , and the verification key  $\text{vk} = (\text{cp}, g, h, \widehat{g}, \mathbf{v}, \Omega, \text{crs})$ .

**Sig.Sign**( $\text{vk}, \text{sk}, \mathbf{m} = (m_1, \dots, m_\ell)$ ): The signing algorithm takes as input  $\text{vk} = (\text{cp}, g, h, \widehat{g}, \mathbf{v}, \Omega, \text{crs})$ ,  $\text{sk} = \omega$ , and messages to be signed  $\mathbf{m} = (m_1, \dots, m_\ell)$ . Choose  $s \xleftarrow{\$} \mathbb{Z}_p$  and compute  $\sigma_1 = g^\omega (v_1^{m_1} \dots v_\ell^{m_\ell} \cdot W)^s$ ,  $\sigma_2 = g^s$ , and  $\sigma_3 = h^s$ . Set  $\mathbf{v} = (\sigma_1, \sigma_2^{m_1}, \dots, \sigma_2^{m_\ell}, \sigma_2, \sigma_3^{m_1}, \dots, \sigma_3^{m_\ell}, \sigma_3, \Omega) \in \mathbb{G}^{2\ell+4}$ , and run the Prove algorithm to generate a proof  $\pi$  where  $\mathbf{v}$  is in the row space of  $\mathbf{M}$ . The QA-NIZK proof  $\pi \in \mathbb{G}$  is described as  $\pi = z_1^\omega (z_2^{m_1} \dots z_{\ell+1}^\ell \cdot z_{\ell+2})^s$ . Output the signature  $\sigma = (\sigma_1, \sigma_2, \sigma_3, \pi) \in \mathbb{G}^4$ .

**Sig.Verify**( $\text{vk}, \sigma, \mathbf{m}$ ): The verification algorithm takes as input  $\text{vk} = (\text{cp}, g, h, \widehat{g}, \mathbf{v}, \Omega, \text{crs})$ ,  $\sigma = (\sigma_1, \sigma_2, \sigma_3, \pi)$ , and  $\mathbf{m} = (m_1, \dots, m_\ell)$ . Output 1 if the following holds, and 0 otherwise.

$$\begin{aligned} & e(\Omega, \widehat{g}_{2\ell+4})^{-1} \\ & = e(\pi, \widehat{g}_z) e(\sigma_1, \widehat{g}_1) e(\sigma_2, \widehat{g}_2^{m_1} \dots \widehat{g}_{\ell+1}^{m_\ell} \cdot \widehat{g}_{\ell+2}) e(\sigma_3, \widehat{g}_{\ell+3}^{m_1} \dots \widehat{g}_{2\ell+2}^{m_\ell} \cdot \widehat{g}_{2\ell+3}) \end{aligned}$$

The following theorem was given in [39].

**Theorem 1** ([39]). *The LMPY signature scheme is existentially unforgeable under chosen-message attacks (EUF-CMA) if the SXDH assumption holds in  $(\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T)$ .*

We remark that a signature is publicly re-randomizable, i.e., for a valid signature-message pair  $\sigma = (\sigma_1, \sigma_2, \sigma_3, \pi)$  and  $\mathbf{m} = (m_1, \dots, m_\ell)$ , and a randomness  $s \in \mathbb{Z}_p$ ,  $(\sigma_1 \cdot (v_1^{m_1} \cdots v_\ell^{m_\ell} \cdot W)^s, \sigma_2 \cdot g^s, \sigma_3 \cdot h^s, \pi \cdot (z_2^{m_1} \cdots z_{\ell+1}^\ell \cdot z_{\ell+2})^s)$  is also a valid signature on  $\mathbf{m}$ .

## 2.4 Sigma Protocols

Sigma protocols are three-move honest-verifier zero-knowledge protocols. First the prover takes as input a statement and a witness, and sends a commitment  $\text{com}$  to the verifier. Next, the verifier, that also takes as input the statement, sends a challenge  $\text{chall}$  to the prover. Finally, the prover sends a response  $\text{resp}$  to the verifier. We require special soundness, where given two accepted transcripts  $(\text{com}_1, \text{chall}_1, \text{resp}_1)$  and  $(\text{com}_2, \text{chall}_2, \text{resp}_2)$  with the condition  $\text{com}_1 = \text{com}_2$  and  $\text{chall}_1 \neq \text{chall}_2$ , there is an extractor  $\text{Extract}$  that takes as input a statement  $s$  and two transcripts above, and outputs a witness  $\omega$  that satisfies  $L(s, \omega) = 1$ . Moreover, we also require special honest verifier zero knowledge (SHVZK), where there is a simulator  $\text{Sim}$  that takes as input a statement  $s$  and a challenge  $\text{chall}$ , and outputs a transcript  $(\text{com}, \text{chall}, \text{resp})$  that is indistinguishable from transcripts produced by the prover and the verifier as above. See [39] for the formal definition.

Moreover, we require sigma protocols to have quasi unique responses. Informally, for a statement  $s$ , and first two moves of the protocol,  $\text{com}$  and  $\text{chall}$ , no adversary can find responses  $\text{resp}$  and  $\text{resp}'$  which are both accepted but  $\text{resp} \neq \text{resp}'$ . If the success probability is zero, then it is called unique responses, or it is also known as strict soundness [55]. See [30] for the formal definition. Faust et al. [30] proved that if a sigma protocol has quasi unique responses, then the NIZK proof system derived via the Fiat-Shamir transformation is simulation-sound. Simulation soundness guarantees that even after seeing accepting proofs produced by the simulator, for both true and false statements, soundness holds. In our group signature scheme, the underlying NIZK proof system is required to be simulation sound for providing CCA anonymity where an adversary is allowed to access the opening oracle.

## 2.5 Complete Subtree Method

In this section, we introduce the Complete Subtree (CS) method [48]. Let  $\mathcal{N}$  be the set of all signers, and  $\mathcal{R} \subset \mathcal{N}$  be the set of revoked signers. By using the CS method,  $\mathcal{N} \setminus \mathcal{R}$  is divided into  $\text{num}$  disjoint sets such as  $\mathcal{N} \setminus \mathcal{R} = S_1 \cup \dots \cup S_{\text{num}}$ , and  $\text{num} = O(R \cdot \log(N/R))$  where  $N = |\mathcal{N}|$  and  $R = |\mathcal{R}|$ .

**Definition 4 (CS Algorithm).** *The CS algorithm takes as input a binary tree BT and a set of revoked signers  $\mathcal{R}_t$  where  $i \in \mathcal{R}_t$  when a signer with index  $i$  is revoked at time  $t$ , and outputs a set of nodes. The description of CS is given below.*

$\text{CS}(\text{BT}, \mathcal{R}_t) :$   
 $\mathbf{X}, \mathbf{Y} \leftarrow \emptyset;$   
 $\forall i \in \mathcal{R}_t$   
    Add  $\text{Path}(i)$  to  $\mathbf{X};$   
 $\forall x \in \mathbf{X}$   
    If  $x_{\text{left}} \notin \mathbf{X}$  then add  $x_{\text{left}}$  to  $\mathbf{Y};$   
    If  $x_{\text{right}} \notin \mathbf{X}$  then add  $x_{\text{right}}$  to  $\mathbf{Y};$   
    If  $|\text{RL}_t| = 0$  then add  $\text{root}$  to  $\mathbf{Y};$   
Return  $\mathbf{Y};$

In our group signature scheme, a signer, who has the identity  $\text{ID}$ , and whose path is  $\{u_0, u_1, \dots, u_\ell\}$ , has LMPY signatures on messages  $(\text{ID}, u_j)$  as certificates for all  $u_j \in \{u_0, u_1, \dots, u_\ell\}$ . The revocation list at time  $t$  also contains LMPY signatures on messages  $(t, u'_j)$  for all  $u'_j \in \{u'_0, u'_1, \dots, u'_{\text{num}}\}$  which is determined by the CS method. If the signer is not revoked at  $t$ , then there exists a node  $u$  such that  $u \in \{u_0, u_1, \dots, u_\ell\} \cap \{u'_0, u'_1, \dots, u'_{\text{num}}\}$ . The signer proves that the knowledge of two LMPY signatures on  $(\text{ID}, u)$  and  $(t, u')$ , and  $u = u'$ . Here, the current time  $t$  is not required to be hidden, and is not a witness.

### 3 Revocable Group Signatures

In this section, we give the syntax and correctness definitions of revocable group signature. We use the LPY definitions [40,41] which are modified from the Kiayias-Yung (KY) model [34] to match the revocation functionality.

A revocable group signature scheme  $\mathcal{R}\text{-}\mathcal{GS}$  consists of 6 algorithms (Setup, Join, Revoke, Sign, Verify, Open) as follows:

#### Definition 5 (Revocable Group Signature).

**Setup**( $1^\lambda, N$ ): *The setup algorithm takes as inputs a security parameter  $\lambda \in \mathbb{N}$  and a maximal number of members  $N \in \mathbb{N}$ , and outputs a group public key  $\text{gpk}$ , the group manager (GM) private key for revocation  $\mathcal{S}_{\text{GM}}$ , and the opening authority (OA) private key for opening  $\mathcal{S}_{\text{OA}}$ . Moreover, the algorithm initializes a public state  $St$  comprising a set data structure  $St_{\text{users}} = \emptyset$  and a string data structure  $St_{\text{trans}} = \epsilon$ .*

**Join** $^{\text{GM}, \mathcal{U}_i}$ : *The interactive protocol for joining between GM and a signer  $\mathcal{U}_i$  (whose identity is  $\text{ID}_i$ ) involves two interactive Turing machines  $J_{\text{user}}$  and  $J_{\text{GM}}$  which execution is denoted as  $[J_{\text{user}}(\text{gpk}), J_{\text{GM}}(St, \text{gpk}, \mathcal{S}_{\text{GM}})]$ .  $\mathcal{U}_i$  obtains a membership secret  $\text{sec}_i$  and a membership certificate  $\text{cert}_i$ . We assume that  $\text{ID}_i$  is contained in  $\text{sec}_i$ . If the protocol is successfully done, GM updates  $St_{\text{users}} \leftarrow St_{\text{users}} \cup \{\text{ID}_i\}$  and  $St_{\text{trans}} \leftarrow St_{\text{trans}} \parallel \langle i, \text{transcript}_i \rangle$ .*

**Revoke**( $\text{gpk}, \mathcal{S}_{\text{GM}}, t, \mathcal{R}_t \subset St_{\text{users}}$ ): *The revocation algorithm takes as input  $\text{gpk}$ ,  $\mathcal{S}_{\text{GM}}$ , a revocation epoch  $t$ , and a set of revoked signers  $\mathcal{R}_t \subset St_{\text{users}}$ , and outputs an updated revocation list  $\text{RL}_t$  which contains  $\mathcal{R}_t$ .*

$\text{Sign}(\text{gpk}, t, RL_t, \text{cert}, \text{sec}, M)$ : The signing algorithm takes as input  $\text{gpk}$ , a time  $t$ ,  $RL_t$ ,  $\text{cert}$ ,  $\text{sec}$ , and a message  $M$  to be signed, and outputs  $\perp$  if  $\text{ID} \in \mathcal{R}_t$ , and a group signature  $\Sigma$ , otherwise.

$\text{Verify}(\text{gpk}, t, RL_t, \Sigma, M)$ : The verification algorithm takes as input  $\text{gpk}$ ,  $t$ ,  $RL_t$ ,  $\Sigma$ , and  $M$ , and outputs 1 or 0 which mean valid or invalid, respectively.

$\text{Open}(\text{gpk}, \mathcal{S}_{\text{OA}}, t, \Sigma, M, St)$ : The opening algorithm takes as input  $\text{gpk}$ ,  $\mathcal{S}_{\text{OA}}$ ,  $t$ ,  $\Sigma$ ,  $M$ , and  $St := (St_{\text{users}}, St_{\text{trans}})$ , and outputs  $\perp$  if the opening is failure, and  $i$  such that  $\text{ID}_i \in St_{\text{users}} \cup \{\perp\}$ , otherwise.

Next, we define correctness. Let  $St$  be a public state, and  $St$  is said to be valid if it can be reached from  $St = (\emptyset, \epsilon)$  by a Turing machine having oracle access to  $J_{\text{GM}}$ . A state  $St'$  is said to be extended another state  $St$  if it can be reached from  $St$ . As in [34, 40, 41] we use the notation  $\text{cert}_i \stackrel{\text{gpk}}{=} \text{sec}_i$  to express that there exist coin tosses  $\varpi$  for  $J_{\text{GM}}$  and  $J_{\text{user}}$  such that, for some valid state  $St'$ , the execution of  $[J_{\text{user}}(\text{gpk}), J_{\text{GM}}(St, \text{gpk}, \mathcal{S}_{\text{GM}})](\varpi)$  provides  $J_{\text{user}}$  with  $\langle i, \text{cert}_i, \text{sec}_i \rangle$ .

**Definition 6 (Correctness).** We say that a revocable group signature scheme  $\mathcal{R}\text{-GS}$  is correct if:

1. In a valid state  $St = (St_{\text{users}}, St_{\text{trans}})$ , the condition  $|St_{\text{users}}| = |St_{\text{trans}}|$  holds, and no two entries of  $St_{\text{trans}}$  can contain certificates with the same tag.
2. If  $[J_{\text{user}}(\text{gpk}), J_{\text{GM}}(St, \text{gpk}, \mathcal{S}_{\text{GM}})]$  is honestly run by both parties and  $\langle i, \text{cert}_i, \text{sec}_i \rangle$  is obtained by  $J_{\text{user}}$ , then  $\text{cert}_i \stackrel{\text{gpk}}{=} \text{sec}_i$  holds.
3. For each  $t$  and any  $\langle i, \text{cert}_i, \text{sec}_i \rangle$  satisfying condition 2,  $\text{Verify}(\text{gpk}, t, RL_t, \text{Sign}(\text{gpk}, t, RL_t, \text{cert}, \text{sec}, M), M) = 1$  holds if  $i \notin \mathcal{R}_t$ .
4. For any  $\langle i, \text{cert}_i, \text{sec}_i \rangle$  resulting from the interaction  $[J_{\text{user}}(\cdot, \cdot), J_{\text{GM}}(\cdot, St, \cdot, \cdot)]$  for some valid state  $St$ , any  $t$  s.t.  $i \notin \mathcal{R}_t$ ,  $\text{Open}(\text{gpk}, \mathcal{S}_{\text{OA}}, t, \Sigma, M, St) = i$  holds where  $\Sigma \leftarrow \text{Sign}(\text{gpk}, t, RL_t, \text{cert}, \text{sec}, M)$ .

Three security definitions, misidentification, non-frameability, and anonymity [41] are given in the full version of this paper.

## 4 Proposed Revocable Group Signature Scheme

In this section, we give our proposed revocable group signature scheme. In our group signature scheme, two LMPY signature schemes with  $\ell = 2$  are setup. Let  $\text{sk} = \omega$  and  $\text{vk} = (\text{cp}, g, h, \hat{g}, \mathbf{v}, \Omega, \text{crs})$ , where  $\text{crs} = (\{z_i\}_{i=1}^4, \hat{g}_z, \{\hat{g}_j\}_{j=1}^8)$ , are for the first scheme, and  $\text{sk}' = \omega'$  and  $\text{vk}' = (\text{cp}, g', h', \hat{g}', \mathbf{v}', \Omega', \text{crs}')$ , where  $\text{crs}' = (\{z'_i\}_{i=1}^4, \hat{g}'_z, \{\hat{g}'_j\}_{j=1}^8)$ , are for the second scheme. The first scheme signs  $\mathbf{m} = (\text{ID}, u)$  where  $\text{ID}$  is the identity of a signer, and  $u$  is a node of the binary tree. Let  $(\sigma_1, \sigma_2, \sigma_3, \pi)$  be its signature. The second scheme signs  $\mathbf{m}' = (t, u)$  where  $t$  is the current time and  $u$  is a node of the binary tree. Let  $(\sigma'_1, \sigma'_2, \sigma'_3, \pi')$  be its signature.

In the signing algorithm, first, the signer re-randomizes  $(\sigma_1, \sigma_2, \sigma_3, \pi)$  and  $(\sigma'_1, \sigma'_2, \sigma'_3, \pi')$ , and let  $\tilde{\sigma} = (\tilde{\sigma}_1, \tilde{\sigma}_2, \tilde{\sigma}_3, \tilde{\pi})$  and  $\tilde{\sigma}' = (\tilde{\sigma}'_1, \tilde{\sigma}'_2, \tilde{\sigma}'_3, \tilde{\pi}')$  be the signatures after re-randomization. Then,  $(\tilde{\sigma}_2, \tilde{\sigma}_3)$  and  $(\tilde{\sigma}'_2, \tilde{\sigma}'_3)$  are independent from

the signed messages and other signatures. Thus, these can be directly included in a group signature. Informally, for the NP language  $L_{\text{LMPY}}$  induced by the relation  $R_{\text{LMPY}}(\text{vk}, \text{vk}', (\tilde{\sigma}, \mathbf{m}), (\tilde{\sigma}', \mathbf{m}')) = 1$  iff  $\text{Sig.Verify}(\text{vk}, \tilde{\sigma}, \mathbf{m}) = 1$ ,  $\text{Sig.Verify}(\text{vk}', \tilde{\sigma}', \mathbf{m}') = 1$ , and  $m_2 = m'_2 (= u)$ , the verifier accepts if  $L_{\text{LMPY}}(\text{s}, (\tilde{\sigma}, \mathbf{m}), (\tilde{\sigma}', \mathbf{m}')) = 1$  where the statement  $\text{s}$  here is the verification equation of the LMPY scheme, and  $((\tilde{\sigma}_1, \tilde{\pi}), (\text{ID}, u))$  and  $((\tilde{\sigma}'_1, \tilde{\pi}'), u)$  are witnesses. We convert the sigma protocol via the Fiat-Shamir transformation. To hide the other part, the signer encrypts  $(\tilde{\sigma}_1, \tilde{\pi})$  and  $(\tilde{\sigma}'_1, \tilde{\pi}')$  via the Cramer-Shoup encryption scheme [23] by using the public key of the group manager. We remark that, in the original Cramer-Shoup scheme, designated verifier NIZK proofs are employed for the validity check of ciphertexts. That is, the validity of ciphertexts can be checked by the decryptor who has the decryption key. Since group signatures are required to be publicly verifiable, as in [39, 49] we employ publicly verifiable NIZK proofs constructed from sigma protocols via the Fiat-Shamir transformation for the validity check. Moreover, for efficiency purposes, we employ the Cramer-Shoup encryption scheme with a randomness-reuse variant [12], and a randomness  $\theta$  is re-used for encrypting plural messages. In addition to these LMPY signatures, the signer also encrypts  $V_{\text{ID}} = v_1^{\text{ID}}$  and  $V_u = v_2^u$ . The former is required to search the corresponding certificate  $\text{cert}_i = (i, \text{Path}(i), V_{\text{ID}}, \{(\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3}, \pi_j, V_u^{(j)})\}_{u_j \in \text{Path}(i)})$  from joining transcripts in the open algorithm. The latter is also required to search  $j$  such that  $V_u = V_u^{(j)} = v_2^{u_j}$  for obtaining  $u_j \in \text{Path}(i)$  in the open algorithm. Finally, the signer proves the knowledge of  $\text{ID}$ ,  $u$ , and  $\theta$ , and also proves that two LMPY signatures sign on the same node  $u$ .

We give our revocable group signature scheme as follows.

**Setup**( $1^\lambda, N$ ): Choose bilinear groups  $\text{cp} = (\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T, p)$  where  $p > 2^\lambda$  and  $g \stackrel{\$}{\leftarrow} \mathbb{G}$  and  $\widehat{g} \stackrel{\$}{\leftarrow} \widehat{\mathbb{G}}$ . Choose  $g' \stackrel{\$}{\leftarrow} \mathbb{G}$  and  $\widehat{g}' \stackrel{\$}{\leftarrow} \widehat{\mathbb{G}}$ . Choose a hash function  $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$  which is modeled as a random oracle.

1. *Generate Two Key Pairs of the LMPY Scheme*: Choose  $\mathbf{v} = (v_1, v_2, W) \stackrel{\$}{\leftarrow} \mathbb{G}^3$  and  $\mathbf{v}' = (v'_1, v'_2, W') \stackrel{\$}{\leftarrow} \mathbb{G}^3$ . Choose  $a, a' \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ , and compute  $h = g^a$  and  $h' = g'^{a'}$ . Set  $\ell = 2$  in the LMPY signature scheme, and set matrices  $\mathbf{M}$  and  $\mathbf{M}'$  as follows.

$$\mathbf{M} = \begin{pmatrix} g & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h \\ v_1 & g & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & \\ v_2 & 1_{\mathbb{G}} & g & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h & 1_{\mathbb{G}} & 1_{\mathbb{G}} & \\ W & 1_{\mathbb{G}} & 1_{\mathbb{G}} & g & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h & 1_{\mathbb{G}} & \end{pmatrix}, \mathbf{M}' = \begin{pmatrix} g' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h' \\ v'_1 & g' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & 1_{\mathbb{G}} & \\ v'_2 & 1_{\mathbb{G}} & g' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & \\ W' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & g' & 1_{\mathbb{G}} & 1_{\mathbb{G}} & h' & 1_{\mathbb{G}} & \end{pmatrix}$$

Run  $\text{QA.KeyGen}(\text{cp}, \mathbf{M})$  and  $\text{QA.KeyGen}(\text{cp}, \mathbf{M}')$  of the QA-NIZK argument, and get  $\text{crs} = (\{z_i\}_{i=1}^4, \widehat{g}_z, \{\widehat{g}_j\}_{j=1}^8)$  and  $\text{crs}' = (\{z'_i\}_{i=1}^4, \widehat{g}'_z, \{\widehat{g}'_j\}_{j=1}^8)$ . Set  $\text{sk} = \omega$ ,  $\text{vk} = (\text{cp}, g, h, \widehat{g}, \mathbf{v}, \Omega = h^\omega, \text{crs})$ ,  $\text{sk}' = \omega'$ , and  $\text{vk}' = (\text{cp}, g', h', \widehat{g}', \mathbf{v}', \Omega' = h'^{\omega'}, \text{crs}')$ .

2. *Generate a Key Pair of the Cramer-Shoup Scheme (with a Randomness-reuse Variant)*: Choose  $x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x'_z, y'_z, x'_\sigma, y'_\sigma \xleftarrow{\$} \mathbb{Z}_p$  and compute  $X_z = g^{x_z} h^{y_z}$ ,  $X_\sigma = g^{x_\sigma} h^{y_\sigma}$ ,  $X_{ID} = g^{x_{ID}} h^{y_{ID}}$ ,  $X_u = g^{x_u} h^{y_u}$ ,  $X'_z = g^{x'_z} h^{y'_z}$ , and  $X'_\sigma = g^{x'_\sigma} h^{y'_\sigma}$ .

3. Output  $\text{gpk} = (\text{vk}, \text{vk}', X_z, X_\sigma, X_{ID}, X_u, X'_z, X'_\sigma)$ ,  $\mathcal{S}_{\text{GM}} = (\text{sk}, \text{sk}')$ , and  $\mathcal{S}_{\text{OA}} = (x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x'_z, y'_z, x'_\sigma, y'_\sigma)$ .

$\text{Join}^{\text{GM}, \mathcal{U}_i}$ : A signer  $\mathcal{U}_i$  and GM run the following interactive protocol.

1.  $\mathcal{U}_i$  chooses  $\text{ID}_i \xleftarrow{\$} \mathbb{Z}_p$ , computes  $V_{\text{ID}} = v_1^{\text{ID}_i}$ ,  $Z_{\text{ID}} = z_2^{\text{ID}_i}$ ,  $\widehat{G}_{2, \text{ID}} = \widehat{g}_2^{\text{ID}_i}$ , and  $\widehat{G}_{5, \text{ID}} = \widehat{g}_5^{\text{ID}_i}$ , and sends  $(V_{\text{ID}}, Z_{\text{ID}}, \widehat{G}_{2, \text{ID}}, \widehat{G}_{5, \text{ID}})$  to GM.
2. If  $V_{\text{ID}}$  has been appeared in transcripts of  $St$ , then GM aborts. Otherwise, GM checks the following equations hold.

$$\begin{aligned} e(V_{\text{ID}}, \widehat{g}_2) &= e(v_1, \widehat{G}_{2, \text{ID}}), e(Z_{\text{ID}}, \widehat{g}_2) = e(z_2, \widehat{G}_{2, \text{ID}}) \\ e(V_{\text{ID}}, \widehat{g}_5) &= e(v_1, \widehat{G}_{5, \text{ID}}), \end{aligned}$$

If all tests pass, GM samples a fresh index  $i \in \mathbb{Z}_p$  and a fresh leaf node (then  $\text{Path}(i)$  is fixed), and sends  $i$  to  $\mathcal{U}_i$ . Otherwise, GM aborts.

3. *Prove the Knowledge of  $\text{ID}_i$* :  $\mathcal{U}_i$  runs an interactive zero-knowledge proof of knowledge of  $\text{ID}_i = \log_{v_1}(V_{\text{ID}})$  in interaction with GM. We employ the Cramer-Damgård-MacKenzie transformation [22] which converts a sigma protocol into a perfect zero-knowledge proof of knowledge. Let  $\pi_K(\text{ID})$  be the interaction transcript.

4. *Generate LMPY Signatures as a Certificate*: For all  $u_j \in \text{Path}(i)$ , GM chooses  $s_j \xleftarrow{\$} \mathbb{Z}_p$  and computes a LMPY signature on messages  $(\text{ID}_i, u_j)$  by using  $\text{sk} = \omega$  such that compute  $V_u^{(j)} = v_2^{u_j}$  and  $\sigma_{j,1} = g^\omega(V_{\text{ID}} \cdot V_u^{(j)} \cdot W)^{s_j}$ ,  $\sigma_{j,2} = g^{s_j}$ ,  $\sigma_{j,3} = h^{s_j}$ , and  $\pi_j = z_1^\omega(Z_{\text{ID}} \cdot z_3^{u_j} \cdot z_4)^{s_j}$ . Finally, GM sends  $\text{cert}_i = (i, \text{Path}(i), V_{\text{ID}}, \{(\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3}, \pi_j, V_u^{(j)})\}_{u_j \in \text{Path}(i)})$  to  $\mathcal{U}_i$ .

5. Finally, GM stores  $\text{transcript}_i = ((Z_{\text{ID}}, \widehat{G}_{2, \text{ID}}, \widehat{G}_{5, \text{ID}}), \pi_K(\text{ID}), \text{cert}_i)$  and  $\mathcal{U}_i$  stores  $(\text{cert}_i, \text{sec}_i)$  where  $\text{sec}_i = \text{ID}_i$ .

$\text{Revoke}(\text{gpk}, \mathcal{S}_{\text{GM}}, t, \mathcal{R}_t \subset St_{\text{users}})$ : Run  $Y \leftarrow \text{CS}(\text{BT}, \mathcal{R}_t)$ . For all  $u_j \in Y$ , GM

chooses  $s_j \xleftarrow{\$} \mathbb{Z}_p$  and computes a LMPY signature on messages  $(t, u_j)$  by using  $\text{sk}' = \omega'$  such that  $\sigma'_{j,1} = g^{\omega'}(v_1^t \cdot v_2^{u_j} \cdot W)^{s_j}$ ,  $\sigma'_{j,2} = g^{s_j}$ ,  $\sigma'_{j,3} = h^{s_j}$ , and  $\pi'_j = z_1^{\omega'}(z_2^t \cdot z_3^{u_j} \cdot z_4)^{s_j}$ . Output  $RL_t = (t, Y, \{(\sigma'_{j,1}, \sigma'_{j,2}, \sigma'_{j,3}, \pi'_j, u_j)\}_{u_j \in Y})$ .

$\text{Sign}(\text{gpk}, t, RL_t, \text{cert}_i, \text{sec}_i, M)$ : Let  $u$  be a node where  $u \in \text{Path}(i) \cap Y$ . If  $u$  does not exist, then output  $\perp$ . Let  $(\sigma_1, \sigma_2, \sigma_3, \pi)$  be a LMPY signature on  $(\text{ID}_i, u)$  contained in  $\text{cert}_i$ , and let  $(\sigma'_1, \sigma'_2, \sigma'_3, \pi')$  be a LMPY signature on  $(t, u)$  contained in  $RL_t$ . Choose  $s, s' \xleftarrow{\$} \mathbb{Z}_p$  and re-randomize signatures as follows.

$$\begin{aligned} \widetilde{\sigma}_1 &= \sigma_1 \cdot (v_1^{\text{ID}_i} \cdot v_2^u \cdot W)^s, \widetilde{\sigma}_2 = \sigma_2 \cdot g^s, \widetilde{\sigma}_3 = \sigma_3 \cdot h^s, \widetilde{\pi} = \pi \cdot (z_2^{\text{ID}_i} \cdot z_3^u \cdot z_4)^s \\ \widetilde{\sigma}'_1 &= \sigma'_1 \cdot (v_1^t \cdot v_2^u \cdot W)^{s'}, \widetilde{\sigma}'_2 = \sigma'_2 \cdot g^{s'}, \widetilde{\sigma}'_3 = \sigma'_3 \cdot h^{s'}, \widetilde{\pi}' = \pi' \cdot (z_2^t \cdot z_3^u \cdot z_4)^{s'} \end{aligned}$$

Choose  $\theta \xleftarrow{\$} \mathbb{Z}_p$  and compute a ciphertext of the Cramer-Shoup encryption  $C_{\text{CS}} = (C_1, C_2, C_z, C_\sigma, C_{\text{ID}}, C_u, C'_z, C'_\sigma)$  such that  $C_1 = g^\theta$ ,  $C_2 = h^\theta$ ,  $C_z =$



$\tilde{\pi} \cdot X_z^\theta$ ,  $C_\sigma = \tilde{\sigma}_1 \cdot X_\sigma^\theta$ ,  $C_{ID} = v_1^{\text{ID}_i} \cdot X_{ID}^\theta$ ,  $C_u = v_2^u \cdot X_u^\theta$ ,  $C'_z = \tilde{\pi}' \cdot X_z'^\theta$ , and  $C'_\sigma = \tilde{\sigma}'_1 \cdot X_\sigma'^\theta$ . Here, the randomness  $\theta$  is re-used. Then, prove the knowledge of  $(\text{ID}_i, \theta, u)$ . Namely, choose  $r_{\text{ID}}, r_\theta, r_u \xleftarrow{\$} \mathbb{Z}_p$ , and compute  $R_1 = g^{r_\theta}$ ,  $R_2 = h^{r_\theta}$ ,  $R_3 = v_1^{r_{\text{ID}}} \cdot X_{\text{ID}}^{r_\theta}$ ,  $R_4 = v_2^{r_u} \cdot X_u^{r_\theta}$ , and  $R_5$  and  $R_6$  such that

$$\begin{aligned}
 R_5 &= (e(X_z, \hat{g}_z)e(X_\sigma, \hat{g}_1))^{r_\theta} (e(\tilde{\sigma}_2, \hat{g}_2)e(\tilde{\sigma}_3, \hat{g}_5))^{-r_{\text{ID}}} (e(\tilde{\sigma}_2, \hat{g}_3)e(\tilde{\sigma}_3, \hat{g}_6))^{-r_u} \\
 R_6 &= (e(X'_z, \hat{g}'_z)e(X'_\sigma, \hat{g}'_1))^{r_\theta} (e(\tilde{\sigma}'_2, \hat{g}'_3)e(\tilde{\sigma}'_3, \hat{g}'_6))^{-r_u}
 \end{aligned}$$

Compute  $c \leftarrow H(\text{gpk}, C_{\text{CS}}, \tilde{\sigma}_2, \tilde{\sigma}_3, \tilde{\sigma}'_2, \tilde{\sigma}'_3, R_1, \dots, R_6, M)$ ,  $s_{\text{ID}} = r_{\text{ID}} + c \cdot \text{ID}_i$ ,  $s_\theta = r_\theta + c \cdot \theta$ , and  $s_u = r_u + c \cdot u$ .<sup>1</sup>

Output a group signature  $\Sigma = (C_{\text{CS}}, \tilde{\sigma}_2, \tilde{\sigma}_3, \tilde{\sigma}'_2, \tilde{\sigma}'_3, c, s_{\text{ID}}, s_\theta, s_u) \in \mathbb{G}^{12} \times \mathbb{Z}_p^4$ .

**Verify(gpk, t, RL<sub>t</sub>, Σ, M):** Compute  $\bar{R}_1 = g^{s_\theta} \cdot C_1^{-c}$ ,  $\bar{R}_2 = h^{s_\theta} \cdot C_2^{-c}$ ,  $\bar{R}_3 = v_1^{s_{\text{ID}}} \cdot X_{\text{ID}}^{s_\theta} \cdot C_{ID}^{-c}$ ,  $\bar{R}_4 = v_2^{s_u} \cdot X_u^{s_\theta} \cdot C_u^{-c}$ , and  $\bar{R}_5$  and  $\bar{R}_6$  such that

$$\begin{aligned}
 \bar{R}_5 &= (e(X_z, \hat{g}_z)e(X_\sigma, \hat{g}_1))^{s_\theta} (e(\tilde{\sigma}_2, \hat{g}_2)e(\tilde{\sigma}_3, \hat{g}_5))^{-s_{\text{ID}}} (e(\tilde{\sigma}_2, \hat{g}_3)e(\tilde{\sigma}_3, \hat{g}_6))^{-s_u} \\
 &\quad \times (e(C_z, \hat{g}_z)e(C_\sigma, \hat{g}_1)e(\tilde{\sigma}_2, \hat{g}_4)e(\tilde{\sigma}_3, \hat{g}_7)e(\Omega, \hat{g}_8))^{-c} \\
 \bar{R}_6 &= (e(X'_z, \hat{g}'_z)e(X'_\sigma, \hat{g}'_1))^{s_\theta} (e(\tilde{\sigma}'_2, \hat{g}'_3)e(\tilde{\sigma}'_3, \hat{g}'_6))^{-s_u} \\
 &\quad \times (e(C'_z, \hat{g}'_z)e(C'_\sigma, \hat{g}'_1)e(\tilde{\sigma}'_2, \hat{g}'_2 \cdot \hat{g}'_4)e(\tilde{\sigma}'_3, \hat{g}'_5 \cdot \hat{g}'_7)e(\Omega', \hat{g}'_8))^{-c}
 \end{aligned}$$

Output 1 if  $c = H(\text{gpk}, C_{\text{CS}}, \tilde{\sigma}_2, \tilde{\sigma}_3, \tilde{\sigma}'_2, \tilde{\sigma}'_3, \bar{R}_1, \dots, \bar{R}_6, M)$  and 0 otherwise.

**Open(gpk, S<sub>OA</sub>, t, Σ, M, St):**

1. If  $\text{Verify}(\text{gpk}, t, RL_t, \Sigma, M) = 0$ , then output  $\perp$ .
2. Otherwise, decrypt  $(C_1, C_2, C_z, C_\sigma, C_{ID}, C_u, C'_z, C'_\sigma)$  by using  $(x_z, y_z, x_\sigma, y_\sigma, x_{ID}, y_{ID}, x_u, y_u, x'_z, y'_z, x'_\sigma, y'_\sigma)$  such that  $\sigma_1 = C_\sigma \cdot C_1^{-x_\sigma} \cdot C_2^{-y_\sigma}$ ,  $\pi = C_z \cdot C_1^{-x_z} \cdot C_2^{-y_z}$ ,  $V_{\text{ID}} = C_{ID} \cdot C_1^{-x_{\text{ID}}} \cdot C_2^{-y_{\text{ID}}}$ ,  $V_u = C_u \cdot C_1^{-x_u} \cdot C_2^{-y_u}$ ,  $\sigma'_1 = C'_\sigma \cdot C_1^{-x'_\sigma} \cdot C_2^{-y'_\sigma}$ , and  $\pi' = C'_z \cdot C_1^{-x'_z} \cdot C_2^{-y'_z}$ .
3. Search  $V_{\text{ID}}$  from in the database of joining transcripts and get  $(\hat{G}_{2, \text{ID}}, \hat{G}_{5, \text{ID}})$ . If there is no such entry, then output  $\perp$ .
4. Let  $V_{\text{ID}}$  be contained in  $\text{cert}_i$  where  $\text{cert}_i = (i, \text{Path}(i), V_{\text{ID}}, \{(\sigma_{j,1}, \sigma_{j,2}, \sigma_{j,3}, \pi_j, V_u^{(j)})\}_{u_j \in \text{Path}(i)})$ . Search  $j$  such that  $V_u = V_u^{(j)}$  and obtain  $u_j \in \text{Path}(i)$ . If there is no such  $j$ , then output  $\perp$ .
5. Check whether  $(\sigma_1, \tilde{\sigma}_2, \tilde{\sigma}_3, \pi)$  and  $(\sigma'_1, \tilde{\sigma}'_2, \tilde{\sigma}'_3, \pi')$  are valid LMPY signatures as follows.

$$\begin{aligned}
 e(\Omega, \hat{g}_8)^{-1} &= e(\pi, \hat{g}_z)e(\sigma_1, \hat{g}_1)e(\tilde{\sigma}_2, \hat{G}_{2, \text{ID}} \cdot \hat{g}_3^{u_j} \cdot \hat{g}_4)e(\tilde{\sigma}_3, \hat{G}_{5, \text{ID}} \cdot \hat{g}_6^{u_j} \cdot \hat{g}_7) \\
 e(\Omega', \hat{g}'_8)^{-1} &= e(\pi', \hat{g}'_z)e(\sigma'_1, \hat{g}'_1)e(\tilde{\sigma}'_2, \hat{g}'_2 \cdot \hat{g}'_3^{u_j} \cdot \hat{g}'_4)e(\tilde{\sigma}'_3, \hat{g}'_5 \cdot \hat{g}'_6^{u_j} \cdot \hat{g}'_7)
 \end{aligned}$$

<sup>1</sup> We can easily see that the underlying sigma protocol has unique responses. Let all values, except  $\text{resp} = (s_{\text{ID}}, s_\theta, s_u)$ , be fixed. Then, assume that an accepted response  $(s'_{\text{ID}}, s'_\theta, s'_u) \neq (s_{\text{ID}}, s_\theta, s_u)$  exists. Then, from  $g^{s_\theta} \cdot C_1^{-c} = g^{s'_\theta} \cdot C_1^{-c}$ ,  $s_\theta = s'_\theta$  holds. From  $v_1^{s_{\text{ID}}} \cdot X_{\text{ID}}^{s_\theta} \cdot C_{ID}^{-c} = v_1^{s'_{\text{ID}}} \cdot X_{\text{ID}}^{s'_\theta} \cdot C_{ID}^{-c}$  and  $s_\theta = s'_\theta$ ,  $s_{\text{ID}} = s'_{\text{ID}}$  holds. From  $v_2^{s_u} \cdot X_u^{s_\theta} \cdot C_u^{-c} = v_2^{s'_u} \cdot X_u^{s'_\theta} \cdot C_u^{-c}$  and  $s_\theta = s'_\theta$ ,  $s_u = s'_u$  holds. Thus,  $(s'_{\text{ID}}, s'_\theta, s'_u) = (s_{\text{ID}}, s_\theta, s_u)$  holds and this shows that the sigma protocol has unique responses, and the NIZK proof system converted by the Fiat-Shamir transformation is simulation sound.

If the above equations hold, then output  $i$  and  $\perp$  otherwise.

**Security Analysis.** Intuitively, security against misidentification attacks hold as follows. Due to revocation functionality the winning condition of the adversary is changed from  $i^* \notin U^a$  to  $i^* \notin U^a \setminus \mathcal{R}_{t^*}$  where  $i^*$  is the opening result of the group signature output by the adversary at time  $t^*$ ,  $U^a$  is the set of signers who joined via  $Q_{a\text{-join}}$  queries, and  $\mathcal{R}_{t^*}$  is the set of revoked signers at time  $t^*$ . Remark that  $i$  may be  $\perp$ . We divide the condition  $i^* \notin U^a \setminus \mathcal{R}_{t^*}$  to (1)  $i^* \notin U^a$  and (2)  $i^* \in \mathcal{R}_{t^*}$ . The first case is the same as that of the proof given by Libert et al. [39], i.e., the adversary produces a valid group signature whose opening result is in outside of the set of adversarially-controlled signers. This case is reduced to the unforgeability of the LMPY signature scheme on some  $(\text{ID}, u)$  where  $\text{ID}$  was not chosen in interactions between the adversary and the  $Q_{a\text{-join}}$  oracle. The second case is that the adversary can produce a valid group signature whose opening result is in the set of revoked signers. This case is also reduced to the unforgeability of the LMPY signature scheme on some  $(t^*, u)$  where  $u$  is a node and the signature is not contained in the revocation list  $RL_{t^*}$ . Since the LMPY signature is unforgeable under the SXDH assumption, Theorem 2 holds. For security against framing attacks, in the join protocol, a signer chooses  $\text{ID}$  and it is unknown to the group manager. Thus, from a forged signature output by the adversary of framing attacks, we can construct an algorithm that extracts such an unknown identity and uses it to solve the SDL problem. Moreover, due to the soundness of the QA-NIZK argument and the CCA security of the Cramer-Shoup encryption scheme, our scheme is anonymous. We give the security proofs of following theorems in the full version of this paper.

**Theorem 2.** *The proposed group signature scheme is secure against misidentification attacks if the SXDH assumption holds in  $(\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T)$  in the random oracle model.*

**Theorem 3.** *The proposed group signature scheme is secure against framing attacks under the SDL assumption in the random oracle model.*

**Theorem 4.** *The proposed group signature scheme is anonymous in the random oracle model if the SXDH assumption holds in  $(\mathbb{G}, \widehat{\mathbb{G}}, \mathbb{G}_T)$ .*

## 5 Implementation

In this section, we give our implementation results of the proposed scheme. We implemented our scheme by employing the Barreto-Lynn-Scott (BLS) curves [9] over a 455-bit prime field (BLS455), which has around 128-bit security due to Barbulescu-Duquesne's security estimation [7, 8]. We also implemented by employing the Barreto-Naehrig (BN) curves [10] over a 382-bit prime field (BN382). Although BN382 cannot ensure 128-bit security from the security estimation, we also give the results of BN382 because of its small size of signature, actually we will show that the signature size is smaller than that of Ohara et al.

scheme. Remark that, since our proposed scheme is based on the SXDH problem, we need to employ elliptic curves which have type 3 pairings, such as BN and BLS curves. Our implementation environment was as follows: CPU: Core i7-7700K(4.20 GHz), and gcc 6.3.0. We employed the RELIC library [4].

Table 2 summarizes benchmarks of elliptic curve and pairing operations on the BLS455 and BN382 in our environment. Here,  $Mul(\mathbb{G}_1, Type)$ ,  $Mul(\mathbb{G}_2, Type)$ , and  $Exp(\mathbb{G}_T, Type)$  are scalar multiplication in  $\mathbb{G}_1$  and  $\mathbb{G}_2$ , and exponentiation in  $\mathbb{G}_T$ , respectively. If a base point is previously known and fixed, then Type is set as K, and U otherwise. We note that we always use Type U for exponentiations in  $\mathbb{G}_T$  since the RELIC library does not support Type K in  $\mathbb{G}_T$ .

**Table 2.** Benchmarks of group operations

Operation	Time ( $\mu$ sec) BLS455 / BN382
$Mul(\mathbb{G}_1, U)$	248.729/209.145
$Mul(\mathbb{G}_1, K)$	131.631/109.539
$Mul(\mathbb{G}_2, K)$	326.377/264.513
$Exp(\mathbb{G}_T, U)$	743.482/632.369
Miller loop	669.451/577.744
Pairing final exp.	779.022/418.154
Total	1448.473/995.898

Table 3 shows that the running times of our scheme. Here, we set  $N = 8192$ , and assumed that 10% of users are revoked ( $R = 819$ ). In the Sign, Verify, and Open algorithms, since some values are fixed and unnecessary to recompute during time  $t$ , these values are firstly precomputed then used at Sign/Verify/Open in our implementation. In the Table 3, “Precompute” rows show timings of the precomputation phase, and “Online” rows show the actual Sign/Verify/Open timings after the precomputation. Remark that the Open algorithm calls the Verify algorithm, and thus the running times of the Open algorithm contains those of the Verify algorithm.

Next, we evaluated the signature size in Table 4. In our scheme, a signature contains 16 group elements (12 elements in  $\mathbb{G}$  and 4 elements in  $\mathbb{Z}_p$ ) whereas in the Ohara et al. scheme, a signature contains 18 group elements (5 elements in  $\mathbb{G}$  and 13 elements in  $\mathbb{Z}_p$ ). When the BLS455 is employed, the sizes of the scalar value in  $\mathbb{Z}_p$ , an element in  $\mathbb{G}$ , an element in  $\widehat{\mathbb{G}}$ , and an element in  $\mathbb{G}_T$  were 39 bytes, 58 bytes, 115 bytes, and 456 bytes, respectively. Then, the actual signature size of our scheme is slightly larger than that of the Ohara et al. scheme, our signature size is 852 bytes whereas that of the Ohara et al. scheme is 797 bytes. This is because, the signature of our scheme contains many elements in  $\mathbb{G}$  compared to that of Ohara et al. scheme, and the size of element in  $\mathbb{G}$  is slightly but significantly larger than that of a value in  $\mathbb{Z}_p$  on BLS455. On the other hand, when the BN382 is employed, the size of a value in  $\mathbb{Z}_p$  is almost the same as

**Table 3.** Implementation Results

Algorithm		Time (msec)
		BLS455/BN382
<b>Setup</b>		21.217/18.011
<b>Join</b>	GM	39.212/32.185
	User	6.701/5.611
	Total	45.913/37.796
<b>Sign</b>	Precompute	10.660/8.540
	Online	10.667/8.443
<b>Verify</b>	Precompute	15.195/11.745
	Online	15.380/12.532
<b>Revoke</b>		16.508/13.755
<b>Open</b>	Precompute	15.182/11.734
	Online	28.732/23.403

that of an element in  $\mathbb{G}$ , indeed, the sizes of a scalar value in  $\mathbb{Z}_p$ , an element in  $\mathbb{G}$ , an element in  $\widehat{\mathbb{G}}$ , and an element in  $\mathbb{G}_T$  are 48 bytes, 49 bytes, 97 bytes, and 384 bytes, respectively. In this case, our signature size is 780 bytes whereas that of the Ohara et al. scheme is 869 bytes. Remark that, as mentioned in the beginning of this section, the BN382 seems to have less than 128-bit security. Therefore, we can say that, if such a slightly lower-level security is accepted, our signature size is smaller than that of the Ohara et al. scheme.

**Table 4.** Signature size

Scheme	$\mathbb{G}$	$\mathbb{Z}_p$	Signature Size (Bytes)
Ohara et al. [49]	5	13	797 (BLS455)/869 (BN382)
Our scheme	12	4	852 (BLS455)/780 (BN382)

## 6 Conclusion

In this paper, we proposed a revocable group signature scheme with scalability secure under simple assumptions in the random oracle model. We implemented our scheme, and showed that our scheme is feasible in practice. In addition to further reduce the signature size, we can consider the following as future works. Bootle et al. [16] considered full dynamicity where signers can join and leave a group at any time, and mentioned that the LPY model is not fully dynamic. Moreover, Emura et al. [28] considered hiding the number of revoked signers since if the number is revealed then one may guess the reason behind such

circumstances, and it may lead to harmful rumors. Kiayias and Zhou [35] and Chow et al. [21] proposed hidden identity-based signatures where the opening just requires the secret key of the OA, and does not require any other secret members list. Thus, the membership list can be hidden from the OA. Considering these improvements is left as an important open problem in this paper.

**Acknowledgement.** This work was partially supported by the JSPS KAKENHI Grant Number JP16K00198.

## References

1. Intel Enhanced Privacy ID (EPID) Security Technology. <https://software.intel.com/en-us/articles/intel-enhanced-privacy-id-epid-security-technology>
2. Intel Software Guard Extensions (Intel SGX). <https://software.intel.com/en-us/sgx>
3. Akane, M., Nogami, Y., Morikawa, Y.: Fast ate pairing computation of embedding degree 12 using subfield-twisted elliptic curve. *IEICE Trans.* **92-A**(2), 508–516 (2009)
4. Aranha, D.F., Gouvêa, C.P.L.: RELIC is an Efficient LIBrary for Cryptography. <https://github.com/relic-toolkit/relic>
5. Attrapadung, N., Emura, K., Hanaoka, G., Sakai, Y.: A revocable group signature scheme from identity-based revocation techniques: achieving constant-size revocation list. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) *ACNS 2014*. LNCS, vol. 8479, pp. 419–437. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-07536-5\\_25](https://doi.org/10.1007/978-3-319-07536-5_25)
6. Attrapadung, N., Emura, K., Hanaoka, G., Sakai, Y.: Revocable group signature with constant-size revocation list. *Comput. J.* **58**(10), 2698–2715 (2015)
7. Barbulescu, R., Duquesne, S.: Updating key size estimations for pairings. *IACR Cryptology ePrint Archive 2017:334* (2017)
8. Barbulescu, R., Duquesne, S.: Updating key size estimations for pairings. *J. Cryptol.* (2018). <https://doi.org/10.1007/s00145-018-9280-5>
9. 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). [https://doi.org/10.1007/3-540-36413-7\\_19](https://doi.org/10.1007/3-540-36413-7_19)
10. 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). [https://doi.org/10.1007/11693383\\_22](https://doi.org/10.1007/11693383_22)
11. Begum, N., Nakanishi, T., Sadiq, S., Islam, M.E.: Implementation of a revocable group signature scheme with compact revocation list using accumulator. In: *CANDAR*, pp. 610–615 (2016)
12. Bellare, M., Boldyreva, A., Kurosawa, K., Staddon, J.: Multirecipient encryption schemes: how to save on bandwidth and computation without sacrificing security. *IEEE Trans. Inf. Theor.* **53**(11), 3927–3943 (2007)
13. Bichsel, P., Camenisch, J., Neven, G., Smart, N.P., Warinschi, B.: Get shorty via group signatures without encryption. In: Garay, J.A., De Prisco, R. (eds.) *SCN 2010*. LNCS, vol. 6280, pp. 381–398. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15317-4\\_24](https://doi.org/10.1007/978-3-642-15317-4_24)

14. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004). [https://doi.org/10.1007/978-3-540-28628-8\\_3](https://doi.org/10.1007/978-3-540-28628-8_3)
15. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM CCS, pp. 168–177 (2004)
16. 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). [https://doi.org/10.1007/978-3-319-39555-5\\_7](https://doi.org/10.1007/978-3-319-39555-5_7)
17. Brickell, E., Li, J.: Enhanced privacy ID from bilinear pairing for hardware authentication and attestation. In: IEEE SocialCom, pp. 768–775 (2010)
18. 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). [https://doi.org/10.1007/3-540-45708-9\\_5](https://doi.org/10.1007/3-540-45708-9_5)
19. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991). [https://doi.org/10.1007/3-540-46416-6\\_22](https://doi.org/10.1007/3-540-46416-6_22)
20. Cheon, J.H.: Discrete logarithm problems with auxiliary inputs. *J. Cryptol.* **23**(3), 457–476 (2010)
21. Chow, S.S.M., Zhang, H., Zhang, T.: Real hidden identity-based signatures. In: Kiayias, A. (ed.) FC 2017. LNCS, vol. 10322, pp. 21–38. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-70972-7\\_2](https://doi.org/10.1007/978-3-319-70972-7_2)
22. Cramer, R., Damgård, I., MacKenzie, P.D.: Efficient zero-knowledge proofs of knowledge without intractability assumptions. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 354–372. Springer, Heidelberg (2000). [https://doi.org/10.1007/978-3-540-46588-1\\_24](https://doi.org/10.1007/978-3-540-46588-1_24)
23. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.* **33**(1), 167–226 (2003)
24. 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). [https://doi.org/10.1007/11958239\\_13](https://doi.org/10.1007/11958239_13)
25. Derler, D., Slamanig, D.: Highly-efficient fully-anonymous dynamic group signatures. In: ACM AsiaCCS, pp. 551–565 (2018)
26. Emura, K., Hayashi, T.: Road-to-vehicle communications with time-dependent anonymity: a lightweight construction and its experimental results. *IEEE Trans. Veh. Technol.* **67**(2), 1582–1597 (2018)
27. Emura, K., Hayashi, T., Ishida, A.: Group signatures with time-bound keys revisited: a new model and an efficient construction. In: ACM AsiaCCS, pp. 777–788 (2017)
28. Emura, K., Miyaji, A., Omote, K.: An  $r$ -hiding revocable group signature scheme: group signatures with the property of hiding the number of revoked users. *J. Appl. Math.* **2014**, 983040:1–983040:14 (2014)
29. Fan, C.-I., Hsu, R.-H., Manulis, M.: Group signature with constant revocation costs for signers and verifiers. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 2011. LNCS, vol. 7092, pp. 214–233. Springer, Heidelberg (2011). [https://doi.org/10.1007/978-3-642-25513-7\\_16](https://doi.org/10.1007/978-3-642-25513-7_16)

30. Faust, S., Kohlweiss, M., Marson, G.A., Venturi, D.: On the non-malleability of the Fiat-Shamir transform. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 60–79. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-34931-7\\_5](https://doi.org/10.1007/978-3-642-34931-7_5)
31. Furukawa, J., Imai, H.: An efficient group signature scheme from bilinear maps. IEICE Trans. **89-A**(5), 1328–1338 (2006)
32. Groth, J.: Fully anonymous group signatures without random Oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-76900-2\\_10](https://doi.org/10.1007/978-3-540-76900-2_10)
33. 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). [https://doi.org/10.1007/978-3-540-78967-3\\_24](https://doi.org/10.1007/978-3-540-78967-3_24)
34. Kiayias, A., Yung, M.: Secure scalable group signature with dynamic joins and separable authorities. IJSN **1**(1/2), 24–45 (2006)
35. Kiayias, A., Zhou, H.: Hidden identity-based signatures. IET Inf. Secur. **3**(3), 119–127 (2009)
36. 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). [https://doi.org/10.1007/978-3-662-46803-6\\_4](https://doi.org/10.1007/978-3-662-46803-6_4)
37. Kumar, V., Li, H., Park, J.J., Bian, K., Yang, Y.: Group signatures with probabilistic revocation: a computationally-scalable approach for providing privacy-preserving authentication. In: ACM CCS, pp. 1334–1345 (2015)
38. 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). [https://doi.org/10.1007/978-3-642-54631-0\\_20](https://doi.org/10.1007/978-3-642-54631-0_20)
39. Libert, B., Mouhartem, F., Peters, T., Yung, M.: Practical “signatures with efficient protocols” from simple assumptions. In: ACM AsiaCCS, pp. 511–522 (2016)
40. 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). [https://doi.org/10.1007/978-3-642-32009-5\\_34](https://doi.org/10.1007/978-3-642-32009-5_34)
41. 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). [https://doi.org/10.1007/978-3-642-29011-4\\_36](https://doi.org/10.1007/978-3-642-29011-4_36)
42. 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). [https://doi.org/10.1007/978-3-662-48000-7\\_15](https://doi.org/10.1007/978-3-662-48000-7_15)
43. Libert, B., Vergnaud, D.: Group signatures with verifier-local revocation and backward unlinkability in the standard model. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 498–517. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-10433-6\\_34](https://doi.org/10.1007/978-3-642-10433-6_34)
44. 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). [https://doi.org/10.1007/978-3-642-00468-1\\_26](https://doi.org/10.1007/978-3-642-00468-1_26)
45. Nakanishi, T., Funabiki, N.: Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 533–548. Springer, Heidelberg (2005). [https://doi.org/10.1007/11593447\\_29](https://doi.org/10.1007/11593447_29)

46. Nakanishi, T., Funabiki, N.: A short verifier-local revocation group signature scheme with backward unlinkability. In: Yoshiura, H., Sakurai, K., Rannenberg, K., Murayama, Y., Kawamura, S. (eds.) IWSEC 2006. LNCS, vol. 4266, pp. 17–32. Springer, Heidelberg (2006). [https://doi.org/10.1007/11908739\\_2](https://doi.org/10.1007/11908739_2)
47. Nakanishi, T., Funabiki, N.: Revocable group signatures with compact revocation list using accumulators. *IEICE Trans.* **98-A**(1), 117–131 (2015)
48. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001). [https://doi.org/10.1007/3-540-44647-8\\_3](https://doi.org/10.1007/3-540-44647-8_3)
49. Ohara, K., Emura, K., Hanaoka, G., Ishida, A., Ohta, K., Sakai, Y.: Shortening the Libert-Peters-Yung revocable group signature scheme by using the random Oracle methodology. *IACR Cryptology ePrint Archive* 2016:477 (2016)
50. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 111–126. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-29485-8\\_7](https://doi.org/10.1007/978-3-319-29485-8_7)
51. Rahaman, S., Cheng, L., Yao, D.D., Li, H., Park, J.J.: Provably secure anonymous-yet-accountable crowdsensing with scalable sublinear revocation. In: PoPETs, vol. 2017, no. 4, pp. 384–403 (2017)
52. Sadiyah, S., Nakanishi, T.: Revocable group signatures with compact revocation list using vector commitments. *IEICE Trans.* **100-A**(8), 1672–1682 (2017)
53. Slamanig, D., Spreitzer, R., Unterluggauer, T.: Adding controllable linkability to pairing-based group signatures for free. In: Chow, S.S.M., Camenisch, J., Hui, L.C.K., Yiu, S.M. (eds.) ISC 2014. LNCS, vol. 8783, pp. 388–400. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-13257-0\\_23](https://doi.org/10.1007/978-3-319-13257-0_23)
54. Slamanig, D., Spreitzer, R., Unterluggauer, T.: Linking-based revocation for group signatures: a pragmatic approach for efficient revocation checks. In: Phan, R.C.-W., Yung, M. (eds.) Mycrypt 2016. LNCS, vol. 10311, pp. 364–388. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-61273-7\\_18](https://doi.org/10.1007/978-3-319-61273-7_18)
55. Unruh, D.: Quantum proofs of knowledge. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 135–152. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-29011-4\\_10](https://doi.org/10.1007/978-3-642-29011-4_10)



# **Network Security**



# Fast Flux Service Network Detection via Data Mining on Passive DNS Traffic

Pierangelo Lombardo<sup>(✉)</sup>, Salvatore Saeli, Federica Bisio, Davide Bernardi, and Danilo Massa

aizoOn Technology Consulting, Strada del Lionetto 6, 10146 Turin, Italy  
{pierangelo.lombardo,salvatore.saeli,federica.bisio,davide.bernardi,danilo.massa}@aizoongroup.com

**Abstract.** In the last decade, the use of fast flux technique has become established as a common practice to organise botnets in Fast Flux Service Networks (FFSNs), which are platforms able to sustain illegal online services with very high availability. In this paper, we report on an effective fast flux detection algorithm based on the passive analysis of the Domain Name System (DNS) traffic of a corporate network. The proposed method is based on the near-real-time identification of different metrics that measure a wide range of fast flux key features; the metrics are combined via a simple but effective mathematical and data mining approach. The proposed solution has been evaluated in a one-month experiment over an enterprise network, with the injection of pcaps associated with different malware campaigns, that leverage FFSNs and cover a wide variety of attack scenarios. An in-depth analysis of a list of fast flux domains confirmed the reliability of the metrics used in the proposed algorithm and allowed for the identification of many IPs that turned out to be part of two notorious FFSNs, namely Dark Cloud and SandiFlux, to the description of which we therefore contribute. All the fast flux domains were detected with a very low false positive rate; a comparison of performance indicators with previous works show a remarkable improvement.

**Keywords:** Automated security analysis · Malware detection  
Network security · Passive traffic analysis · Botnet · Fast flux

## 1 Introduction

During the last few years, the number of cyberattacks with relevant financial impact and media coverage has been constantly growing. As a result, many companies and organizations have been reinforcing investment to protect their networks, with a resultant increase in the research on this topic [1].

Over the last two decades, botnets have represented one of the most prominent sources of threats on the internet: they are networks of compromised computers (popularly referred to as zombies or bots), which are controlled by a

remote attacker (bot herder). Botnets provide the bot herder with massive resources (bandwidth, storage, processing power), allowing for the implementation of a wide range of malicious and illegal activities, like spam, distributed denial-of-service attacks, spreading of malware (such as ransomware, exploit kits, banking trojans, etc.) [16, 18–20, 22].

A common practice for bot herders is to organise their bots in Fast Flux Service Networks (FFSNs): some bots, chosen from a pool of controlled machines, are used as front-end proxies that relay data between a (possibly unaware) user and a protected hidden server. The technique behind these structures is the *fast flux*, i.e., the rapid and repeated changing of an internet host and/or name server resource record in a Domain Name System (DNS) zone, resulting in rapid changes of the IP addresses to which the domain resolves. FFSNs make the tracing and the recovery of all the infected components extremely difficult, thus allowing for a very high availability for illegal online services related to phishing, dumps stores, and distribution of ransomware, info stealers, and click fraud [21, 26, 28, 32, 33, 35].

FFSNs have been known to cybersecurity experts for more than one decade [22, 32], but in the last few years it has been obtaining a spotlight [17, 18, 20, 24, 31, 34]. The renewed interest is related to the studies of large botnets (e.g., Dark Cloud, also known as Zbot network, and the most recent SandiFlux) which make massive usage of fast flux [2, 21, 25]. The standard approach to FFSNs detection is via the so-called *active* DNS analysis, i.e., by actively querying some domains and by collecting and analysing the answers: this strategy has been widely explored and allows for extensive analyses of botnets [17, 22–29].

Instead, the algorithm described in the present work relies on *passive* analysis of the DNS traffic of a single network: it detects the fast flux domains without interaction with the network traffic, thus making the algorithm completely transparent inside and outside the monitored network; in particular, it cannot be uncovered by the attackers, who often control the authoritative name servers responsible for responding to DNS queries about their fast flux domains [30]. The proposed detection approach has been evaluated in a 30-day-long experimental session over the network described in Sect. 5. The performance is much higher compared to a state-of-the-art analogous method [33]. Moreover, the analysis was performed near-real-time: the average execution time of the algorithm was 25 s, while the average time between two subsequent runs of the algorithm was 3 min (see Sect. 3 for more details), meaning that the average detection time for fast flux domains was less than 2 min.

As an additional test of the proposed approach, we examined the IPs — collected via active DNS analysis — associated with a list of fast flux domains gathered from [3–6]. This investigation confirmed the reliability of the metrics used in the fast flux detection method proposed herein and allowed for the identification of more than 10000 IPs, some of which are likely associated with compromised hosts, which turned out to be part of two notorious botnets, namely Dark Cloud and SandiFlux, to the description of which we therefore contribute.

The paper is structured as follows. In Sect. 2, we discuss the most relevant features of FFSNs, with an outline of related works. In Sect. 3, we briefly describe *aramis*, the monitoring platform that contains the fast flux detection method which is the focus of this paper and which is described thoroughly in Sect. 4. Section 5 comprises a detailed discussion of the experimental results of the test of the proposed algorithm, while Sect. 6 contains further investigations on the FFSNs underlying some fast flux domains. Finally, we discuss possible future developments in Sect. 7.

## 2 Background and Related Work

One of the first works providing an overview of the fast flux attacks was the Honeynet project [32]. In order to explain hidden operations executed by botnets, authors gave examples of both single and double fast flux mechanisms: while the first rapidly changes the A records of domains, the latter frequently changes both the A records and the NS records of a domain. The interested reader can find a review and a classification of fast flux attacks in [35].

Content Delivery Network (CDN) and Round-Robin DNS (RRDNS) are legitimate techniques which are used by large websites to distribute the load of incoming requests to several servers. The response to a DNS query is evaluated by an algorithm which chooses a pool of IPs from a large list of available servers whose number can be of the order of thousands (see Sect. 6 for some examples). As a result, the behaviour in terms of DNS traffic is very similar to the one of a FFSN, and indeed CDNs and RRDNSs represent the typical false positives in fast flux detection algorithms [22, 26, 33].

A large number of approaches have been proposed to detect FFSNs and to distinguish them from legitimate CDNs and RRDNSs. Most of them rely on active DNS analysis, which allows for the collection of a large number of IPs associated with a domain, thus simplifying the FFSNs detection, but they require the resolutions of domains that may be associated with malicious activities [20, 22, 24, 26, 28]. These methods, despite being appropriate for a deep analysis of FFSNs, have relevant drawbacks in implementations oriented to the monitoring of corporate networks [30, 33].

Some FFSN detection methods based on passive DNS analysis have been proposed. Some of them analyse the DNS traffic of a whole Internet Service Provider (ISP), thus taking in input the DNS traffic generated by many different networks. Perdisci et al. [30], in particular, performed a large-scale passive analysis of DNS traffic. They extract some relevant features from the DNS traffic and classified the domains via a C4.5 decision tree classifier. Berger et al. [18] and Stevanovic et al. [34] proposed two other approaches to analyse the DNS traffic of an ISP. Both methods are based on a tool called DNSMap and classify the bipartite graphs formed by the collected fully qualified domain names and the associated IPs. The first method searches for generic malicious usage of DNS, while the latter focuses on FFSNs.

Soltanaghaei and Kharrazi [33], finally, proposed a method for passive DNS analysis of a network which requires a history for each domain to be evaluated

and achieved 94.44% detection rate and 0.001% false positive rate in their best experiment. Our algorithm employs a similar approach, but, with a more careful choice of the metrics achieves better results, while performing a near-real-time analysis (see Sects. 4 and 5 for details).

### 3 aramis

The proposed fast flux detection technique is included in a commercially available network security monitoring platform called *aramis* (Aizoon Research for Advanced Malware Identification System) [7, 19]. This software automatically identifies different types of malware and attacks in near-real-time, it is provided with dedicated hardware<sup>1</sup>, and its structure can be outlined in four phases:

1. Collection: sensors located in different nodes of the network gather data, preprocess them in real-time and send the results to a NoSQL database.
2. Enrichment: data are enriched in the NoSQL database using the information obtained from the aramis Cloud Service, which collects intelligence from various open-source intelligence sources and from internally managed sources.
3. Analysis: stored data are processed by means of two types of analysis: (i) advanced cybersecurity analytics which highlight specific patterns of attacks, among which Domain Generation Algorithms (DGAs) [19] and fast flux, and (ii) a machine learning engine which spots deviations from the usual behaviour of each node of the network.
4. Visualization: results are presented in dashboards to highlight anomalies.

The cycle of the four phases restarts after a time  $\Delta t$  which slightly depends on the traffic flow analysed and amounts to  $182 \pm 36$  s on the network described in Sect. 5. A time  $\Delta t$  of this magnitude is the best trade-off between the near-real-time requirement and the need of a large amount of data in order to have statistically significant results.

## 4 Detection Method

The aim of the proposed detection method is the near-real-time identification of malicious fast flux via the passive monitoring of the DNS traffic of a single network. To this purpose, the method is composed of three steps of analysis. (i) Filtering: queries which are known to be non-malicious (e.g., popular domains, known CDNs, local domains, etc.) are removed. (ii) Metrics identification: some key indicators are calculated over the queries remaining after filters. (iii) Identification: the metrics are used to identify malicious fast flux among the queries.

The three steps have been constructed by combining information on the FFSNs — acquired from the literature — with a simple but effective mathematical and data mining approach. The parameters of the model have been estimated over a *validation set* formed by 30-days of DNS traffic captured from the network described in Table 1, and by 12 pcaps associated with different malware campaigns that leverage FFSNs, collected from the public repository [8].

<sup>1</sup> E5-2690 2.9 GHz  $\times$  2 (2 sockets  $\times$  16 cores) 16 x 8GB RAM, 1.1TB HDD.

**Table 1.** Validation-network description

	30-days total	One-hour average
N. of machines	261	-
N. of connections	80 M	111 k
N. of resolved A-type DNS queries	12 M	17 k
N. of unique resolved A-type DNS queries	381 k	527

#### 4.1 Filters

The algorithm receives resolved DNS requests of type A (which return 32-bits IPv4 addresses, in accordance with [9]) collected near-real-time from the monitored network. The first step consists in the application of the filters reported in Table 2 to the retrieved queries.

**Table 2.** Filters description

Type	Description
White list domains	Domains known to be trusted, e.g., the ones associated with crypto currencies (if their use is allowed in the network): the underlying peer-to-peer networks are, in many respects, similar to botnets
Popular domains	Top 100 domains collected inside the network under analysis, web URLs of the 500 world biggest companies provided by Forbes [10] and top 10000 domains in the world provided by Alexa [11]
Configuration words	Domains containing certain substrings (e.g., related to network system and structure) represent congenital traffic
Overloaded DNS	In order to provide anti-spam or anti-malware techniques, DNS queries are sometimes overloaded, thus causing possible noise
Local and corporate domains	These domains represent a high percentages of the legitimate DNS traffic in a corporate network.
CDNs and RRDNSs	These are the most common sources of false positives in fast flux detection algorithms; aramis (see Sect. 3) includes a function (with a structure similar to that of the proposed fast flux detection algorithm) which periodically updates a white list with the main CDNs and RRDNSs detected in the monitored network, thus allowing for a substantial speed up
Queries with large TTL	According to the literature, malicious fast flux are characterised by a short TTL [22, 28, 33], therefore queries with a TTL larger than 1800 s are filtered

#### 4.2 Metrics Identification

The DNS requests that survive the filters described in Sect. 4.1 are integrated with the history of the previous 30 days, saved locally. This allows for a more accurate evaluation of the behaviour of the domains, however an assessment is

already possible when the first answer is received. Among the remaining domains there are many new emerging CDNs<sup>2</sup> and, in order to distinguish them from the FFSNs — which is the main challenge in malicious fast flux detection — we identified some key indicators. Some of these indicators can be already evaluated after a single query (we call them *static metrics*), while others need a certain history (*history-based metrics*). The information regarding Autonomous Systems (ASs) and public networks used in the following metrics are retrieved from [12].

**Static Metrics.** The metrics described in this section are evaluated over all the IPs collected.

*Maximum Answer Length.* A relevant metric for the detection of malicious fast flux is the number of IPs returned in a single A query. In particular, we consider the maximum  $m_{\text{al}}$  of such value: a malicious fast flux is believed to typically have a  $m_{\text{al}}$  larger than a legitimate fast flux [22, 35].

*Cumulative Number of IPs.* Malicious fast flux typically employ a larger number of IPs ( $n_{\text{IP}}$ ) compared with CDNs, due to the lower reliability of each single node [33].

*Cumulative Number of Public Networks.* Since the botnet underlying a malicious fast flux contains infected machines which are typically distributed quite randomly in different networks, the same is expected to be true for the IPs retrieved by the related queries [22, 35]. For this reason a malicious fast flux typically has a number of public networks ( $n_{\text{net}}$ ) larger than a legitimate CDN.

*Cumulative Number of ASs.* For the same reason described above, FFSNs typically have a number of ASs ( $n_{\text{AS}}$ ) larger than legitimate CDNs.

*AS-Fraction.* The analysis of some preliminary fast flux pcaps revealed that, despite being in general very useful, in some cases the absolute number of AS was not a distinctive feature, while its ratio with the number of IPs was more appropriate. For this reason we defined the metric

$$f_{\text{AS}} = \frac{n_{\text{AS}} - 1}{n_{\text{IP}}}, \quad (1)$$

which quantifies the degree to which the IPs are dispersed in different AS. This quantity takes values from  $f_{\text{AS}} = 0$  (when all the IPs are in the same AS) to  $f_{\text{AS}} \sim 1$  (when each IP is in a different AS and the number of IPs is large). In order to preserve these properties and to encode the additional information about the typical scales associated with  $n_{\text{AS}}$  for CDNs and FFSNs respectively, we rescaled  $f_{\text{AS}}$  as described below. The first rescaling is<sup>3</sup>

$$x \longrightarrow \theta(n_{\text{AS}} - n_0) \left[ 1 - e^{-\left(\frac{n_{\text{AS}} - n_0}{s}\right)^2} \right] x, \quad (2)$$

where  $x = f_{\text{AS}}$ ,  $\theta(t)$  is the Heaviside step function (i.e.,  $\theta(t)$  is 1 for positive  $t$  and 0 otherwise),  $s$  is a scale representing the average number of ASs in a

<sup>2</sup> The filter mentioned in Table 2 detects a CDN only when it has a sufficient history.

<sup>3</sup> Hereafter, the left and right hand sides of the arrow represent the quantity before and after the rescaling respectively.

typical CDN and  $n_0$  is a threshold for  $n_{AS}$  below which the behaviour is not suspicious from the viewpoint of the number of ASs.<sup>4</sup> The rescaling in Eq. 2 reduces  $f_{AS}$  when its value is comparable with the  $n_{AS}$  expected in a CDN. The second rescaling applies Eq. 2 to the quantity  $x = 1 - f_{AS}$  and reduces it (i.e., increases  $f_{AS}$ ) when  $n_{AS}$  is comparable with that of a typical FFSNs. In this case the scale  $s$  represents the average number of ASs in a typical malicious fast flux, while  $n_0$  is a threshold for  $n_{AS}$  below which we do not increase  $f_{AS}$ .<sup>5</sup>

*IP-Dispersion.* The analysis of the distribution of the retrieved IPs is another way to understand to which degree the structure underlying FFSN is random and chaotic. We transform the set of the  $n$  IPs associated with each query into the corresponding positions in the 32-bits IPv4 address space  $x_1, \dots, x_n$ ,<sup>6</sup> and we define

$$d_{IP} = \frac{1}{l_n} \text{median}(\Delta \mathbf{x}), \quad (3)$$

where  $\Delta \mathbf{x} = \{x_i - x_{i-1}\}_{i=2}^n$ , the  $\{x_i\}$  have been ordered so that  $x_i \geq x_{i-1}$ , and  $l_n$  is the average distance if the  $n$  IPs were uniformly distributed in the whole public IPs address space. The IP-dispersion takes value from  $d_{IP} = 1$  (i.e., when the IPs are uniformly distributed) to  $d_{IP} = 0$  (i.e., when the IPs are clearly subdivided into a few clusters of close addresses). A similar idea was used by Nazario et al. [28], who evaluated the average distance among the  $\{x_i\}$ , but their metric is more sensitive to outliers and it is not normalised in the interval  $[0,1]$ , which is crucial to combine it with the other metrics, as described in Sect. 4.3. The FFSNs analysis described in Sect. 6 confirmed that the indicator in Eq. 3 is able to catch the key distribution properties of IPs in a FFSN.

**History-Based Metrics.** The *history* is constructed by subdividing the queries retrieved from the monitored network in subsequent *chunks*: each chunk contains at least 10 queries and spans a time interval of at least one hour; these two conditions are the minimal requirements to make the metric definitions meaningful from a statistical point of view. The metrics described in this section are evaluated only if it is possible to construct at least two chunks.

*Change in the Set of IPs.* It is a common belief that, while a CDN typically returns IPs taken from a stable IP-pool, a malicious fast flux employs the available nodes in the FFSN, which often evolves quickly, and therefore its IP-pool changes from time to time [22,35]. We defined a metric which measures in a very simple way the change in the IP-pool:

$$c_{IP} = \frac{n_{IP}}{n_{IP}^c} - 1, \quad (4)$$

<sup>4</sup> We set  $s = 2.5$  and  $n_0 = 3$ ; the first is the average  $n_{AS}$  for the top 4 largest CDNs detected in the validation set, while the latter is half the minimum of  $n_{AS}$  detected for a fast flux in the validation set.

<sup>5</sup> We set  $s = 40$  in agreement with Ref. [35], which states that a typical FFSN has a set of IPs distributed among 30–60 ASs, and  $n_0 = 5$ , which is the maximum number of ASs detected for a CDN in the validation set.

<sup>6</sup> To each IP  $n_1.n_2.n_3.n_4$  we associated  $x = 256^3 n_1 + 256^2 n_2 + 256 n_3 + n_4$ .



where  $n_{\text{IP}}^c$  is the number of unique IPs present in the chunk averaged over all chunks, while  $n_{\text{IP}}$  has been defined in Sect. 4.2. This quantity takes the value  $c_{\text{IP}} = 0$  when all the IPs are found in each chunk, i.e., when the IP-pool is stable and it is explored completely in each chunk (and therefore  $n_{\text{IP}}^c = n_{\text{IP}}$ ). On the other hand, if the IP-pool changes substantially from one chunk to the other, the total number of IPs  $n_{\text{IP}}$  is much larger than the average number of IPs  $n_{\text{IP}}^c$  found in a chunk, and therefore  $c_{\text{IP}}$  becomes large (it is unbounded above). The same considerations apply to all the following metrics.

*Change in the Set of Public Networks.* While CDNs typically use IPs taken from the same few public networks, malicious fast flux frequently introduce IPs from new networks [22, 35]. We measure the change in the set of public networks by means of  $c_{\text{net}} = n_{\text{net}}/n_{\text{net}}^c - 1$ , where  $n_{\text{net}}^c$  is the network-analogous of  $n_{\text{IP}}^c$ .

*Change in the Set of ASs.* The generalisation of the previous argument to the next aggregation level brings us to the analysis of the changes in the number of AS involved. We introduce therefore  $c_{\text{AS}} = n_{\text{AS}}/n_{\text{AS}}^c - 1$ , where  $n_{\text{AS}}^c$  is the AS-analogous of  $n_{\text{IP}}^c$ .

*Change in the Answer Length.* Another relevant indicator is the change in the number of IPs retrieved in each query [22, 35]. We measure this change by means of  $c_{\text{al}} = m_{\text{al}}/m_{\text{al}}^c - 1$ , where  $m_{\text{al}}^c$  is the  $m_{\text{al}}$ -analogous of  $n_{\text{IP}}^c$ .

### 4.3 Fast Flux Domains Identification

A preliminary step for fast flux domains identification is the filtering of the queries with  $d_{\text{IP}} = 0$ , because this removes many false positives with no loss in terms of true positives. The next step is the use of the metrics defined in Sect. 4.2 to discriminate among malicious fast flux and CDN. Instead of using a machine learning ‘black box’ classifier, we combine the indicators in a controlled way, in order to encode some other domain knowledge and to allow for an easier interpretation of the results. Foremost we aggregate the static and history-based metrics separately, and finally we combine them into a single anomaly indicator  $A$ , which can straightforwardly be used to classify the queries between fast flux and legit domains.

**Aggregation of the Static Metrics.** We normalised the metrics  $n_{\text{IP}}$ ,  $n_{\text{net}}$ ,  $n_{\text{AS}}$ , and  $m_{\text{al}}$  in the interval  $[0,1]$ , so that for all of them the value 0 corresponds to a typical CDN, while 1 corresponds to the expected behaviour of a malicious fast flux. This is achieved by means of a square-exponential scaling of the form

$$x \longrightarrow 1 - e^{-\left(\frac{x-x_0}{s}\right)^2}, \quad (5)$$

where  $x_0 = 1$  is the minimum value for the metric before the rescaling,  $s$  is different for each metric and represents an intermediate scale between a typical CDN behaviour and a behaviour clearly ascribed to a malicious fast flux.<sup>7</sup> Eq. 5

<sup>7</sup> The values of  $s$  were set based on information retrieved from the literature ([35] and references therein) and the validation set. More in detail, we chose  $s_{\text{IP}} = 24$ ,  $s_{\text{net}} = 12$ ,  $s_{\text{AS}} = 6$ , and  $s_{\text{al}} = 10$ .

rescales  $x = x_0$  (i.e., the smallest possible value for  $x$ ),  $x = s + x_0$  (i.e., a value intermediate between the typical CDN behaviour and the typical malicious behaviour), and  $x \gg s$  (i.e., a value much larger than the scale  $s$ ) to 0, 1/2, and 1 respectively.

After the scaling, all the quantities  $n_{\text{IP}}$ ,  $n_{\text{net}}$ ,  $n_{\text{AS}}$ ,  $m_{\text{al}}$ ,  $f_{\text{AS}}$ , and  $d_{\text{IP}}$  are comparable: they take values in the interval  $[0,1]$  and for each of them a value close to 0 denotes a typical CDN behaviour, while a value close to 1 indicates a very suspicious behaviour. We combined these indicators with a weighted arithmetic mean in a unique static index<sup>8</sup>

$$A_{\text{stat}} = w_{\text{IP}}n_{\text{IP}} + w_{\text{net}}n_{\text{net}} + w_{\text{AS}}n_{\text{AS}} + w_{\text{al}}m_{\text{al}} + w_f f_{\text{AS}} + w_d d_{\text{IP}}. \quad (6)$$

In order to avoid the evaluation of misleading indicators due to lack of data, the metrics  $f_{\text{AS}}$  and  $d_{\text{IP}}$  are evaluated only if a minimum number of IPs is collected, while  $m_{\text{al}}$  is evaluated only if at least one answer contains more than one IP. When one metric is absent, its value is set to 0 (in the absence of data we apply a sort of ‘presumption of innocence’, to reduce false positives), its weight in the evaluation of  $A_{\text{stat}}$  is decreased by a factor 20 (because the innocence assessment is only due to the absence of data), and the other weights are proportionally rescaled so that  $\sum_i w_i = 1$ .

**Aggregation of the History-Based Metrics.** As already mentioned, the metrics  $c_{\text{IP}}$ ,  $c_{\text{net}}$ ,  $c_{\text{AS}}$ , and  $c_{\text{al}}$  defined in Sect. 4.2 are unbounded above. We normalise them in the interval  $[0,1]$  by means of Eq. 5 with  $x_0 = 0$  (as the minimum value for these metrics before the rescaling is 0).<sup>9</sup> After the rescaling, all the metrics take values in the interval  $[0,1]$  and for each of them a value close to 0 corresponds to a very stable behaviour, while a value close to 1 indicates a behaviour with high variability over time. We combine then in a unique indicator three of the history-based metrics (the fourth, i.e.,  $c_{\text{al}}$  is instead used in Eq. 8) with a weighted arithmetic mean<sup>10</sup>

$$A_{\text{dyn}} = w'_{\text{IP}}c_{\text{IP}} + w'_{\text{net}}c_{\text{net}} + w'_{\text{AS}}c_{\text{AS}}. \quad (7)$$

**Final Aggregation.** We combine the indicators  $A_{\text{stat}}$ ,  $A_{\text{dyn}}$ , and  $c_{\text{al}}$  into a single anomaly indicator  $A$ , which should be used to classify the queries between fast flux and legit domains. In order to reduce false positives, we differentiate on the basis of the quantity  $f_{\text{AS}}$ , and we define

$$A = \begin{cases} \sum_i w_i A_i & \text{if } f_{\text{AS}} \geq 0.5 \\ \prod_i (A_i)^{w_i} & \text{if } f_{\text{AS}} < 0.5 \end{cases}, \quad (8)$$

<sup>8</sup> The weights reflect the importance of the corresponding metric in the correct classification in the validation set; the optimal values are  $w_{\text{IP}} = w_{\text{net}} = 0.03$ ,  $w_{\text{AS}} = 0.13$ ,  $w_{\text{al}} = 0.09$ ,  $w_f = 0.54$ , and  $w_d = 0.18$ .

<sup>9</sup> The values of  $s$  were set based on information retrieved from the literature and the validation set. More in detail, we chose  $s_{\text{IP}} = s_{\text{net}} = 1$  and  $s_{\text{AS}} = s_{\text{al}} = 0.5$ .

<sup>10</sup> The weights reflect the importance of the corresponding metric in the validation set; the optimal values are  $w'_{\text{IP}} = 0.07$ ,  $w'_{\text{net}} = 0.23$ , and  $w'_{\text{AS}} = 0.7$ .

where  $\{A_i\} = \{A_{\text{stat}}, A_{\text{dyn}}, c_{\text{al}}\}$  and  $\{w_i\}$  are the related weights.<sup>11</sup> Analogously to the averages in Eqs. 6 and 7, when one metric is absent, its value  $A_i$  is set to 0 (not anomalous), its weight  $w_i$  is decreased by a factor 20, and the other weights are proportionally rescaled so that  $\sum_j w_j = 1$ .

Note that in Eq. 8 a (weighted) arithmetic mean is used when the AS-fraction is large, while a (weighted) geometric mean is used when the AS-fraction is small; this implies that in the latter case a value close to 0 for one of the indicators  $A_i$  gives a stronger penalty to  $A$ .

The detection of malicious fast flux has thus been reduced to a very simple one-dimension classification problem: only queries with  $A > A_{\text{th}}$  are labeled as fast flux, where the optimal threshold ( $A_{\text{th}} = 0.25$ ) has been found by maximizing the performance on the validation set. In order to increase the readability of the results, we applied a sigmoid-shaped rescaling which maps  $A = 0$  and  $A = 1$  onto themselves and  $A_{\text{th}}$  onto 0.5.

## 5 Experimental Evaluation

The fast flux detection algorithm described in Sect. 4 was evaluated over a test set comprising 30 days of ordinary traffic of the network described in Table 3 with the injection of fast flux traffic which covers all the most relevant fast flux attack scenarios (see Table 4 for a complete list).

**Table 3.** Test-network description

	30-days total	One-hour average
N. of machines	391	-
N. of client machines	286	-
N. of connections	398 M	552 k
N. of resolved A-type DNS queries	75 M	104 k
N. of unique resolved A-type DNS queries	1.3 M	1.9 k

The fast flux traffic has been injected in the network via 47 pcaps — collected from the public repositories [3–5] — which are associated with 9 different malware campaigns. Table 4 provides a brief description of each malware campaign with the following information:

- the category, i.e., the malware type associated with the campaign
- the name of the campaign
- the list of the domains present in each pcap of the campaign, with the anomaly indicator  $A$  associated by the algorithm to each of them
- the average value of  $A$  for each campaign.

<sup>11</sup> An optimisation procedure on the validation set produced similar weight for the three quantities:  $w_{\text{stat}} = 0.27$ ,  $w_{\text{dyn}} = 0.38$ , and  $w_{\text{al}} = 0.35$ .

In order to rule out overfitting, we used the test set (i.e., the network traffic described in Table 3 and the pcaps described in Table 4) only to test the performance of the algorithm, while we used another set (see Sect. 4) to define the algorithm and the values of its parameters.

**Table 4.** Malware description (the underlined domains are farther analysed in Sect. 6)

Category	Campaign	Domains ( $A$ )	$\langle A \rangle$
Banking Trojan	ZBOT	<u>miscapoerasun.ws</u> (0.85)	0.85
Banking Trojan	Dreambot	<u>rahmatullah.at</u> (0.89); <u>ardshinbank.at</u> (0.92)	0.91
Banking Trojan	Ursnif	<u>widmwdndghdk.com</u> (0.90); <u>bnvmcnjgheht.com</u> (0.85); <u>qqweerr.com</u> (0.85)	0.87
VBA Dropper	Doc Dropper Agent <sup>a</sup>	<u>aassmncnnc.com</u> (0.90); <u>iiieejrjr.com</u> (0.87); <u>ghmchdkenee.com</u> (0.88)	0.88
Ransomware	Locky	<u>thedarkvpv.net</u> (0.83); <u>nsaflow.info</u> (0.91); <u>mrscrowe.net</u> (0.93); <u>sherylbro.net</u> (0.87); <u>scottfranch.org</u> (0.90); <u>gdiscoun.org</u> (0.90)	0.89
Ransomware	Nymaim	<u>iqbppddvjq.com</u> (0.91); <u>dannrnysvp.com</u> (0.91); <u>pmjpdwys.com</u> (0.93); <u>vqmfco.com</u> (0.86); <u>gbfseis.com</u> (0.91); <u>iuzngzhl.com</u> (0.97); <u>vpvqskazjvco.com</u> (0.84); <u>jauudedqnm.com</u> (0.93); <u>dybgsb.com</u> (0.93); <u>tuzhohg.com</u> (0.93); <u>sxrhysqdp.com</u> (0.86); <u>arfbqcc.com</u> (0.93); <u>dannrnysvp.com</u> (0.87)	0.90
Banking Trojan	Zeus Panda	<u>farvictor.co</u> (0.89); <u>fardunkan.co</u> (0.89); <u>bozem.co</u> (0.84); <u>farmacyan.co</u> (0.87); <u>fargugo.co</u> (0.90); <u>manfam.co</u> (0.85)	0.87
Banking Trojan	GOZI ISFB	<u>qdkngjibqnwehiqwrzbudwe.com</u> (0.80); <u>jnossidjfnweqrfew.com</u> (0.90); <u>zxcuiniqhweizsds.com</u> (0.86); <u>huwikacjajsneqwe.com</u> (0.92); <u>efojowufjaowudawd.com</u> (0.92); <u>onlyplacesattributionthe.net</u> (0.90); <u>nvvnfjvnfcdnj.net</u> (0.86); <u>popoiuiuntnt.net</u> (0.89); <u>zzzzmmsnsns.net</u> (0.80); <u>popoosneneee.net</u> (0.83); <u>liceindividualshall.net</u> (0.87); <u>roborobonsnsn.net</u> (0.93)	0.87
Ransomware	GandCrab	<u>zonealarm.bit</u> (0.90)	0.90

<sup>a</sup> Doc.Dropper.Agent-6332127-0 [13].

Table 4 clearly shows that the proposed method successfully detected all the fast flux domains with a high anomaly indicator. In fact the value of  $A$  averaged over all campaigns is equal to 0.89.

**Table 5.** Results

	$A > 0$	$A > 0.5$
True Positives ( $T_P$ )	47 (100%)	47 (100%)
False Negatives ( $F_N$ )	0 (0%)	0 (0%)
False Positives ( $F_P$ )	6 (<0.001%)	4 (<0.001%)

In Table 5 we summarise the performance of the algorithm: in the second column we consider the total number of outputs of the algorithm (i.e., the number of domains with  $A > 0$ ) while in the third column we report the number of outputs labeled as fast flux (i.e., the number of domains with  $A > 0.5$ ). On the rows we report the following quantities

- True Positives rate ( $T_P$ ): the number of unique fast flux domains detected;
- False Negatives rate ( $F_N$ ): the number of unique fast flux domains incorrectly labeled as legit;
- False Positives rate ( $F_P$ ): the number of unique legit domains incorrectly labeled as fast flux.

All rates are given as absolute values and as percentages for each type on the corresponding number of unique domains in input.

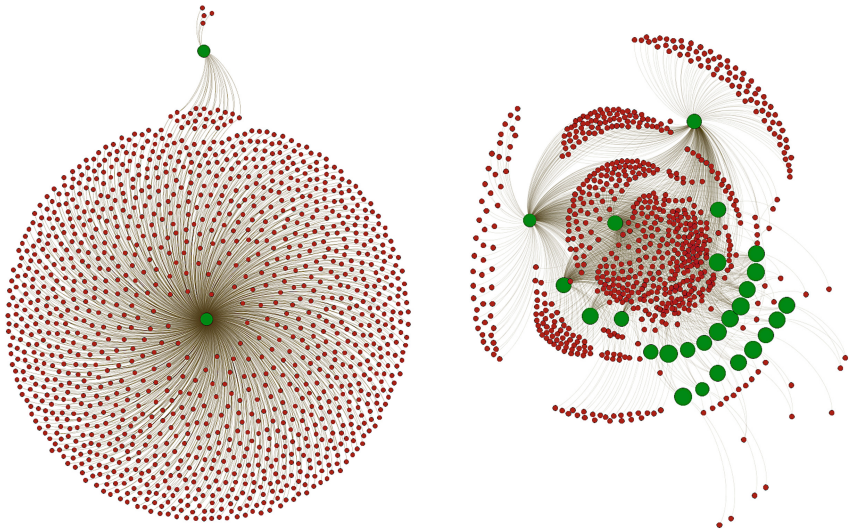
A remarkable result is the absence of false negatives: this determines indeed a 100% recall, also known as detection rate,  $R = T_P / (T_P + F_N)$ . In order to evaluate the algorithm also with a metric that takes into account the false positives rate  $F_P$ , we computed the F-score  $F = 2PR / (P + R)$  (where  $P = T_P / (T_P + F_P)$ ), obtaining  $F = 95.9\%$ .

As a comparison, [33] obtained  $R = 94.4\%$  and  $F = 89.5\%$  in their best experimental result. We can therefore conclude that the proposed method is able to detect queries to fast flux domains in a corporate network in near-real-time and with high anomaly indicators, limiting false positives at the same time.

## 6 Fast Flux Service Networks Analysis

As an in-depth analysis of the algorithm described in Sect. 4, we examined the IPs associated to a list of fast flux domains. The IPs were collected via active DNS analysis and precisely with an FFSN-spanner which resolved systematically domains taken from a list of malicious domains; these domains were gathered via a scouting activity from the public repositories [3–6]. With the purpose of hiding the FFSN-spanner activity from the bot herders, we randomized the sequence of the queries and the waiting times among two subsequent queries, while implementing an anonymization technique based on the use of the Tor network. In order to overcome a limitation of the DNSPort resolver [14], which returns only the first answer for domain lookup, we adopted *ttdnsd*, the Tor TCP DNS Daemon. This solution allows for making arbitrary DNS requests by converting any UDP request into a TCP connection, which is given to Tor through the SOCKS

port. The request is then forwarded anonymously through the Tor network and reaches one of the ‘open’ recursive name servers via the Tor Exit node.



**Fig. 1.** Bipartite graph representing the IPs (*small red circles*) associated with each domain (*large green circle*) in the pre-migration scenario. An arc indicates that the IP has been given in answer to a query resolving the domain. (Color figure online)

Over the period 09.03.2018–19.04.2018, we collected 10747 IPs associated with 55 domains<sup>12</sup>: 7640 are fictitious IPs, related to the Nymaim campaign, while the remaining 3017 IPs are likely associated with compromised hosts. The IPs of the first group (*Nymaim-fake-IPs*), which are translated into real IPs<sup>13</sup> by the decoder algorithm in the malware that use them [15], are strictly related to the *C&C Network* described in [25]. The IPs of the latter group (*real IPs*) showed instead a relevant change in the behaviour on the 26.03.2018: this was probably the last part of the migration described in [2].

The pre-migration scenario is described in Fig. 1: it can be noted that different domains (*large green circles*) share some IPs (*small red circles*). In Fig. 2 we

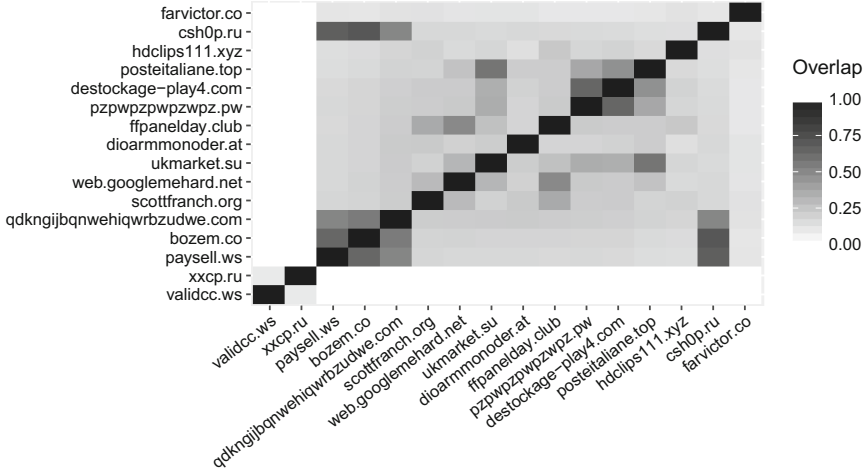
<sup>12</sup> Some domains are reported in Table 4, others in Fig. 2; the remaining domains are odqndpqowdnqwpodn.com, moncompte-carrefour.org, 0768.ru, allianzbank.org, commerzb.co, db-ag.co, druhok.com, form.xbeginner.org, ihalbom.com, ingdirectverifica.com, lloyds-personal.com, mein-advanzia.info, point.charitablex.org, postofficegreat.com, ransomware.bit, redluck0.com, safe.bintrust.org, sunyst.co, dfplajngru.com, mer.arintrueed.org, www.ico-teleqram.net, clo.arotamarid.org, www.translationdoor.com, vr-b.co, vr-b.cc.

<sup>13</sup> An analysis on some pcaps associated with iuzngzhl.com, arlfbqcc.com, and vpvqskazjvco.com revealed that the corresponding real IPs are based on the SandiFlux FFSN described below.

represent the overlap  $O_{ij}$  among all the pairs  $(i, j)$  of the top 16 domains observed before the migration (excluded the ones associated with the Nymain campaign), defined as

$$O_{ij} = \frac{|X_i \cap X_j|}{|X_i \cup X_j|}, \tag{9}$$

where  $X_i$  is the pool of IPs associated with the  $i$ -th domain and  $|X|$  is the the cardinality of  $X$ .

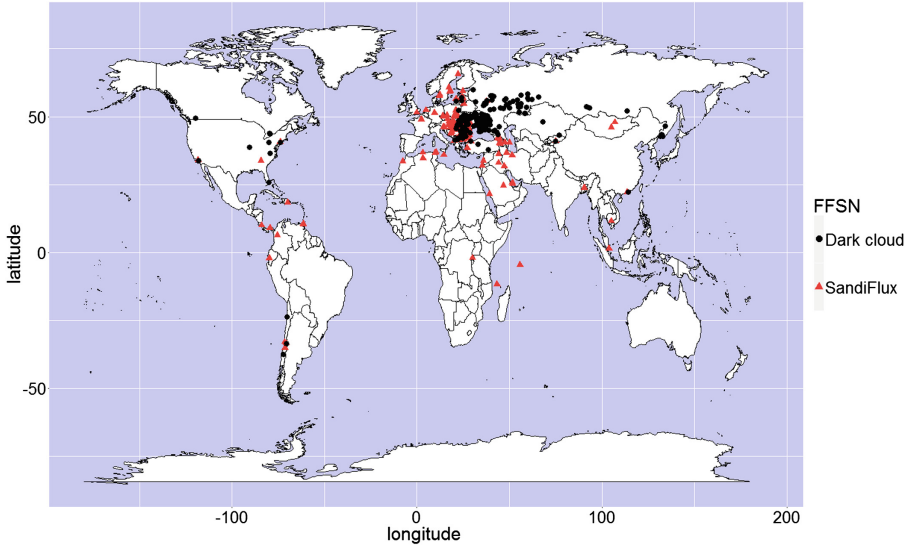


**Fig. 2.** Overlap representation  $O_{ij}$  (defined in Eq. 9) among all the pairs  $(i, j)$  of the top 16 domains (for number of retrieved IPs) in the pre-migration scenario. Darker tones represent larger overlaps.

Both Figs. 1 and 2 show a clear subdivision of the domains in two independent clusters. The analysis of the fast flux domains revealed that the clusters correspond to two large FFSNs, namely Dark Cloud (on the left in Figs. 1 and 2) and SandiFlux (on the right). Indeed, in the first cluster we recognised domains associated with Dumps Stores that leverage Dark Cloud [21], while in the latter we found domains associated with the GandCrab campaigns, which leverage SandiFlux [2]. It is worth noting that the sets of IPs in the two FFSNs that we identified are highly overlapped, but no IP is shared among the two groups.

The pre-migration subdivision in two different FFSNs is reflected in the different geolocation of the relative IPs: Fig. 3 shows that, while the IPs retrieved from SandiFlux are mainly localised in central-east Europe, the ones retrieved from Dark Cloud are based in eastern Europe and Russia.

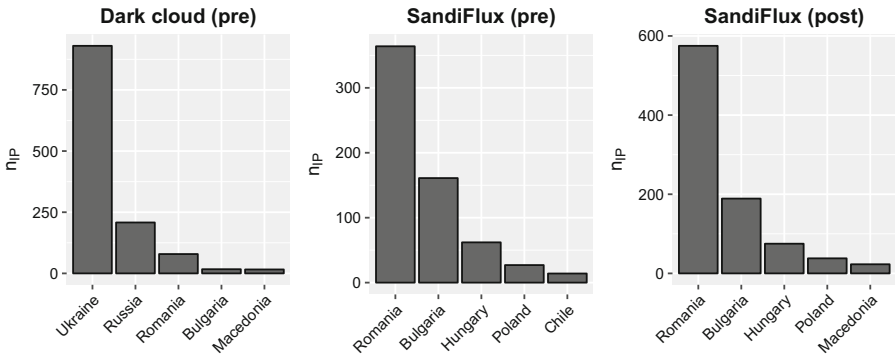
After the migration all the domains appeared to leverage a unique FFSN, which we recognised as SandiFlux. In Fig. 4 we further investigate the top 5 countries represented before the migration (on the left) and after (on the right): Ukraine and Russia confirmed to be the most represented countries in Dark



**Fig. 3.** Geolocation of the IPs retrieved for the FFSNs before the migration

Cloud [21], while SandiFlux’s IPs are found mainly in Romania and Bulgaria both before and after the migration. Figures 3 and 4 are based on the IP-geoloc tables downloaded from Maxmind [12].

The FFSNs described above are a good testing ground for the metrics introduced in Sect. 4.2: in Table 6 we report a summary of some of these metrics evaluated over the FFSNs and three large CDNs. Note that two of the CDNs we observed (namely, *www.nationalgeographic.it* and *cdn.wetransfer.net*) have a very large number of IPs, making thus  $n_{IP}$  a misleading indicator in these cases.



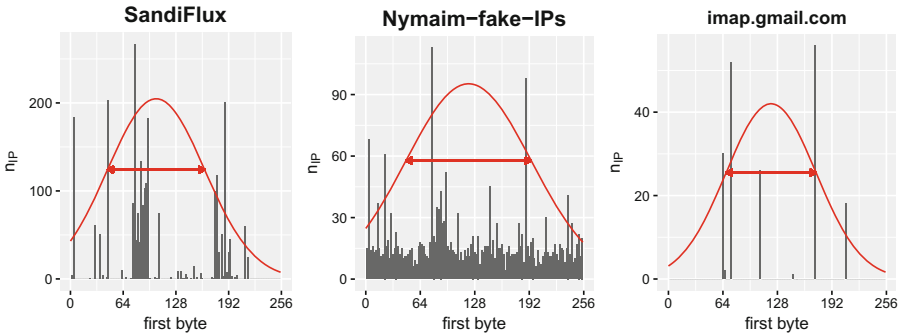
**Fig. 4.** Histogram of the number of IPs localised in the top 5 countries before the migration (on the left) and after (on the right)



This is not a problem for the proposed algorithm, since, as explained in Sect. 4,  $n_{IP}$  is used in combination with many other metrics.

**Table 6.** Summary of some relevant metrics

	$n_{IP}$	$n_{AS}$	$n_{IP}^{resc}$	$n_{AS}^{resc}$	$c_{AS}^{resc}$	$f_{AS}^{resc}$	$d_{IP}$
Dark Cloud	1276	221	1	1	1	1	$1.1 \cdot 10^{-3}$
SandiFlux	1831	354	1	1	1	1	$1.3 \cdot 10^{-3}$
Nymaim-fake-IPs	7640	1767	1	1	1	1	0.77
www.nationalgeographic.it	2589	1	1	0	0	0	$2.8 \cdot 10^{-6}$
cdn.wetransfer.net	2852	1	1	0	0	0	$3.1 \cdot 10^{-6}$
neo4j.com	33	1	0.83	0	0	0	$5.8 \cdot 10^{-4}$



**Fig. 5.** Histogram of the frequencies of the first byte in the IP-pool associated with two groups of fast flux domains and with one large CDN (bin-size = 2)

In Fig. 5 we represent the histogram of the frequencies of the first byte in the IP-pool of SandiFlux and Nymaim-fake-IPs and one medium-size CDN. A clear difference between botnets can be noticed, in particular among the Nymaim-fake-IPs, where the IP-distribution is not so far from a uniform random distribution and the CDN ‘imap.gmail.com’, where the IP-distribution has a few high peaks. Figure 5 clearly shows that simple indicators as the mean and the variance (represented by the corresponding Gaussian distribution) do not catch the nature of the distribution, while the metric  $d_{IP}$  defined in Eq. 3 is much more appropriate. In particular  $d_{IP} = 1.3 \cdot 10^{-3}$  for SandiFlux and  $d_{IP} = 0.77$  for Nymaim-fake-IPs, while the CDN ‘imap.gmail.com’ has  $d_{IP} = 5.0 \cdot 10^{-8}$ . Note that the encoding of the IPs is not a problem for the detection algorithm: in fact it increases the IP dispersion, thus fastening the detection.

## 7 Conclusions

In this paper, we proposed a fast flux detection method based on the passive analysis of the DNS traffic of a corporate network. The analysis is based on

*aramis* security monitoring system. The proposed solution has been evaluated over the LAN of a company, with the injection of 47 pcaps associated with 9 different malware campaigns that leverage FFSNs and cover a wide variety of attack scenarios. All the fast flux domains were detected with a very low false positive rate and the comparison of performance indicators with a state-of-the-art work shows a remarkable improvement. An in-depth active analysis of a list of malicious fast flux domains confirmed the reliability of the metrics used in the proposed algorithm and allowed for the identification of more than 10000 IPs, some of which are likely associated with compromised hosts. These IPs turned out to be part of two notorious botnets, namely Dark Cloud and SandiFlux, to the description of which we therefore contribute.

As a future development, we plan to introduce in the algorithm a metric related to the use of reserved IPs, which we observed to be extensively present in SandiFlux. Another planned development is the inspection of the overlap in terms of IPs among the most suspicious domains, as we saw that many IPs are shared among domains in the same FFSN.

## References

1. [https://www.acs.org.au/content/dam/acs/acs-publications/ACS.Cybersecurity\\_Guide.pdf](https://www.acs.org.au/content/dam/acs/acs-publications/ACS.Cybersecurity_Guide.pdf)
2. <https://www.proofpoint.com/us/threat-insight/post/sandiflux-another-fast-flux-infrastructure-used-malware-distribution-emerges>
3. <https://www.hybrid-analysis.com/>
4. <https://packettotal.com/>
5. <https://www.reverse.it/>
6. <https://virustotal.com/>
7. <http://www.aramisec.com>
8. <https://www.malware-traffic-analysis.net/>
9. <https://tools.ietf.org/html/rfc1035>
10. <http://www.forbes.com>
11. <http://www.alexa.com>
12. <https://dev.maxmind.com/geoip/>
13. <http://blog.talosintelligence.com/2017/07/threat-roundup-0630-0707.html>
14. <https://www.torproject.org/docs/tor-manual.html.en>
15. <https://www.cert.pl/en/news/single/nymaim-revisited/>
16. Alieyan, K., Almomani, A., Manasrah, A., Kadhum, M.M.: A survey of botnet detection based on DNS. *Neural Comput. Appl.* **28**(7), 1541–1558 (2017)
17. Almomani, A.: Fast-flux hunter: a system for filtering online fast-flux botnet. *Neural Comput. Appl.* **29**(7), 483–493 (2018)
18. Berger, A., D’Alconzo, A., Gansterer, W.N., Pescapé, A.: Mining agile DNS traffic using graph analysis for cybercrime detection. *Comput. Netw.* **100**, 28–44 (2016)
19. Bisio, F., Saeli, S., Lombardo, P., Bernardi, D., Perotti, A., Massa, D.: Real-time behavioral DGA detection through machine learning. In: 2017 International Carnahan Conference on Security Technology (ICCST), pp. 1–6. IEEE (2017)
20. Chahal, P.S., Khurana, S.S.: TempR: application of stricture dependent intelligent classifier for fast flux domain detection. *Int. J. Comput. Netw. Inf. Secur.* **8**(10), 37 (2016)

21. Crowder, W., Dunker, N.: Dark cloud network facilitates crimeware. [https://www.riskanalytics.com/wp-content/uploads/2017/10/Dark\\_Cloud\\_Network\\_Facilitates\\_Crimeware.pdf](https://www.riskanalytics.com/wp-content/uploads/2017/10/Dark_Cloud_Network_Facilitates_Crimeware.pdf)
22. Holz, T., Gorecki, C., Rieck, K., Freiling, F.C.: Measuring and detecting fast-flux service networks. In: NDSS (2008)
23. Hsu, C.-H., Huang, C.-Y., Chen, K.-T.: Fast-flux bot detection in real time. In: Jha, S., Sommer, R., Kreibich, C. (eds.) RAID 2010. LNCS, vol. 6307, pp. 464–483. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15512-3\\_24](https://doi.org/10.1007/978-3-642-15512-3_24)
24. Jiang, C.B., Li, J.S.: Exploring global IP-usage patterns in fast-flux service networks. JCP **12**(4), 371–379 (2017)
25. Katz, O., Perets, R., Matzliach, G.: Digging deeper - an in-depth analysis of a fast flux network (2017). <https://www.akamai.com/us/en/multimedia/documents/white-paper/digging-deeper-in-depth-analysis-of-fast-flux-network.pdf>
26. Lin, H.T., Lin, Y.Y., Chiang, J.W.: Genetic-based real-time fast-flux service networks detection. Comput. Netw. **57**(2), 501–513 (2013)
27. Martinez-Bea, S., Castillo-Perez, S., Garcia-Alfaro, J.: Real-time malicious fast-flux detection using DNS and bot related features. In: 2013 Eleventh Annual International Conference on Privacy, Security and Trust (PST), pp. 369–372. IEEE (2013)
28. Nazario, J., Holz, T.: As the net churns: fast-flux botnet observations. In: 2008 3rd International Conference on Malicious and Unwanted Software, MALWARE 2008, pp. 24–31. IEEE (2008)
29. Passerini, E., Paleari, R., Martignoni, L., Bruschi, D.: FluXOR: detecting and monitoring fast-flux service networks. In: Zamboni, D. (ed.) DIMVA 2008. LNCS, vol. 5137, pp. 186–206. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-70542-0\\_10](https://doi.org/10.1007/978-3-540-70542-0_10)
30. Perdisci, R., Corona, I., Giacinto, G.: Early detection of malicious flux networks via large-scale passive DNS traffic analysis. IEEE Trans. Dependable Secure Comput. **9**(5), 714–726 (2012)
31. Ruohonen, J., Leppänen, V.: Investigating the agility bias in DNS graph mining. In: 2017 IEEE International Conference on Computer and Information Technology (CIT), pp. 253–260. IEEE (2017)
32. Salusky, W., Danford, R.: Know your enemy: fast-flux service networks. HoneyNet Proj. 1–24 (2007)
33. Soltanaghaei, E., Kharrazi, M.: Detection of fast-flux botnets through DNS traffic analysis. Scientia Iranica. Trans. D Comput. Sci. Eng. Electr. **22**(6), 2389 (2015)
34. Stevanovic, M., Pedersen, J.M., D’Alconzo, A., Ruehrup, S.: A method for identifying compromised clients based on DNS traffic analysis. Int. J. Inf. Secur. **16**(2), 115–132 (2017)
35. Zhou, S.: A survey on fast-flux attacks. Inf. Secur. J. Glob. Perspect. **24**(4–6), 79–97 (2015)



# Beyond Cookie Monster Amnesia: Real World Persistent Online Tracking

Nasser Mohammed Al-Fannah<sup>1</sup>(✉), Wanpeng Li<sup>2</sup>, and Chris J. Mitchell<sup>1</sup>

<sup>1</sup> Information Security Group, Royal Holloway, University of London, Egham, UK  
nasser@alfannah.com, me@chrismitchell.net

<sup>2</sup> School of Computing, Mathematics and Digital Technology,  
Manchester Metropolitan University, Manchester, UK  
W.Li@mmu.ac.uk

**Abstract.** *Browser fingerprinting* is a relatively new method of uniquely identifying browsers that can be used to track web users. In some ways it is more privacy-threatening than tracking via cookies, as users have no direct control over it. A number of authors have considered the wide variety of techniques that can be used to fingerprint browsers; however, relatively little information is available on how widespread browser fingerprinting is, and what information is collected to create these fingerprints in the real world. To help address this gap, we crawled the 10,000 most popular websites; this gave insights into the number of websites that are using the technique, which websites are collecting fingerprinting information, and exactly what information is being retrieved. We found that approximately 69% of websites are, potentially, involved in first-party or third-party browser fingerprinting. We further found that third-party browser fingerprinting, which is potentially more privacy-damaging, appears to be predominant in practice. We also describe *FingerprintAlert*, a freely available browser add-on we developed that detects and, optionally, blocks fingerprinting attempts by visited websites.

**Keywords:** Browser fingerprinting · Online tracking · Privacy

## 1 Introduction

A number of authors have discussed the very wide variety of readily available attributes collectable by websites from a visiting browser, enabling websites to uniquely identify browsers and potentially track them; this is known as *browser fingerprinting* [1, 9, 24, 31]. Although the range of retrievable attributes, as well as methods for retrieving them, have been widely discussed, relatively little has been published regarding the real-world prevalence of browser fingerprinting, who is deploying it, and the types of attributes collected to achieve it. This issue clearly merits further investigation, and has motivated the work described.

Browser fingerprinting is becoming an increasingly serious privacy concern despite some apparently benign applications (see Sect. 2.2). Its virtually

permanent nature<sup>1</sup> is something that might be subject to future regulation, much as the use of cookies has recently received the attention of regulators in Europe. Its use is virtually invisible to users and there is no direct way of preventing it. Moreover, we found that the four browsers used by more than 88% of web users<sup>2</sup> (i.e. Chrome, Internet Explorer, Firefox and Edge) do almost nothing to help mitigate fingerprinting<sup>3</sup>, alert the user to its occurrence, or even provide information about it in user help documents.

We examined the fingerprinting behaviour of the 10,000 most visited websites. We aimed to discover how many websites deploy browser fingerprinting, whether directly or through third-parties. We also examined which attributes are collected. Further, to help raise awareness of this issue, we developed a browser add-on that alerts users whenever a visited website attempts to fingerprint their browser; users can also opt to enable a fingerprinting blocking feature.

The remainder of the paper is organized as follows. Section 2 describes tracking and browser fingerprinting, and reviews relevant prior art. In Sect. 3 the collection of data from 10,000 websites is described; the results obtained are reported in Sect. 4 and analysed in Sect. 5. In Sect. 6 we discuss the relationship with the prior art. Section 7 describes the *FingerprintAlert* add-on, and the paper ends with discussion and conclusions in Sect. 8.

## 2 Background

### 2.1 Online Tracking

Online tracking (or web tracking) is the process of monitoring a user's online activities; entities that perform tracking are known as *trackers* [22]. The methodology used in our study, like that of many other studies, cannot conclusively determine if a website is actually tracking users; we simply observe whether they collect attributes from browsers that would allow them to track via browser fingerprinting. In line with common usage, we refer to recipients of fingerprintable data (whether first- or third-party) as trackers.

In practice, the most common motive for online tracking is to enable online behavioural advertising. This describes the practice by web advertising companies of tracking users' online activities in order to display personalised and targeted advertisements [40]. Additionally, tracking is used as a tool for market research [24]. There are two main approaches to online tracking—stateful tracking involving the use of cookies<sup>4</sup>, and stateless tracking, including the use

<sup>1</sup> Some browser attributes change over time (e.g. browser version) but uniquely identifying browsers is usually still possible [41], and uniquely identifying the hosting platform is also possible if a different browser is used [8].

<sup>2</sup> The most commonly used browser data was retrieved from <https://www.netmarketshare.com/browser-market-share.aspx> [accessed on 01/07/2018].

<sup>3</sup> Firefox has a limited set of options to thwart fingerprinting.

<sup>4</sup> A web cookie is a small amount of data sent by a website as part of an HTTP response and then stored by the browser. The browser then provides the contents of the cookie back to the same server in subsequent HTTP requests [6].

of browser fingerprinting [24] as defined in Sect. 2.2. In this paper, following the seminal work of Eckersley [9], we focus on the latter.

In some ways, browser fingerprinting is a more reliable method of tracking than the use of cookies [23], and it appears that browser fingerprinting is increasingly being used for this purpose. Unlike browser fingerprinting, cookies are stored on user devices and so can be controlled or deleted by users. In particular, the use of a *private browsing mode*<sup>5</sup> as provided by many browsers, whilst limiting the use of cookies does very little to protect users against browser fingerprinting [4]. Furthermore, while modern browsers provide a user-selectable *Do Not Track* option, this apparently does not prevent widespread tracking [2].

## 2.2 Browser Fingerprinting

Browser fingerprinting enables user web activity to be tracked. It relies on learning properties of a browser and its host platform, including both hardware properties and software state (cf. the term *device fingerprinting* [18]). Browser fingerprinting typically involves a web server performing some combination of: (a) collecting and analysing information contained in HTTP request headers, and (b) downloading JavaScript to the browser which collects and sends back information gathered from browser APIs. Examples of collected information include: screen resolution, CPU/GPU model, and names of installed fonts<sup>6</sup>. As in these examples, collectable attributes relate to both browser and host platform.

Tracking web users has long been possible by using cookies. However, the absence of a cookie (e.g. because it has been deleted by the user) means that the device can no longer be tracked [9]. By contrast, browser fingerprinting requires no files to be stored on the user's device, its effectiveness partly depends on the browser, and users have virtually no control over it [4]. It can be used for tracking web users by creating a unique ID derived by combining collected attributes [20].

Four widely discussed uses of browser fingerprinting are: targeted advertising [2, 22]; social media sharing [22, 33]; analytics services [2, 22]; and web security [2, 38]. Of course, browser fingerprinting has other uses, e.g. to act as a second layer of authentication [9] or to enhance the effectiveness of CAPTCHAs [3]. However, even in these cases the server gets the benefit, and the user is often not informed that fingerprinting is in use [43]. Determining the exact reason(s) why a website deploys browser fingerprinting is extremely difficult.

Browser fingerprinting websites perform it either as a first-party or a third-party (or both). That is, a website may download JavaScript to the browser, which can send the collected attributes back to either its own site (first-party fingerprinting) or to a third-party site (third-party fingerprinting) [35]. It is even possible that some website operators are not aware that a third-party is performing browser fingerprinting via their website [10]. This could arise because third-party fingerprinting sites typically provide client websites with the JavaScript

<sup>5</sup> Modes of this type, which have various names, are intended to enhance the privacy properties of the browser [42].

<sup>6</sup> A demonstration of the wide range of information collectable from any browser is available at <https://fingerprintable.org/test>.

which collects and sends the attributes used for fingerprinting and in return, the third-party site provides a range of services to the client website (e.g. data analytics or social plugins). As a result, some website operators may not know what data the third-party JavaScript collects from user browsers, or what it might be used for.

In the context of tracking, first-party fingerprinting gives relatively little information to a website—it merely enables multiple visits by the same browser to be linked, and gives no information about other visited websites. If the user identity is known by other means (e.g. because the user logs in) it can also indicate when this user is employing multiple devices [2]. Third-party fingerprinting, on the other hand, is much more privacy-damaging in that it enables browsers (and hence users) to be tracked across multiple websites. Later in this paper we report on the websites that perform the majority of third-party tracking.

### 2.3 Previous Work

Back in 2010, Eckersley [9] first described how the collection of a range of apparently trivial and readily-available browser attributes, such as time zone, screen resolution, set of installed plugins, and operating system version, could be combined to uniquely identify a browser; he gave this process the name browser fingerprinting. Since then, many other authors, including Mowery et al. [27,28], Boda et al. [7], Olejnik et al. [32], Fifield et al. [16], Takei et al. [36] and Mulazzani et al. [29], have described a range of ways of enhancing its effectiveness. In parallel, and motivated by the threat to user privacy posed by browser fingerprinting, a number of authors, e.g. Nikiforakis et al. [30], Fiore et al. [17] and FaizKhademi et al. [11] have proposed ways of limiting its effectiveness.

The BrowserLeaks website (<https://www.browserleaks.com>) and Alaca et al. [5] catalogue a wide range of types of information that could be used for browser fingerprinting. Upathilake et al. [39] have also classified some of the most widely used methods for fingerprinting. Browser fingerprinting is clearly very effective; for example, in a large-scale study, Laperdrix et al. [20] observed that an average of 86% of desktop and mobile browsers possess a unique fingerprint; other studies [9,27] have reported similar results (80–90%). It is important to note that some of the attributes that can be used for fingerprinting vary between desktop and mobile platforms; as a result the efficiency of fingerprinting also varies between platform types [20]. For example, a device model name can be retrieved from a mobile browser *user agent* but not from its desktop counterpart.

We conclude this brief review of the prior art by summarising previous work with a similar scope to that of this paper, namely examining the prevalence and nature of browser fingerprinting. In 2015, Libert [23] published the results of a study of third-party HTTP requests utilized for browser fingerprinting. Acar et al. [2] performed a large-scale study of fingerprinting focussing mainly on detection by whether a site examined the set of installed fonts. More recently, Le et al. [21] followed a similar approach, but based detection on use of the canvas API rather than the installed fonts. Englehard et al. [10] performed one of the most comprehensive studies in this area, although they focussed on tracking

in general and not just on stateless (fingerprinting-based) tracking. Englehardt et al. examined the JavaScript downloaded by websites to browsers, a potentially rich source of information, using their own tool, OpenWPM. According to the authors, this tool performs better than many other similar tools such as FPDetective [2]. However, the use of automated tools to examine JavaScript has limitations, in that tools can only look for scripts they are programmed to identify, regardless of the nature of data collected by a tracker. Metwalley et al. [26] also examined the prevalence of tracking; however, they looked at a relatively limited number of websites (500) and aimed to detect all types of online tracking via passive measurements rather than looking specifically at fingerprinting.

## 2.4 Motivation

Despite the fact that browser fingerprinting has been extensively studied, relatively little information has appeared on its prevalence and the browser attributes that are collected in practice. To the authors' knowledge, no other study has listed all the browser fingerprinting attributes that are collected by a large set of real-world websites. This observation motivates the work described in the sequel, in which we describe a study of the fingerprinting behaviour of the 10,000 most popular websites. Unlike the work of Englehardt et al. [10] and Acar et al. [2], we chose not to examine the JavaScript itself, but instead monitor the data that is actually transferred back from the browser. While adopting a somewhat similar method, the scale of the study is more than an order of magnitude larger than the study of Metwalley et al. [26].

One important motive for understanding better the prevalence and nature of browser fingerprinting is to help in developing tools that inform the user about fingerprinting, and also enable users to exert control over the degree to which fingerprinting is possible. To this latter end, in Sect. 7 we describe *Fingerprint-Alert*, a browser add-on developed as part of the study, which makes users aware whenever a website is collecting information usable for browser fingerprinting. It also allows all detected fingerprinting to be blocked.

## 3 Data Collection Methodology

### 3.1 Data Gathering

The main objectives of the data collection exercise were to assess the number of websites performing browser fingerprinting, and what types of data are being collected for this purpose. To achieve our objectives, we decided to crawl a large number of well-used websites and to test their data gathering behaviour. We chose 10,000 sites, as this seemed both sufficiently many to generate representative results, and also a manageable number so we could analyse the considerable volumes of data generated. We only looked at the data transmitted, rather than analysing the downloaded JavaScript, for two main reasons: manual analysis of JavaScript on this scale was infeasible, and automated analysis, as noted above,



has limitations. Moreover, the data that is sent was the key issue of concern for us, not so much how it is gathered.

We used a simple method to decide whether a web server is performing browser fingerprinting. To try to “normalize” web server behaviour, we looked only at the interactions that occur when a browser initially visits the homepage of the website, rather than other information gathering exercises that might occur (e.g. when a user tries to log in). So, a website that sends any fingerprinting browser attributes back to its, or a third-party, server at a first visit has been *deemed* to be engaged in browser fingerprinting; the precise criterion used to decide whether a site is fingerprinting is given in Sect. 3.3.

### 3.2 Experimental Set Up

In order to select which websites to crawl, we retrieved the top 10,000 websites from the freely available Majestic list of the one million most visited websites<sup>7</sup>. We wrote a program to crawl the homepages of these websites to discover if they employ browser fingerprinting techniques at the point when the website is first loaded (i.e. prior to any interaction). This of course means that we missed websites that employ interaction-triggered fingerprinting. The crawler was created using Selenium WebDriver<sup>8</sup>, a Python script, the *FingerprintAlert* add-on, and the Chrome browser (details of the crawler software components and the device used can found in Appendices A.2 and A.3). The Python script instructs Selenium to visit the 10,000 websites in the list, wait for each to fully load, and then wait for a further short period before moving to the next website.

The delay is included because, in preparatory work, we manually visited 50 websites on the list and found that some only relayed information after a delay ranging from one second to several minutes following the full loading of the page. Such waits seem likely to be both to allow the various elements of the web page to be loaded and executed and to take account of dynamic content being continuously loaded (e.g. advertisements). We set the short delay to 3 s; this was a fairly arbitrary choice, although it was long enough to cause a number of websites to transmit data, although not sufficiently long to make the crawling process significantly more time consuming.

The add-on collects and stores all data that is relayed from the browser to one or more web servers using the GET, POST or HEAD HTTP methods<sup>9</sup> [15], i.e. the commonly used means by which information, including attributes used for fingerprinting, is relayed from browser to server. Whether or not the data was sent SSL/TLS-protected, i.e. using HTTPS [34], was also recorded.

<sup>7</sup> Majestic is a website specializing in web usage statistics, and provides a daily-updated list of the top one million websites, <https://majestic.com/reports/majestic-million> [accessed on 09/10/2017].

<sup>8</sup> Selenium is open-source software used to automate browsers for testing purposes—see <https://www.seleniumhq.org>.

<sup>9</sup> The quantity of data that can be relayed using GET or HEAD is very limited, whereas POST allows the transmission of very large volumes (megabytes) of data.

The crawling process took approximately 300 h to complete. It took this long for several reasons, including that some websites took several minutes to fully load, and that Selenium occasionally crashed. In such cases, the crawler was restarted manually, where we re-crawled websites after a crash to ensure we did not miss any data.

### 3.3 Data Processing

Prior to the full crawling process we initially crawled a smaller sample (approximately 1,000 of the websites) to test the crawler. In this process we indiscriminately collected all data sent (if any) from the browser to web servers. Manual examination of the collected data revealed it included information unrelated to the visiting device or the browser (e.g. the URLs of displayed advertisements), i.e. of no interest to this study. Most importantly for our purposes, we were able to identify fingerprinting attributes that had unique formats or values (e.g. screen resolution:  $1920 \times 1080$ ) that made automatic detection possible. Using these preliminary findings, we programmed our crawler to automatically detect a set of 17 attributes (as listed in Appendix A.1). The crawler used regular expressions to examine relayed data and match them with the prepopulated attributes.

The presence of one or more of these attributes in data returned by the browser was used to determine whether or not a website was engaged in fingerprinting. This set of 17 attribute types includes many of the attributes whose use for fingerprinting is most widely discussed, so we believe that the presence or absence of an attribute of one of these types is a reasonable indicator of whether fingerprinting is being performed.

However, other attributes are much more complex, and hence are difficult to automatically identify. In subsequent manual analysis of the recorded data, we were able to identify many additional attributes because they were labelled by name in the captured data. To perform this task automatically would have been extremely difficult because some sections of the recorded data were not parsed, and the substrings of the data that were parsed varied in format (unsurprisingly given the absence of any standards for data formats for transferred attributes).

In order to manually identify fingerprinting attributes in the collected data, we first used publicly available scripts to retrieve a large set of fingerprinting attributes from the browser that was used to run the experiments (the scripts we used can be found at <https://github.com/fingerprintable>). We then attempted to match these values with the values in the collected data. Once we completed the matching, we manually inspected the matches found; this was necessary to ensure that the matches found were genuine and not coincidental similarities in strings or numbers. In most cases the match was confirmed by finding labels followed by the expected values in the collected data.

### 3.4 Challenges Addressed

We faced a number of challenges in both implementing crawling and processing the collected data. First, websites are unlikely to admit use of browser

fingerprinting, and so we can only attempt to judge their behaviour based on the types of information retrieved from the browser, and when it was collected. As mentioned earlier, there is a wide range of attributes that, when put together, can be used to create a unique device fingerprint. Identifying and monitoring all such attributes is very challenging, especially since new attributes seem to arise frequently (given continuously evolving browser functionality). Moreover, many websites cause the browser to send a series of data strings back to the server; automatically, or even manually, identifying what these data represent is highly non-trivial. It was not always possible to parse the data sent since there is no standard for such data transmissions; indeed, some websites may deliberately obfuscate the data they send. It was therefore impossible to fully interpret all the data. Fortunately, there are certain attributes that are easily identifiable because of their special format and range of values, such as screen resolution (e.g.  $1920 \times 1080$ ), fonts (e.g. Arial), or geolocation coordinates (e.g. 51.4167,  $-0.5667$ ).

It is very difficult to determine the minimum number of attributes needed to produce a unique fingerprint. Fingerprint uniqueness depends on many factors, including the range of values of an attribute, how often it changes, and how different it is between one browser/platform and another. As a result, we made the simplifying assumption that a website is deemed a tracker if it causes a browser to send at least one of the 17 attributes given in Appendix A.1.

As our crawler was Selenium-based, it suffered from the known crashing problem [10] on certain websites, e.g. when it was unable to fully load all the elements of a website. In such cases the crawler had to be manually restarted. On average, Selenium crashed once in every 155 visited websites. Moreover, Chrome add-ons are limited to 5 MB of storage and so, to ensure that the collected data did not reach that limit, we programmed the crawler to stop after every 200 visited websites, yielding an average of 3 MB of collected data. However, Selenium usually crashed before reaching the 200-website limit.

The 10,000 websites took an average of 19s to fully load. Our tests were performed using an Internet connection with a minimum bandwidth of 40 mbps, and so connection limitations are unlikely to be the reason for the loading delays. The time to load a website noticeably increased as we went through the list of crawled websites, i.e. the less popular websites loaded more slowly. So, in future similar experiments, we would recommend that crawlers should not timeout until at least 20s have elapsed.

## 4 Results

The data collected in this study, as well as the tools we used for data collection and analysis, are available at <https://github.com/fingerprintable>. The dataset includes the contents of all HTTP messages sent by and to the crawled websites that attempted fingerprinting. This includes the data retrieved from the visiting device (i.e. the device used for data gathering), as well as the domain names of the sender and receiver of the data. Figure 1 shows a sample of a complete block of data from amongst those collected in our study.

```

{
  "toURL": "<tracker URL is shown here>",
  "referer": "<visited website URL is shown here>",
  "method": "POST",
  "data":
  "fp={\"os\": \"2\", \"browser\": \"Chrome61,0,3163,100\", \"fonts\": \"
  undefined\", \"screenInfo\": \"1280*800*24\", \"plugins\": \"Portable
  Document Format::internal-pdf-viewer::Chrome PDF
  Plugin|:mhjfbmdgcfjbbpaeojofohoefgiehjai::Chrome PDF
  Viewer|:internal-nacl-plugin::Native Client|Shockwave Flash 24.0
  r0::internal-not-yet-present::Shockwave Flash|Enables Widevine
  licenses for playback of HTML audio/video content. (version:
  1.4.8.1008)::widevinecdmadapter.plugin::Widevine Content
  Decryption Module\"};"
}

```

Fig. 1. Excerpt of collected data

Using a combination of automated parsing and manual inspection, we detected the transmission of 284 different attribute types. We further detected 1,914 distinct fingerprinters. 70 websites (i.e. 0.7%) timed out (e.g. because the website did not respond) during the crawling process and thus were fully, or partially, excluded from our findings. Overall, 6,876 (68.8%) of the crawled websites collected data from visiting browsers (as first- or third-parties) that could be used for browser fingerprinting. We refer to such websites as *fingerprinting websites*; of course, despite the name, the fingerprinting websites might not actually be using the collected data for fingerprinting.

Fingerprinting is most commonly performed by third-party sites; 84.5% of the 6,876 sites collecting data sent it only to third-parties. Of the rest, 2.4% were exclusively first-party fingerprinters, with the other 13.1% using both first- and third-party data collection. Over the 6,876 fingerprinting websites, data was sent to an average of 3.42 domains. The largest number of different data-collecting websites to which data was sent for a single visited website was 42.

Fingerprinting websites collected an average of 1.75 KB of data. The third-party websites that collected the most data were yandex (2.9 MB), optimizely (2.8 MB) and casalemedia (2.1 MB). Figure 2 shows the top 10 third-party websites in terms of collected data volume for a single visiting browser.

Of the attributes we can automatically detect, the three most frequently collected were: screen/browser resolution, language, and charset (i.e. character encoding). We found that fingerprinters collected, on average, 5 of the 17 pre-populated attributes. Figure 3 summarises the 10 most frequently collected attribute types. The most widely used fingerprinting third-party was google-analytics<sup>10</sup> (see <https://github.com/fingerpritable> for a complete list of fingerprinting third-parties); google-analytics provides web analytics as well as other web-based services to websites. DoubleClick<sup>11</sup> (Google's online advertising service) was the website that collected the largest volume of data overall.

As noted above, amongst the collected data we were able to identify 284 fingerprinting attributes, which we divided into six categories (see Table 1). The full list of 284 attributes can be found in Appendix B.

<sup>10</sup> <https://analytics.google.com>.

<sup>11</sup> <https://www.doubleclickbygoogle.com>.

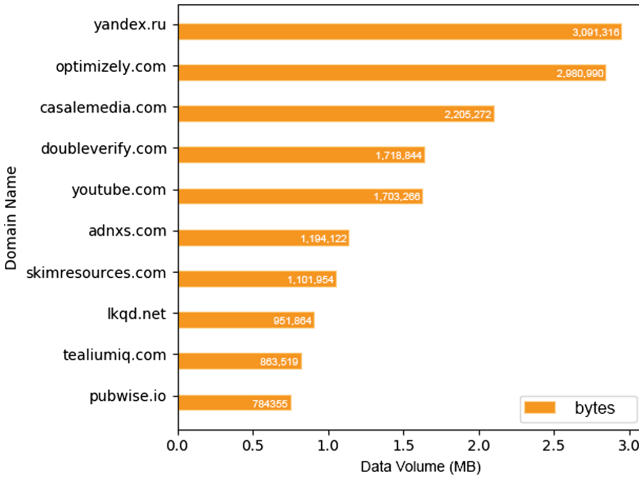


Fig. 2. Top 10 fingerprinters in terms of collected data volume per browser

Table 1. Summary of identified fingerprinting attributes

Attribute type	WebGL	Features	Media	Miscellaneous	Input/Output	Network	Total
Count	114	64	41	35	20	10	<b>284</b>

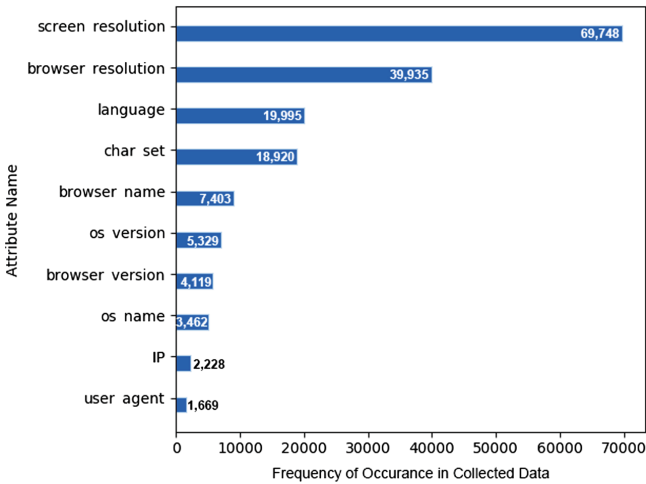


Fig. 3. Top 10 collected attributes

## 5 Analysis

**Processing Collected Data.** The crawler logged every website that relayed data if one, or more, of the 17 pre-programmed attributes were detected. We

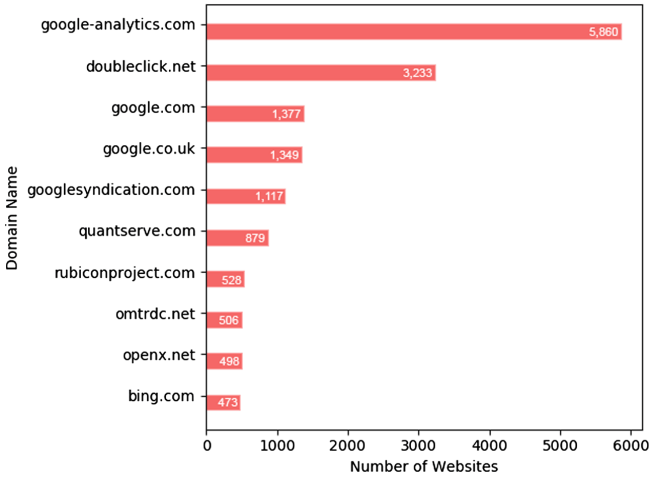
examined random samples of the collected data to identify the presence of any false positives. We found some HTTP messages that contained data that were incorrectly matched with one of the 17 attributes. We wrote a script to remove such records (e.g. if the string `1280088.jpeg` matched with the screen resolution width 1280). This filtering reduced the number of false positives. However, in general, identifying false positives (if any) in the filtered data is non-trivial since the ability to fingerprint browsers typically depends on both the number and type of collected attributes. For example, Mowery et al. [28] have demonstrated that the canvas API alone could be enough to fingerprint a browser, and Laperdrix et al. [20] demonstrated a seemingly successful method of fingerprinting based on a specific set of just 17 attributes.

**Undetected Fingerprinting.** As noted in Sect. 3.2, the crawler only visited the *homepages* of the 10,000 websites. Websites we reported as not deploying browser fingerprinting might nevertheless still be doing so on other pages. Moreover, the attribute collection reported here was unprompted (i.e. no clicking, cursor movements or typing was involved) except for loading of the web page. Through manual visits to selected websites, we found that some websites only cause the browser to send fingerprinting attributes when there are further interactions. Moreover, some websites only retrieved attributes when a user submits a form or logs in, and such cases would be too complex (if not impossible) to capture automatically. The focus of this study is fingerprinting that targets everyone, including those engaged in casual browsing.

**Prevalence of Fingerprinting.** Our study confirms the findings of Englehardt and Narayanan [10] that fingerprinting is commonplace, at least by widely-used websites, and yet there are a relatively small number of entities actually collecting and processing attributes (mainly third-party trackers). Indeed, the top five third-party fingerprinting domains (see Fig. 4) are all part of a single company, Google Inc. This finding is consistent with Libert [23], who found that 78.07% of the top one million websites send data to a Google-owned domain.

We found that 68.8% of the top 10,000 websites are potentially engaged in fingerprinting, although previous studies yielded rather different results. For example, in 2013, Nikiforakis et al. [31] found that only 0.4% of the top 10,000 websites deployed fingerprinting. A year later, Acar et al. [1] reported that 5% of the top 100,000 websites deployed browser fingerprinting using the canvas API. It thus seems likely that both the prevalence of browser fingerprinting and the number of attributes being collected for this purpose have significantly increased.

**Fingerprinting Attributes.** We attempted to find the fingerprinting attributes reported by Alaca et al. [5] and the BrowserLeaks website in the collected data, including attributes not in the list of 17 attribute types detectable by the crawler. This gave us an indication of the range of attributes that are collected in the real world, as opposed to those discussed in the literature, and also helped us improve the functioning of the add-on described in the Sect. 7.



**Fig. 4.** Top third-party fingerprinting domains

As reported above, we were able to identify the collection of 284 attributes, a much larger number than those reported by previous studies. This is partly explained by the fact that previous studies have searched for a smaller number of attributes; for example Eckersley [9] and Cao et al. [8] looked for just 10 and 53 respectively. The significantly higher number we found also seems likely to be a result of the growing use of browser fingerprinting [2, 31], and the fact that we monitored the HTTP messages transmitted between visited websites and potential trackers as opposed to detecting the presence of pre-identified fingerprinting scripts, as previously widely performed. Most of the attributes we were able to identify are collectable by BrowserLeaks.com. However, BrowserLeaks can also collect many attributes that we did not find any websites to be collecting, including many of the browser features collectable by *Modernizr*<sup>12</sup>.

**Deployment of HTTPS.** Some fingerprinting websites do not use HTTPS to send the fingerprinting attributes which are thus transmitted in plaintext; this is a potentially significant user privacy threat. Of the 1,914 distinct fingerprinters we detected, as many as 683 used only HTTP for attribute transmission, 274 mixed use of HTTP and HTTPS, and the remaining 957 used only HTTPS. That is, 50% of the fingerprinting websites used HTTP at least in some cases for transmitting what could be construed as personally identifiable information. Seemingly, the use of HTTP is more common in less popular websites, as Merzdovnik et al. [25] reported that as many as 60% of the top 100,000 websites performing fingerprinting used HTTP. We identified a fingerprinting website that

<sup>12</sup> A JavaScript library that help websites detect the availability of css and html5 features in a visitor's browser <https://modernizr.com>.

used the WebSocket protocol<sup>13</sup> as well as HTTP. These results apply only to the use of HTTP/HTTPS for transmitting browser attributes, not to whether or not the visited website uses HTTPS.

**Fingerprint IDs.** Some websites cause a browser to send a value that is explicitly labelled *fingerprint* or *fp*, along with fingerprinting attributes. These values are typically strings of alphanumeric characters that appear to function as platform/user identifiers. Evidently, some first- and third-party trackers share such user identifiers [12], allowing them to compile extensive profiles of users. This also means that a website or a tracker could acquire user- or platform-related information without any prior interaction with that user. Such ID-sharing practices clearly make browser fingerprinting-based tracking more privacy-threatening.

## 6 Relationship to the Prior Art

Our study, like that of Libert [23], examined HTTP requests; however, whereas Libert examined only third-party tracking, we also considered first-party tracking, i.e. by the visited website itself. Moreover, we focussed on browser fingerprinting and not on tracking via cookies, a topic that has been extensively examined in the prior art (e.g. Felten et al. [13], Krishnamurthy et al. [19] and Mayer et al. [24]). A further difference between the work described here and several previous studies, including that of Englehardt et al. [10], is that they examined the fingerprinting scripts while we examined the data relayed back to server via HTTP. Most significantly, and as discussed in Sect. 5, we detected a much higher level of browser fingerprinting than previously reported; indeed, our results suggest that fingerprinting is becoming ubiquitous.

Given that this is a rapidly changing and evolving area, it is important to repeat studies frequently, and so one contribution of our work is to reveal the current state of the art. We do not claim that the approach we have adopted is better than other approaches, but it does have the advantage of being based purely on the data itself, and not on the many and various scripts that might be used to fingerprint browsers. Our study has enabled us to give an up to date, fairly comprehensive, and large-scale list of the attributes being used in practice for browser fingerprinting.

## 7 Browser Add-on

**Overview.** As part of the research described here, we developed *Fingerprint-Alert*<sup>14</sup>, a browser add-on compatible with desktop versions of Chrome and Firefox for both Windows and macOS. Based on the preliminary crawling described in Sect. 3.3, we programmed the add-on to detect the same 17 attributes. It is

<sup>13</sup> It is a relatively new full-duplex TCP communication protocol [14].

<sup>14</sup> <https://chrome.google.com/webstore/detail/ielakmofegkdlnlppfikmbceaajdofu>  
<https://addons.mozilla.org/en-US/firefox/addon/fingerprintalert>.



activated whenever a web page is loaded, and checks whether any of these pre-specified attributes are being relayed back to a web server. If the add-on detects such activity, it displays an alert that includes both the sending and receiving URLs. The add-on also provides a detailed report of detected activities, including data relayed and the corresponding destination(s). Finally, the add-on offers a user-selectable option to automatically block detected fingerprinting attempts. If selected, an HTTP message including any of the monitored attributes will be blocked from being relayed back to a remote server. Despite only detecting 17 attributes, these attributes are typically transmitted alongside other attributes which are also blocked, given that they are in the same HTTP message.

**Blocking Feature.** Websites typically send collected data in a series of HTTP messages, and *FingerprintAlert* blocks those messages that contain at least one of the 17 attributes. We found that these attributes are typically transmitted in the same HTTP message as a large number of other fingerprinting attributes, which are also blocked as a result.

As with any add-on that interferes with browser behaviour, the blocking feature of *FingerprintAlert* might cause unexpected results or even break some websites. To ensure it does not cause significant usability issues, we tested it on the 50 most visited websites from our list. We enabled the blocking feature, and spent around two minutes on each website performing actions such as signing up, logging in and clicking on links. During the tests we did not observe any unexpected behaviour or errors except for some glitches on two websites (e.g. unable to load support chat window). Nonetheless, in the unlikely event that the add-on damages a user's experience at a website, the blocking option or the notifications option can easily be disabled. The add-on will continue to record detected fingerprinting attempts even if both these options are disabled.

**Challenges.** Detecting newer or obscure fingerprinting attributes is an obstacle that faces all privacy add-ons [10]. Moreover, websites could choose to conceal transmitted attributes, e.g. using encryption, or use fingerprinting attributes that are not publicly known. Additionally, it is difficult to automatically detect all fingerprinting attribute values, as they may be similar to other data or have no specific set of values. On the other hand, detecting and examining scripts executed on websites is likely to be hindered by changes in code, syntax and execution. For that reason, the add-on notifies the user if any HTTP message sent to a server is found to contain one or more of the selected set of 17 attributes.

**Other Add-ons and Future Improvements.** The add-on complements, rather than replaces, other add-ons that mitigate fingerprinting, such as those that monitor and block fingerprinting scripts (e.g. *Ghoesrty*<sup>15</sup> and *Privacy Badger*<sup>16</sup>). The main purpose of our add-on is to make users aware of

<sup>15</sup> <https://www.ghostery.com>.

<sup>16</sup> <https://www.eff.org/privacybadger>.

fingerprinting attempts as they happen and the identity of domains collecting the fingerprinting data, and as a result increase their awareness of how widespread such practices are. The results of our study could also help in developing new tools designed to thwart fingerprinting. In the future, we aim to improve *FingerprintAlert* by increasing the number of automatically-detectable attributes. This can be achieved by further in-depth examination of the formats and values of attributes that are currently undetectable. Since the crawler is based on the add-on, any future crawls would also be made more effective by such improvements.

## 8 Discussion and Conclusions

Cookies are familiar to many users, especially with the introduction of regulations on their use, such as the so-called cookie law<sup>17</sup> covering tracking whether using cookies or any other technology. These regulations have caused many websites to announce the use of cookies. However, while users can disable local storage of cookies, cookies can be selectively deleted, and cookies expire, browser fingerprinting is virtually outside of user control and is much more permanent; it is thus significantly more threatening to user privacy.

Many authors, e.g. Nikiforakis et al. [30] and Torres et al. [37], have described means of reducing the effectiveness of fingerprinting through browser add-ons or by adjusting user-configurable browser settings. Previously described add-ons typically either hide certain attributes or fabricate their values. While such add-ons can be helpful, they also have well-known limitations; exhibiting an unrealistic set of attributes values is also fingerprintable [31] and could negatively affect the browsing experience (e.g. if screen resolution values are manipulated).

We have shown that browser fingerprinting is being conducted on a significantly larger scale than previously reported, involving the transmission of large volumes of browser and device-specific data to trackers. We also reported on the large number of fingerprinting attributes collected. As other authors have described, browser fingerprinting has significant negative implications for user privacy, and it is therefore important that the web user community is made aware of its prevalence and potential effectiveness. To this end we have developed *FingerprintAlert*, that informs users when fingerprinting is occurring and can also block it. If web user privacy is to be preserved, fingerprinting technology needs to be made user-controllable so users can limit the degree to which they are tracked. Our browser add-on contributes to this by providing users with the option to block browser fingerprinting. In the longer term it may be necessary for regulators to examine ways of limiting the degree to which users are tracked using fingerprinting, and/or for browser manufacturers to find ways of developing browsers that limit how easily one user can be distinguished from another.

---

<sup>17</sup> [http://ec.europa.eu/ipg/basics/legal/cookies/index\\_en.htm](http://ec.europa.eu/ipg/basics/legal/cookies/index_en.htm) [accessed on 14/04/2018].

**Ethical Issues.** Clearly any experiment involving real world websites raises potential ethical issues. However, no data relating to individuals were accessed, no vulnerabilities in websites were discovered or exploited, and all websites were accessed as intended by their providers. Websites were crawled only once, except in cases of a crawler crash where an additional visit was required. All the results are publicly available, as described in Sect. 4.

## Appendix

### A Crawling Components and Environment

#### A.1 Prepopulated List of Attributes

Resolution, OS, OS Version, User-Agent, Browser Name, Browser Version, WebGL Renderer, WebGL Vendor, WebGL Version, GPU, GPU Vendor, Installed Plugins, Language, Geolocation, City, IP Addresses, and Charset.

#### A.2 Crawler Software Components

Component	Details
Browser add-on	FingerprintAlert 1.0
Programming language	Python 3.6.3
Automation tool	Selenium 3.8.1

#### A.3 Computing Environment

Component	Details
Device	MacBook Pro (10.1.1)
OS	MacOS Sierra 12.1
Browser	Chrome 62.0.3202.94

### B Attributes Collected by Fingerprinters

#### B.1 WebGL

aliased line width range, aliased point size range, alpha bits, angle instanced arrays, antialiasing, blue bits, depth bits, experimental-webgl, ext blend min max, ext disjoint timer query, ext frag depth, ext shader texture lod, ext srgb, ext texture filter anisotropic, fragment shader high float precision, fragment shader high float precision range max, fragment shader high float precision range min, fragment shader high int precision, fragment shader high int precision range max, fragment shader high int precision range min, fragment shader low float

precision, fragment shader low float precision range max, fragment shader low float precision range min, fragment shader low int precision, fragment shader low int precision range max, fragment shader low int precision range min, fragment shader medium float precision, fragment shader medium float precision range max, fragment shader medium float precision range min, fragment shader medium int precision, fragment shader medium int precision range max, fragment shader medium int precision range min, green bits, max 3d texture size, max anisotropy, max array texture layers, max color attachments, max combined fragment uniform components, max combined texture image units, max combined vertex uniform components, max cube map texture size, max draw buffers, max fragment input components, max fragment uniform blocks, max fragment uniform components, max fragment uniform vectors, max program texel offset, max render buffer size, max samples, max texture image units, max texture lodbias, max texture size, max transform feedback interleaved components, max transform feedback separate attribs, max transform feedback separate components, max uniform block size, max uniform buffer bindings, max varying components, max varying vectors, max vertex attribs, max vertex output components, max vertex texture image units, max vertex uniform blocks, max vertex uniform components, max vertex uniform vectors, max view port dims, min program texel offset, oes element index uint, oes standard derivatives, oes texture float, oes texture float linear, oes texture half float, oes texture half float linear, oes vertex array object, performance caveat, red bits, renderer, shading language version, stencil bits, unmasked renderer webgl, unmasked vendor webgl, vendor, version, vertex shader high float precision, vertex shader high float precision range max, vertex shader high float precision range min, vertex shader high int precision, vertex shader high int precision range max, vertex shader high int precision range min, vertex shader low float precision, vertex shader low float precision range max, vertex shader low float precision range min, vertex shader low int precision, vertex shader low int precision range max, vertex shader low int precision range min, vertex shader medium float precision, vertex shader medium float precision range max, vertex shader medium float precision range min, vertex shader medium int precision, vertex shader medium int precision range max, vertex shader medium int precision range min, webgl, webgl compressed texture s3tc, webgl compressed texture s3tc srgb, webgl debug renderer info, webgl debug shaders, webgl depth texture, webgl draw buffers, webgl lose context, webgl2, webkit ext texture filter anisotropic, webkit webgl compressed texture s3tc, webkit webgl depth texture, webkit webgl lose context.

## B.2 Features

adblock, application cache, background size, blending, bluetooth, border image, border radius, box shadow, budget, canvas winding, credentials, css animations, css columns, css gradients, css reflections, css transforms, css transforms 3dc, css transitions, drag and drop, flex box, flex box legacy, font face, generated content, get battery, get game pads, get user media, hash change, history, hsla, img hash, inline svg, installed fonts, installed plugins, java enabled, js, media devices, mime

types, multiple bgs, opacity, permissions, post message, presentation, register protocol handler, request media key system access, request midi access, rgba, send beacon, service worker, shockwave flash, smil, svg, svg clip paths, text shadow, towebp, unregister protocol handler, usb, vibrate, web sql database, web workers, webkit get user media, webkit persistent storage, webkit temporary storage, webrtc, websockets.

### B.3 Media

ac-base latency, ac-channel count, ac-channel count mode, ac-channel interpretation, ac-max channel count, ac-number of inputs, ac-number of outputs, ac-sampler ate, ac-state, an-channel count, an-channel count mode, an-channel interpretation, an-fft size, an-frequency bin count, an-max decibels, an-min decibels, an-number of inputs, an-number of outputs, an-smoothing time constant, audio ogg, avc1.42c00d, avc1.42e01e (mp4a.40.2), codecs1, dynamiccompressor, h264, hybridoscillator, mp3, mp4a.40.2, mpeg, opus, oscillator, theora, video mp4, video ogg, vorbis (ogg), vorbis (vp8), vorbis (vp9), vorbis (wav), wav, webm, wm4a.

### B.4 Miscellaneous

app code name, battery level, charging, charging time, charset, collect time, cookie enabled, cpu cores, discharging time, do not track, geolocation, graphics card vendor, hardware concurrency, has timezone mismatch, incognito, indexed db, js heap size limit, languages, local storage, navigator, online, open data base, platform, product, product sub, referrer, renderer, session storage, timestamp, timezone, total js heap size, used js heap size, user agent, vendor, vendor sub.

### B.5 Network

downlink, effectivetype, is proxied, is tor, is using tor exit node, local ip, onchange, public ipv4, public ipv6, rtt.

## References

1. Acar, G., Eubank, C., Englehardt, S., Juárez, M., Narayanan, A., Díaz, C.: The web never forgets: persistent tracking mechanisms in the wild. In: ACM SIGSAC 2014, Scottsdale, AZ, USA, 3–7 November 2014, pp. 674–689. ACM (2014)
2. Acar, G., et al.: FPDetective: dusting the web for fingerprinters. In: ACM SIGSAC CCS 2013, Berlin, Germany, 4–8 November 2013, pp. 1129–1140. ACM (2013)
3. Al-Fannah, N.M.: Making defeating captchas harder for bots. In: Computing Conference 2017, London, UK, 18–20 July 2017, pp. 775–782. IEEE Computer Society, July 2017
4. Al-Fannah, N.M., Li, W.: Not all browsers are created equal: comparing web browser fingerprintability. In: Obana, S., Chida, K. (eds.) IWSEC 2017. LNCS, vol. 10418, pp. 105–120. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-64200-0\\_7](https://doi.org/10.1007/978-3-319-64200-0_7)

5. Alaca, F., van Oorschot, P.C.: Device fingerprinting for augmenting web authentication: classification and analysis of methods. In: ACSAC 2016, Los Angeles, CA, USA, 5–9 December 2016, pp. 289–301. ACM (2016)
6. Barth, A., Berkeley, U.: HTTP State Management Mechanism. RFC 6265, RFC Editor, April 2011
7. Boda, K., Földes, Á.M., Gulyás, G.G., Imre, S.: User tracking on the web via cross-browser fingerprinting. In: Laud, P. (ed.) NordSec 2011. LNCS, vol. 7161, pp. 31–46. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-29615-4\\_4](https://doi.org/10.1007/978-3-642-29615-4_4)
8. Cao, Y., Li, S., Wijmans, E.: (Cross-)browser fingerprinting via OS and hardware level features. In: NDSS 2017, San Diego, California, USA, February 26–March 1 2017. The Internet Society (2017)
9. Eckersley, P.: How unique is your web browser? In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 1–18. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-14527-8\\_1](https://doi.org/10.1007/978-3-642-14527-8_1)
10. Englehardt, S., Narayanan, A.: Online tracking: a 1-million-site measurement and analysis. In: ACM SIGSAC CCS 2016, Vienna, Austria, 24–28 October 2016, pp. 1388–1401. ACM (2016)
11. FaizKhademi, A., Zulkernine, M., Weldemariam, K.: FPGuard: detection and prevention of browser fingerprinting. In: Samarati, P. (ed.) DBSec 2015. LNCS, vol. 9149, pp. 293–308. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-20810-7\\_21](https://doi.org/10.1007/978-3-319-20810-7_21)
12. Falahrastegar, M., Haddadi, H., Uhlig, S., Mortier, R.: Tracking personal identifiers across the web. In: Karagiannis, T., Dimitropoulos, X. (eds.) PAM 2016. LNCS, vol. 9631, pp. 30–41. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-30505-9\\_3](https://doi.org/10.1007/978-3-319-30505-9_3)
13. Felten, E.W., Schneider, M.A.: Timing attacks on web privacy. In: ACM CCS 2000, Athens, Greece, 1–4 November 2000, pp. 25–32. ACM (2000)
14. Fette, I., Melnikov, A.: The WebSocket Protocol. RFC 6455, RFC Editor, December 2011
15. Fielding, R., et al.: Hypertext Transfer Protocol - HTTP/1.1. RFC 2616, RFC Editor, June 1999
16. Fifield, D., Egelman, S.: Fingerprinting web users through font metrics. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 107–124. Springer, Heidelberg (2015). [https://doi.org/10.1007/978-3-662-47854-7\\_7](https://doi.org/10.1007/978-3-662-47854-7_7)
17. Fiore, U., Castiglione, A., Santis, A.D., Palmieri, F.: Countering browser fingerprinting techniques: constructing a fake profile with Google chrome. In: NBiS 2014, Salerno, Italy, 10–12 September 2014, pp. 355–360. IEEE Computer Society (2014)
18. Franklin, J., McCoy, D.: Passive data link layer 802.11 wireless device driver fingerprinting. In: USENIX Security 2006, Vancouver, BC, Canada, July 31–August 4, 2006. USENIX Association (2006)
19. Krishnamurthy, B., Wills, C.E.: Generating a privacy footprint on the internet. In: ACM SIGCOMM IMC 2006, Rio de Janeiro, Brazil, 25–27 October 2006, pp. 65–70. ACM (2006)
20. Laperdrix, P., Rudametkin, W., Baudry, B.: Beauty and the beast: diverting modern web browsers to build unique browser fingerprints. In: IEEE S&P 2016, San Jose, CA, USA, 22–26 May 2016, pp. 878–894. IEEE Computer Society (2016)
21. Le, H., Fallace, F., Barlet-Ros, P.: Towards accurate detection of obfuscated web tracking. In: IEEE MN 2017, Naples, Italy, 27–29 September 2017, pp. 1–6. IEEE (2017)

22. Lerner, A., Simpson, A.K., Kohno, T., Roesner, F.: Internet jones and the raiders of the lost trackers: an archaeological study of web tracking from 1996 to 2016. In: USENIX Security 2016, Austin, TX, USA, 10–12 August 2016. USENIX Association (2016)
23. Libert, T.: Exposing the invisible web: an analysis of third-party http requests on 1 million websites. *Int. J. Commun.* **9**, 18 (2015). <http://ijoc.org/index.php/ijoc/article/view/3646>
24. Mayer, J.R., Mitchell, J.C.: Third-party web tracking: policy and technology. In: IEEE S&P 2012, San Francisco, California, USA, 21–23 May 2012, pp. 413–427 (2012)
25. Merzdochnik, G., et al.: Block me if you can: a large-scale study of tracker-blocking tools. In: IEEE EuroS&P 2017, Paris, France, 26–28 April 2017, pp. 319–333. IEEE (2017)
26. Metwalley, H., Traverso, S., Mellia, M., Miskovic, S., Baldi, M.: The online tracking horde: a view from passive measurements. In: Steiner, M., Barlet-Ros, P., Bonaventure, O. (eds.) TMA 2015. LNCS, vol. 9053, pp. 111–125. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-17172-2\\_8](https://doi.org/10.1007/978-3-319-17172-2_8)
27. Mowery, K., Bogenreif, D., Yilek, S., Shacham, H.: Fingerprinting information in JavaScript implementations. In: W2SP 2011, Oakland, CA, USA, 26 May 2011, vol. 2, pp. 180–193 (2011)
28. Mowery, K., Shacham, H.: Pixel perfect: fingerprinting canvas in HTML5. In: W2SP 2012, San Francisco, CA, USA, 24 May 2012. IEEE Computer Society (2012)
29. Mulazzani, M., et al.: Fast and reliable browser identification with JavaScript engine fingerprinting. In: W2SP 2013, San Francisco, CA, USA, 24 May 2013, vol. 5 (2013)
30. Nikiforakis, N., Joosen, W., Livshits, B.: PriVaricator: Deceiving fingerprinters with little white lies. In: WWW 2015, Florence, Italy, 18–22 May 2015, pp. 820–830. ACM (2015)
31. Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., Vigna, G.: Cookieless monster: exploring the ecosystem of web-based device fingerprinting. In: IEEE S&P 2013, Berkeley, CA, USA, 19–22 May 2013, pp. 541–555. IEEE Computer Society (2013)
32. Olejnik, L., Acar, G., Castelluccia, C., Diaz, C.: The leaking battery. In: Garcia-Alfaro, J., Navarro-Arribas, G., Aldini, A., Martinelli, F., Suri, N. (eds.) DPM/QASA -2015. LNCS, vol. 9481, pp. 254–263. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-29883-2\\_18](https://doi.org/10.1007/978-3-319-29883-2_18)
33. Portokalidis, G., Polychronakis, M., Keromytis, A.D., Markatos, E.P.: Privacy-preserving social plugins. In: USENIX Security 2012, Bellevue, WA, USA, 8–10 August 2012, pp. 631–646. USENIX Association (2012)
34. Rescorla, E.: HTTP over TLS. RFC 2818, RFC Editor, May 2000
35. Roesner, F., Kohno, T., Wetherall, D.: Detecting and defending against third-party tracking on the web. In: USENIX NSDI 2012, San Jose, CA, USA, 25–27 April 2012, pp. 155–168. USENIX Association (2012)
36. Takei, N., Saito, T., Takasu, K., Yamada, T.: Web browser fingerprinting using only cascading style sheets. In: BWCCA 2015, Krakow, Poland, 4–6 November 2015, pp. 57–63. IEEE Computer Society (2015)
37. Torres, C.F., Jonker, H., Mauw, S.: FP-Block: Usable web privacy by controlling browser fingerprinting. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9327, pp. 3–19. Springer, Cham (2015). [https://doi.org/10.1007/978-3-319-24177-7\\_1](https://doi.org/10.1007/978-3-319-24177-7_1)

38. Unger, T., Mulazzani, M., Fruhwirt, D., Huber, M., Schrittwieser, S., Weippl, E.R.: SHPF: enhancing HTTP(S) session security with browser fingerprinting. In: ARES 2013, Regensburg, Germany, 2–6 September 2013, pp. 255–261. IEEE Computer Society (2013)
39. Upathilake, R., Li, Y., Matrawy, A.: A classification of web browser fingerprinting techniques. In: NTMS 2015, Paris, France, 27–29 July 2015, pp. 1–5. IEEE (2015)
40. Ur, B., Leon, P.G., Cranor, L.F., Shay, R., Wang, Y.: Smart, useful, scary, creepy: perceptions of online behavioral advertising. In: SOUPS 2012, Washington, DC, USA, 11–13 July 2012, p. 4. ACM (2012)
41. Vastel, A., Laperdrix, P., Rudametkin, W., Rouvoy, R.: FP-STALKER: tracking browser fingerprint evolutions. In: IEEE S&P 2018, San Fransisco, CA, USA, 21–23 May 2018, pp. 1–14. IEEE (2018)
42. Zhao, B., Liu, P.: Private browsing mode not really that private: dealing with privacy breach caused by browser extensions. In: IEEE/IFIP DSN 2015, Rio de Janeiro, Brazil, 22–25 June 2015, pp. 184–195 (2015)
43. Zimmeck, S., Li, J.S., Kim, H., Bellovin, S.M., Jebara, T.: A privacy analysis of cross-device tracking. In: USENIX Security 2017, Vancouver, BC, Canada, 16–18 August 2017, pp. 1391–1408. USENIX Association (2017)





# Cyber-Risks in the Industrial Internet of Things (IIoT): Towards a Method for Continuous Assessment

Carolina Adaros Boye<sup>(✉)</sup>, Paul Kearney, and Mark Josephs

Birmingham City University, Birmingham B4 7XG, UK

carolina.adarosboye@mail.bcu.ac.uk, {paul.kearney,mark.josephs}@bcu.ac.uk

**Abstract.** Continuous risk monitoring is considered in the context of cybersecurity management for the Industrial Internet-of-Thing. Cyber-risk management best practice is for security controls to be deployed and configured in order to bring down risk exposure to an acceptable level. However, threats and known vulnerabilities are subject to change, and estimates of risk are subject to many uncertainties, so it is important to review risk assessments and update controls when required. Risks are typically reviewed periodically (e.g. once per month), but the accelerating pace of change means that this approach is not sustainable, and there is a requirement for continuous monitoring of cybersecurity risks. The method described in this paper aims to alert security staff of significant changes or trends in estimated risk exposure to facilitate rational and timely decisions. Additionally, it helps predict the success and impact of a nascent security breach allowing better prioritisation of threats and selection of appropriate responses. The method is illustrated using a scenario based on environmental control in a data centre.

**Keywords:** Internet of Things · Industrial IoT  
Industrial Control Systems · Cyber-security · Control systems  
Risk analysis

## 1 Introduction

The US National Institute for Standards and Technology (NIST) defines risk monitoring as “maintaining ongoing awareness of an organisation’s risk environment, risk management program, and associated activities to support risk decisions” [6]. Nevertheless, it is not unusual for risk monitoring to be done as a discrete activity, once over a period of one, or even several, months with a low level of integration with the operational processes.

The Industrial Internet of Things (IIoT) present important concerns regarding cyber security including risks where consequences go beyond the realm of information systems to interact with the physical world. Therefore, it is advantageous to have timely information about the possible development of a threat scenario.

Suspicious events that can be detected during operation occur frequently and can overload Security Operations Centre (SOC) personnel with data. Introducing a risk-based approach to automated threat detection would allow prioritising security resources and improve decision making. Many methods fail to do this by focusing only on the threat and on the direct consequences, detaching the analysis from the operational impacts, and from the business context. The solution proposed in this research contributes to improving IoT cyber security by monitoring risks continuously. The main idea is to provide a holistic view of the potential impacts of an attack considering how the consequences at an operational level can affect business processes and strategic objectives. The aim of developing this method is to provide relevant, accurate and timely information about cyber-security risks in IoT systems.

While the focus of the method is to adjust risk indicators in near real time, it is necessary to have a good level of understanding about the variables that will be used in the calculations.

The method considers a variety of inputs divided in two groups: dynamic and static. Dynamic inputs will provide (near) real time information about the state of the system to a risk calculation engine that will update the key risk indicators. This should shorten response times for allocation and adjustment of security controls. Continuous updates to the risk treatment plan will procure a better integration between operational processes, risk management, and security processes.

As one of the main “blind-spots” in IoT security is the physical layer, it is proposed to use anomaly detection techniques to monitor variables that can be correlated with possible security issues. Examples are electricity consumption, server performance, and other side-channel information. Establishing direct correlations between an anomaly and its root cause will be challenging and in many cases it will require the involvement of an expert. Also, it may be more difficult to obtain these data compared with other dynamic inputs such as software and network monitoring, because not all of them will be necessarily provided by already available detection tools. Addressing this is among the main challenges that this research project will face.

At the current stage of this research, a conceptual model has been developed with the potential to be adapted to different sorts of IoT systems. Section 2 of this paper explains the problem and current gaps that are addressed. Section 3 gives an example of a use case to provide a setting for explaining the method. Section 4 gives a the general description of the method. Section 5 provides a threat scenario based on the case described in Sect. 3 and explains how the method would work in this case. Section 6 mentions relevant considerations and future challenges of this project and Sect. 7 provides the conclusions.

## 2 Outline of the Problem

Although there are many expectations about introducing new technologies in the industrial control system domain there are also many legacy systems that

cannot be easily replaced. These systems are still widely used and deployed and will need to coexist with the concept of Industry 4.0. One important concern is that their original design did not consider security sufficiently for the current levels of connectivity. Industrial Control Systems (ICS) is closely related to IoT in the sense that they both fit within the definition of an “ecosystem of interconnected devices and services that exchange and process data” [9]. Throughout this paper, the term IoT will be used under the understanding that Industrial Control Systems fit among this definition. Some authors will refer to these systems as Industrial IoT (IIoT). Examples of use cases in IoT can be found but are not restricted to the following industries [21]:

- Transportation
- Health-care
- Government
- Public safety and military
- Retail and hospitality
- Food and farming
- Manufacturing and heavy industry
- Entertainment and sports
- Energy and utilities
- Finance and banking
- Education
- Information and communications technology

In most of these industries, performance, time to market and cost pressures have been a priority over cyber-security [21]. The lack of standards and regulations, and poor security awareness of manufacturers and users has not helped to improve this situation. In the past (and in some cases, still in the present) electro-mechanical or cyber-physical systems based their security mostly on isolation and perimeter security. The circumstances have changed and the vulnerability of these systems has increased. Even critical systems that are isolated from public networks present risks. For example, the malware Stuxnet, discovered in 2010, was allegedly infiltrated to an Iranian nuclear plant through an USB drive connected to one of the computer terminals. This terminal was connected to the control system and was used as foothold to spread the malware to the Siemens PLCs of the plant. This is an example of the “air gap myth” which proves that isolation by itself is insufficient.

Although attacks on IoT systems are nothing new, the amount of connected IoT systems currently exceeds the human population [10, 14], giving more opportunities to attackers. An industrial report released this year based on the study of different attack vectors in industries reveals that in 82% of the cases an internal attacker could have penetrated the industrial system from the corporate network. Significant flaws in network segmentation and separation of privilege were also found, among many other vulnerabilities [25]. Attacks on Symantec’s IoT honeypots almost doubled in less than a year [29]. According to Cisco, no industry vertical is safe from cyber-attacks [3] and it is believed that IoT devices

“are becoming the attack infrastructure of the future” [1]. Cyber-crime has become to be known as a profitable business and cyber-weapons also have started to be commonly used by nations for surveillance and national intelligence. Smart TV’s have already been known to be part of plans to develop tools for espionage, and successful cases of sabotage of national critical infrastructure have been attributed to nation states.

Because IoT systems are based, in part, on computer and network systems, they inherit all their security issues, as well as presenting additional cyber-risks. Their complex architecture increases their attack-surface [16,21] by the addition of devices that interact with the physical world. The variety of hardware involved will have distinctive requirements and constraints which makes security more challenging. In many cases, typical security mechanisms could be not feasible or be insufficient [26]. Limitations in memory and processing capabilities, as well as real time response requirements present constraints to encryption and authentication processes. Also, special attention regarding physical security is required as often the systems have components distributed in a wide area. The use of wireless communications has an inherent risk enhanced by the variety of protocols and enabling technologies. There are fewer standards, regulations, and overall less experience in IoT security [21] and manufacturers tend to have less knowledge in the matter than professionals from the software development world [4]. Whereas in information systems the main concern usually is related to confidentiality, in IoT systems implications of a cyber attack go beyond information theft. Risks can include also damage of physical assets, and even threat to human life [26]. For example by compromising the integrity or availability of critical systems such as life support equipment or systems working in safety-critical environments.

Currently, an important amount of available literature proposes solutions for particular aspects of IoT security such as authentication, secure communications, and attack modelling. These solutions are relevant, but not sufficient by themselves to provide acceptable levels of security. Security issues of one layer of an IoT system cannot be solved in another [16]. This means that different solutions need to be integrated and effectiveness monitored in the context of the overall system.

Security should be implemented as a combination of processes, technology, and people [6] so it is important to consider all these factors in the equation. Automated tools can help to deal with big data and recognise patterns of behaviour, but these patterns need to be put in the context of the operational and business processes and their objectives. The input of experts is essential to achieve this.

Several methods have been developed for IoT cyber security risk assessments based on existing techniques, including game theory [28], fuzzy logic [17], and Bayesian Networks [31]. Some of these methods are general and others focus on specific type of system. A review of 24 existing cyber security risk assessment

methods applied for SCADA systems was done in [2] where the main opportunities of improvement that were found are the following:

1. Addressing context establishment
2. Overcoming attack or failure orientation<sup>1</sup>
3. Accounting for the human factor
4. Capture and formalisation of expert opinion
5. Improvement of the reliability of probabilistic data
6. Evaluation and validation
7. Tool support

Risk analysis continues to be understood as a discrete activity, often done using spreadsheets or other tools which are not integrated with operations and are fed manually with data. Nevertheless, the NIST recommends transitioning to near real time risk management [7]. With new threats and vulnerabilities been discovered on a regular basis, it is likely that many of the data used in a risk assessment would expire in a short period of time. This would make the results irrelevant. Very little academic work has been done related to real time or continuous risk evaluation.

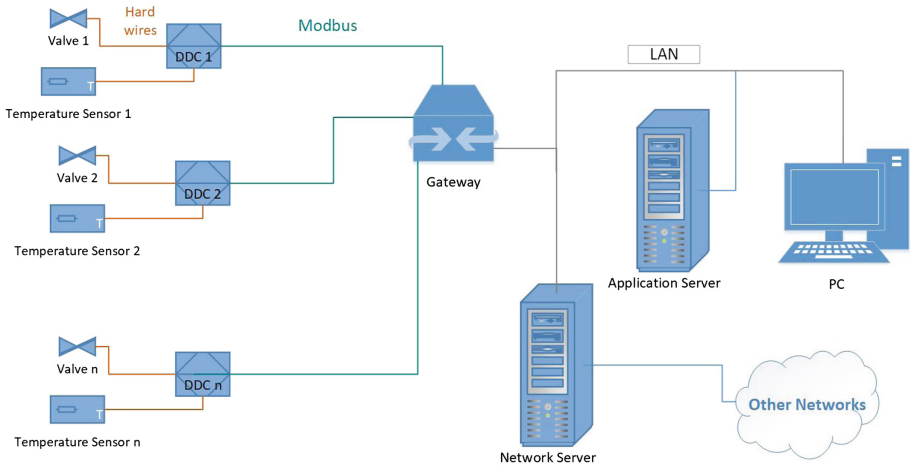
A limited amount of research proposing dynamic or real time cyber security risk evaluation methods has been published [12,13] but most of them are not specific for IoT or IIoT. Other models for real time risk assessment reviewed were mainly focused on threat and anomaly detection and did not consider the impacts. Anomaly detection can be useful to detect threats in an IoT system by comparing variables with a model of their expected behaviour. However, the picture will be incomplete if this information is detached from its context. Several publications about anomaly detection in IoT, Industrial Control, and SCADA systems propose techniques such as machine learning [11,17,20], data mining [24], statistical analysis [8,32], and hybrid methods [19]. The work published by Zhang et al. [33] on incident prediction and risk assessment for industrial control systems considers both real time processing and asset valuation, but, it only provides proof of concept through simulation experiments on a single type of system. In conclusion, there is a lack of risk assessment methods for IoT that are both holistic and dynamic and that have been tested in different scenarios.

### 3 Example of a Risk Scenario in IoT System

A simplified temperature control system will be used to illustrate the method. Temperature control is use case that can be found in domestic, commercial, and industrial environments. Nevertheless, in different domains the system will typically present different characteristics, types of technology, and architectures. This research, rather than in IoT domestic or consumer devices, is focused on Industrial Systems. Temperature control can have different purposes in industries. For example, avoiding products such as food and chemicals to decompose

---

<sup>1</sup> This means basing the analysis only on known attack mechanisms and failure modes.



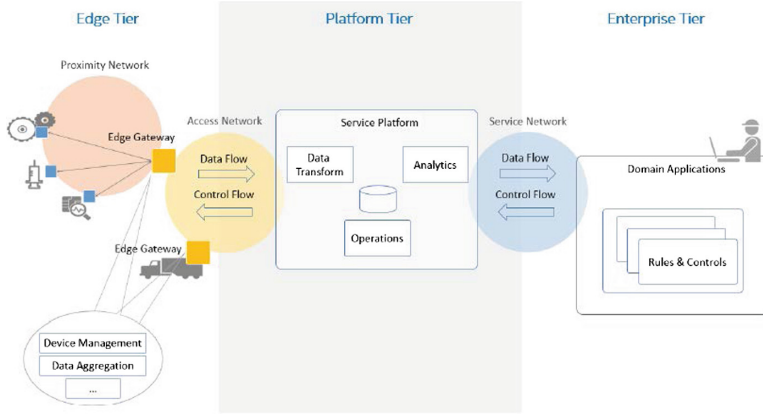
**Fig. 1.** Diagram of temperature control system

or degrade, or allowing different process to perform in optimal conditions. A temperature control malfunction will have different consequences depending on the business processes involved. Understanding these consequences is crucial when assessing risks, and also a key part of finding possible signals of compromise.

The scenario developed in this example corresponds to the temperature control system for a Data Centre, and is shown in Fig. 1. The scenario was validated with an engineer that works in a consultancy in Chile who has over ten years of work experience in configuration, installation, and maintenance of Industrial Control Systems.

A data centre, ideally should operate in an environment with a temperature between 24 and 27 °C [23]. This includes a margin of error, as servers typically can tolerate up to 30 °C. At higher temperatures, servers do not achieve their best performance, and their fans will need to spin at their maximum rate, increasing power consumption. To avoid the temperature surpassing an established limit, Direct Digital Control (DDC) devices are used which are connected directly to temperature sensors and to the control valves for the heating and cooling systems. The controllers communicate with a Building Management System (BMS) that runs in an application server. A local PC located in the same premises is in charge to run the control and monitoring software interface. The BMS sends alerts via email and SMS messages when an event requires attention.

Figure 2 shows a diagram from the Industrial Internet Reference Architecture (IIRA) [18] that will be used to describe the system based on the definition of three tiers. The edge tier collects data from the real world through the proximity network where sensors and actuators are connected. The platform tier exchanges, consolidates, and processes data from the other tiers. This can include commands generated in the enterprise tier to control variables in the edge tier. The enterprise tier implements domain-specific applications and provides user interfaces.



**Fig. 2.** Three tier architecture pattern from the IIRA implementation viewpoint

The following specification represents the system “as is”, before applying any risk treatment:

**Edge Tier:** comprises sensors, actuators and DDCs. The sensors and actuators are hard wired to the controllers providing inherent trust. They use electrical signals to communicate. Thus, they are not “smart”. The DDCs have a keyboard allowing authentication through a 4 digit security code. In the perimeter network, the protocol used by the controllers is Modbus (other protocols commonly used in these systems are Bacnet and Lonworks). The controllers are connected to a gateway that converts the signals to a standard internet protocol (TCP/IP) and connects to a Local Area Network (LAN).

**Platform Tier:** comprises the BMS software installed in an application server that processes operational data. The access network which connects the Platform tier to the Edge Tier is the same as the service network, corresponding to the LAN. The system is insulated from other networks for security reasons, except for the connection to an email server that allows sending alerts to operators in case of certain events and the connection to a service that sends SMS alerts. There are no firewalls or any network monitoring and detection mechanisms in place. The network server has separate cards for the LAN and other networks.

**Enterprise Tier:** comprises a monitoring and control software running in a PC terminal. Authentication is done through user and password without enforcing a secure credentials. There are no defences against brute force attacks in place. Privilege separation options include user, administrator and engineer roles. There is no remote connection, therefore the software only can be accessed within the perimeter. Remote monitoring is based only on the alerts sent by the BMS system.

**Physical Security:** authorised personnel is authenticated through an ID card, a 4 digit password, and their digital print. Special authorisation is required

for visitors and contractors which need to register. Although they should be accompanied by an authorised member of staff at all times, some contractors might be left alone for small periods of time, as sometimes they require to work there for several hours. The hardware of the control systems has often physical ports open. Personnel only visit the data centre when it is necessary, but there is nobody permanently in the area.

**Cyber-security policies and practices:** before the risk assessment, the BMS was considered in the cyber-security policies of the data centre. Some isolated cyber-security controls were in place, such as some degree of authentication, and the control for physical access described. There is not a clear differentiation of roles and privileges and most users just share credentials. This includes contractors. Network security is based only on isolation, and for this reason malware detection is not considered important. At the enterprise platform level the software registers and stores event logs but they are not monitored. Regarding the configuration of the temperature control settings, there is no registration of any changes or events and there are no configuration management policies in place. Backups of the system are done every six months, but there are no assurance processes to audit this or any other cyber-security practice.

## 4 Description of the Method Proposed

This project aims to make use of different sources of data to analyse cyber-risks in a continuous basis, integrating this activity with the operational process. The objective of the method is to generate useful and meaningful information for decision makers. A “decision maker” is any actor that is in position to make a decision that can affect security. These decisions can be related to business operations that can cause collateral effects in security or to security management itself. Figure 3 shows a general view of the method. The Security Operations Centre (SOC) which is the area that monitors and deals with security issues on an organisation will be provided with a more comprehensive view of attack vectors, by including IoT and operation technologies (industrial systems) in their scope. They will also be able to establish priorities for alerts regarding to the level of risk involved. The risk analysts will be allowed to monitor risks continuously, evaluating the effectiveness of security measures and control, and providing up-to-date inputs to decision making processes that involve or affect security.

While the whole purpose of risk management is to improve decision making, the landscape changes too quickly to have a picture of the situation without



**Fig. 3.** Illustration of the method



expecting it to vary in a short time. Different internal and external factors, will continuously shape the degree of risk. New information can modify the levels of uncertainty regarding occurrence of an event, and also internal and external changes can affect risks factors. This means that security plans based on previous evaluations may become quickly obsolete. In the case of IoT and IIoT, there are more attack vectors and less visibility of the system from end-to-end in comparison with IT systems. Thus, analysing, monitoring and managing risks is critical.

Figure 4 shows a conceptual model describing the main building blocks of the method proposed. The idea is that the results should generate decisions that affect the risk treatment plan of the organisation, modifying the situation of the security levels of the IIoT system. The inputs for the risk calculations will come from three main sources: detection tools, system’s variables, and a knowledge base. The first two type of sources are categorised as “dynamic inputs” and would be transmitted in a continuous stream. The data that is stored in the knowledge base is categorised as “static inputs” which either remains unchanged or is subject to eventual updates. The risk calculation engine will process the information about threats, vulnerabilities and impacts and issue alerts in the event of any condition that might change the risk scores.

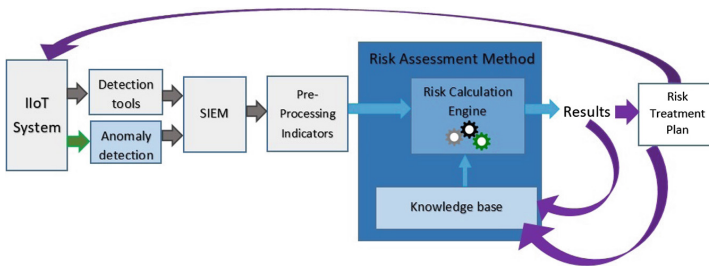


Fig. 4. Building blocks of the method

### 4.1 Dynamic Inputs

These inputs are captured in run time from tools for malware detection, intrusion detection, network traffic analysis, and logs monitoring, and threat intelligence sharing. An anomaly detection tool is in charge of analysing data that can reveal signs of threats, including operational data. A module based on a SIEM software will be used to process the dynamic inputs that will feed the risk calculation engine. The anomaly detection tool will help detecting issues related to the edge tier.

Currently, there are existent commercial tools for anomaly detection based on machine learning. While normal operation conditions of the system and business rules could, in theory, eventually be learned by artificial intelligence (AI), this will need a training period and stable conditions over time. Therefore, known business rules and thresholds should be previously set. Also it is

necessary to include expert's knowledge regarding other variables like dependencies between processes, impacts at different levels, regulations and strategic decisions. Therefore, while machine learning and AI techniques will contribute to anomaly detection, it is proposed to combine it with other methods. It is not aimed in this project to work with tools that work following a 100% unsupervised dynamic. The human factor and experts' knowledge are elements that require to be acknowledged in risk management [2].

## 4.2 Static Inputs

Static inputs correspond to the data that cannot be collected online. The reasons could be that there are no tools available to collect this information on real time or, that the data just does not change continuously. For example, the valuation of an asset, quantification of impacts, and risk acceptance criteria. These inputs will be provided during the set up process of the tool. The data will be stored in a knowledge base and updated in a periodic basis or after events. An example of such events is a change in the risk appetite of the organisation. The provenance of this data will be diverse, some inputs will be loaded from tools or data bases and others manually. Examples of inputs stored in the knowledge base are: asset inventory, asset and impact valuation, initial threat quantification, traceability between threats and assets, and between processes and business objectives, and definition of normal and abnormal states of operation.

## 4.3 Risk Calculation Engine

The risk calculation engine would be in charge to process the inputs and generate the results. This engine will be implemented in a software tool. The risk calculation engine would be based on different standards to quantify vulnerabilities, threats, and impacts, the three variables that define a cyber-risk. The FAIR method (Factor analysis of information Risks) [30] is a useful start point to understand the different factors that are involved in a cyber-risk. This method defines the vulnerability as a combination of the "threat capability", the resourcefulness of the threat agents to act against an asset, and the "control strengths", the probability that the current controls resist the attack. While this definition is conceptually useful, the quantification of these two factors, as defined by this method can present problems. Both are defined according to their position within a probabilistic distribution of the threat population, meaning that it is necessary to be able to make plausible assumptions about the possible threat agents. The Common Vulnerability Scoring System (CVSS) [5], provides a score from zero to ten depending on eight variables which are related to the vulnerability. These variables are: attack vector, attack complexity, privileges required, user interaction, scope, confidentiality, integrity, and availability. Additionally, temporal and environmental metrics help giving a score are used for more accuracy within an specific context.

Different methods were reviewed for quantification of threats. Some methods base the probability of a threat event in the frequency of occurrence [15,30].

This approach can be plausible in environments that maintain similar conditions over time. Nevertheless, in cyber-security assuming that threats will behave in the future following the same trend they have done in the past is dangerous. It has to be considered that often new attack mechanisms and zero-day vulnerabilities appear. The model proposed by SANDIA [22] analyses and scores threats by building a profile according to seven different attributes: intensity, stealth, time, technical personnel, cyber-knowledge, kinetic knowledge, and access. Different combinations of this attributes are used to describe 8 different threat profiles, where 1 represents the highest level of threat and 8 the lowest.

To calculate the impact, it is common transforming every consequence into monetary values, because it is an useful way to add up and compare impacts of diverse nature, such as time loss and reputation damage, among others. As a mean of normalising this value, it will be suggested within this method to use a ratio between the total impact of a risk and a referential budget that the organisation will define according to its risk appetite.

The quantification of risks done in the initial assessment would be subject to continuous updates during operation mode. This updates will be related to the threat analysis, which is the factor that presents the higher levels of uncertainty. The threat value then, is the risk component that will be subject to change dynamically according to the information provided by the dynamic inputs. When events that imply possible threats are detected, they will have an effect of modifying the quantification of the corresponding threat values, and therefore, the risk scores.

#### 4.4 Results

Continuous re-calculations of Key Risk Indicators (KRI) will be performed in run time for monitoring purposes, and stored in a data base for ex-post analysis. The risk analyst and SOC operator will have different views of the KRI according to their roles. The risk analyst will be more interested in monitoring the behaviour of the risks and evaluating the effectiveness of current controls, as a mean to make better informed security-related decisions. The SOC operator will be more focused on alerts and any indicator of an attack developing in any of its stages. This is explained through an example in Sect. 5. Risks are based on uncertainty. Thus, in the cases of an imminent attack, this is not considered a risk but an issue. In most occasions a cyber-attack will not take place in a single instance but it will follow a sequence of stages. Detection of an issue such as an unauthorised access or malware presence, can help to avoid the risk of an attack progressing into further stages, like privilege escalation, maintaining foothold, and establishing command and control capabilities.

#### 4.5 Initial Risk Assessment and Continuous Monitoring Dynamic

The method will consider two stages: the initial risk assessment and the continuous risk assessment. In the first stage the initial KRI are calculated and the risk

monitoring tool is configured according to the context. This activity will condition the success of the continuous risk assessment. Therefore, it is crucial to develop a good understanding of the likelihood and impacts of a breach. Different forms and questionnaires based on standards (e.g. ISO 27005) will be developed to capture expert’s opinion and guide the set-up process of the tool in a way that cyber risks can be mapped with their impacts at all levels of the organisation, including the business point of view. The second stage is the continuous risk assessment which consists in the recalculation of risk scores according to the information provided by the detection tools.

## 5 Demonstration of the Method Through the Example

To demonstrate how the continuous risk assessment method would work, a threat scenario was built using the example of Sect. 3. The initial risk assessment and tool configuration stage will consist in evaluating the system “as is”. After this, a risk treatment plan is developed, incorporating controls to mitigate risks and residual risks are formally accepted. Then, the continuous risk assessment process starts.

### 5.1 Initial Risk Assessment and Tool Configuration

In this stage risks are identified, quantified and evaluated. It is expected that after this assessment a risk treatment plan will be developed, incorporating controls. The next stage will provide the means to monitor the effectiveness of these measures in a continuous basis. The risk management approach will follow the process described by the ISO27005 standard shown in Fig. 5.

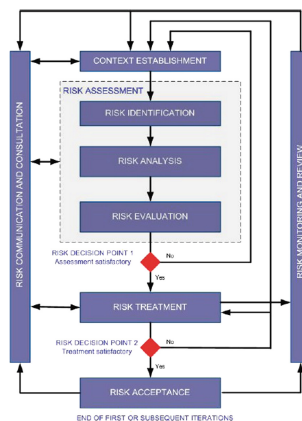


Fig. 5. Overview of the risk management process, ISO27005

**Context Establishment.** In this stage it is defined the scope, including assets that need protection. To illustrate the method some of the attack vectors related

to the temperature control system will be reviewed. The assets considered in the assessment belong to both the data centre domain and the company business domain, because the temperature control system can be used as a bridge to other systems. Other control systems in the data centre such as CCTV, fire alarm system, electrical supply and UPS, as well as servers and network equipment are included in the monitoring. Their dependencies with the temperature control system need to be established at this point. Operational and business processes, as well as support processes such as IT, finance, and human resources should be considered. To quantify impacts, a series of possible consequences were listed including damages to assets implying replacement or reparation, effort spent in recovery, downtime, fines, and compensations to customers and third parties, and damage in brand reputation. This last can be reflected in loss of revenue or need to expend in marketing strategies to recover the trust of the customers. All the impacts are quantified in monetary values. To put this value in context according to its significance to the organisation, an impact indicator is calculated to evaluate the impact in terms of the monthly budget that the company assigns for cybersecurity.

The risk acceptance criteria is established to define a risk frontier depending on the organisation's risk appetite. Once established this frontier, any risk that is out of limits is reviewed and included in the risk treatment plan. The decision of retaining any risk outside of these limits needs to be formally approved by senior management. We will suppose this company defined that any risk that is rated medium or higher or whose potential impact exceeds a predefined value should be reviewed. The roles involved in this risk assessment were: risk analyst, GRC manager, SOC analyst, Security manager, data centre manager and operators, the asset owners of all assets identified in the assessment, and senior management (CSO, CIO, CEO).

**Risk Assessment.** In this stage, risks are identified, analysed, and evaluated. The analysis process consist in understanding possible attack paths, and consequences of a breach, and making plausible assumptions that allow identifying and quantifying threats, vulnerabilities, and potential impacts. Threats and vulnerabilities are linked with possible assets compromised and impacts at different levels. The likelihood is estimated based on the threat and vulnerability levels, and the total impact is quantified considering the effects on the operational processes as well as in the business.

From the analysis of the system "as is", it was found possible that internal personnel or a contractor could download malicious code in the computer terminal using a USB drive. Thus, bridging the air-gap. The motivations of the attacker which could be many, for example, been bribed by a competitor to sabotage the servers. The malicious code could have different purposes, which means that there are different risks. The present example will focus on two risks: Risk 1 will correspond to the manipulation of the temperature to increase failure rates of the servers and Risk 2 is related to the use of the BMS as a bridge to access the business networks. Each of the two risks is analysed independently.

Risk 1 was analysed obtaining a low level of vulnerability and a low level of threat. The impact was also considered low, since the system could be easily reset and reconfigured in the case of been compromised. As a result, this risk was rated as low. Risk 2 was rated as medium, because, although the threat and vulnerability were also considered low, the impact was considered high as critical processes and data could be compromised. Both risks were evaluated by comparing them with the risk acceptance criteria (risk frontier) concluding that only Risk 2 needed to be included in the risk treatment plan.

**Risk Treatment.** After the risk assessment, a risk treatment plan is developed, to address all the risks that exceeded the risk frontier choosing from a set of three possible actions: reduce, avoid, or share. A fourth possible action is retaining the risk, which can be consider under senior management approval. In the case of Risk 2, using the BMS to access other networks, it was decided to reduce the risk by introducing an firewall in the local network.

**Tool Configuration.** Residual risks are analysed and evaluated after the risk treatment plan is put in action. The information is loaded in the knowledge base, including the scores of all the risks identified. As Risk 2 was reduced, it is re-evaluated and defined as low which means that it is within acceptable limits. At this point, all the risk scores should be below the risk frontier. There could be two reasons why some risks might not meet this condition. One is that the control defined in the risk treatment plan has not been fully implemented yet, and the other is that there is a formal authorisation of senior management to retain certain risk. Each risk is linked with the events that might change the current scores in order to start the continuous monitoring process.

## 5.2 Continuous Risk Assessment

In the previous stage, an initial iteration of the information security risk management process was done. The following step consists in monitoring the risks and perform subsequent iterations of the whole process. Through the continuous risk assessment, it is intended that these iterations will be repeated in short intervals of time and whenever a certain development of events requires it, rather than in a periodic basis.

For the current example, we will imagine that a contractor introduces malicious code through an USB drive in the computer terminal to change the temperature control settings. The malware will work in a similar way as Stuxnet, changing temperature setting of the DDCs and disguising this action by displaying the original set values of temperature. Therefore, there will not be any condition that triggers an alert directly related to overheating. For example, the set value is 25 °C, and all the instances of the system display 25°, but the temperature will really be 50 °C. The malicious actions are scheduled to take place at hours where is more unlikely to be staff on site to notice this, allowing persistence of the attack and increasing the potential damage. But, while the malware might remain unnoticed, the anomaly detection system will detect an unusual behaviour in other processes which are linked to this. The fans of the servers will

spin faster, increasing the electrical power consumption and the servers might not perform in their best capacity. Considerable outliers in the energy consumption levels and in the performance of the servers will trigger an alert by the anomaly detection system. The risk calculation engine will process this information and modify the risk indicators of all the risks that can be related to this event, including Risk 1. An alert will be sent to the SOC operator to investigate the situation and call for action. In parallel, another alert will be sent to the risk analyst indicating that Risk 1 has now been rated as high. The risk analyst will also have the information about the processes that the affected servers are running and how they impact the business.

The previous example shows how a risk that was initially considered low changes to a higher value dynamically, through the development of events. The threat score is amplified when suspicious events are detected resulting in a higher risk value. It has to be noticed that this would lead to two courses of action. First, the SOC operator can generate an immediate response regarding remediation and recovery actions. Second, the risk analyst will generate all the necessary actions to develop a risk treatment plan to establish controls to avoid this risk becoming again an issue in the future. Examples of these actions are blocking all unused physical ports by default, restricting the privileges to download software, and adding malware detection and stricter regulations regarding not leaving any third party unattended in the perimeter.

## 6 Considerations and Challenges

The development of this method will not be exempt of challenges. There are still unsolved issues in this project which need to be tackled in future stages of this project for the method and tool to have a practical application.

Because IoT and IIoT systems are very heterogeneous, the method only can be tested in a limited amount of systems. Tailoring guidelines can be provided for adapting the method to different use cases, and it would be a matter of further research to confirm its applicability in different contexts and scenarios. Big data issues including processing, storage and retrieval of information will also be a challenge, as well as the development of interfaces between tools and normalisation of the data.

As much as there might be a lot of ground in common with regular IT systems, this project aims to tackle the particular requirements of IIoT. One of the challenges of this is that the amount of processes, stakeholders, dependencies with other systems, and assets involved is bigger. Also, there might be more expectation for these systems to have automated security controls. Nevertheless, it must be recognised that the autonomy of any system will always be within certain limits. Establishing these limits, mapping all the processes affected, as well as providing appropriate rules and training mechanisms to the anomaly detection system is part of the challenge.

False positives is a known problem of detection tools which would affect, as well this method. It is necessary to find solutions that do not undermine the

ability of the method to alert when there is a real threat. The user will have the mission of calibrating the tool by identifying and giving feedback about any misinterpretation of the data. A case management system would be a possible alternative to support a continuous improvement mechanism for the method. Overall, it is important to understand the data in order to avoid providing misleading results. An example of this is the huge amount of “noise” that failed attack attempts can cause which may lead to think that there is a developing threat when actually according to [22] one attribute that increases the threat is, precisely, stealth. Attacks that are easy to detect and stop might not be a threat at all!

Another aspect is to distinguish cyber-attacks from other issues such as physical attacks or malfunctions of the system. It is considered on the best interest of an organisation to know about any potential threat even if it is not caused by a cyber-attack. Therefore, the detection of an issue whose causes end to be from a different nature, rather than been dismissed, should be reported to the relevant stakeholders. Although the scope of the method is to monitor cyber-risks, from the risk management point of view, other types of risk can also be of interest. For example, in [27] it is argued that physical attacks and cyber-attacks should not be treated separately proposing 4 types of attacks: physical-only, cyber only, cyber-enabled physical and physical-enabled cyber.

## 7 Conclusions

The presence of IoT in several industries and the increasing amount of cyber threats predicts a growth in the demand for cyber security solutions. Developing methods to maintain cyber-situational awareness through a continuous risk monitoring process can support rational and well informed decisions. The approach proposed takes into account the context of the system, as well as the business objectives and priorities. By linking the potential threats with the impacts and vulnerabilities it is possible to do a better prioritisation of security resources. Shorter iterations for risk assessments will make it possible to react in a more timely manner to changes in the environment. The underlying principle is that risk management should not be detached from the system’s operations, it should be integrated, since both processes serve as input to each other.

Currently there are not widely used and tested solutions to evaluate IIoT cyber security risks in run time that include an holistic perspective of the system. The present paper gives a general description of a solution that has the potential of addressing several gaps of existent risk assessment approaches. Considering the context establishment and capturing expert’s opinion is addressed on the initial assessment and tool configuration, and subject to updates. The feedback loop of the system requires experts to be involved in the process as well as to give input to calibrate the method. The method also goes beyond attack or failure orientation because it is not limited to known attack mechanisms. By including anomaly detection and other tools it allows issuing alerts under any event that diverts the system’s behaviour from what is consider normal. This is relevant,



because it is not feasible to analyse all the possible attack mechanisms. The combination of different tools to support this method, as well as the development of the risk calculation engine as an automated tool, will allow the method to be implemented in a practical and effective way. If decision makers are well informed of cyber-security risks, this will allow better application of policies and control mechanisms, improving the overall security of the system.

## References

1. Boddy, S., Shattuck, J.: Threat Analysis Report. The Hunt for IoT. The Growth and Evolution of Thingbots Ensures Chaos (2018). <https://www.f5.com/labs/articles/threat-intelligence/the-hunt-for-iot-the-growth-and-evolution-of-thingbots-ensures-chaos>. Accessed 25 June 2018
2. Cherdantseva, Y., Burnap, P., Blyth, A., Eden, P., Jones, K., Soulsby, H., Stoddart, K.: A review of cyber security risk assessment methods for SCADA systems. *Comput. Secur.* **56**, 1–27 (2016)
3. Cisco: Cisco 2017 annual security report. Technical report (2017)
4. Cook, E., Kearney, P.: Security challenges and cybercrime. *J. Inst. Telecommun. Prof.* **9**, 22–25 (2015)
5. Common vulnerability scoring system sig. <https://www.first.org/cvss/>. Accessed 09 Apr 2018
6. Dempsey, K., et al.: Information security continuous monitoring (ISCM) for federal information systems and organizations: National institute of standards and technology special publication 800–137 (2012)
7. Dempsey, K., Ross, R., Stine, K.: Supplemental guidance on ongoing authorization (2014)
8. Desnitsky, V., Kotenko, I., Nogin, S.: Detection of anomalies in data for monitoring of security components in the internet of things. In: 2015 XVIII International Conference on Soft Computing and Measurements (SCM), pp. 189–192. IEEE (2015)
9. ENISA: Security Recommendations for IoT in the context of Critical Information Infrastructures (2017). <https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot>. Accessed 09 Apr 2018
10. Gartner newsroom (2017). <http://www.gartner.com/newsroom/id/3598917>. Accessed 30 July 2017
11. Greensmith, J.: Securing the internet of things with responsive artificial immune systems. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 113–120. ACM (2015)
12. Henrie, M.: Cyber security risk management in the SCADA critical infrastructure environment. *Eng. Manag. J.* **25**(2), 38–45 (2013)
13. Huang, H., Xie, D.: Real-time network risk evaluation paradigm-inspired by immune. In: 2015 11th International Conference on Natural Computation (ICNC), pp. 786–790. IEEE (2015)
14. IBM Institute for Business Value: Internet of threats. securing the internet of things for industrial and utility companies (2018)
15. ISO/IEC: Iso/iec 27005:2011. information security risk management (2011)
16. Jing, Q., Vasilakos, A.V., Wan, J., Lu, J., Qiu, D.: Security of the internet of things: perspectives and challenges. *Wirel. Netw.* **20**(8), 2481–2501 (2014)
17. Kotenko, I., Saenko, I., Ageev, S.: Countermeasure security risks management in the internet of things based on fuzzy logic inference. In: 2015 IEEE on Trust-com/BigDataSE/ISPA, vol. 1, pp. 654–659. IEEE (2015)

18. Lin, S.W., et al.: Industrial internet reference architecture. Technical report, Industrial Internet Consortium (IIC) (2015)
19. Linda, O., Manic, M., Vollmer, T.: Improving cyber-security of smart grid systems via anomaly detection and linguistic domain knowledge. In: 2012 5th International Symposium on Resilient Control Systems (ISRCS), pp. 48–54. IEEE (2012)
20. Liu, C., Zhang, Y., Zeng, J., Peng, L., Chen, R.: Research on dynamical security risk assessment for the internet of things inspired by immunology. In: 2012 Eighth International Conference on Natural Computation (ICNC), pp. 874–878. IEEE (2012)
21. Macaulay, T.: RIoT Control: Understanding and Managing Risks and the Internet of Things. Morgan Kaufmann, San Francisco (2016)
22. Mateski, M., et al.: Cyber threat metrics. Sandia National Laboratories (2012)
23. Moss, D.L.: Data center operating temperature: The sweet spot (2011)
24. Pan, S., Morris, T., Adhikari, U.: Developing a hybrid intrusion detection system using data mining for power systems. *IEEE Trans. Smart Grid* **6**(6), 3104–3113 (2015)
25. Positive Technologies: Industrial companies attack vectors (2018). <https://www.ptsecurity.com/upload/corporate/ww-en/analytics/ICS-attacks-2018-eng.pdf>. Accessed 25 June 2018
26. Sadeghi, A.R., Wachsmann, C., Waidner, M.: Security and privacy challenges in industrial internet of things. In: 2015 52nd ACM/EDAC/IEEE on Design Automation Conference (DAC), pp. 1–6. IEEE (2015)
27. Smith, B.J., Sholander, P.E., Phelan, J.M., Wyss, G.D., Varnado, G.B., Depoy, J.M.: Risk assessment for physical and cyber attacks on critical infrastructures. Technical report, Sandia National Laboratories (2005)
28. Spyridopoulos, T., Maraslis, K., Tryfonas, T., Oikonomou, G., Li, S.: Managing cyber security risks in industrial control systems with game theory and viable system modelling. In: 2014 9th International Conference on System of Systems Engineering (SOSE), pp. 266–271. IEEE (2014)
29. Symantec: Istr-internet security threat report. Technical report (2017)
30. The Open Group: Fair - ISO/IEC 27005 cookbook (2010)
31. Wang, J., Fan, K., Mo, W., Xu, D.: A method for information security risk assessment based on the dynamic bayesian network. In: 2016 International Conference on Networking and Network Applications (NaNA), pp. 279–283. IEEE (2016)
32. Yang, Y., McLaughlin, K., Sezer, S., Littler, T., Im, E.G., Pranggono, B., Wang, H.: Multiattribute SCADA-specific intrusion detection system for power networks. *IEEE Trans. Power Deliv.* **29**(3), 1092–1102 (2014)
33. Zhang, Q., Zhou, C., Tian, Y.C., Xiong, N., Qin, Y., Hu, B.: A fuzzy probability bayesian network approach for dynamic cybersecurity risk assessment in industrial control systems. *IEEE Trans. Industr. Inf.* **14**(6), 2497–2506 (2017)

## Author Index

- Abdelraheem, Mohamed Ahmed 171  
Adaros Boye, Carolina 502  
Afonso, Vitor 47  
Al-Fannah, Nasser Mohammed 481  
Alkharashi, Abdulwhab 387  
Altuncu, Enes 85  
Andersson, Tobias 171  
Au, Man Ho 369
- Bernardi, Davide 463  
Bicakci, Kemal 85  
Bisio, Federica 463  
Bodden, Daniël 132  
Bösch, Christoph 25  
Boschini, Cecilia 3  
Boyd, Colin 268  
Buchmann, Johannes 289  
Butin, Denis 289
- Cai, Jiahao 101  
Cai, Quanwei 213  
Camenisch, Jan 3
- Davies, Gareth T. 268  
de Geus, Paulo Lício 47  
Demirel, Denise 289  
Deng, Yuan 67  
Duan, Haixin 67
- Emura, Keita 442
- Gehrmann, Christian 171  
Gjøsteen, Kristian 268  
Glackin, Cornelius 171  
Grégio, André 47  
Guan, Le 213
- Hayashi, Takuya 442  
Hiller, Matthias 25  
Hiromasa, Ryo 328  
Hou, Lin 116  
Hu, Lei 101  
Hu, Yang 231
- Ishizaka, Masahito 422
- Jiang, Fangjie 213  
Jiang, Yao 268  
Josephs, Mark 502
- Kalysch, Anatoli 47  
Kargl, Frank 25  
Kawai, Yutaka 328  
Kearney, Paul 502  
Kiziloz, Hakan Ezgi 85  
Kleber, Stephan 25  
Koseki, Yoshihiro 328  
Kuchta, Veronika 403  
Kuppusamy, Lakshmi 250
- Li, Shimin 308  
Li, Wanpeng 481  
Liang, Bei 308  
Lin, Dongdai 116, 151  
Lin, Jingqiang 213  
Liu, Jianwei 231  
Liu, Joseph K. 369  
Liu, Meicheng 151  
Lombardo, Pierangelo 463  
Luo, Fucui 347
- Markowitch, Olivier 403  
Massa, Danilo 463  
Matousek, Matthias 25  
Matsuura, Kanta 422  
Mitchell, Chris J. 481  
Müller, Tilo 47
- Nepal, Surya 369  
Neven, Gregory 3
- Oliveira, Daniela 47
- Rangasamy, Jothi 250  
Renaud, Karen 387  
Rimba, Paul 369

Saeli, Salvatore 463  
Sahkulubey, Muhammet Sakir 85  
Sahu, Rajeev Anand 403  
Sakzad, Amin 369  
Saraswat, Vishal 403  
Schabhüser, Lucas 289  
Sharma, Gaurav 403  
Sharma, Neetu 403  
Slomka, Frank 25  
Steinfeld, Ron 369  
Sun, Siwei 101

Unterstein, Florian 25  
Uzunay, Yusuf 85  
Uzunkol, Osmanbey 250

Vinterbo, Staal A. 192

Wang, Kunpeng 347  
Wang, Xin 308  
Wang, Xuzi 116  
Wu, Baofeng 116

Xue, Rui 308

Yang, Kun 67  
Yasuda, Satoshi 328  
Yu, Bin 369

Zhang, Chao 67  
Zhang, Xiaojuan 151  
Zhang, Yanting 231  
Zhang, Yingjie 101  
Zhang, Zongyang 231  
Zhuge, Jianwei 67